PythonによるSASデータハンドリング

中嶋　優一
（ノバルティスファーマ株式会社）

Handling SAS Dataset in Python

Yuichi Nakajima
Novartis Pharma K.K

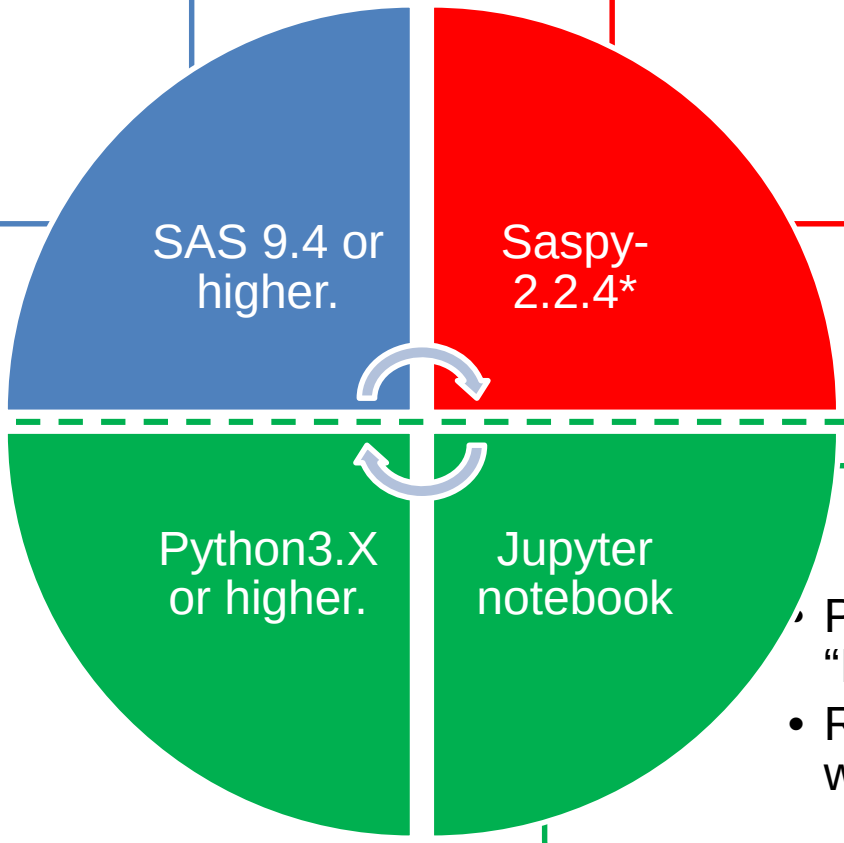
要旨:

SaspyはPythonにおいてSASを利用するためのパッケージライブラリである。本発表ではPythonのパッケージライブラリとして知られるSaspyの導入方法，PythonによるSASデータセットの加工，および基本的なデータ集計・表示方法等を紹介する。

キーワード: Python, Saspy

Pre-requirement

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

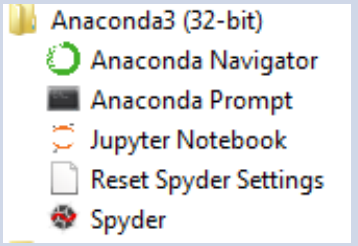
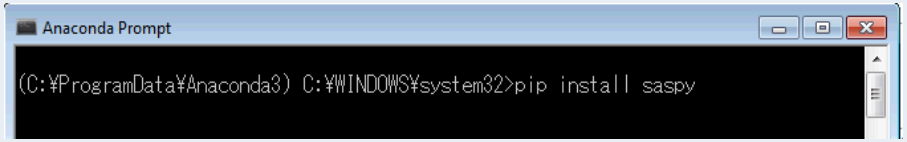
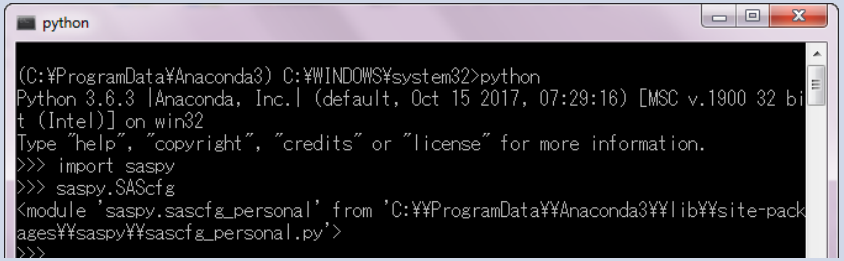


- As of July 2018, v2.2.4 is the latest version.

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

Available from
Anaconda distribution

Steps of Saspy Installation

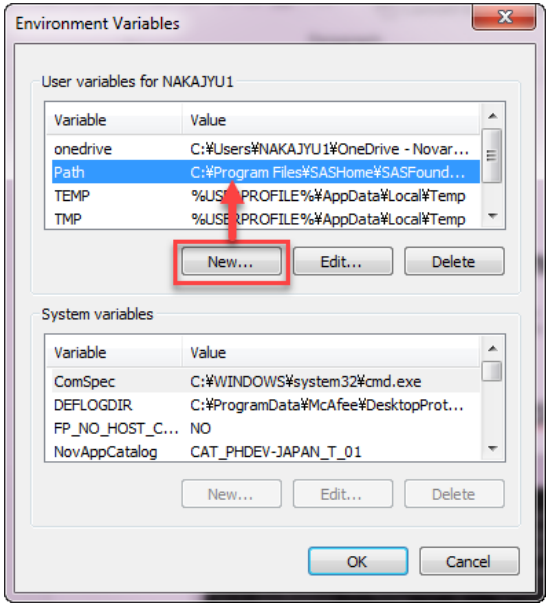
#	Process	Note
1	Make sure to install below contents appropriately. • SAS 9.4 or higher version. • Anaconda3 Distribution	 
2	Open **"Anaconda Prompt"** and install Saspy • Enter *pip install saspy*	
3	After importing saspy (*import saspy*) in python, need to edit SAS configuration files(sascfg.py) to establish corresponding SAS session (See next slide) • Location is available by *saspy.SAScfg*.	
4	Start session to create class instance setting configuration type (Windows) • *sas = saspy.SASsession(cfgname='winlocal')*	

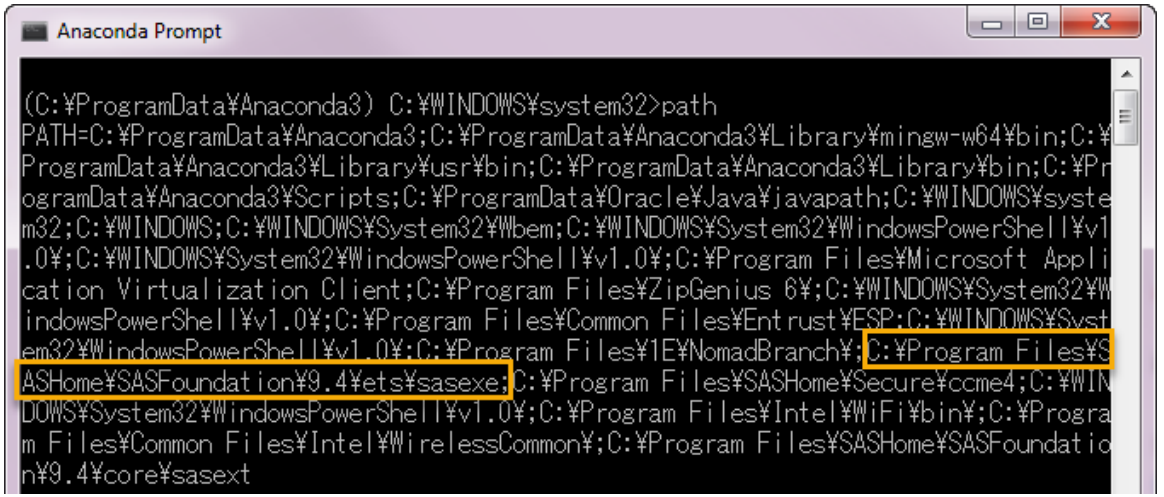
(Option) Update sascfg.py

- If you need to copy **sascfg.py** to **sascfg_personal.py** in order to update configuration for using Windows local SAS.
  - Update SAS_config_names as ‘winlocal’ in sascfg_personal.py.

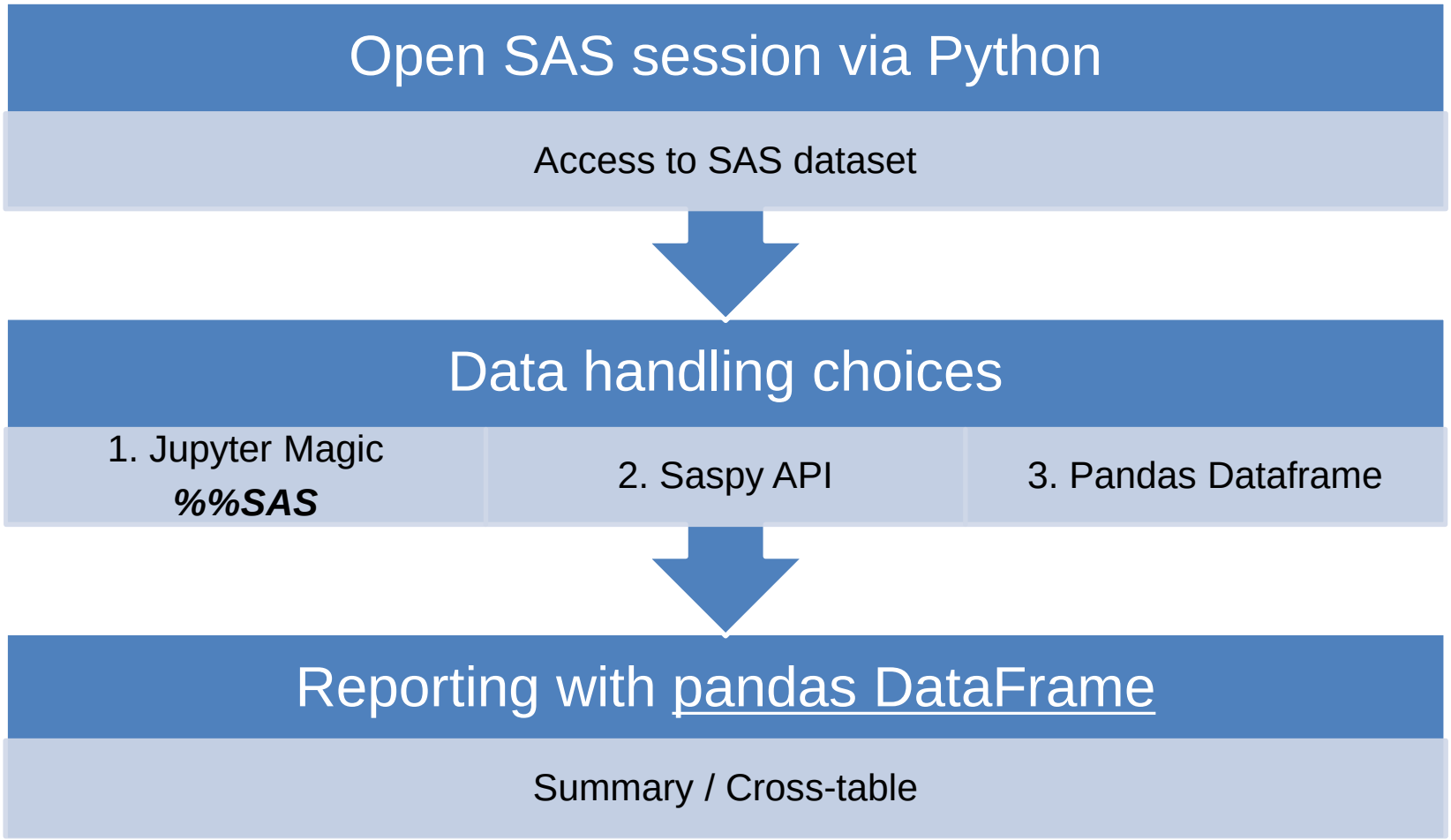
    e.g) SAS_config_names=['winlocal']
  - Update SAS session to specify java.exe file in sascfg_personal.py.

    e.g) winlocal = {'java' : 'C:¥ProgramData¥Oracle¥Java¥javapath¥java.exe', ‘encoding' : 'windows-1252', 'classpath' : cpW}
- To establish local connection, check if system PATH environment variable is added.





Overall process



Convert SAS data to Pandas DataFrame.

- Use Saspy API for read SAS data and convert to pandas DataFrame.

```
In [3]: # Set up file path a.k.a libname in SAS
        sas.saslib('temp', path="C:\\Users\\NAKAJYU1\\Desktop\\tempds")
        # Read SAS dataset
        ae = sas.sasdata('ae', libref='temp')
        # Convert SAS dataset to pandas dataframe
        dfae = sas.sasdata2dataframe('ae', libref='temp')
```

```
In [4]: print(type(ae))
        print(type(dfae))

        <class 'saspy.sasbase.SASdata'>
        <class 'pandas.core.frame.DataFrame'>
```

- Data handling choices Jupyter magic / Saspy API / Pandas

Choices	Description	Example (data sorting) <>: SAS dataset, []: Pandas DataFrame
1. 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 cell.	%%SAS proc sort data = <AAAA> out = <BBBB>; by USUBJID descending AESEV; run;
2. Saspy API	Saspy can setup a SAS session and run analytics from Python.	<AAAA>.sort(by='USUBJID DESCENDING aesev', out='[BBBB]')
3. Pandas DataFrame	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]) **Focus on Pandas for this poster.**

Pandas DataFrame
- data handling vs SAS code-

SAS	What do you want to do?	Pandas DataFrame
proc sort data = <AAAA> out = <BBBB>; by USUBJID descending AESEV; run;	**Data sorting**	{BBBB}={AAAA}.**sort_values**(**by**=['USUBJID', 'AESEV'], **ascending**=[True, False])
data <BB>; set <AA>; keep USUBJID SEX AGE RACE; run;	**Keep specific variable from dataset**	{BB}={AA}[['USUBJID', 'SEX', 'AGE', 'RACE']]
data <DD>; set <AA> <BB> <CC>; run;	**Append dataset**	{DD}={AA}.**append**([BB, CC])
data <BB>; set <AA>; where AESEV = “MODERATE” ; run;	**Extract records with condition**	{BB}={AA}.**loc**[{AA}['AESEV']=='MODERATE']
data <CC>; merge <AA>(in=a) <BB>(in=b); by USUBJID; if a and b; Run;	**Merge two datasets**	{CC} = **pd.merge**({AA},{BB}, **on**='USUBJID', **how**='inner') *# inner: a & b, left: a, right: b, outer: a or b*
data <BB>; set <AA>; by SEX; if first.AEDECOD; run;	**Get first / last observation per variable.**	{BB}={AA}.**groupby**(‘AEDECOD').**first**()

Pandas DataFrame
- data handling vs SAS code-

SAS	What do you want to do?	Pandas DataFrame
proc print data = <AA>; run;	**Show dataset**	{AA}.**head**()
proc contents data = <AA>; run;	**Show dataset contents**	{AA}.**info**()
data <BB>; set <AA>; if AVAL < 10 then AVALCAT1 = "Low"; else AVALCAT1 = "High"; run;	**if / then logic 1**	{BB}['AVALCT1N'] = **np.where**({AA}['AVAL'] < 10, 'Low', 'High')

data <BB>;
set <AA>;
if AVAL < 10 then AVALCT1N = 1;
else if AVAL >= then AVALCT1N = 2;
else AVALCT1N = .;
run;
	if / then logic 2	def **if_else**(x): # Setup function !!
		if x < 10:
		return 1
		elif x >= 10:
		return 2
		else:
		return np.NaN # need import numpy
		
		{BB}['AVALCT1N'] = {AA}['AVAL'].**apply**(**if_else**)

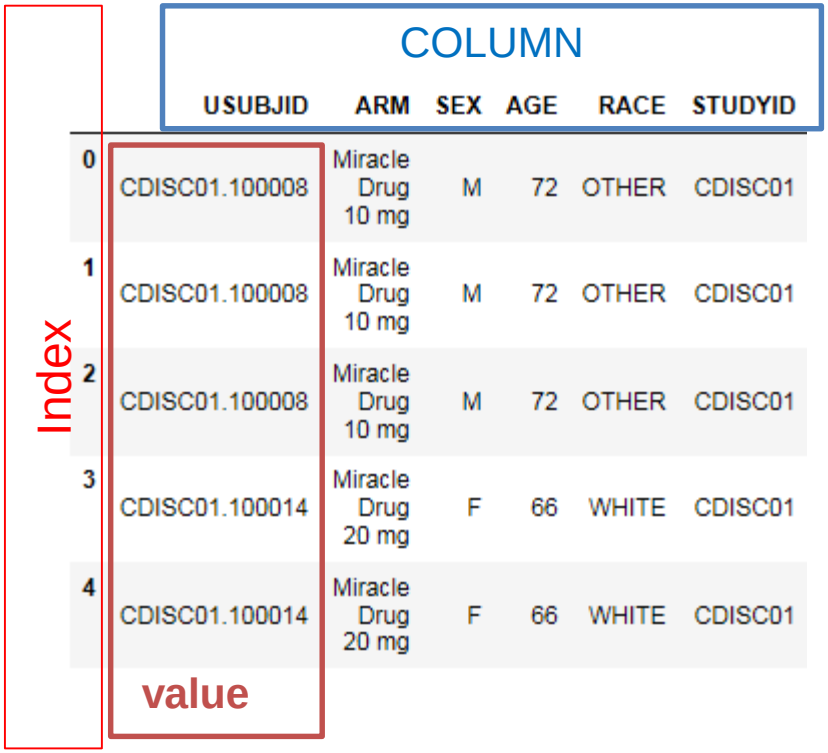
Pandas DataFrame
- Reporting / Summary Table-

1. “pivot_table” Method for cross-table
   - pd.pivot_table(wk, values='USUBJID',index = 'SEX',columns = 'ARM', **aggfunc='count'**)
   - Count every records.

ARM SEX	Miracle Drug 10 mg	Miracle Drug 20 mg	Placebo
F	3.0	5.0	5.0
M	3.0	NaN	NaN

   - pd.pivot_table(wk, values='USUBJID',index = 'SEX',columns = 'ARM', **aggfunc=lambda x:x.nunique()**)
   - Count unique records.

ARM SEX	Miracle Drug 10 mg	Miracle Drug 20 mg	Placebo
F	1.0	1.0	1.0
M	1.0	NaN	NaN



	USUBJID	ARM	SEX	AGE	RACE	STUDYID
0	CDISC01.100008	Miracle Drug 10 mg	M	72	OTHER	CDISC01
1	CDISC01.100008	Miracle Drug 10 mg	M	72	OTHER	CDISC01
2	CDISC01.100008	Miracle Drug 10 mg	M	72	OTHER	CDISC01
3	CDISC01.100014	Miracle Drug 20 mg	F	66	WHITE	CDISC01
4	CDISC01.100014	Miracle Drug 20 mg	F	66	WHITE	CDISC01

** aggfunc option: Several function can be obtained by “count”, “lambda”, “sum”, “mean”, ...

Pandas DataFrame
- Reporting / Summary Table-

2. “groupby” Method for summary table.

– Prepare dataframe to be counted first.

```
dflb=sas.sasdata2dataframe('lb', libref='temp')
dfdm=sas.sasdata2dataframe('dm', libref='temp')
dfdm1=dfdm[['USUBJID', 'ARM','SEX', 'AGE', 'RACE']]
dflb1=dflb[['USUBJID', 'VISIT','LBTEST', 'LBSTRESC', 'LBSTRESN']]
wk = pd.merge(dfdm1, dflb1, on='USUBJID', how='inner')
```

	USUBJID	VISIT	LBTEST	LBSTRESC	LBSTRESN
0	CDISC01.100008	SCREEN	Bilirubin	6.8	6.80
1	CDISC01.100008	WEEK 24	Bilirubin	5.1	5.10
2	CDISC01.100008	SCREEN	Blood Urea Nitrogen	9.28	9.28
3	CDISC01.100008	WEEK 24	Blood Urea Nitrogen	6.43	6.43
4	CDISC01.100008	SCREEN	Glucose	5.5	5.50
5	CDISC01.100008	WEEK 24	Glucose	4.8	4.80
6	CDISC01.100008	SCREEN	Vitamin B12	270	270.00
7	CDISC01.100008	SCREEN	Vitamin B9	126.4	126.40
8	CDISC01.100008	SCREEN	Hematocrit	0.36	0.36
9	CDISC01.100008	WEEK 24	Hematocrit	0.33	0.33
10	CDISC01.100008	SCREEN	Hemoglobin	120	120.00
11	CDISC01.100008	WEEK 24	Hemoglobin	113	113.00
12	CDISC01.100008	SCREEN	Lymphocytes	1.68	1.68
13	CDISC01.100008	WEEK 24	Lymphocytes	1.34	1.34
14	CDISC01.100008	SCREEN	Occult Blood	1+	NaN
15	CDISC01.100008	WEEK 24	Occult Blood	NEGATIVE	NaN
16	CDISC01.100008	SCREEN	pH	8	8.00
17	CDISC01.100008	WEEK 24	pH	8	8.00
18	CDISC01.100008	SCREEN	Glucose	NEGATIVE	NaN

– For summary table: wk.groupby(['LBTEST', 'VISIT'])['LBSTRESN'].**describe()**

		count	mean	std	min	25%	50%	75%	max
LBTEST	VISIT								
Bilirubin	SCREEN	4.0	5.525000	0.850000	5.10	5.1000	5.100	5.5250	6.80
	WEEK 24	4.0	5.100000	1.388044	3.40	4.6750	5.100	5.5250	6.80
Blood Urea Nitrogen	SCREEN	4.0	5.977500	2.283891	4.28	4.5500	5.175	6.6025	9.28
	WEEK 24	4.0	5.537500	0.618729	5.00	5.2700	5.360	5.6275	6.43
Glucose	SCREEN	8.0	3.100000	2.488832	0.00	0.9750	3.000	5.2750	6.40
	WEEK 24	6.0	2.600000	2.184491	0.20	0.8750	2.250	4.3000	5.50

– If the value is character, you can use **value_count()**

```
wk1=wk.loc[wk['LBTEST']=='Occult Blood']
wk1.groupby(['LBTEST', 'VISIT'])['LBSTRESC'].value_counts()
```

```
LBTEST        VISIT    LBSTRESC
Occult Blood  SCREEN   NEGATIVE    3
                       1+          1
              WEEK 24  NEGATIVE    3
Name: LBSTRESC, dtype: int64
```

Conclusion

- With combination of Jupyter magic, Saspy and Pandas DataFrame, SAS datasets can be processed in Python environment.
- Although SAS is still powerful analytic tool in clinical development, Python could support the activities from SAS data handling to reporting.
- Please contact me if you need an example full code. yuichi.nakajima@novartis.com
- References
  - ❖ Saspy: https://sassoftware.github.io/saspy/api.html
  - ❖ Saspy install: https://communities.sas.com/t5/Base-SAS-Programming/Connect-to-SAS-using-Python-saspy/td-p/400730
  - ❖ Pandas DataFrame: https://pandas.pydata.org/pandas-docs/stable/index.html