Paper DV01

Python for Visualization in Solid Tumor Studies

Hengwei Liu, Daiichi Sankyo, Inc., Basking Ridge, New Jersey

ABSTRACT

Visualization is an integral part of the statistical analysis in solid tumor studies. When it comes to the tools for visualization, the programmers do have a choice. There are many tools available: SAS® visual analytics, R shiny and Spotfire, to name just a few. So why Python? Python has a large user community. It is easy to get help online. Python programs for figures are succinct and easy to understand. Python figures have dynamic features such as tooltip and can be saved in many formats. Python can handle compressed SAS datasets and can easily convert them to CSV files. In this paper we show how to use Python to create figures for solid tumor studies.

INTRODUCTION

The following plots are often requested in the analysis for solid tumor studies:

Swimmer plot

The x-axis is time from first dose of treatment. The y-axis is the subject ID. The treatment durations are shown in horizontal bars. The response CR, PR, SD, PD, NE, NON-CR/NON-PD, death and treatment ongoing are shown in different symbols at the time point where they occur. The patients are ordered from the one with longest treatment duration to the one with shortest treatment duration.

Spider plot

The x-axis is treatment duration. The y-axis is percent change from baseline in sum of diameters. Each patient is represented by a line in the figure. The percent change from baseline in sum of diameters in each cycle is shown as a dot.

Waterfall plot

The x-axis is for the patients. The y-axis is the best percent change from baseline in sum of diameters. The patients are ordered from the one with largest best percent change to the one with smallest best percent change.

Forest plot for objective response rate (ORR)

The ORR is defined as the proportion of subjects who achieved a best overall response of complete response (CR) or partial response (PR). It is often calculated for multiple subgroups of patients, with the proportions and confidence intervals presented in a forest plot.

For the swimmer plot, spider plot and waterfall plot, the Python modules required are the matplotlib and pandas. The read_sas function in the pandas module is used to read the ADaM datasets. The pyplot functions in matplotlib is used to do the figures. The pyplot is a collection of functions that can produce with ease the scatter plot, line plot, bar plot and horizontal bar plot. For more information about pyplot see reference [1].

For the forest plot, three additional modules are required: numpy, scipy and zepid graphics.

The development of the Python programs is done in Python 3.9.2 in Linux.

GET INTO THE DETAILS

The SAS datasets to be used to create the figures are ADSL (Subject Level Analysis Dataset), ADTR (Tumor Results Analysis Dataset), and ADRS (Disease Response Analysis Dataset).

They follow the ADaM data structure in ADaM IG 1.1.

Table 1 shows the variables required in each dataset.

Dataset	Variable	Label	Note
ADSL	SUBJID	Subject Identifier for the Study	
ADSL	TRTDURM	Treatment Duration (Months)	This variable is used to draw the horizontal bar in the swimmer plot
ADSL	DTHDY	Day of Death	This variable is used to draw the point for death in the swimmer plot
ADSL	EOTSTT	End of Treatment Status	This variable is used to draw the point for treatment ongoing in the swimmer plot
ADTR	SUBJID	Subject Identifier for the Study	
ADTR	PARAMCD	Parameter Code	There is a parameter for best percent change from baseline in sum of diameters for the waterfall plot; there is a parameter for sum of diameters; the percent change from baseline for this parameter is used in the spider plot.
ADTR	AVAL	Analysis Value	
ADTR	PCHG	Percent Change from Baseline	
ADTR	ABLFL	Baseline Record Flag	
ADTR	ADY	Analysis Relative Day	
ADTR	ANL01FL	Analysis Record Flag 01	
ADTR	PARQUAL	Evaluator	This variable takes the value of either INVESTIGATOR or INDEPENDENT ASSESSOR.
ADTR	TRTP	Planned Treatment	'TRT A' or 'TRT B'
ADRS	SUBJID	Subject Identifier for the Study	
ADRS	PARAMCD	Parameter Code	There is a parameter for the overall response for each cycle. This is used in the swimmer plot. There is a parameter for confirmed best overall response. This is used in the forest plot.
ADRS	AVALC	Analysis Value (C)	
ADRS	ADY	Analysis Relative Day	
ADRS	PARQUAL	Evaluator	See note for PARQUAL in ADTR

Table 1. the Variables Required in Each Dataset

SPIDER PLOT

The set_index is used to set the months as index of the data frame. Group_by is used to draw the line that connects the percent change from baseline in sum of diameters at each assessment.

```
#spider.py

import matplotlib.pyplot as plt

import pandas as pd

df = pd.read_sas('adtr.sas7bdat')

df2=df[(df["PARQUAL"]==b'INVESTIGATOR') & (df['PARAMCD']==b'SUMDIAM') &
(df['ANL01FL']==b'Y')]

df2=df2.assign(ADY2=df2['ADY']/(365.25/12))


df2.loc[df2['ABLFL']==b'Y','ADY2']=0

df2.loc[df2['ABLFL']==b'Y','PCHG']=0

df2.set_index('ADY2',inplace=True)

df2.groupby('SUBJID')['PCHG'].plot(color='red', marker='o')


plt.title('Spider Plot', fontsize=9)

plt.xlabel('Treatment Duration (months)', fontsize=9)

plt.ylabel('Percent Change from Baseline in Sum of Diameters', fontsize=9)


plt.grid(True)

plt.show()
```

A sample output is displayed in Figure 1.

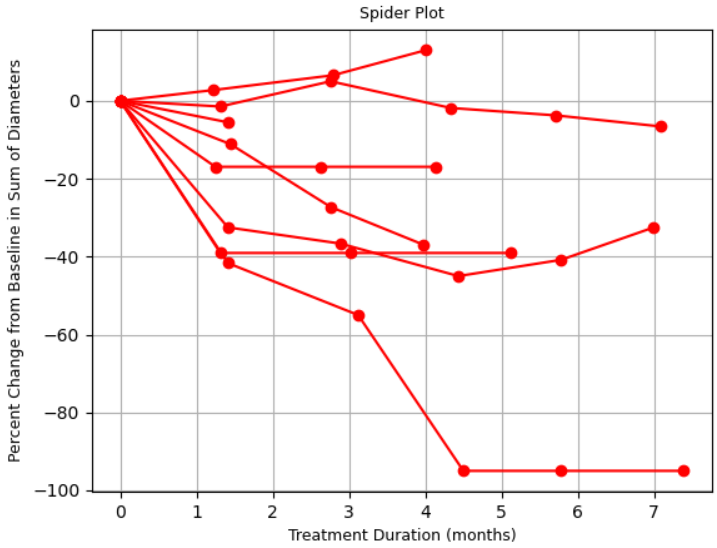

Figure 1. Sample Spider Plot

There are situations where we need to show the treatment information in the spider plot by showing different treatments in different colors. We can do this by creating a dataframe for each treatment, and create the plot for each dataframe.

```
#spider2.py
```

```
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.patches as mpatches

df = pd.read_sas('adtr.sas7bdat')
df=df.assign(ADY2=df['ADY']/(365.25/12))
df.loc[df['ABLFL']==b'Y','ADY2']=0
df.loc[df['ABLFL']==b'Y','PCHG']=0
df.set_index('ADY2',inplace=True)

df2=df[(df["PARQUAL"]==b'INVESTIGATOR') & (df['PARAMCD']==b'SUMDIAM') &
(df['ANL01FL']==b'Y') & (df['TRTP']==b'TRT A')]
df3=df[(df["PARQUAL"]==b'INVESTIGATOR') & (df['PARAMCD']==b'SUMDIAM') &
(df['ANL01FL']==b'Y') & (df['TRTP']==b'TRT B')]
df2.groupby('SUBJID')['PCHG'].plot(color='blue', marker='o')
df3.groupby('SUBJID')['PCHG'].plot(color='red', marker='o')

plt.title('Spider Plot', fontsize=9)
plt.xlabel('Treatment Duration (months)', fontsize=9)
plt.ylabel('Percent Change from Baseline in Sum of Diameters', fontsize=9)
blue_patch = mpatches.Patch(color='blue', label='TRT A')
red_patch = mpatches.Patch(color='red', label='TRT B')
plt.legend(handles=[red_patch, blue_patch])
plt.grid(True)
plt.show()
```

A sample output is displayed in Figure 2.

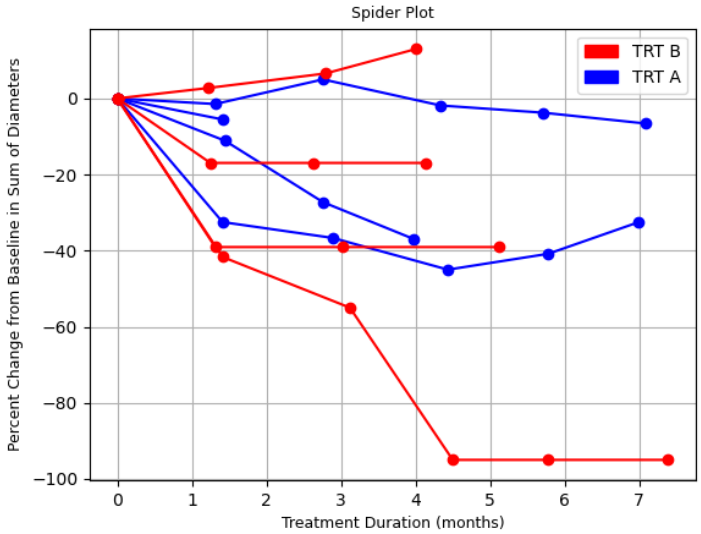

Figure 2. Sample Spider Plot with Two Colors for Two Treatments

WATERFALL PLOT

The sort_values is used with the option ascending=False so that the best percent changes from baseline in sum of diameters are displayed from the largest value to the smallest value.

```
#waterfall.py
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_sas('adtr.sas7bdat')
df2=df[(df["PARQUAL"]==b'INVESTIGATOR') & (df['PARAMCD']==b'BESTPCHG')]
df_sorted=df2.sort_values('AVAL', ascending=False)
plt.bar('SUBJID', 'AVAL', data=df_sorted)

plt.title('Watefall Plot', fontsize=9)
plt.xlabel('Patient', fontsize=9)
plt.ylabel('Best Percent Change from Baseline in Sum of Diameters', fontsize=9)

ax = plt.gca()
ax.axes.xaxis.set_ticks([])
plt.grid(True)
plt.show()
```

A sample output is displayed in Figure 3.

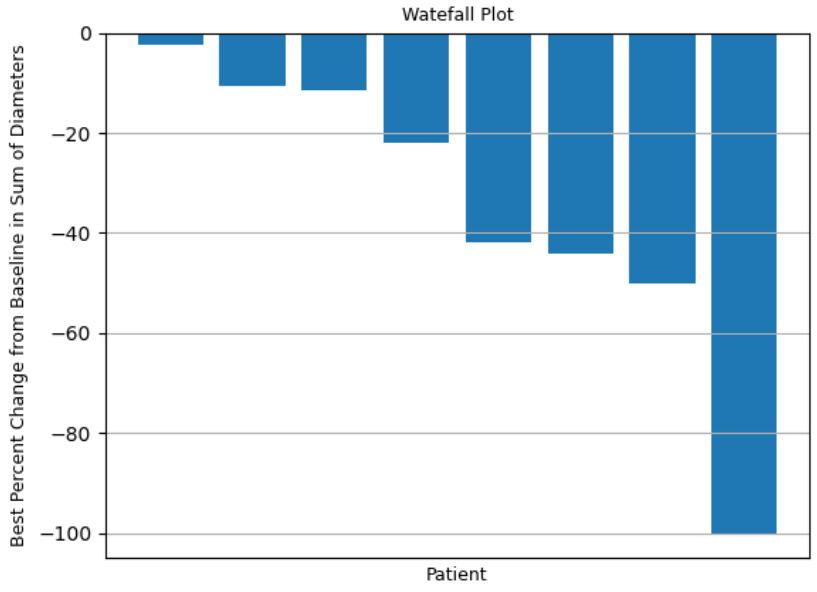

Figure 3. Sample Waterfall Plot

Sometimes we need to show the treatment information in the waterfall plot by showing the bars in different colors based on the treatment the patient received. We can add to the dataframe a column for the colors based on treatment, and use the color= option in the plt.bar.

```
#waterfall2.py
```

```
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.patches as mpatches

df = pd.read_sas('adtr.sas7bdat')
df2=df[(df["PARQUAL"]==b'INVESTIGATOR') & (df['PARAMCD']==b'BESTPCHG')]
df2=df2.assign(COL='red')
df2.loc[df2['TRTP']==b'TRT A','COL']='blue'
df_sorted=df2.sort_values('AVAL', ascending=False)
plt.bar('SUBJID', 'AVAL', data=df_sorted, color='COL')

plt.title('Watefall Plot', fontsize=9)
plt.xlabel('Patient', fontsize=9)
plt.ylabel('Best Percent Change from Baseline in Sum of Diameters', fontsize=9)

blue_patch = mpatches.Patch(color='blue', label='TRT A')
red_patch = mpatches.Patch(color='red', label='TRT B')
plt.legend(handles=[red_patch, blue_patch])

ax = plt.gca()
ax.axes.xaxis.set_ticks([])
plt.grid(True)
plt.show()
```

A sample output is displayed in Figure 4.

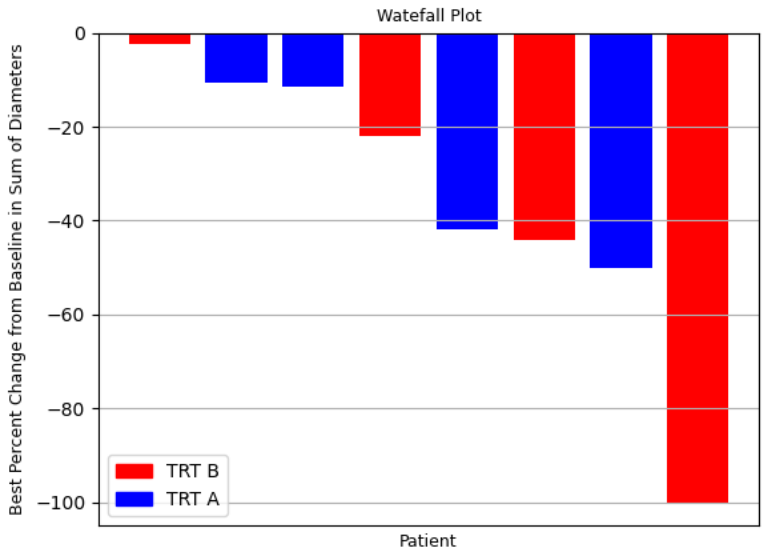

Figure 4. Sample Waterfall Plot with Two Colors for Two Treatments

SWIMMER PLOT

The sort_values is used to sort the treatment duration. The default sorting order for sort_values is ascending=True. So the treatment durations are sorted from the smallest to the largest for the patient IDs in the y-axis.

After the horizontal bars are drawn for the treatment duration, the scatter function is called eight times for death, treatment ongoing, CR, PR, SD, PD, NE and NON-CR/NON-PD respectively. Notice the use of zorder=-1 for the horizontal bar and zorder=1 for the scatter plots. This is to bring the scatter plots to the foreground.

```
#swimmer.py

import matplotlib.pyplot as plt

import pandas as pd


df = pd.read_sas('adsl.sas7bdat')

df3 = pd.read_sas('adrs.sas7bdat')

df2=df3[(df3["PARQUAL"]==b'INVESTIGATOR') & (df3["PARAMCD"]==b"OVRLRESP")]

ongo=df[df["EOTSTT"]==b'ONGOING']


df=df.assign(DTHDM=df["DTHDY"]/(365.25/12))

df2=df2.assign(ADM=df2["ADY"]/(365.25/12))


cr=df2[df2["AVALC"]==b'CR']

pr=df2[df2["AVALC"]==b'PR']

sd=df2[df2["AVALC"]==b'SD']

pd=df2[df2["AVALC"]==b'PD']

ne=df2[df2["AVALC"]==b'NE']

non=df2[df2["AVALC"]==b'NON-CR/NON-PD']


fig, ax=plt.subplots()

plt.subplots_adjust( top=0.9, bottom=0.1)

df.sort_values('TRTDURM', inplace=True)

ax.barh('SUBJID', 'TRTDURM', color="gray", zorder=-1 , data=df)


ax.scatter('DTHDM','SUBJID', data=df, marker='*',color='red', s=30, zorder=1,
label='Death')

ax.scatter('TRTDURM','SUBJID', data=ongo, marker='o',color='green', s=30, zorder=1,
label='Ongoing')

ax.scatter('ADM','SUBJID', data=cr, marker='v',color='blue', s=30, zorder=1,
label='CR')

ax.scatter('ADM','SUBJID', data=pr, marker='^',color='red', s=30, zorder=1,
label='PR')

ax.scatter('ADM','SUBJID', data=sd, marker='s',color='black', s=30, zorder=1,
label='SD')

ax.scatter('ADM','SUBJID', data=pd, marker='p',color='green', s=30, zorder=1,
label='PD')

ax.scatter('ADM','SUBJID', data=ne, marker='+',color='blue', s=30, zorder=1,
label='NE')
```

```
ax.scatter('ADM','SUBJID', data=non, marker='x',color='red', s=30, zorder=1,
label='NON-CR/NON-PD')


plt.title('Swimmer Plot', fontsize=9)
plt.xlabel('Time from First Dose (months)', fontsize=9)
plt.ylabel('Patient', fontsize=9)
plt.legend()


plt.grid(True)
plt.show()
```

A sample output is displayed in Figure 5.

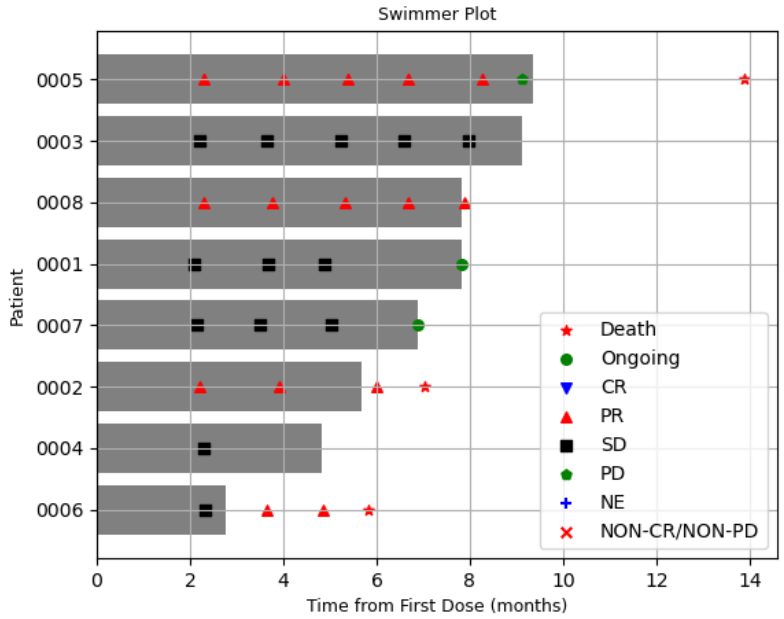

Figure 5. Sample Swimmer plot

FOREST PLOT FOR OBJECTIVE RESPONSE RATE (ORR)

There is a python module statsmodels.stats.proportion that calculates different confidence intervals for binomial proportion (see reference [2]). We use the codes from statsmodels.stats.proportion that calculate the Clopper-Pearson confidence interval to set up a function confint. This function is called for each subgroup of interest and it produces the proportions and confidence intervals. To draw the forest plot zepid graphics is used. (see reference [3]).

```
#forest.py
import numpy as np
from scipy import stats, optimize
import matplotlib.pyplot as plt
import pandas as pd
from zepid.graphics import EffectMeasurePlot
```

```
df = pd.read_sas('adrs.sas7bdat')

def confint(subgrp):
    df2=df[(df["PARQUAL"]==b'INVESTIGATOR') & (df['PARAMCD']==b'CBRSP') & (subgrp)]
    df3=df[(df["PARQUAL"]==b'INVESTIGATOR') & (df['PARAMCD']==b'CBRSP') &
((df['AVALC']==b'PR') | (df['AVALC']==b'CR')) & (subgrp)]

    count=df3.shape[0]
    nobs=df2.shape[0]
    alpha=0.05
    alpha_2=alpha/2
    q_=count/nobs

    ci_low = stats.beta.ppf(alpha_2, count, nobs - count + 1)
    ci_upp = stats.beta.isf(alpha_2, count + 1, nobs - count)

    if np.ndim(ci_low) > 0:
        ci_low[q_ == 0] = 0
        ci_upp[q_ == 1] = 1
    else:
        ci_low = ci_low if (q_ != 0) else 0
        ci_upp = ci_upp if (q_ != 1) else 1
    return q_, ci_low, ci_upp

q_, ci_low, ci_upp=confint(df['SEX']==b'M')
q2_, ci_low2, ci_upp2=confint(df['SEX']==b'F')
q3_, ci_low3, ci_upp3=confint(df['RACE']==b'ASIAN')

labs = ["SEX='M'","SEX='F'" ,"RACE='ASIAN'"]
measure = [q_, q2_, q3_]
lower = [ci_low, ci_low2, ci_low3]
upper = [ci_upp, ci_upp2, ci_upp3]

p = EffectMeasurePlot(label=labs, effect_measure=measure, lcl=lower, ucl=upper)
p.labels(effectmeasure='ORR')
p.colors(pointshape="D")
ax=p.plot(figsize=(7,3), t_adjuster=0.02, max_value=1.1, min_value=0 )
plt.tight_layout()

plt.title("Forest plot for ORR")
plt.show()
```

Figure 6 shows a sample forest plot.

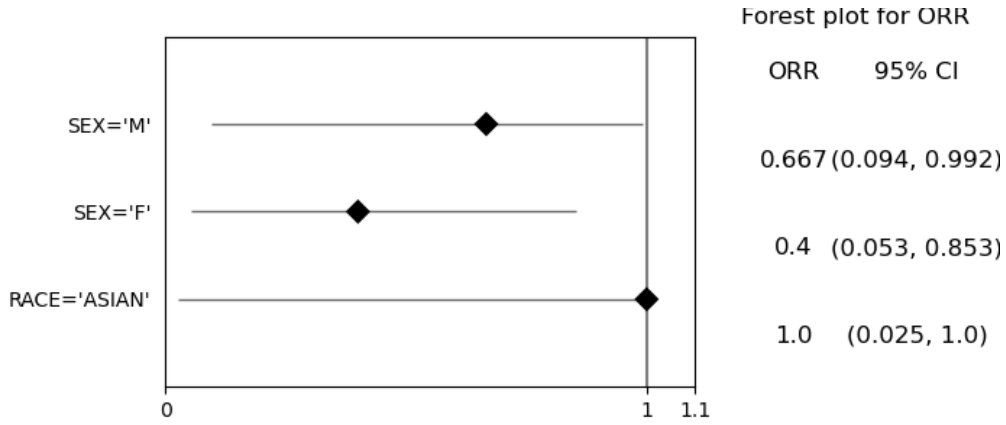

Figure 6. Sample Forest Plot

CONCLUSION

Python can produce the swimmer plot, spider plot, waterfall plot and forest plot for solid tumor studies in short and easily understandable programs. The output can be saved in many formats, e.g., png, jpg, and pdf. It holds advantage over some programming languages that require lengthy codes to create figures. In the statistical programming environment in the pharmaceutical and biotech industry, Python is not as widely employed as in other data science fields. Being a powerful and versatile open-source language Python can surely find more use.

REFERENCES

[1] Pyplot tutorial

https://matplotlib.org/tutorials/introductory/pyplot.html

[2] Source code for statsmodels.stats.proportion

https://www.statsmodels.org/devel/_modules/statsmodels/stats/proportion.html#proportion_confint

[3] zEpid graphics

https://zepid.readthedocs.io/en/latest/Graphics.html

ACKNOWLEDGEMENT

The author thanks the management at Daiichi Sankyo, Inc. for the support.

CONTACT INFORMATION

Hengwei Liu
Daiichi Sankyo, Inc.
211 Mount Airy Road
Basking Ridge, NJ 07920
Hengwei_liu@yahoo.com