In the name of God

Non-parametric with python

Contents :

Preface

What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Source : https://www.python.org/doc/essays/blurb/

Top 10 Most Popular Programming Languages

1. Python

2. JavaScript

3. Java

4. C#

5. C

6. C++

7. Go

8. R

9. Swift

10. PHP

R Vs Python: What's the Difference?

R and Python are both open-source programming languages with a large community. New libraries or tools are added continuously to their respective catalog. R is mainly used for statistical analysis while Python provides a more general approach to data science.

R and Python are state of the art in terms of programming language oriented towards data science. Learning both of them is, of course, the ideal solution. R and Python requires a time-investment, and such luxury is not available for everyone. Python is a general-purpose language with a readable syntax. R, however, is built by statisticians and encompasses their specific language.

Difference between R and Python

Parameter	R	Python
Objective	Data analysis and statistics	Deployment and production
Primary Users	Scholar and R&D	Programmers and developers
Flexibility	Easy to use available library	Easy to construct new models from scratch. I.e., matrix computation and optimization
Learning curve	Difficult at the beginning	Linear and smooth
Popularity of Programming Language. Percentage change	4.23% in 2018	21.69% in 2018
Average Salary	$99.000	$100.000
Integration	Run locally	Well-integrated with app
Task	Easy to get primary results	Good to deploy algorithm
Database size	Handle huge size	Handle huge size
IDE	Rstudio	Spyder, Ipython Notebook
Important Packages and library	tidyverse, ggplot2, caret, zoo	pandas, scipy, scikit-learn, TensorFlow, caret
Disadvantages	Slow High Learning curve Dependencies between library	Not as many libraries as R
Advantages	• Graphs are made to talk. R makes it beautiful • Large catalog for data analysis • GitHub interface • RMarkdown • Shiny	• Jupyter notebook: Notebooks help to share data with colleagues • Mathematical computation • Deployment • Code Readability • Speed • Function in Python

my data is saved in CSV file.

year	g	b	gg	bb	ggg	bbb	gggg	bbbb
48	8782	10680	1760433	992699	0	0	542917	257649
49	9248	10966	1846935	1069331	0	0	623483	297295
50	8706	10602	1874746	1128112	0	0	702481	341018
51	9509	11728	1734310	1069331	161754	97464	797565	390881
52	10025	11748	1825407	1137189	361084	210381	668600	330668
53	18587	22400	1957533	1180938	576045	330293	547629	269295
54	40574	42280	2173074	1348350	739710	411746	524235	273383
56	79978	95446	2321089	1491431	824242	459419	558411	297107
57	97434	114435	2459239	1619235	875516	493394	608754	332835
58	108298	126838	2581835	1732580	917571	528769	634712	354566
59	119662	139835	2625342	1769630	972137	563126	693809	393750
60	109123	127363	3067778	1941547	1063448	626443	674398	438830
61	81163	90840	2899086	1899506	975359	599705	626861	416536
62	92554	103435	3170008	2113369	1078543	671334	572081	414349
63	85329	92196	3308208	2284600	1055466	661631	576781	399662
64	100960	101939	3468923	2525480	1114107	703546	591049	424139
65	36135	41639	3615988	2720028	1236523	782213	628377	450354
66	50376	56610	3828105	2960218	1355589	854653	700784	488771
67	58451	64986	4058853	3173967	1406118	893392	768422	509599
68	69064	77345	4307333	3450374	1498834	968082	825448	550813
69	85223	92756	4537629	3724812	1638550	1086056	944675	628342
70	103345	114151	4790250	4026895	1782133	1213665	1000835	671812
71	108151	119341	5041319	4328327	1892510	1340172	1066653	752813
72	119959	132554	5224343	4563250	2050707	1490871	1192370	838616
73	81307	87557	5270533	4666836	2274665	1724847	1385197	1003113
74	63534	69119	5210412	4652405	2496712	1943259	1476237	1108108
75	97782	43946	5151547	4594053	2621635	2090393	1615458	1304133
76	71299	75765	4996934	4448983	2729990	2225215	1701525	1477244
77	95339	99842	4885675	4352718	2845092	2343720	1816811	1662824
78	100308	104014	4720322	4217915	2880892	2401989	1903939	1800716
79	109723	110711	4560725	4106422	2889797	2404875	2022833	1896793
80	125856	125700	4348771	3938766	2828770	2343746	2050709	1957867
81	144365	142538	4175849	3792588	2748874	2278350	2065265	1998449
82	168240	160822	3924999	3588016	2698328	2255566	2020296	1964854
83	208167	195487	3662507	3366417	2636728	2228875	1940924	1887600
84	226503	212276	3457559	3190358	2514943	2163621	1944256	1873515
85	254035	238387	3215507	2991211	2345168	2025926	1917888	1854697
86	278060	262333	3108860	2897693	2214730	1931383	1917183	1845854
87	284583	268349	3009561	2818290	2100894	1837767	1890567	1805380
88	263246	247983	2952075	2773554	1967906	1740364	1838567	1731272
89	236191	219653	2912264	2742705	1833240	1644404	1832405	1677775
90	230414	221844	2878290	2713462	1733942	1558304	1713585	1546074
91	239148	225305	2898799	2734026	1705332	1539569	1601595	1437252
92	242765	232630	2932282	2769239	1693270	1535135	1567691	1389637
93	226989	232028	3522913	3328079	1110458	1002417	1683347	1594689
94	260266	271658	3610900	3407385	1085222	985951	1696561	1591595
95	323824	337615	3701398	3492795	1068474	979158	1643680	1555271
96	338704	349748	3819488	3603017	1591299	1463060	1130697	1099526

I will explain about data ;

Number of level of education of individuals from 1348 to 1396.

- g : is number of preschool girls
- b : : is number of preschool boys
- gg : is number of primary school girls
- bb : is number of primary school boys
- ggg : is number of middle school girls
- bbb : is number of middle school boys
- gggg : : is number of high school girls
- bbbb : is number of high school boys

I got the data from the website of the Statistics Center of Iran

Top some Python Libraries for Data Science and statistics

1. Pandas

Pandas is an open-source Python package that provides high-performance, easy-to-use data structures and data analysis tools for the labeled data in Python programming language. Pandas stand for *Python Data Analysis Library*. Who ever knew that?

When to use? Pandas is a perfect tool for data wrangling or munging. It is designed for quick and easy data manipulation, reading, aggregation, and visualization.

Pandas take data in a CSV or TSV file or a SQL database and create a Python object with rows and columns called a data frame. The data frame is very similar to a table in statistical software, say Excel or SPSS.

2. NumPy

One of the most fundamental packages in Python, *NumPy* is a general-purpose array-processing package. It provides high-performance multidimensional array objects and tools to work with the arrays. NumPy is an efficient container of generic multi-dimensional data.

NumPy's main object is the homogeneous multidimensional array. It is a table of elements or numbers of the same datatype, indexed by a tuple of positive integers. In NumPy, dimensions are called *axes* and the number of axes is called *rank*. NumPy's array class is called *ndarray* aka *array.*

3. SciPy

The SciPy library is one of the core packages that make up the SciPy stack. Now, there is a difference between SciPy Stack and SciPy, the library. *SciPy* builds on the NumPy array object and is part of the stack which includes tools like Matplotlib, Pandas, and SymPy with additional tools,

SciPy library contains modules for efficient mathematical routines as linear algebra, interpolation, optimization, integration, and statistics. The main functionality of the SciPy library is built upon NumPy and its arrays. SciPy makes significant use of NumPy.

4. Matplotlib

This is undoubtedly my favorite and a quintessential Python library. You can create stories with the data visualized with Matplotlib. Another library from the SciPy Stack, Matplotlib plots 2D figures.

When to use? Matplotlib is the plotting library for Python that provides an object-oriented API for embedding plots into applications. It is a close resemblance to MATLAB embedded in Python programming language.

5. Seaborn

So when you read the official documentation on Seaborn, it is defined as the data visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics. Putting it simply, seaborn is an extension of Matplotlib with advanced features.

So, what is the difference between Matplotlib and Seaborn? Matplotlib is used for basic plotting; bars, pies, lines, scatter plots and stuff whereas, seaborn provides a variety of visualization patterns with less complex and fewer syntax.

6. Scikit Learn

Introduced to the world as a Google Summer of Code project, Scikit Learn is a robust machine learning library for Python. It features ML algorithms like SVMs, random forests, k-means clustering, spectral clustering, mean shift, cross-validation and more... Even NumPy, SciPy and related scientific operations are supported by Scikit Learn with Scikit Learn being a part of the SciPy Stack.

Start with python

At the first we should import libraries in python IDLE .

mabahes.py - E:\disk D\99\mabahes\end\mabahes.py (3.9.0)

File Edit Format Run Options Window Help

```
1  import pandas as pd
2  import math
3  import statistics
4  import numpy as np
5  import scipy.stats
6  from scipy import stats
7  import csv
8  from plotnine import ggplot,facet_grid,aes, labs, geom_point
9  import matplotlib.pyplot as plt
10 from sklearn.linear_model import LinearRegression
11 import statsmodels.api as sm
12 import seaborn as sns
13 from statsmodels.api import OLS
14 from matplotlib import pyplot
15 from scipy.optimize import curve_fit
16 from numpy import arange
17
18
```

In general , We should not use all libraries .

Next step : we must import CSV file .

```
19 ## import data
20 df = pd.read_csv('D:/99/mabahes/end/dd.csv')
21 print(np.array(df))
```

Out put :

```
*Python 3.9.0 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct  5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: E:\disk D\99\mabahes\end\mabahes.py =================
[[   48   8782  10680 1760433  992699       0       0  542917  257649]
 [   49   9248  10966 1846935 1069331       0       0  623483  297295]
 [   50   8706  10602 1874746 1128112       0       0  702481  341018]
 [   51   9509  11728 1734310 1069331  161754   97464  797565  390881]
 [   52  10025  11748 1825407 1137189  361084  210381  668600  330668]
 [   53  18587  22400 1957533 1180938  576045  330293  547629  269295]
 [   54  40574  42280 2173074 1348350  739710  411746  524235  273383]
 [   56  79978  95446 2321089 1491431  824242  459419  558411  297107]
 [   57  97434 114435 2459239 1619235  875516  493394  608754  332835]
 [   58 108298 126838 2581835 1732580  917571  528769  634712  354566]
 [   59 119662 139835 2625342 1769630  972137  563126  693809  393750]
 [   60 109123 127363 3067778 1941547 1063448  626443  674398  438830]
 [   61  81163  90840 2899086 1899506  975359  599705  626861  416536]
 [   62  92554 103435 3170008 2113369 1078543  671334  572081  414349]
 [   63  85329  92196 3308208 2284600 1055466  661631  576781  399662]
 [   64 100960 101939 3468923 2525480 1114107  703546  591049  424139]
 [   65  36135  41639 3615988 2720028 1236523  782213  628377  450354]
 [   66  50376  56610 3828105 2960218 1355589  854653  700784  488771]
 [   67  58451  64986 4058853 3173967 1406118  893392  768422  509599]
 [   68  69064  77345 4307333 3450374 1498834  968082  825448  550813]
 [   69  85223  92756 4537629 3724812 1638550 1086056  944675  628342]
 [   70 103345 114151 4790250 4026895 1782133 1213665 1000835  671812]
 [   71 108151 119341 5041319 4328327 1892510 1340172 1066653  752813]
```

Showed as a matrix because we used numpy.array .

And then

```
24 ## show as a dataframe
25 pddf = pd.DataFrame(df)
26 print(pddf)
27 a = np.array(df)
```

Output :

	year	g	b	gg	bb	ggg	bbb	gggg	bbbb
0	48	8782	10680	1760433	992699	0	0	542917	257649
1	49	9248	10966	1846935	1069331	0	0	623483	297295
2	50	8706	10602	1874746	1128112	0	0	702481	341018
3	51	9509	11728	1734310	1069331	161754	97464	797565	390881
4	52	10025	11748	1825407	1137189	361084	210381	668600	330668
5	53	18587	22400	1957533	1180938	576045	330293	547629	269295
6	54	40574	42280	2173074	1348350	739710	411746	524235	273383
7	56	79978	95446	2321089	1491431	824242	459419	558411	297107
8	57	97434	114435	2459239	1619235	875516	493394	608754	332835
9	58	108298	126838	2581835	1732580	917571	528769	634712	354566
10	59	119662	139835	2625342	1769630	972137	563126	693809	393750
11	60	109123	127363	3067778	1941547	1063448	626443	674398	438830
12	61	81163	90840	2899086	1899506	975359	599705	626861	416536
13	62	92554	103435	3170008	2113369	1078543	671334	572081	414349
14	63	85329	92196	3308208	2284600	1055466	661631	576781	399662
15	64	100960	101939	3468923	2525480	1114107	703546	591049	424139
16	65	36135	41639	3615988	2720028	1236523	782213	628377	450354
17	66	50376	56610	3828105	2960218	1355589	854653	700784	488771

Note : we know use Data Frame so we use pandas library for data to change data frame and call that with pddf name .

Next step :

For example, we get some operations such as R.

```
## mean
d = np.mean(df)
print(d)
## mean
print("var")
a2 = df.var(ddof=1)
print(a2)
## ks test
print('ks test')
print(stats.kstest(df['b'],df['g']))
```

Output:

Mean :

```
year    7.235417e+01
g       1.308601e+05
b       1.322380e+05
gg      3.487624e+06
bb      2.904961e+06
ggg     1.587340e+06
bbb     1.242380e+06
gggg    1.238511e+06
bbbb    1.033069e+06
dtype: float64
```

Variance :

```
var
year    2.022336e+02
g       8.405851e+09
b       7.827607e+09
gg      1.100020e+12
bb      1.261263e+12
ggg     7.165491e+11
bbb     5.722421e+11
gggg    3.251030e+11
bbbb    4.132904e+11
dtype: float64
```

Ks test :

```
ks test
KstestResult(statistic=0.125, pvalue=0.85283384171513)
```

```
41 ## summary
42 print(df.describe())
43 result = scipy.stats.describe(a, ddof=1, bias=False)
44 print(result)
```

Output :

Line 42 we used describe code for obtain summary :

```
        year          g  ...        gggg        bbbb
count  48.000000    48.000000  ...  4.800000e+01  4.800000e+01
mean   72.354167  130860.083333  ...  1.238511e+06  1.033069e+06
std    14.220886   91683.428190  ...  5.701781e+05  6.428766e+05
min    48.000000    8706.000000  ...  5.242350e+05  2.576490e+05
25%    60.750000   70740.250000  ...  6.601280e+05  4.106772e+05
50%    72.500000  102152.500000  ...  1.161534e+06  9.208645e+05
75%    84.250000  226624.500000  ...  1.820710e+06  1.666562e+06
max    96.000000  338704.000000  ...  2.065265e+06  1.998449e+06
```

We can see mean standard deviation and etc .

In the next print :

We used scipy library for obtain summary .

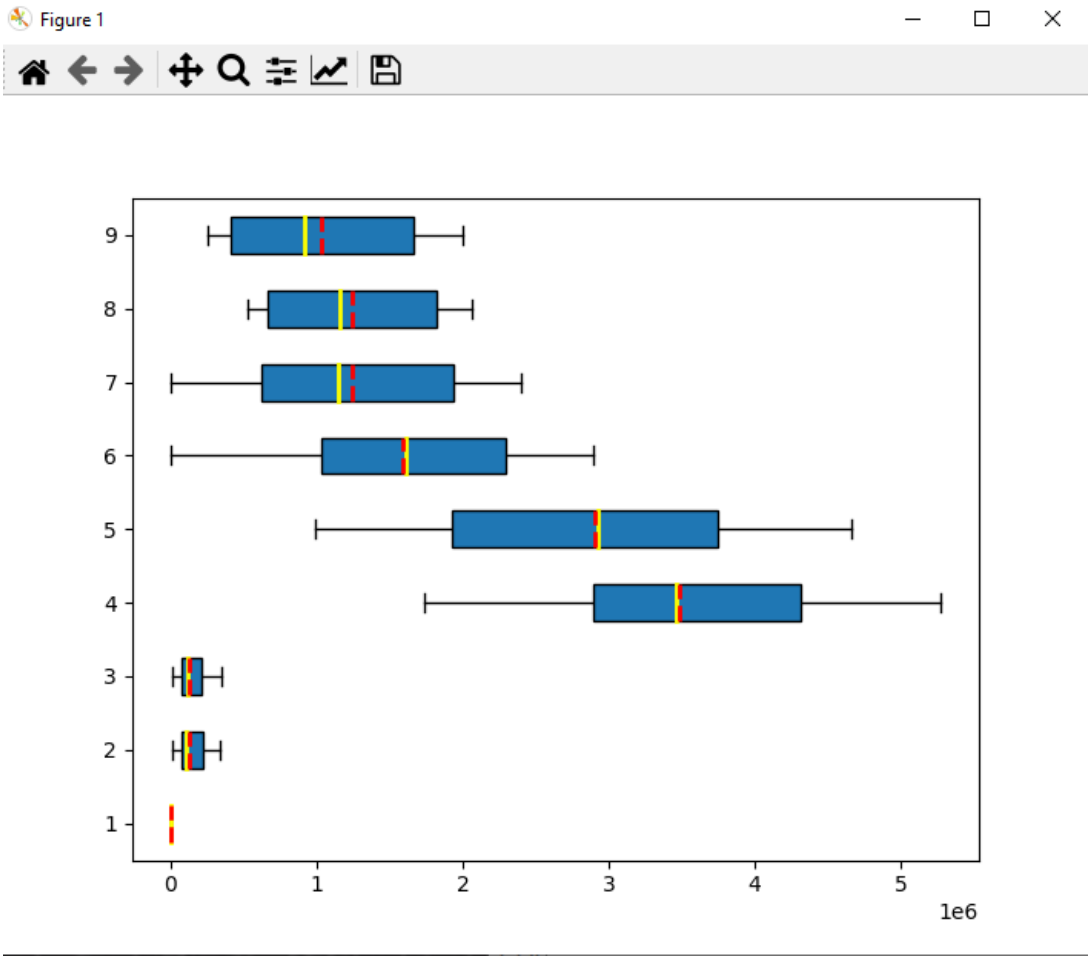
Output :

```
[8 rows x 9 columns]
DescribeResult(nobs=48, minmax=(array([    48,    8706,   10602, 1734310,
992699,       0,       0,
        524235,  257649], dtype=int64), array([    96,  338704,  349748, 52705
33, 4666836, 2889797, 2404875,
       2065265, 1998449], dtype=int64)), mean=array([7.23541667e+01, 1.308
60083e+05, 1.32237979e+05, 3.48762367e+06,
       2.90496108e+06, 1.58733973e+06, 1.24238019e+06, 1.23851138e+0
6,
       1.03306919e+06]), variance=array([2.02233599e+02, 8.40585100e+09,
7.82760667e+09, 1.10002010e+12,
       1.26126255e+12, 7.16549095e+11, 5.72242066e+11, 3.25103019e+1
1,
       4.13290364e+11]), skewness=array([-0.04005161,  0.63203533,  0.639
80655,  0.07310676, -0.1759978 ,
       -0.11150676,  0.02351265,  0.08204523,  0.17942308]), kurtosis=array([
-1.16645346, -0.67844912, -0.31807235, -0.94211753, -1.04715452,
       -0.90356133, -1.27509001, -1.74331754, -1.71329171]))
```

We can see the differences .

Boxplot :

```
# box plot
fig, ax = plt.subplots()
ax.boxplot((df), vert=False, showmeans=True, meanline=True,
           patch_artist=True,
          medianprops={'linewidth': 2, 'color': 'yellow'},
          meanprops={'linewidth': 2, 'color': 'red'})
plt.show()
```



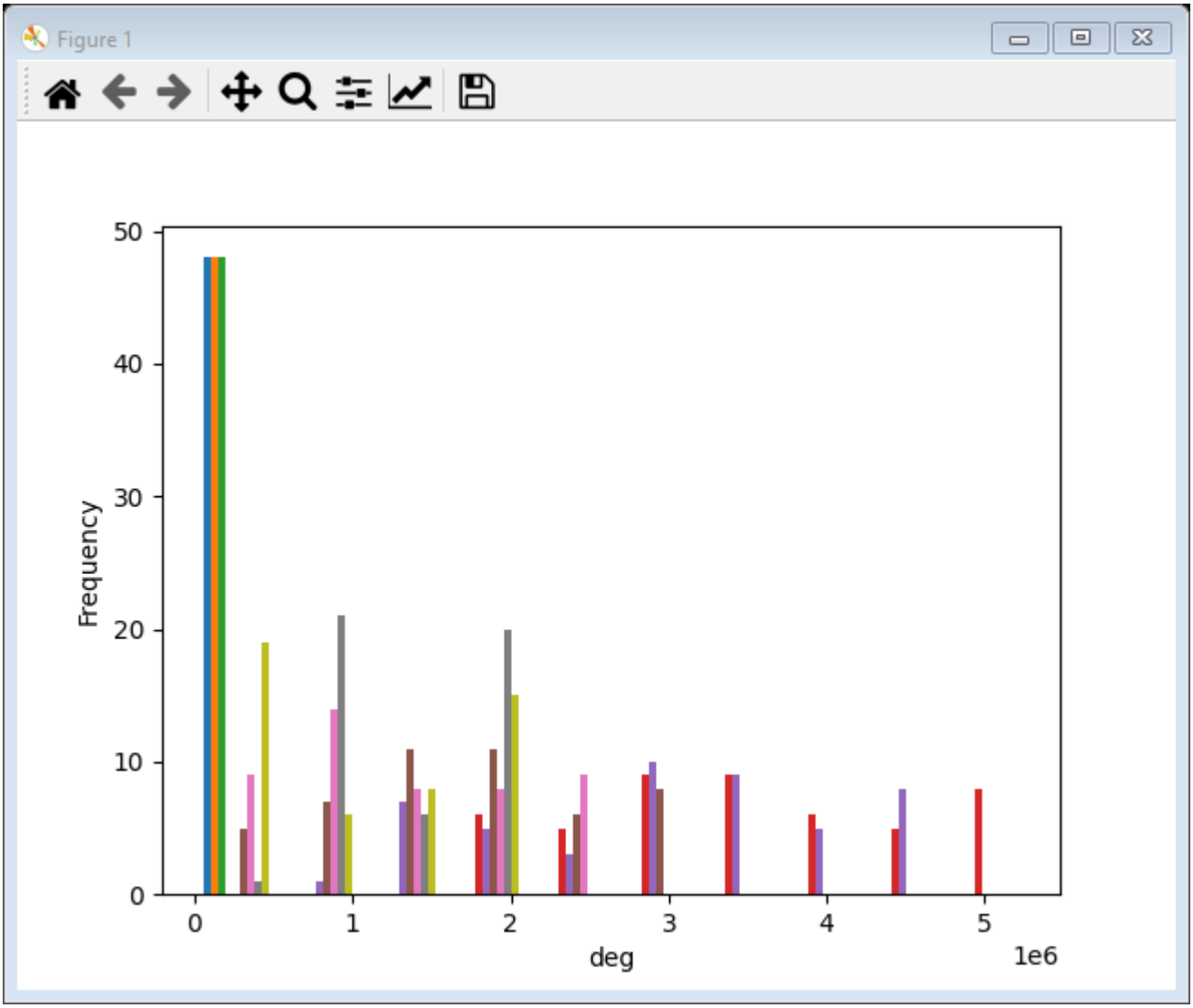
For each column we drew a boxplot

- **The mean** is the red dashed line.
- **The median** is the yellow dashed line.

Histogram :

```
#histogram
fig, ax = plt.subplots()
ax.hist(df, cumulative=False)
ax.set_xlabel('deg')
ax.set_ylabel('Frequency')
plt.show()

ax.set_ylabel('Frequency')
plt.show()
```

Output :



Draw plot for all columns :

```
######    plot    ########

plt.plot(df['g'])
plt.show()

plt.plot(df['b'])
plt.show()

plt.plot(df['gg'])
plt.show()

plt.plot(df['bb'])
plt.show()

plt.plot(df['ggg'])
plt.show()

plt.plot(df['bbb'])
plt.show()

plt.plot(df['gggg'])
plt.show()

plt.plot(df['bbbb'])
plt.show()
```

Respectively :

G :

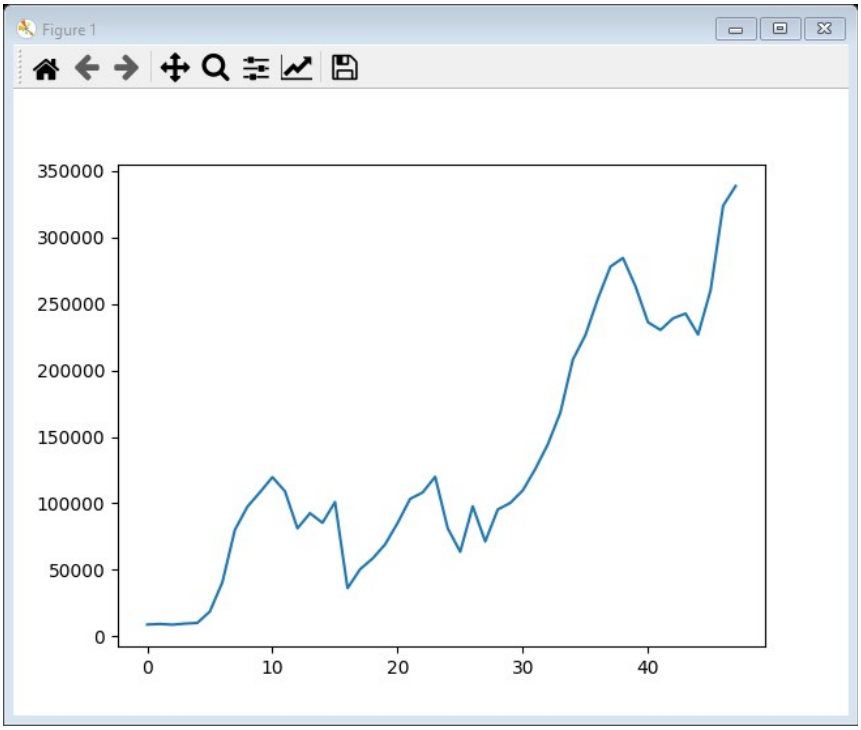


Figure 1
350000
300000
250000
200000
150000
100000
50000
0
0
10
20
30
40

b:

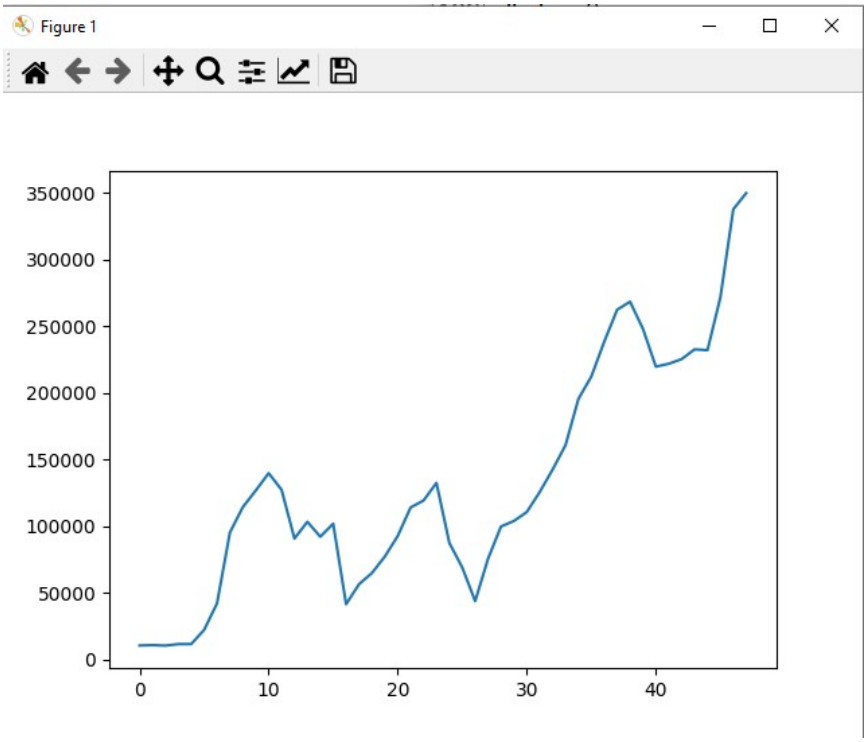


Figure 1
350000
300000
250000
200000
150000
100000
50000
0
0
10
20
30
40

gg:

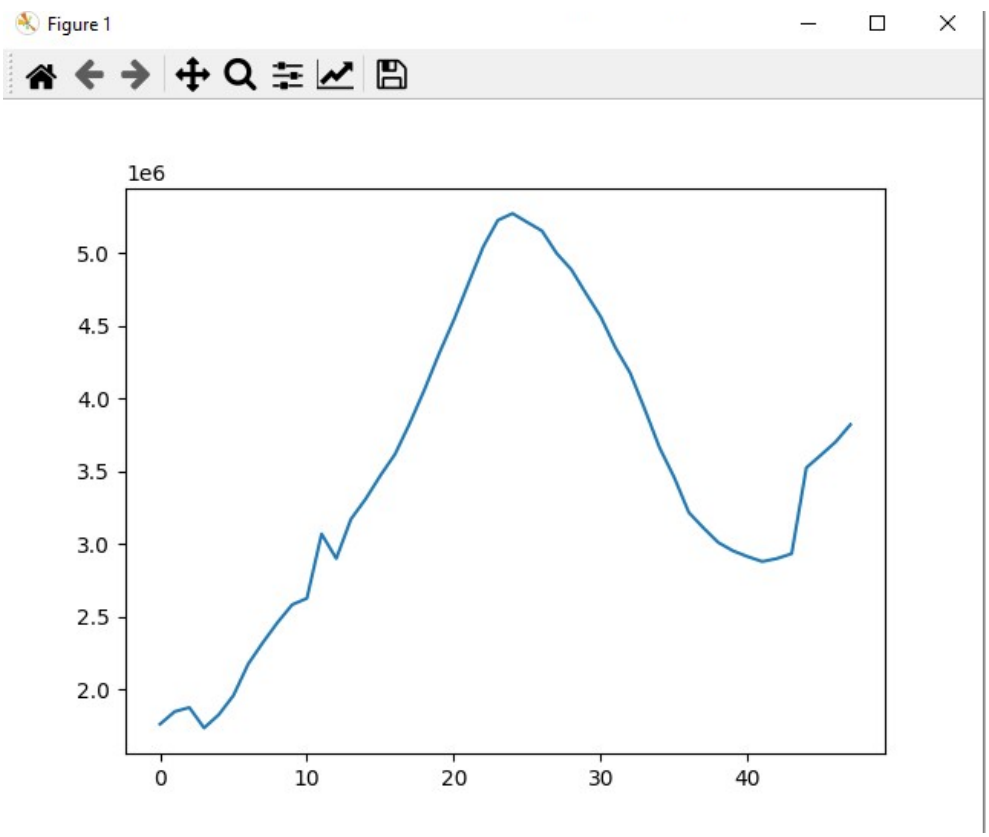


Figure 1
1e6
5.0
4.5
4.0
3.5
3.0
2.5
2.0
0
10
20
30
40

bb:

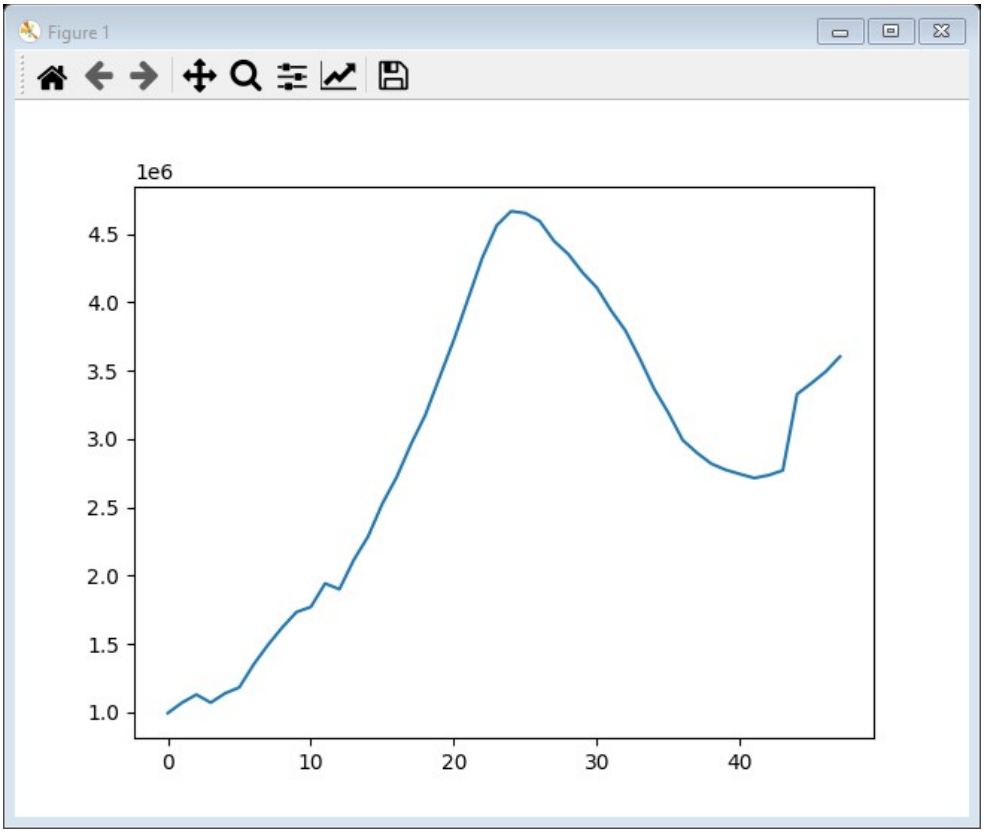
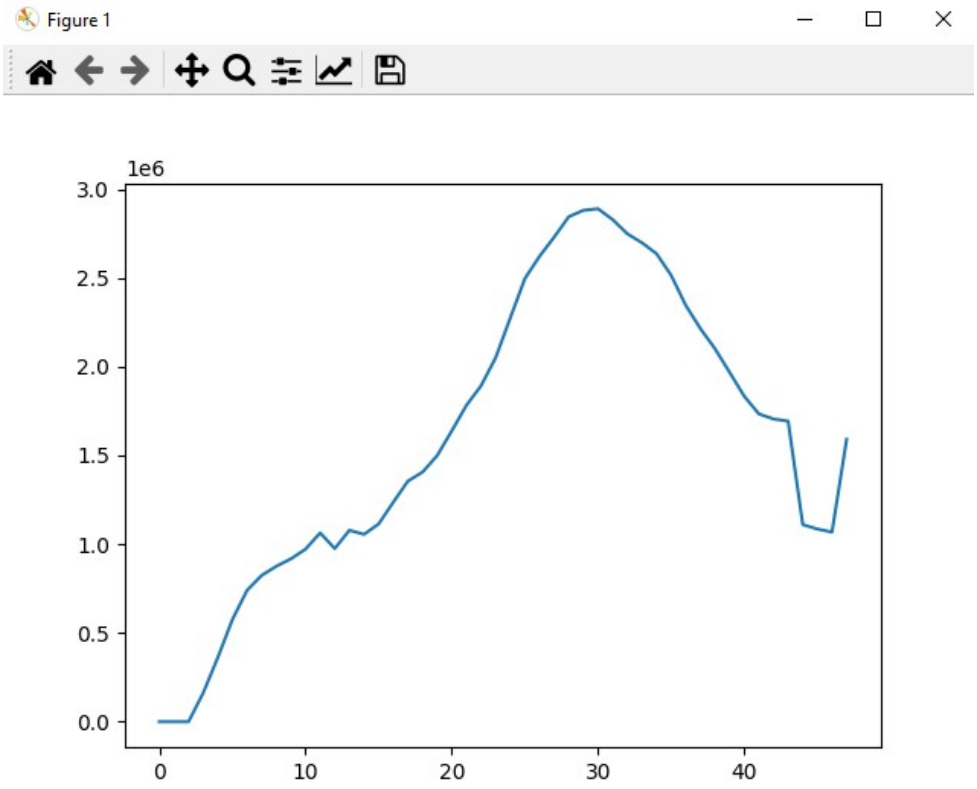
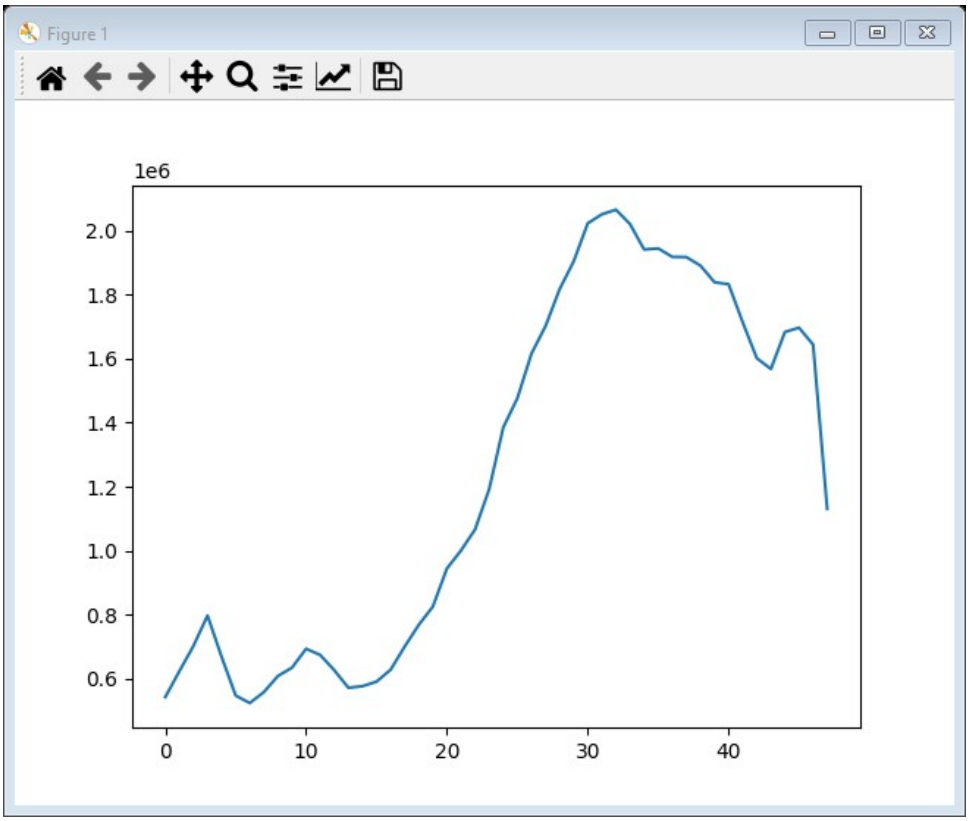


Figure 1
1e6
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
10
20
30
40

ggg:



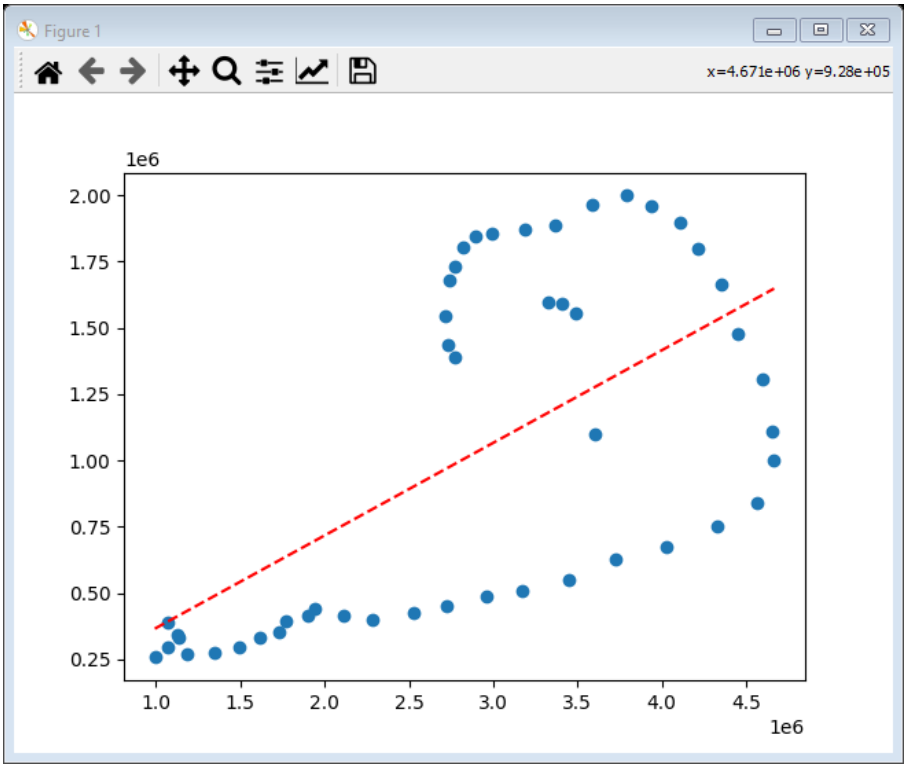
bbbb:



Although

Creat curve and draw scatter plot :

```
### Curve
def objective(x, a, b):
        return a * x + b
# choose the input and output variables
data = df.values
# choose the input and output variables
x, y = data[:, 4], data[:, -1]
# curve fit
popt, _ = curve_fit(objective, x, y)
# summarize the parameter values
a, b = popt
print('y = %.5f * x + %.5f' % (a, b))
# plot input vs output
pyplot.scatter(x, y)
# define a sequence of inputs between the smallest an
x_line = arange(min(x), max(x), 1)
# calculate the output for the range
y_line = objective(x_line, a, b)
# create a line plot for the mapping function
pyplot.plot(x_line, y_line, '--', color='red')
pyplot.show()
```



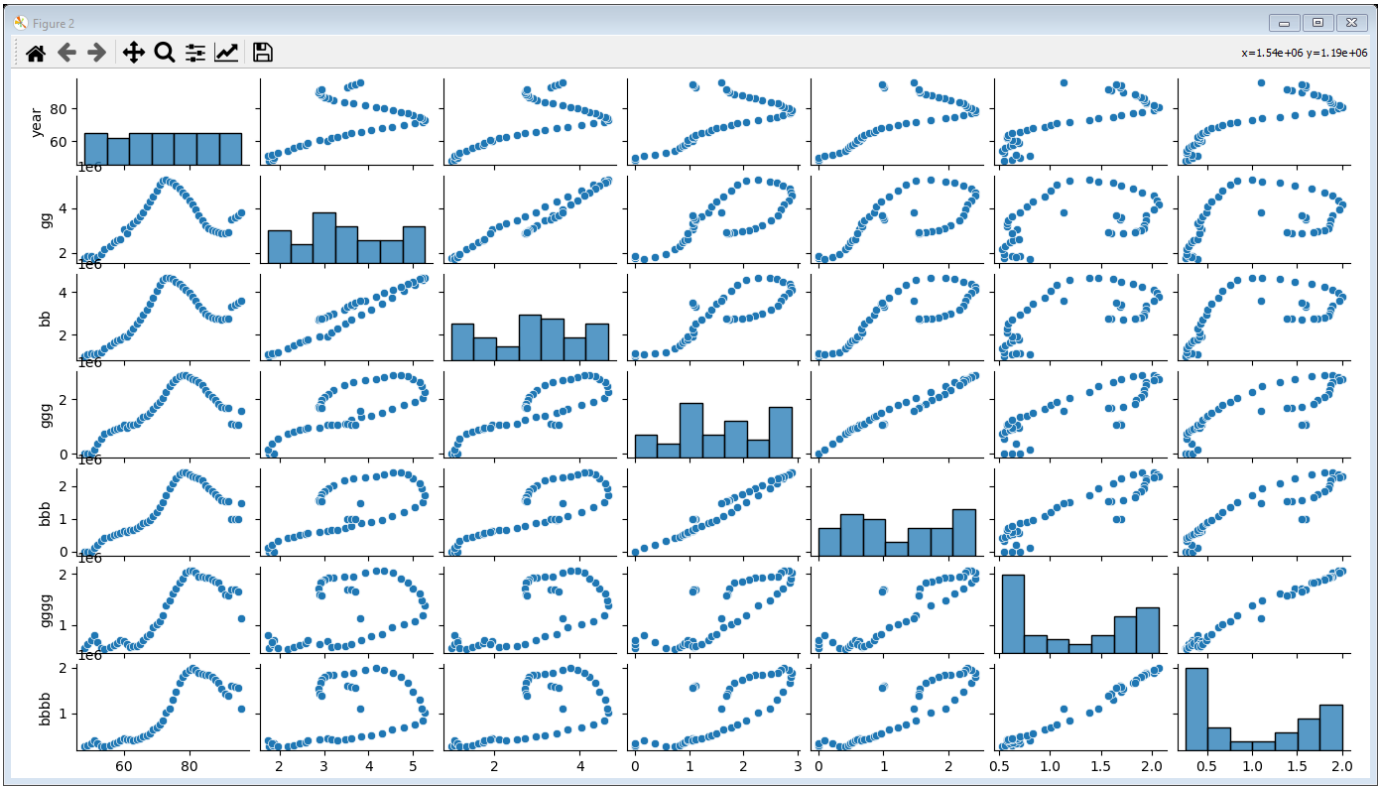
Use simple.eda such as R *

corr() is used to find the pairwise correlation of all columns in the dataframe. Any na values are automatically excluded

In this part , we had to use (import seaborn as sns

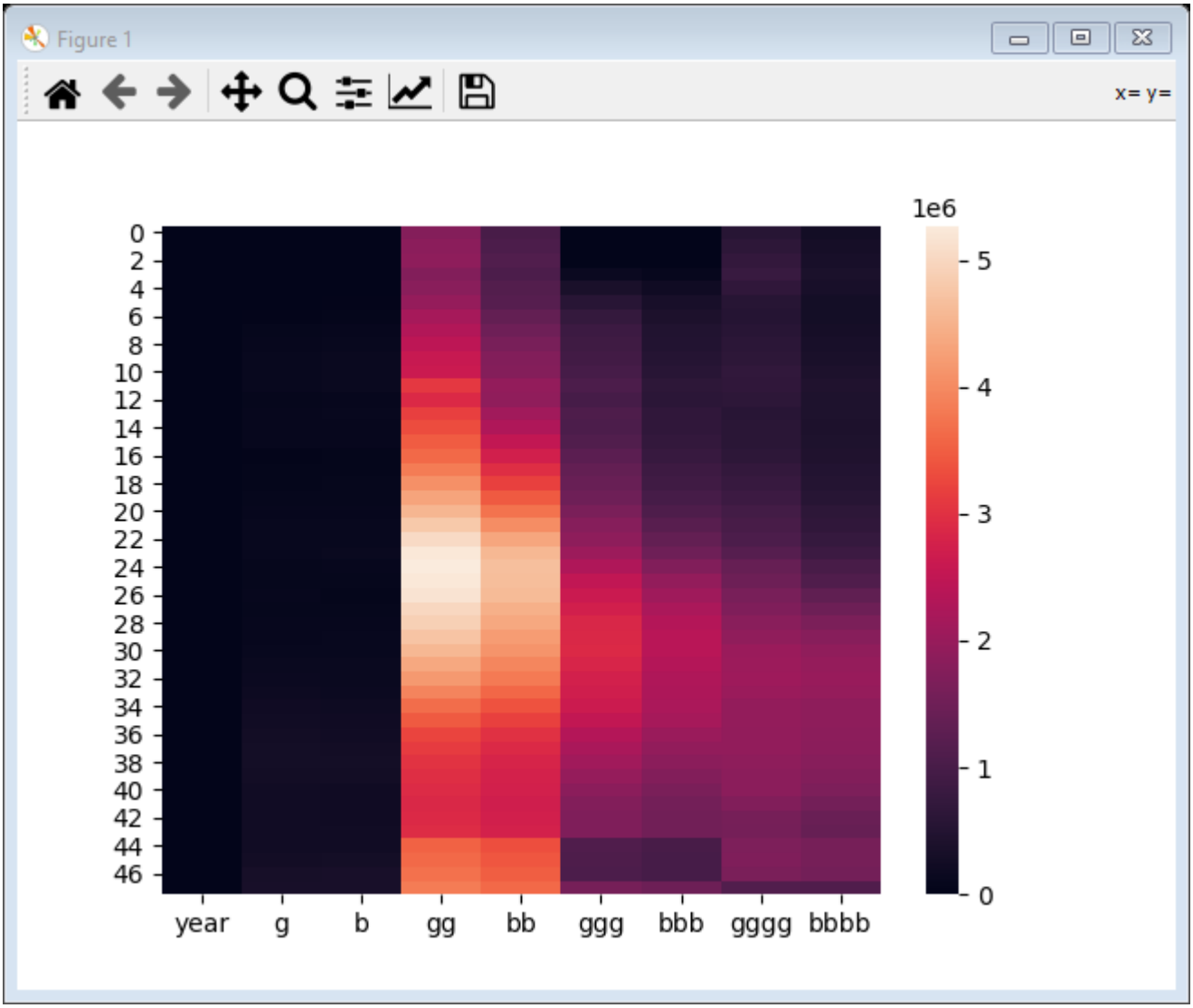
import matplotlib.pyplot as plt) library for run code.

```
f, ax = plt.subplots(figsize=(10, 8))
corr = df.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
##
s = df.drop(['b','g'], axis=1)
sns.pairplot(s)

plt.show()
```



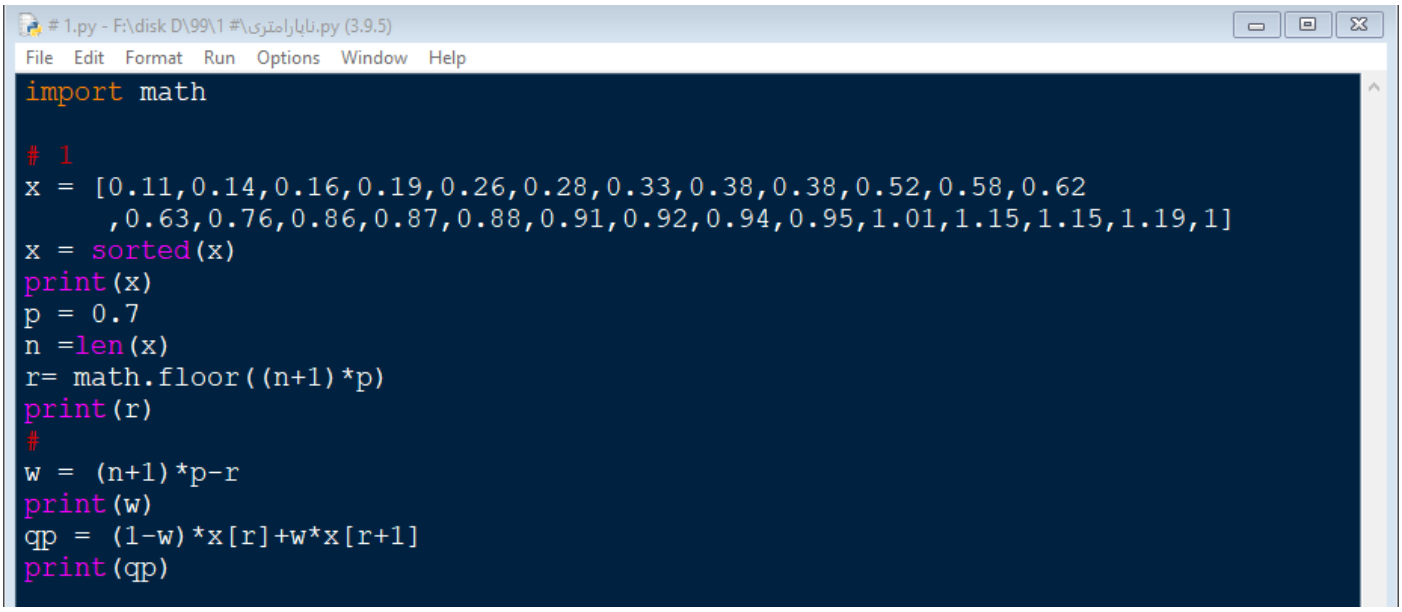
Heat map :

```
import csv
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

## import data
df = pd.read_csv('E:/disk D/99/mabahes/end/dd.csv')
uniform_data = df
ax = sns.heatmap(uniform_data)
plt.show()
```

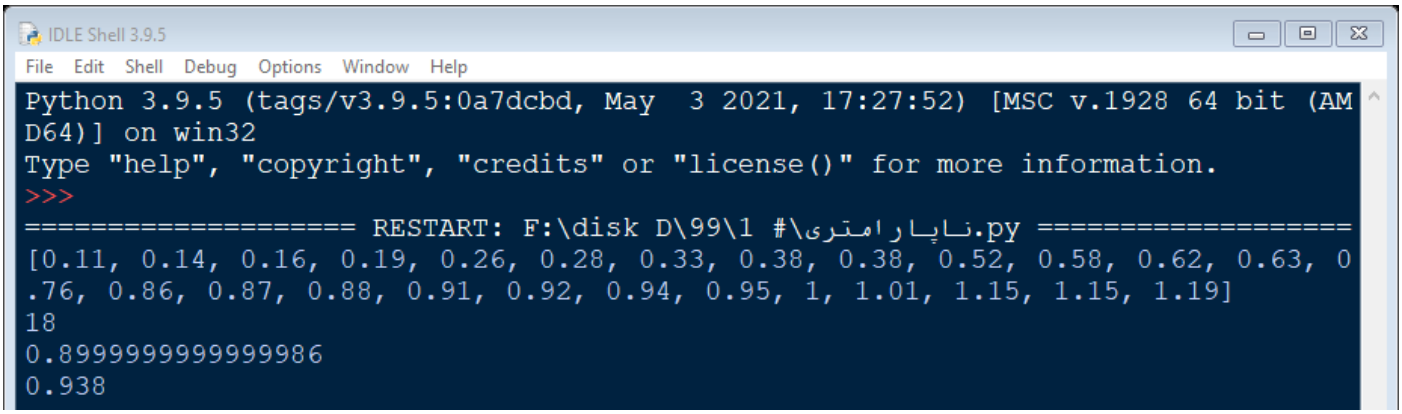


Example number 1 :

Code :

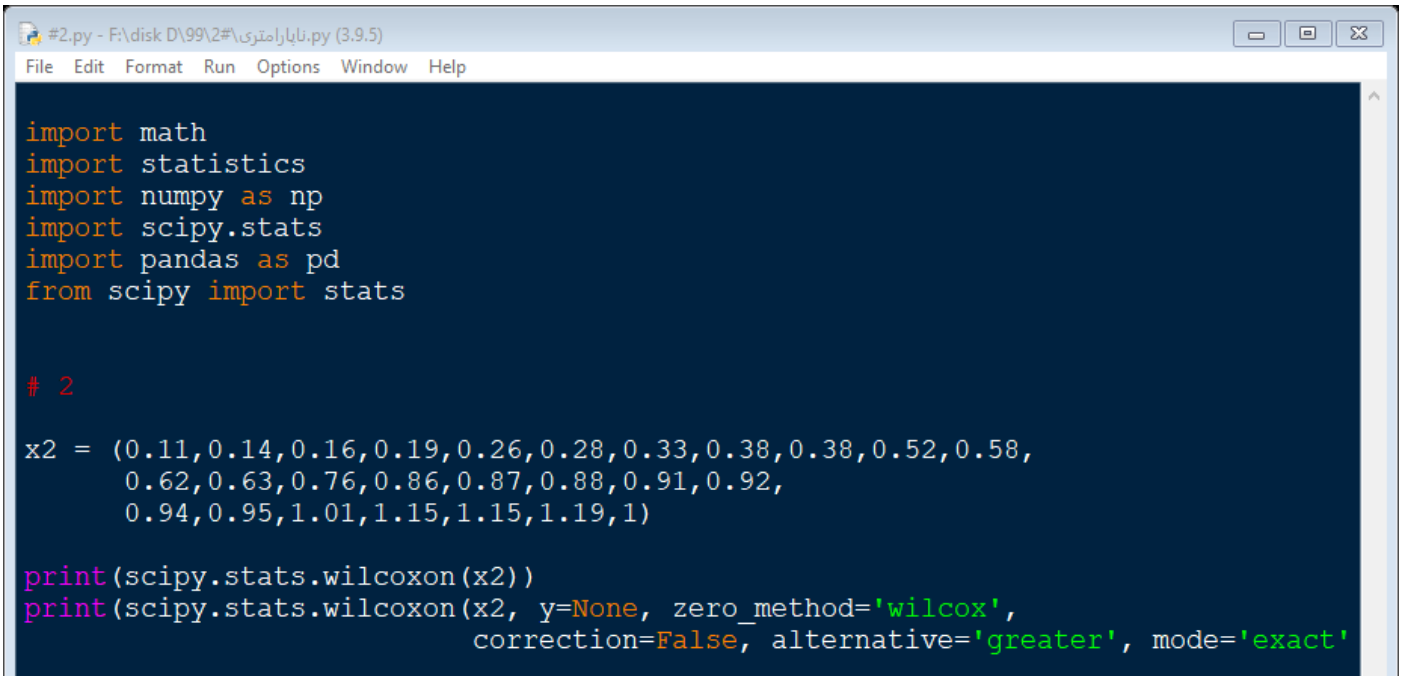


Output :

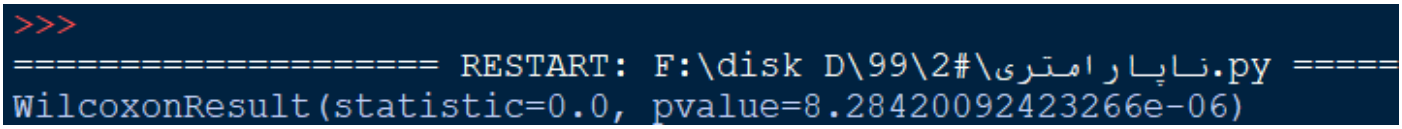


Example number 2 :

Code :



Output :



Example number 7 :

Code :

```
#7.py - F:\disk D\99\7#\ناپارامتری.py (3.8.10)
File  Edit  Format  Run  Options  Window  Help
import math
import statistics
import numpy as np
import scipy.stats
import pandas as pd
from scipy import stats


##7
a = np.array([75,69.8,85.7,74,69,83.3,68.9,77.8,72.2,77.4])
b = np.array([85.4,83.1,80.2,74.5,70,81.5,75.4,78,85.4,80.4])
d = b-a
d = d[d!=0]
print("d")
print(d)

n = len(d)
print("n")
print(n)
print("B")
B= len(d[d>0])
print(B)
#n=10000
#p=10/19
#k=0
l1 =scipy.stats.binom.cdf(B,n,1/2)
l2 =1 - scipy.stats.binom.cdf(B-1,n,1/2)
l1 = min(scipy.stats.binom.cdf(B,n,1/2),1 - scipy.stats.binom.cdf(B-1,n,1/2))
pval = 2*(l1)
print(l1)
```

Output:

```
========== RESTART: F:\disk D\99\7#\ناپارامتری.py ==
d
[10.4 13.3 -5.5  0.5  1.  -1.8  6.5  0.2 13.2  3. ]
n
10
B
8
0.0546875
```

Example number 7 :

Code :

#9.py - F:\disk D\99\9#\ناپارامتری.py (3.8.10)

File Edit Format Run Options Window Help

```
import numpy as np
import tkinter
import math
import statistics
import pandas as pd
# matrix rank
from numpy.linalg import matrix_rank
import numpy.linalg as npl
import qnorm

A = [1,5,7,8,13,15,20,21,23,24,25,27,28,29,30]
B = [2,3,4,6,9,10,11,12,14,16,17,18,19,22,26]
n =len(A)
m =len(B)
N = n+m
C= (B,A)
R = npl.matrix_rank(C)
WB = sum(list(range(1, n)))
print('WB ',WB)
WAB = (WB - n)*(n+1)/2
print('WAB',WAB)
EWB = n*(n+1)/2
VWB=n*m*(n+1)/12
print('VWB',VWB)
vv = WB+(1/2)-(EWB)/math.sqrt(VWB)
pval = qnorm.quantile_normalize(vv)
```

Output :

IDLE Shell 3.8.10

File Edit Shell Debug Options Window Help

```
Python 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021, 11:48:03) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: F:\disk D\99\9#\ناپارامتری.py ==========
WB  105
WAB 720.0
VWB 300.0
```

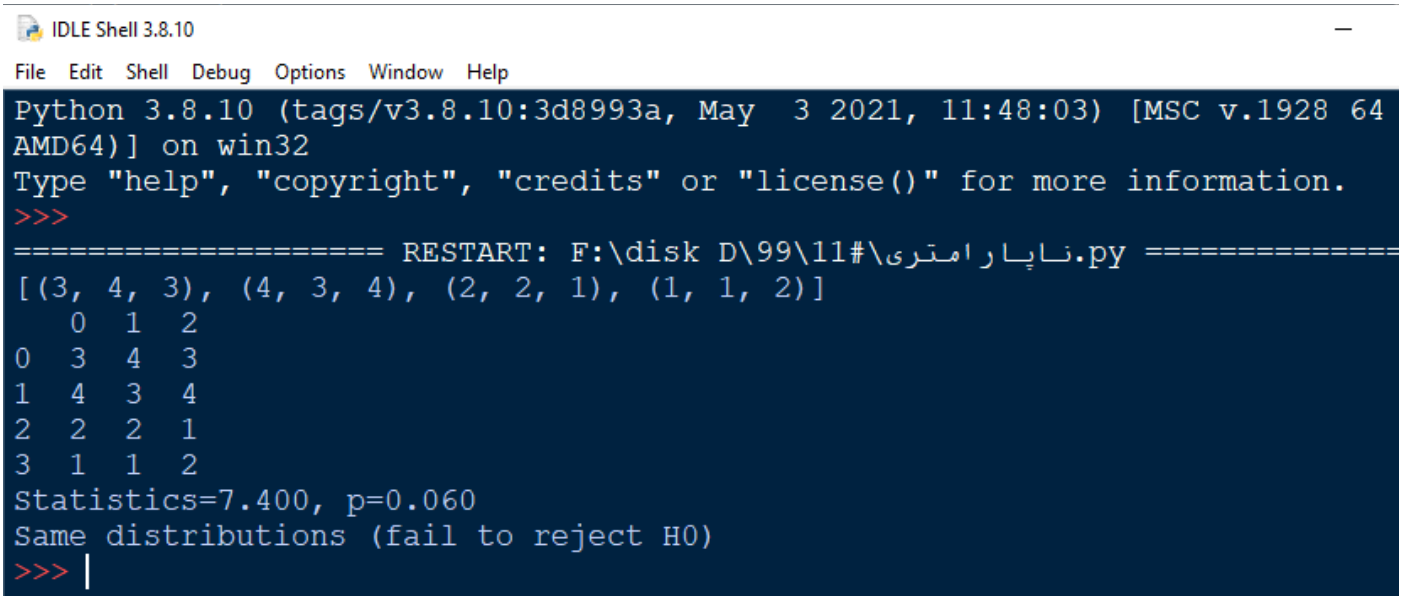
Example number 11 :

Code :

#11.py - F:\disk D\99\11#\ناپارامتری.py (3.8.10)

File Edit Format Run Options Window Help

```
import math
import statistics
import numpy as np
import scipy.stats
import pandas as pd
from scipy import stats

#
from scipy.stats import friedmanchisquare

## 11
x1=(3,4,3)
x2 = (4,3,4)
x3 = (2,2,1)
x4 = (1,1,2)
X = [x1,x2,x3,x4]

print(X)
df =pd.DataFrame(X)
print(df)


stat, p = friedmanchisquare(x1, x2, x3,x4)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
        print('Same distributions (fail to reject H0)')
else:
        print('Different distributions (reject H0)')

#friedman.test(x)
```


IDLE Shell 3.8.10
File Edit Shell Debug Options Window Help
Python 3.8.10 (tags/v3.8.10:3d8993a, May 3 2021, 11:48:03) [MSC v.1928 64
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: F:\disk D\99\11#\ناپارامتری.py ==========
[(3, 4, 3), (4, 3, 4), (2, 2, 1), (1, 1, 2)]
0 1 2
0 3 4 3
1 4 3 4
2 2 2 1
3 1 1 2
Statistics=7.400, p=0.060
Same distributions (fail to reject H0)
>>>

Example number 12 :

Code :

#12.py - F:\disk D\99\12#\ناپارامتری.py (3.8.10)

File Edit Format Run Options Window Help

```
import math
import statistics
import numpy as np
import scipy.stats
import pandas as pd
from scipy import stats

import matplotlib.pyplot as plt
from scipy.stats import kstest


# scipy.stats.kstest(rvs, cdf, args=(), N=20, alternative='


x= (0.621,0.503,0.203,0.477,0.710,0.581,0.329,0.480,0.554,0
ks = stats.kstest(x, 'norm', alternative='greater')
print(ks)
#seaborn.ecdfplot(data=None, *, x=None, y=None, hue=None, w
#, complementary=False, palette=None,
#  hue_order=None, hue_norm=None, log_scale=None, legend=Tr
import seaborn as sns

sns.ecdfplot(x)
plt.show()
```

Output :

IDLE Shell 3.8.10

File Edit Shell Debug Options Window Help

```
Python 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021, 11:48:03) [MSC v.1928 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: F:\disk D\99\12#\ناپارامتری.py ==========
KstestResult(statistic=0.2388520680899867, pvalue=0.27554425661824383)

========== RESTART: F:\disk D\99\12#\ناپارامتری.py ==========
```

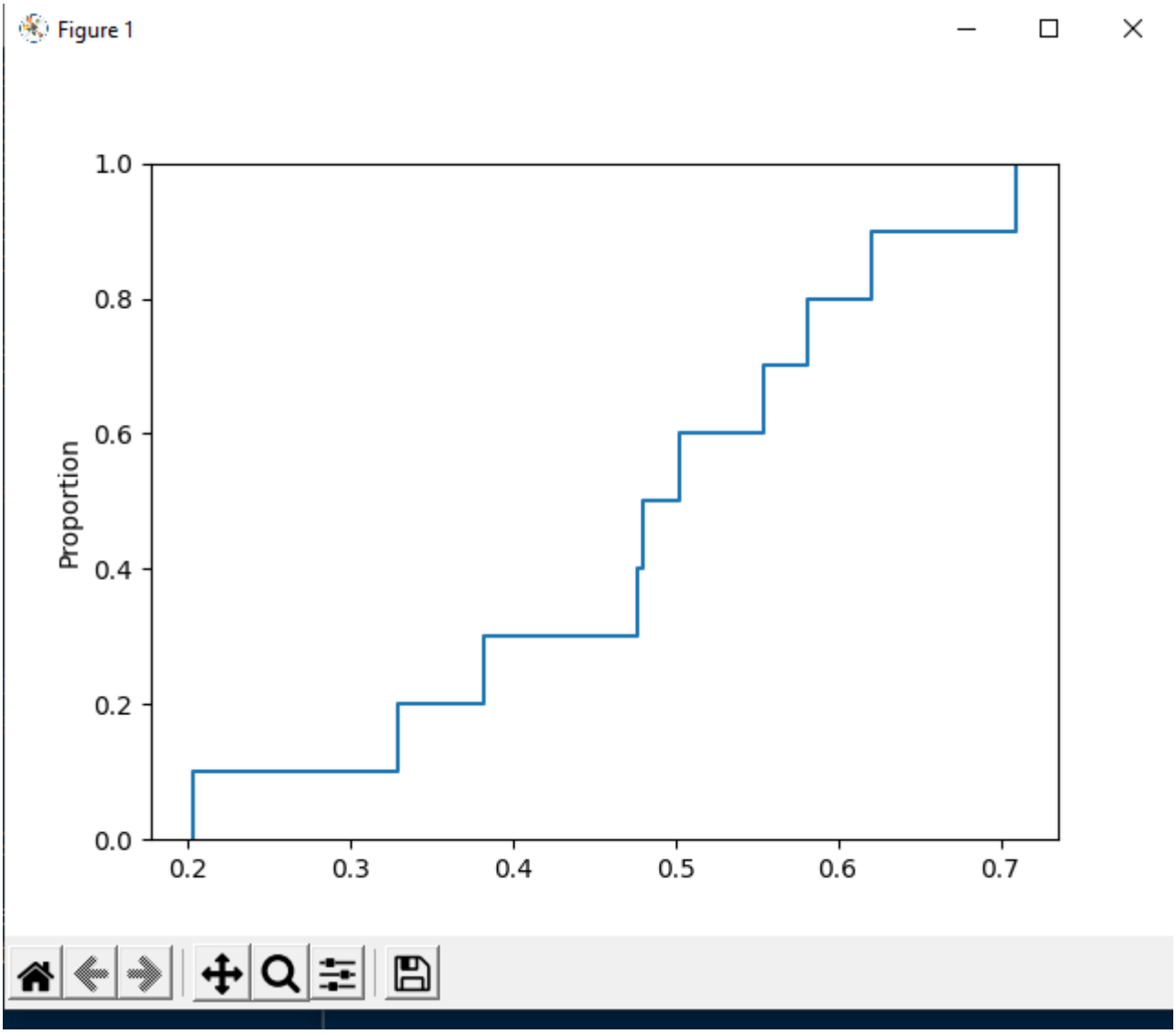


Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
Proportion
0.2
0.3
0.4
0.5
0.6
0.7

Example number 13 :

Code :

#13.py - F:\disk D\99\13#\ناپارامتری.py (3.8.10)

File Edit Format Run Options Window Help

```
import math
import statistics
import numpy as np
import scipy.stats
import pandas as pd
from scipy import stats

import matplotlib.pyplot as plt
x= (10.3,11.2,11.5,11.9,12.8)
y= (10.4,11.8,12.5,12.6,13.8,13.9)
print(stats.kstest(x,y))
#seaborn.ecdfplot(data=None, *, x=None, y=None, hue=None, weights=None, stat='pr
#, complementary=False, palette=None,
#  hue_order=None, hue_norm=None, log_scale=None, legend=True, ax=None, **kwargs
import seaborn as sns
sns.ecdfplot(x)
sns.ecdfplot(y)

plt.show()
```

Output :

```
Python 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021, 11:48:03) [MSC v.1928 64 bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: F:\disk D\99\13#\ناپارامتری.py ==========
KstestResult(statistic=0.4666666666666667, pvalue=0.47402597402597413)
```

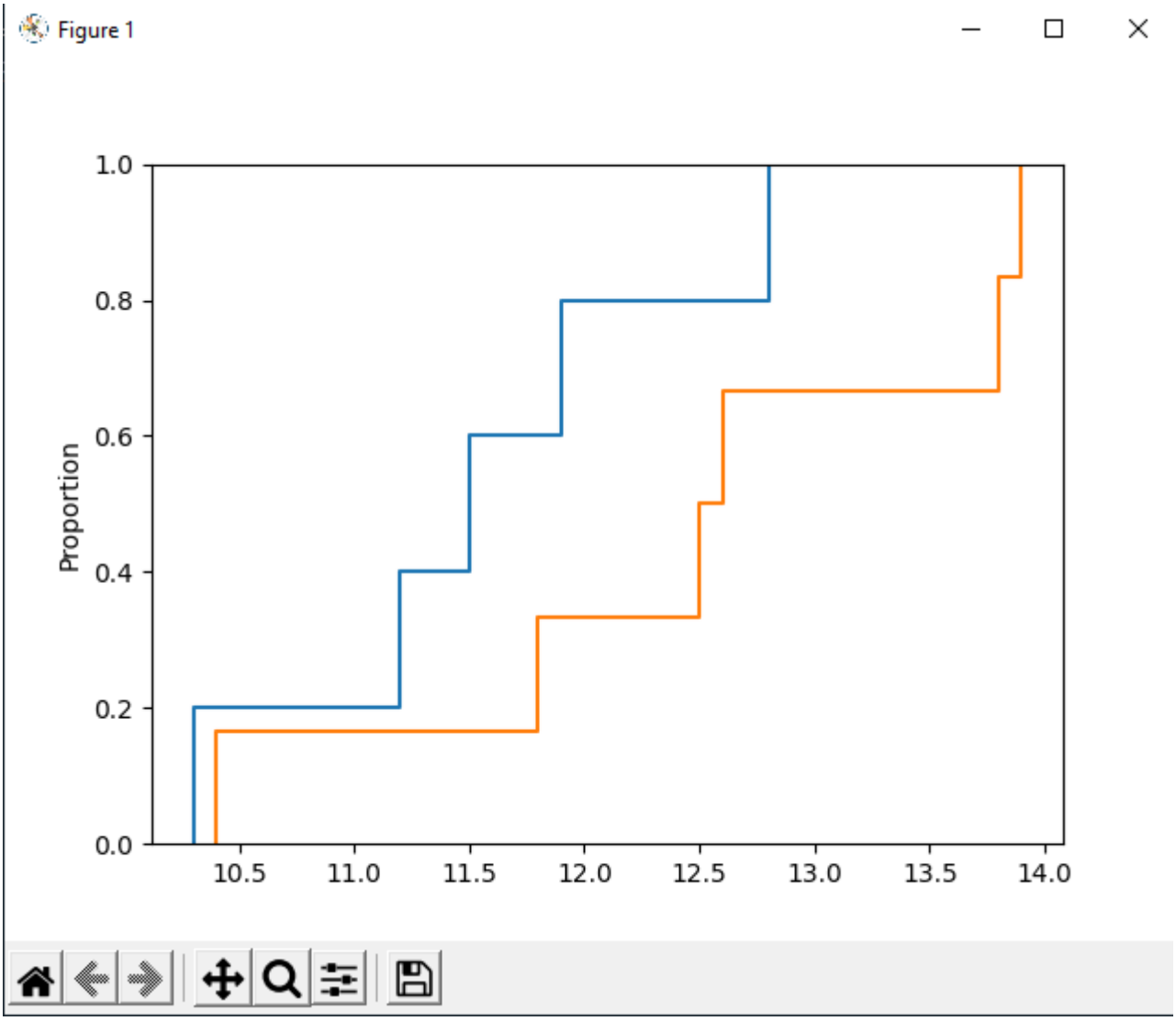


Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
Proportion
10.5
11.0
11.5
12.0
12.5
13.0
13.5
14.0

Conclusion :

We can reach these results :

- Python is fast , flexible and accurate .
- Ability to use multiple libraries simultaneously .
- Use a variety of editors for example : anaconda , pycharm or VScode and etc .

(but in this project ,I use IDLE)

- Use data with NAN quantities with specific codes .