

PhUSE EU connect 2018



Utilization of Python in clinical study by SASPy

Yuichi Nakajima, Principal Programmer, Novartis Pharma K.K
PhUSE EU connect 2018
November 06, 2018

Disclaimer

- The opinions expressed in this presentation and on the following slides are solely those of the presenter and not necessarily those of Novartis.
- Novartis does not guarantee the accuracy or reliability of the information provided herein.

How do you get to Osaka from Tokyo ?



- Jupyter Magic
- SASPy
- Pandas DataFrame

Target audience...



Python Experiences



SAS Experiences

- SAS programmer who wants to learn Python from scratch.

Outline

- Introduction of SASPy
 - Pre-requirements
 - SASPy Installation
- Use of Python in clinical study
 - Data handling / reporting / visualization
 - Possible application example
- Summary
- Conclusion

Outline

- Introduction of SASPy
 - Pre-requirements
 - SASPy Installation
- Use of Python in clinical study
 - Data handling / reporting / visualization
 - Possible application example
- Summary
- Conclusion

Objective

- This presentation will cover
 - How to establish SAS session in Python.
 - How to read SAS dataset and process in Python.
 - How to create summary tables and plots by Python.
- Then discuss a possible usage of Python for clinical study, meaning how Python can support daily reporting activity for SAS programmers.

Introduction

- **Why Python?**

- Python is the one of the most popular programming languages in recent years. It is now getting used in machine learning and AI.
- Big advantage of Python is that plenty of Python libraries to implement various analysis and it can be a “one stop shop” for programming.

- **What is SASPy?**

- SASPy is the module which provides Python Application Programming Interfaces (APIs) to the SAS system.

Pre-requirement

- Focus on “**Windows PC SAS**” connection.
- See reference for other connection type.

- As of July 2018, v2.2.4 is the latest version.

SAS 9.4 or higher.

SASPy-2.2.4

Python3.X or higher.

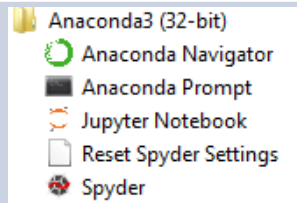
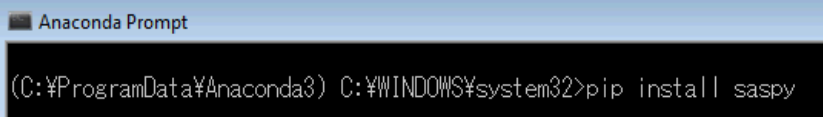
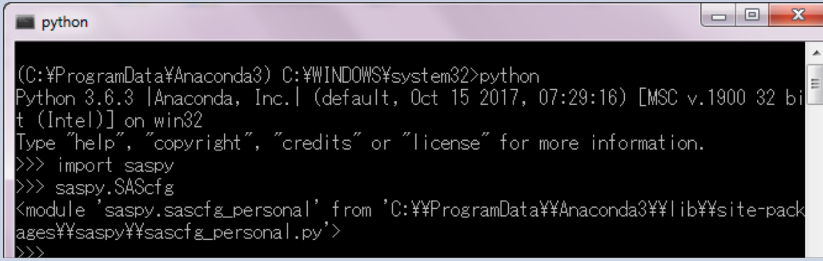
Jupyter notebook

- Previously called “IPython Notebook”.
- Run Python on the web browse.

Available from
[Anaconda distribution](#)

SASPy Installation

- Installation process

#	Process	Note
1	Download SASPy API, then make sure to install below contents appropriately. - SAS 9.4 or higher version. - Anaconda3 Distribution	
2	Open “ Anaconda Prompt ” and install SASPy Enter <code>pip install saspy</code>	
3	After importing saspy (<code>import saspy</code>) in python, need to edit SAS configuration files(sascfg.py) to establish corresponding SAS session (See next slide) File location is obtained by <code>saspy.SAScfig</code>.	
4	Start session to create class instance setting configuration type (Windows) <code>sas = saspy.SASsession(cfgname='winlocal')</code>	

SASPy Installation (Cont.)

- To optimize configuration file to your system environment, you need to create **sascfg_personal.py** instead of **sascfg.py**.

- Usually there are 3 modifications required in sascfg_personal.py.

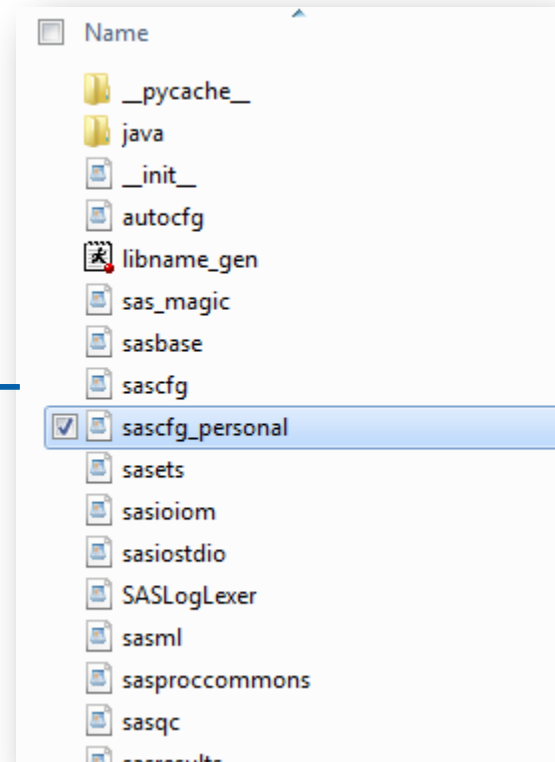
- Update SAS_config_names as 'winlocal' in sascfg_personal.py.

```
SAS_config_names=['winlocal']
```

- Update SAS session to specify java.exe file in your PC.

```
winlocal = {'java' : 'C:\ProgramData\Oracle\Java\javapath\java.exe',  
            'encoding' : 'windows-1252',  
            'classpath' : cpW}
```

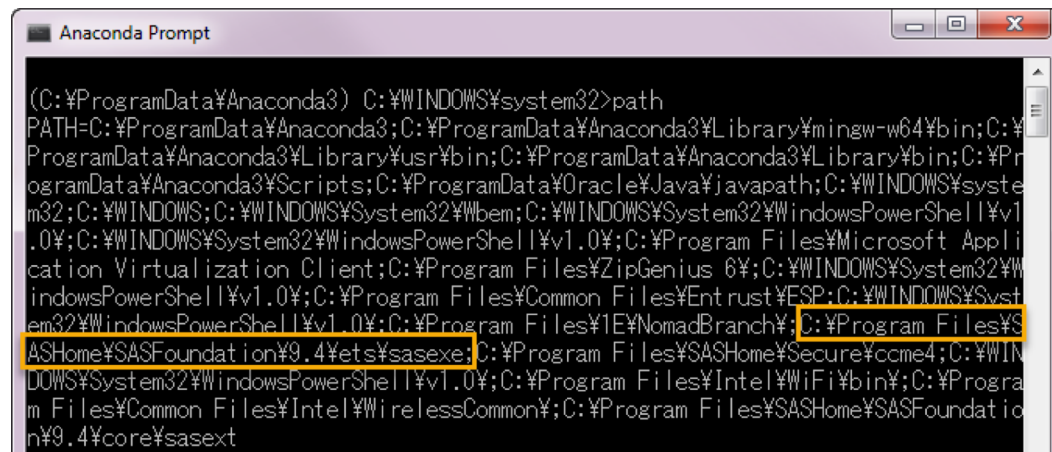
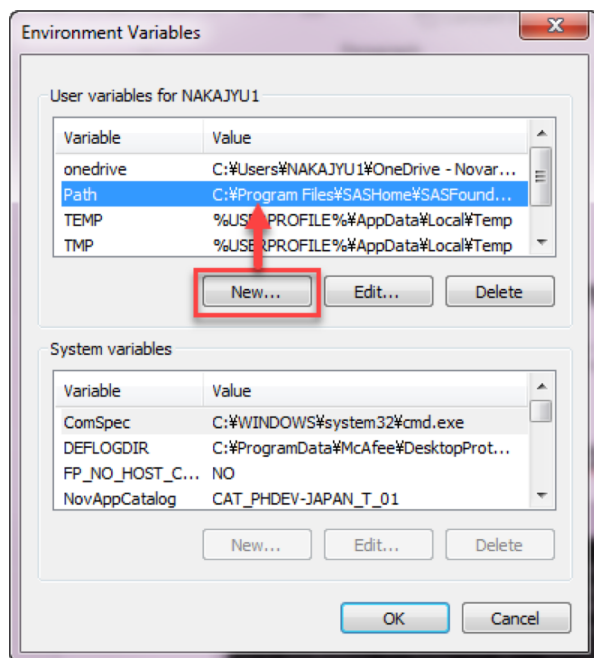
- Windows client class path. Make sure location links are corresponded to your **own** environment.



```
cpW = "C:\\Program Files\\SASHome\\SASDeploymentManager\\9.4\\products\\deploywiz_94476_prt_xx_sp0_1\\deploywiz\\sas.svc.connection.jar"  
cpW += ";C:\\Program Files\\SASHome\\SASDeploymentManager\\9.4\\products\\deploywiz_94476_prt_xx_sp0_1\\deploywiz\\log4j.jar"  
cpW += ";C:\\Program Files\\SASHome\\SASDeploymentManager\\9.4\\products\\deploywiz_94476_prt_xx_sp0_1\\deploywiz\\sas.security.sspi.jar"  
cpW += ";C:\\Program Files\\SASHome\\SASDeploymentManager\\9.4\\products\\deploywiz_94476_prt_xx_sp0_1\\deploywiz\\sas.core.jar"  
cpW += ";C:\\ProgramData\\Anaconda3\\Lib\\site-packages\\saspy\\libname\\connection.jar"
```

SASPy Installation (Cont.)

- Finally to establish local connection, check whether system PATH environment variable is added correctly.
 - “C:\Program Files\SASHome\SASFoundation\9.4\core\sasext”.



Outline

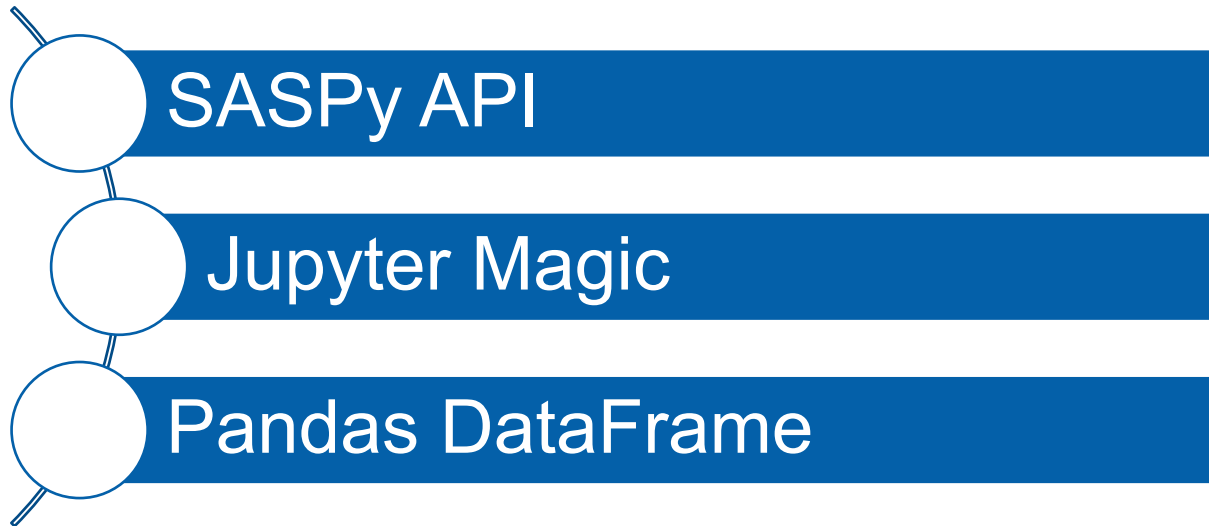
- Introduction of SASPy
 - Pre-requirements
 - SASPy Installation
- Use of Python in clinical study
 - Data handling / reporting / visualization
 - Possible application example
- Summary
- Conclusion

How to handle SAS dataset in Python ?

- After establish SAS session in Jupyter notebook,

```
In [1]: import saspy  
  
In [2]: sas = saspy.SASsession(cfgname='winlocal')  
SAS Connection established. Subprocess id is 17596
```

- you need to select the best choice or combination in data handling from three possible options.



How to handle SAS dataset in Python ?

Choices	Description	Example code of data sorting in Jupyter notebook cell <>: SAS dataset, []: Pandas DataFrame
Jupyter magic	Magic commands is utility command such provided by IPython. Set “%%SAS” on the top of cell, then SAS code can be effective within that python cell.	%%SAS libname temp “xxxxxxx”; proc sort data = temp.<AAAA> out = work.<BBBB>; by USUBJID descending AESEV; run;
SASPy API	SASPy can setup a SAS session and run analytics from Python.	sas.saslib('temp', path="xxxxxxx") <AAAA>=sas.sasdata('<AAAA>', libref='temp' <BBBB> = <AAAA>.sort(by='USUBJID DESCENDING AESEV', out='temp')
Pandas DataFrame (DF)	Pandas is a third party package to handle one dimension data (Vector: Series) and 2 dimension data (Matrix: DataFrame), Set “import Pandas”	[BBBB]=[AAAA].sort_values(by=['USUBJID', 'AESEV'], ascending=[True, False])

Pandas DataFrame

	USUBJID	SITEID	VISIT
0			
1			
2			
3			
...			

- Whereas SASPy can handle SAS dataset directly, this presentation is mainly focusing on **Pandas DF** in order for following data handling, reporting and visualization.
- Advantage of Jupyter magic will be referred in later slide.

Data handling and reporting in Jupyter using SAS dataset

1. SASPy

1) Import SASPy and establish SAS session

```
- >>> import saspy  
- >>> sas = saspy.SASsession(cfgname='winlocal')
```

2) Convert SAS datasets to Pandas DF

```
- >>> sas.saslib('temp', path="C:\\Users\\....")  
- >>> dfae = sas.sasdata2dataframe('ae', libref='temp')  
- >>> dfdm = sas.sasdata2dataframe('dm', libref='temp')
```

Assume
CDISC
dummy
datasets

2. Pandas

1) Import Pandas and merge two DF

```
- >>> import pandas as pd  
- >>> wk = pd.merge(dfdm, dfae, on='USUBJID', how = 'inner')
```

“right”, “left”,
“outer” can be
selected for
merge.

Data handling and reporting in Jupyter using SAS dataset

2. Pandas

2) With a very simple code, summary result is available by `pivot_table()` function.

```
- >>> pd.pivot_table(wk,  
    values = 'AGE',  
    columns = ['ARM'],  
    aggfunc = 'describe')
```

Select several
aggfunc
options.

ARM	Miracle Drug 10 mg	Miracle Drug 20 mg	Placebo
25%	70.000000	66.0	80.0
50%	71.000000	66.0	80.0
75%	72.000000	66.0	80.0
count	6.000000	5.0	5.0
max	72.000000	66.0	80.0
mean	71.000000	66.0	80.0
min	70.000000	66.0	80.0
std	1.095445	0.0	0.0

When aggfunc = 'count'

ARM	Miracle Drug 10 mg	Miracle Drug 20 mg	Placebo
AGE	6	5	5

Data handling and reporting in Jupyter using SAS dataset

2. Pandas

3) Show AE summary table

```
- >>> pd.pivot_table(wk,
                        values      = 'USUBJID',
                        index       = ['AEBODSYS', 'AEDECOD'],
                        columns     = ['ARM'],
                        aggfunc     = lambda x:x.nunique(),
                        fill_value  = 0 )
```

	ARM	Miracle Drug 10 mg	Miracle Drug 20 mg	Placebo
AEBODSYS	AEDECOD			
Cardiac disorders	Palpitations	1	0	0
	Constipation	0	0	1
Gastrointestinal disorders	Diarrhoea	0	1	0
	Haemorrhoids	0	1	0
	Nausea	0	0	1
	Vomiting	0	1	0
General disorders and administration site conditions	Fatigue	0	0	1
Metabolism and nutrition disorders	Decreased appetite	1	0	0
Musculoskeletal and connective tissue disorders	Arthralgia	0	0	1
	Muscle spasms	1	0	0
Nervous system disorders	Dizziness	1	0	0
	Headache	0	1	0
Psychiatric disorders	Agitation	1	0	0
	Anxiety	1	0	1

Note that option
aggfunc = 'count'
doesn't consider
duplication.

Matplotlib for data visualization

- In order to plot data by Matplotlib, first generate 1)figure and 2)sub plot. At least one sub plot must be created.

```
import matplotlib.pyplot as plt
```

1) Call figure instance

```
fig = plt.figure()
```

2) Call subplot

```
ax = fig.add_subplot(111)
```

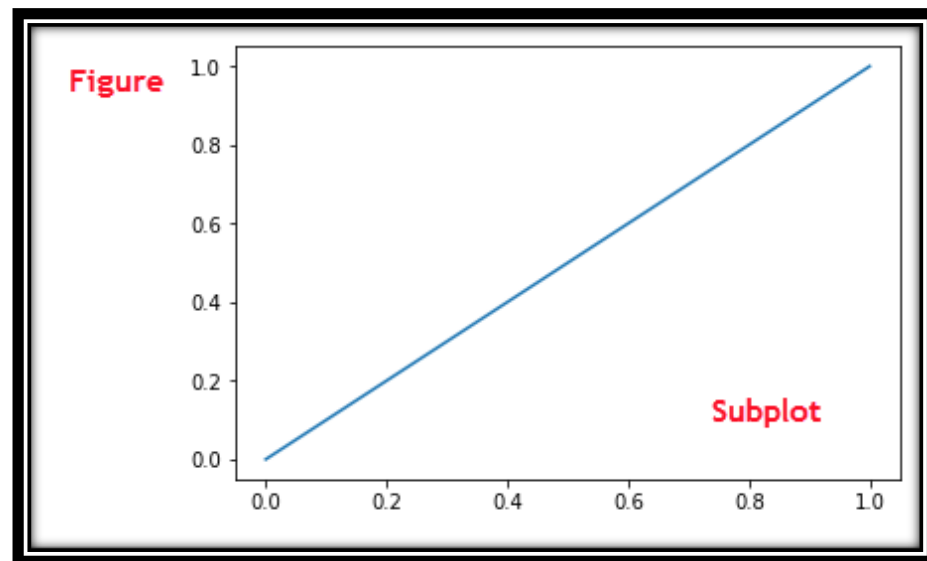
```
dat = [0, 1]
```

Line plot by plot function

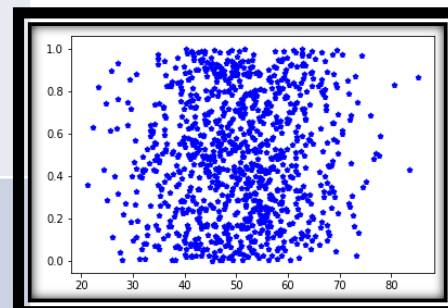
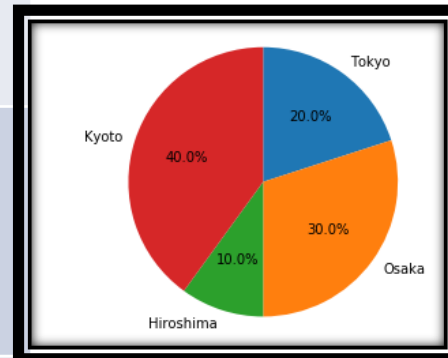
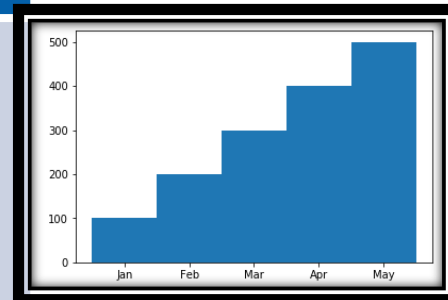
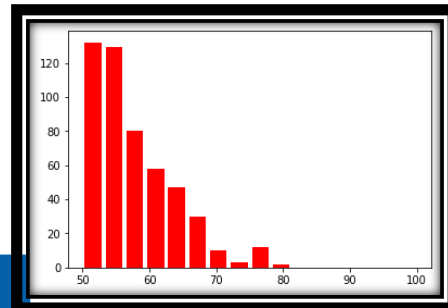
```
ax.plot(dat)
```

Display with show function

```
plt.show()
```



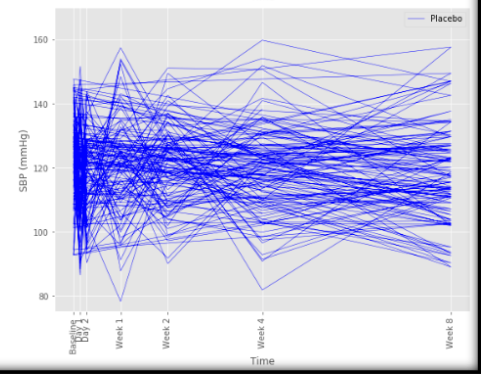
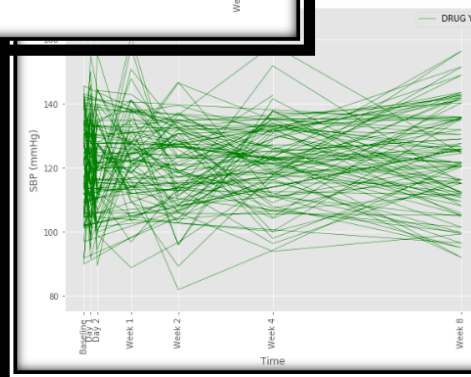
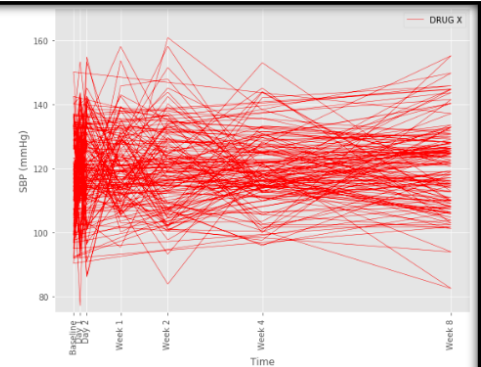
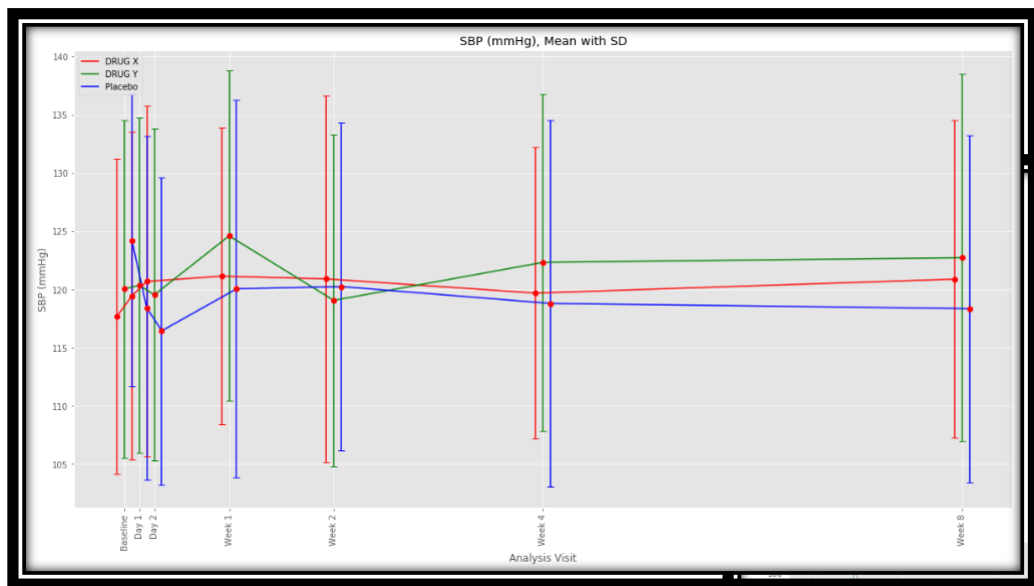
Matplotlib for data visualization



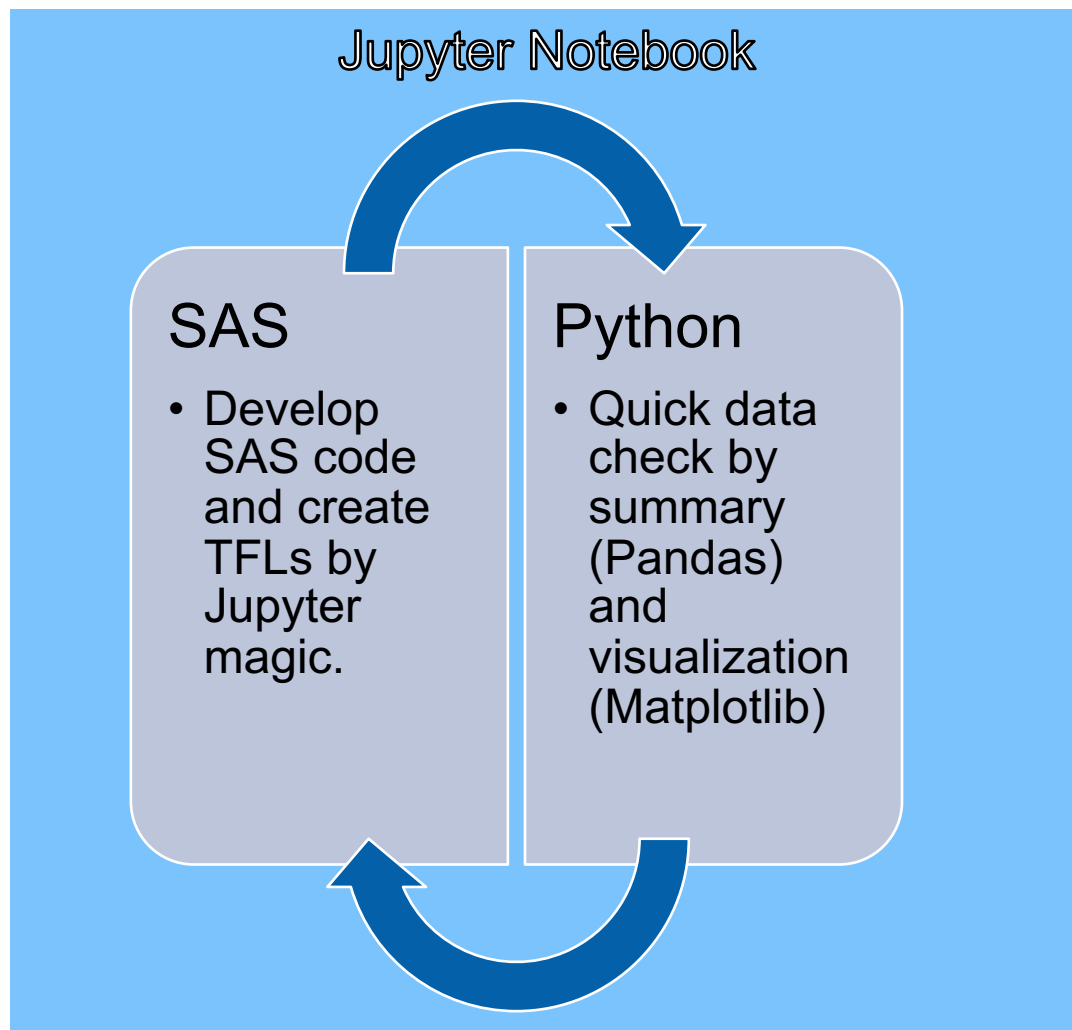
Plot type	Function	Description	Quick examples
To try below examples, run following code first. <pre> >>> import numpy as np >>> import matplotlib.pyplot as plt >>> x1 = np.array([1, 2, 3, 4, 5]) >>> x2 = np.array([100, 200, 300, 400, 500]) >>> x = np.random.normal(50, 10, 1000) >>> y = np.random.rand(1000) </pre>			
Histograms	hist()	Compute and draw the histogram of x.	<pre> >>> plt.hist(x, bins=16, range=(50, 100), rwidth=0.8, color='red') </pre>
Bar charts	bar()	The bars are positioned at x with the given alignment. Their dimensions are given by width and height.	<pre> >>> plt.bar(x1, x2, width=1.0, linewidth=3, align='center', tick_label=['Jan', 'Feb', 'Mar', 'Apr', 'May']) </pre>
Pie charts	pie()	Make a pie chart of array x. The fractional area of each wedge is given by x/sum(x).	<pre> >>> plt.pie(x3, labels=['Tokyo', 'Osaka', 'Hiroshima', 'Kyoto'], counterclock=False, startangle=90, autopct="%1.1f%%") >>> plt.axis('equal') </pre>
Scatter plot	scatter()	A scatter plot of y vs x with varying marker size and/or color.	<pre> >>> plt.scatter(x, y, s=15, c='blue', marker='*', linewidth='2') </pre>

Matplotlib for data visualization

- Indeed, using CDISC dataset, typical clinical study plots can be created.



Possible application example 1



- By using SASPy, **Jupyter notebook can be an editor for SAS programming.**
- Jupyter Magic is enable Python to submit SAS code in Jupyter notebook cell.
- Thus interactive data check and visualization can be added to your SAS programming.

Possible application example 1

- Suggestion of application

SASPy

- Establish connection to SAS
- Read SAS data from TFL.

Jupyter Magic

- Set up SASPy
- SAS code in Jupyter cell by %%SAS.
- Thus TFL data can be processed for

Pandas

- Apply Pandas functions to SAS data
- If dataset are truly analysis ready, it can be converted Pandas DF and apply functions for quick summaries.

```
import saspy

sas = saspy.SASsession(cfgname='winlocal')

%%SAS sas
Jupyter Magic

%let templib=C:\Users\NAKAJYU1\Desktop\tempds;
libname outlib "&templib.";
data _temp;
  RC = filename("filenm", "&templib.");
  length FNAME $400 ;
  DID = dopen("filenm") ;
  if DID gt 0 then do;
    NUM = dnum(DID);
    call symput("1", trim(left(put(NUM, best.))));
    do i = 1 to NUM ;
      FNAME = dread( DID , i ) ;
      output ;
    end;
  end;
run;

%macro loop;
  %do i = 1 %to &1.;
    data _null_;
      set _temp;
      where i = &i.;
      call symputx("item", FNAME);
    run;

    libname %scan(&item., 1, ".") XPORT "&templib./&item" ;
    proc copy in = %scan(&item., 1, ".") out = outlib;run;
  %end;
%mend;
%loop;
```

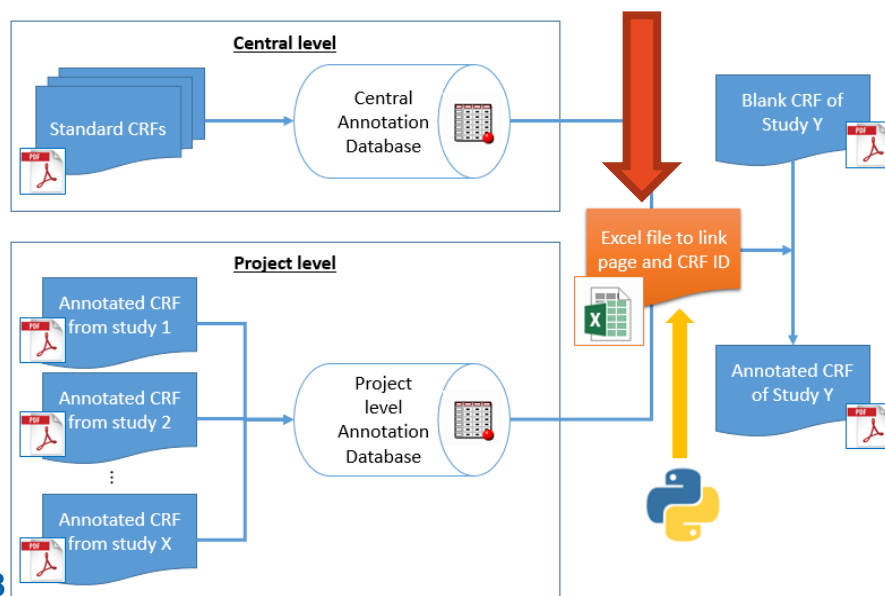
in clinical study.

be processed for

ook cell by %%SAS.
Jupyter notebook.

Possible application example 2

- There are many strength in Python and handing PDF file is one of advantage of using Python.
- In Novartis, aCRF is created in semi-automated process. Only manual process is creation of below Excel file. The idea is to make this excel file automated in Python.



Possible application example 2

pdfminer

- To extract comment box from PDF file, 3rd party Python library “pdfminer” is able to extract text box from PDF.
- Go through every page of blank crf and get text information from each pages.

Pandas

- Store those text data into Pandas DF until last page of PDF.

SASPy

- Then convert Pandas DF to SAS datasets, to process text data, Pandas DF needs to be converted to SAS dataset by SASPy `df2sd()` function.

Possible application example 2

Jupyter Magic

- Process obtained SAS data by Jupyter notebook.
- Obtained information in SAS dataset will be merged central annotation database by SAS code in order to keep study individual information only.

Pandas

- Converted merge SAS dataset to csv file and export file by `to_csv()` function. You can also select SASPy or Jupyter magic (`proc export`) directly to export csv file.

The image shows a Novartis clinical trial form on the left and a SAS dataset view on the right. A blue arrow points from the form to the dataset view. The form has fields for ID, Center No., Subject No., and SCREENING Visit Name. A red box highlights the 'SVG01' label. The SAS dataset view shows a table with columns 'page' and 'value'. The first two rows are highlighted with red boxes: row 1 with 'Study' and row 2 with 'SVG01'.

page	value
1	Study
2	SVG01
3	ID
4	Center No.
5	Subject No.
6	Visit Name
7	Date of visit
8	DD-MON-YYYY
9	Epoch/Verified
10	Approved
11	Locked
12	Frozen
13	Document Number
14	Page Name
15	Subevent#
16	Page Number
17	Status
18	Version
19	Study
20	ID

Merge
annotation DB



Possible application example 2

Actual code

```
# Import libraries
import sys
import saspy
import pandas as pd
from pdfminer.converter import PDFPageAggregator
from pdfminer.layout import PDFPageInterpreter
from pdfminer.pdfinterp import PDFPageInterpreter

# Set parameters
laparams = LAParams(detect_vertical=True)
resource_manager = PDFResourceManager()
device = PDFPageAggregator(resource_manager, laparams=laparams)
interpreter = PDFPageInterpreter(resource_manager, device)

# Open pdf file
f = open('C:\\Users\\nakajyul\\')

# Initialize
page_num = 0
init = []
total = pd.DataFrame(init).T

# Process pdf page until end of
for page in PDFPage.get_pages(f):
    interpreter.process_page(page)
    layout = device.get_result()
    boxes = find_textboxes_recu

    page_num += 1
    page = str(page_num)
    temp1 = pd.DataFrame(init).T

    # Get page text into DataFrame
    for box in boxes:
        temp=pd.DataFrame([page])
        temp1=temp1.append(temp)
        total=temp1.append(temp1)

    # Apply labels to column
    rslt=total.rename(columns={0: " "})
    # Remove page break
    rslt=rslt.replace( '\n', ' ', re

# Convert dataframe to SAS data
pdftext=sas.df2sd(rslt,"pdftext")
f.close()

# Jupyter magic command, run SAS command in cell
%%SAS
libname temp "C:\\Users\\nakajyul\\OneDrive - Novartis Pharma AG\\work\\02_NON PROJECT\\python\\SAS";

proc sort data = temp.pdftext out = wk1 nodupkey;
    by value page;
run;
data wk2;
    set wk1;
    length crfid $30.;
    pagenum = input(page, best.);
    crfid = strip(value);
run;

proc sql noprint ;
    select max ( pagenum ) into :maxn from wk2 ;
quit ;
%let maxn = &maxn ;
proc sort data = temp.central_annotation_database out = wk3 nodupkey;
    by crfid;
run;

data wk4;
    merge wk2(in=a) wk3(in=b);
    by crfid;
    if a and b;
    keep pagenum crfid;
    proc sort;
        by pagenum crfid;
run;

data dummy;
    do pagenum = 1 to &maxn.;
        output;end;
run;

data temp.rslt;
    merge dummy wk4;
    by pagenum;
run;
```

Outline

- Introduction of SASPy
 - Pre-requirements
 - SASPy Installation
- Use of Python in clinical study
 - Data handling / reporting / visualization
 - Possible application example
- **Summary**
- Conclusion

Summary

- **SASPy** can establish SAS session in Python.
- The advantages of use of Python with SAS datasets
 - **Jupyter magic** which allows SAS script in Jupyter notebook
 - Easy to convert SAS dataset to **Pandas DF**, then easy to display by **Matplotlib**. Quick data review can be performed by simple Python code.
 - **Pdfminer** is capable to read PDF file and process extracted data in Python.

Summary

- Summary of Python libraries in this presentation

Python library	Description
SASPy	It provides Python APIs to the SAS system and establish SAS session and run analytics from Python through a combination of object-oriented methods and Python magic.
Pandas	It is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive.
Matplotlib	It is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
Numpy	NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.
pdfminer	It attempts to reconstruct complex structures by guessing from its positioning.

Outline

- Introduction of SASPy
 - Pre-requirements
 - SASPy Installation
- Use of Python in clinical study
 - Data handling / reporting / visualization
 - Possible application example
- Summary
- Conclusion

Conclusion

- Both programming languages have each advantage and disadvantage respectively. With the combination of SAS and Python via SASPy, it can maximize the value of programming.
- There is no doubt SAS is the most popular software in clinical study data analysis. However, acquiring Python programming skill additionally will enable SAS programmer **to expand the capability and opportunity of your daily work.**

Let's find a better solution,
It's all up to you !

References

- pandas <https://pandas.pydata.org/>
- matplotlib <https://matplotlib.org/>
- SASPy <https://sassoftware.github.io/saspy/>
- pdfminer <https://media.readthedocs.org/pdf/pdfminer-docs/latest/pdfminer-docs.pdf>
- CDISC <https://www.cdisc.org/>
- Yuichi Nakajima, “ Handling SAS Dataset in Python“, SAS User Forum Japan 2018,
https://www.sas.com/content/dam/SAS/ja_jp/doc/event/sas-user-groups/2018/poster-yn.pdf
- Yuichi Nakajima, “Data Visualization by Python: Data from Pandas to Matplotlib“, PharmaSUG SDE Japan 2018
<https://www.pharmasug.org/proceedings/tokyo2018/posters/PharmaSUG-Tokyo-2018-PO03.pdf>

Thank you !

