**DATA SCIENCE BOOTCAMP: SAS To Python**

1. **Set directory/path**

**SAS:** libname mydata ' path\\name'

**Python:** import os

os.chdir("path\\name")

1. **Importing Packages**

**Python:** import pandas as pd

from pandas import Series, DataFrame, Index

1. **Import csv file**

**SAS:** proc import datafile = “path\\name\\filename.csv” dbms =csv out = a1 replace;

getnames = yes; run;

**Python:** a1 = pd.read\_csv('path\\name\\filename.csv')

1. **Structure of Data**

**SAS:** proc contents data = a1; run;

**Python:** a1.info(), where a1 is dataframe

1. **Appending dataset**

**SAS:** proc append base = a1 data=a2 force; run;

/\* Append a2 to a1 dataset \*/

/\* force is optional, which forces the append when variables are different\*/

**Python:** import pandas as pd

a1.append(a2, ignore\_index = TRUE)

1. **Computing Descriptive Statistics for Multiple Variables**

**SAS:** proc univariate data = a1;

var x y z; run; ; x,y,z are variables/columns in a1 dataset

/\*;‘var’ statement is optional, if not mentioned then it gives for all numerical variables.\*/

**Python:** a1.describe() ; for all numerical variables

Or a1[‘x’].describe() ; for variable specific

1. **Sort dataset**

**SAS:** proc sort data = a1 nodup out=a2;

by x y; run;

**Python:** a1.sort\_values(by=[‘x’,’y’], ascending=False)

1. **Ranking values with single column**

**SAS:** proc rank data = a1 out = a2 descending;

var x; run;

**Python:** a1[‘Rank’] = a1[“x”].rank(method=’average’, ascending=False)

1. **Ranking values with multiple columns**

**SAS:** proc rank data = a1 out = a2 descending;

var x y; run;

**Python:** col1 = a1[“x”].astype(str)

col1 = a1[“x”].astype(str)

a1['Rank'] = (col1+col2).astype(int).rank(method='dense', ascending=False).astype(int)

1. **Correlation Check**

**SAS:** proc corr data=a1 plots=matrix;

var x y; run;

**Python:** a1.corr (method=’pearson’)

1. **Finding Mean**

**SAS:** proc means data=a1; /\*same for proc summary\*/

var x; class y z;

output out=a2; run;

**Python:** a1[‘x’].describe()

1. **Frequency Check**

**SAS:** proc freq data=a1;

table x / nocol nopercent nocum; run;

**Python:** a1[‘x].value\_counts()

1. **Frequency Check (with missing)**

**SAS:** proc freq data=a1;

table x / nocol nopercent nocum missing; run;

**Python:** a1[‘x].value\_counts(dropna=FALSE)

Or, a1.isnull().sum()

1. **Replace Missing value:**

**Python:** a2 = a1[[x,y]].fillna(a1.z.mean()) ; replace missing value of variable x & y by mean of column z

1. **Dropping or keeping vars**

**SAS:** data a2; set a1 (keep = x); run; ;keep

data a2; set a1 (drop = x); run; ;drop

**Python:** a2 = pd.DataFrame (a1 , columns = ['x’, ‘z’]) ;keep

a1.drop(a1.columns[[1, 69]], axis=1, inplace=True) ;drop col 2 and 60 by indexing

1. **Renaming Vars**

**SAS:** data a2(rename=(x=x1 y=y1));

set a1; run;

**Python:** a2 = a1.rename(index=str, columns={'x' : 'x1’ , ‘y’ : ‘y1’})

1. **Sub-setting based on values of obs**

**SAS:** **data** a2;

set a1;

where x = some\_value; **run**;

Python: a1.loc[a1[‘x'] == some\_value]

1. **IF-THEN statements**

**SAS: data** a2; set a1;

if x = **12** then cat= ‘entry’ ; else cat = ‘exit’;

**run**;

**Python**: a1[‘cat’] = pd.np.where(a1.x == 12,"entry",”exit”))

1. **Merging datasets horizontally:**

**SAS** (using data step): data new\_table;

merge table1, table2, table3,…,tablen;

by var1, var2, var3,..., varn; run;

**Python**: import pandas as pd

pd.merge (table1, table2, ***on = [’ var1’, ‘var2’,’ var3’,…,’ varn’]***)

1. **Defined type of merging – Left, Right, Inner, Outter or full**

**SAS** (using sql): Proc sql;

create table new\_table as

select var1, var2, var3,..., varn

from table1 as a

left join table2 as b

on a.key = b.key

order by var1, var2, var3,..., varn; quit;

**Python**: import pandas as pd

**pd.merge** (table1, table2, on = [’ var1’, ‘var2’,’ var3’,…,’ varn’], ***how = ‘left’***)

# you can change the ***how =*** ‘ ’ to the type of join you want to execute on the tables.

1. **Merging datasets vertically:**

**SAS** (using data step): data new\_table;

set table1, table2, table3…tablen; run;

OR

**SAS** (using sql): proc sql;

create table new\_table as

select var1, var2, var3,..., varn

from table1

union corresponding all

select var1, var2, var3,..., varn

from table1; quit;

**Python**: new\_table = table1.**append**(table2)

Or new\_table = **pd.concat**([table1, table2, table3,…,tablen])

1. **Data Aggregation and Grouping:**

**SAS** (using sql): Proc sql;

create table new\_table as

select var1, ***sum***(var2)

from table1

group by var1

order by var1, var2; quit;

/\* here my aggregating function is sum and I want to group by var1 \*/

**Python**: new\_table = table1.**groupby**([var1]) [var2].sum()

new\_table = table1.**groupby**([’ var1’, ‘var2’,…]) [’ var1’, ‘var2’,…].***aggregating\_function***

1. **Plotting:**

List of SAS libraries for plotting:

1. GCHART bar, pie, block, donut, star charts
2. GPLOT scatter, bubble, line, area, box, regression plots
3. G3D 3-D scatter and surface plots
4. GCONTOUR contour plots
5. GMAP maps with user-defined data

**SAS:**

/\* plot a vertical bar chart \*/ /\* change the vbar to hbar for a horizontal view\*/

Eg1. Vertical Bar Chart proc gchart data= table1;

vbar var1 / type=percent;

run; quit;

/\* scatter plot for var1  and var2 \*/

Eg2. proc gplot data= table1;

title ‘Scatter Plot Example’;

plot var1 \* var2; run; quit;

**Python**: Import matplotlib as plt

fig = plt.figure()

# to create subplots

ax1 = fig.add\_subplot(2 , 2 , 1)

ax2 = fig.add\_subplot(2 , 2 , 2)

ax3 = fig.add\_subplot(2 , 2 , 3)

ax4 = fig.add\_subplot(2 , 2 , 4)

# above code creates a subplot of 4 charts in 1 figure with ax1 being the first (tope left) graph and ax4 being the last (bottom right) graph.

# Python by default plots on the last graph if the subplot is not mentioned. It is not necessray to create subplots but for the sake of covering this we are creating the graphs in sub plots.

# You can just plot on the fig or have just 1 subplot for simplicity.

*Generic form:*

matplotlib.pyplot.hist(*x*, *bins=None*, *range=None*, *density=None*)

Eg1: Histogram on sub plot 2 with bins of length 10 and line cololur black

ax2.hist(randn(100), bins= 10, color='k', alpha=0.3)

Eg2: Scatter on sub plot 3 with x = var1 and y = var2 and colour of scatter dots = green

ax3.scatter(var1, var2, c=’g’, label = ‘scatter example’)

1. **Date Operations:**

* **Get System date:** /\* calling todays’ date and time \*/

**SAS** (datastep): *call symput ('timenow',put (time(),time.)); call symput ('datenow',put (date(),date9.));*

**Python** : from datetime import date, time, datetime, timedelta

today\_date = date.today()

* **Increment Date:** /\* increment date by a fixed period \*/

**SAS**: INTNX(interval, start-from, increment );

Eg1. INTNX(‘Month’, '25Dec2017'd, 1); /\* = 25Jan2017 \*/

INTNX(‘Week, '26Jan2018'd, -1); /\* = 19Jan2018 \*/

**Python**: from datetime import datetime

from datetime import timedelta

# add a day datetime.now() + timedelta(days = 1)

# subtract a week datetime.now() + timedelta(weeks = -1)

# add 1 day and 1 month datetime.now() + timedelta (days = 1, month= 1)

* **Time interval between two dates:** /\* interval between D1 and D2 \*/

**SAS**: INTCK(interval, start\_date, end\_date, method)

Eg1. INTCK(‘Year’, ‘01Jan2010’d, ‘01Jan2020’) /\* =10 \*/

**Python**: from datetime import datetime

from datetime import timedelta

time\_interval = (D1 – D2)

#to find out month interval: time\_interval.months

#to find out days interval: time\_interval.days

1. **String Operations:**
2. **/\* find length of string var1 \*/**

**SAS**: length\_var = length(var1)

**Python:** var1= “this is just an example of a string variable”

len(var1)

1. **/\* find len of string var1 without trailing balnks \*/**

**SAS**: length\_var = length(var1)

**Python:** var1= “WI Analytics ”

len(var1.strip())

# lstrip() : remove leading blanks

# rstrip() : remove trailing blanks

# strip() : remove all blanks

1. **/\* finding specific chars in string var1 \*/**

Suppose, we want to find at what position ‘WI’ appears in string var1

**SAS:** v\_find = find(var1, ‘ WI‘, ‘i’)

\*‘i’ method makes the search case-insensetive

**Python:** var1.find(‘WI’)

\*can also be done using index() to do this

1. **/\* finding substrings \*/**

**SAS:** substring = substr(var1 , i , m)

,where ‘i’ is the index position from where the substring starts and ‘m’ is the defined length of the substring needed.

**Python:** var1[ x : y ]

,where x is the start index point and y is the stop index point. Please refer to indexing on how to do this better.

1. **Concatenation:**

In SAS there are 3 functions commonly used for concatenating:

CAT – Simple concatenation

CATT – Concatenation with trailing blanks trimmed

CATS – Concatenation with leading and trailing blanks trimmed

CATX – CATS with an option of using a delimiter to separate the strings

**SAS:** data table;

set table;

concat\_var= catx(‘ ‘,var1, var2, var3); run;

/\* above example concatenates with space as delimiter \*/

**Python:** concat\_var = var1 + var2 + var3 +….+ varn

# if any variable is numeric then the string variable would be repeated as many times for an ‘\*’ operator and would return an error for ‘+’ operator as the numeric variable.

User can define the numeric var as string by simply using str(var) function

Eg: var1 = ‘WI’ and var2 = 3

concat\_var = var1 + var2 🡪 Error

concat\_var = var1 \* var2 🡪 WI WI WI

concat\_var = var1 +str( var2) 🡪 WI3