Data manipulation: Plotting and Visualization

AAA-Python Edition

Plan

- 1- matplotlib
- 2- Plotting with Pandas and seaborn
- 3- Interactive and dynamic graphics

1- matplotlib

Basic Plotting

```
%matplotlib inline
```

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

Importing pyplot module from matplotlib (as plt)

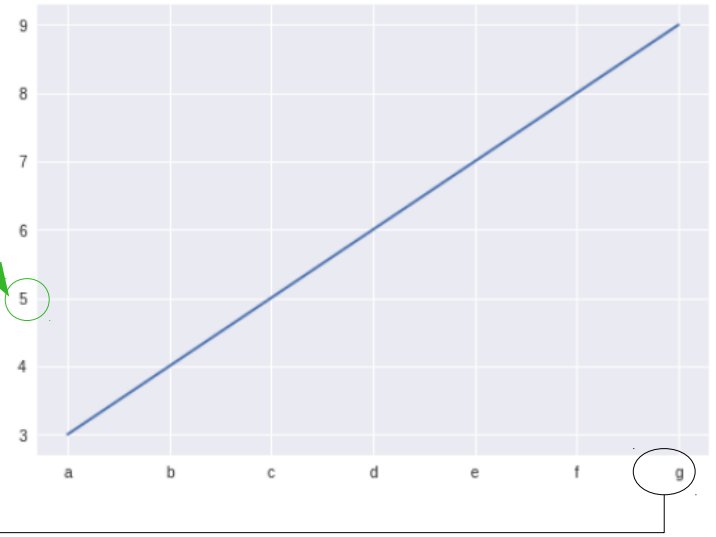
You have to use this command, to be able to see all the plots (embed an image after each plot)

ser1

```
a    3
b    4
c    5
d    6
e    7
f    8
g    9
dtype: int64
```

```
# plotting ser1 values as a line
plt.plot(ser1)
```

```
[<matplotlib.lines.Line2D at 0x7f273ec0f940>]
```



The series **values** are the **y** values, the series **index** values, are the **x** values

[By Amina Delali]

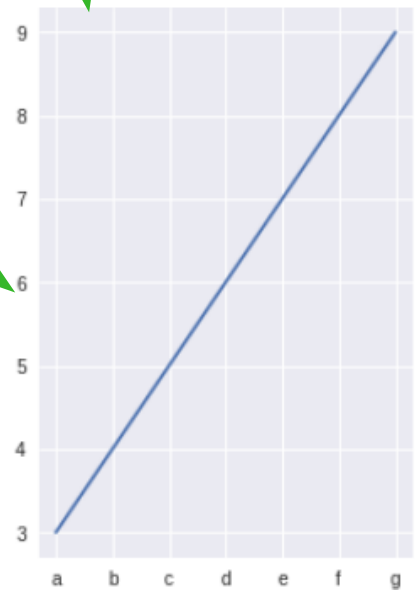
1- matplotlib

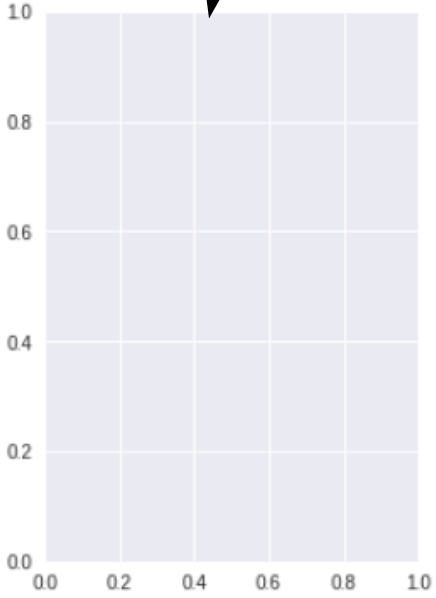
Basic Plotting

```
1  # annother way to plot lines (figures)
2  # using figures, axis and subplots
3
4  # creating a figure instance
5  myFig= plt.figure()
6  # adding to the figure a subplot:
7  # divinding the corresponding region
8  # in (1,1) ==> 1 line x 2 columns
9  # the third value 1: position (number of the subplot)
10 axis1= myFig.add_subplot(1,2,1)
11 axis2= myFig.add_subplot(1,2,2)
12 # plotting in the first position
13 axis1.plot(ser1)
```

To divide the region in **4** subplots just use: **add_subplot(2,2**,"position")

To plot in this part just use: **axis2.plot(ser1)**



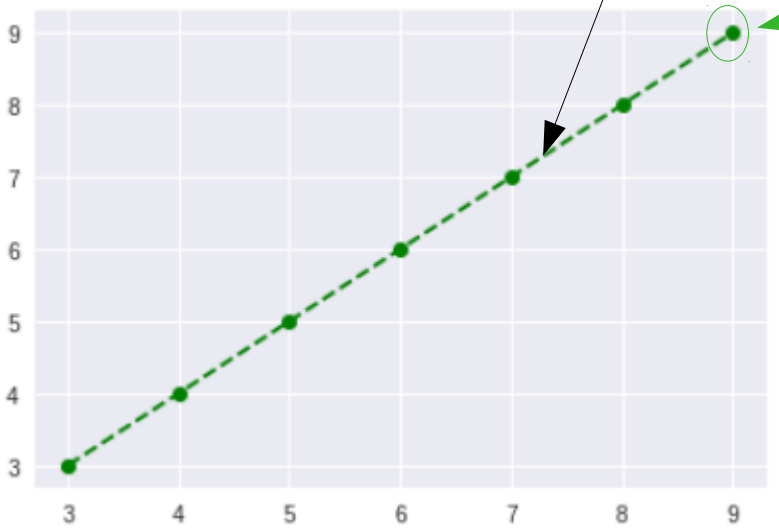


[By Amina Delali]

1- matplotlib

Customizing plots

```
# you can specify : the color, the line_style
# and the marker type of the plot
plt.plot(ser1, color='g', linestyle='dashed', marker='o')
```



colors

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

character	description
``'-'``	solid line style
``'--'``	dashed line style
``'-.'``	dash-dot line style
``':'``	dotted line style
``'.'``	point marker
``','``	pixel marker
``'o'``	circle marker
``'v'``	triangle_down marker
``'^'``	triangle_up marker
``'<'``	triangle_left marker

(From google colab help)

Some line styles and markers

[By Amina Delali]

1- matplotlib

Basic Plotting

```
1  myFig2= plt.figure()
2  axis1= myFig2.add_subplot(1,1,1)
3  # modify the values of x to be shown
4  axis1.set_xticks([0,2,4,6])
5  # modify theire names
6  axis1.set_xticklabels(list("ABCD"))
7  # adding a title to the plot
8  axis1.set_title ("Plotting ser1")
9  # ploting axis1 wiht a label "range"
10 axis1.plot(ser1,label="range",marker="o")
11 # adding a legend
12 axis1.legend(loc="best")
```

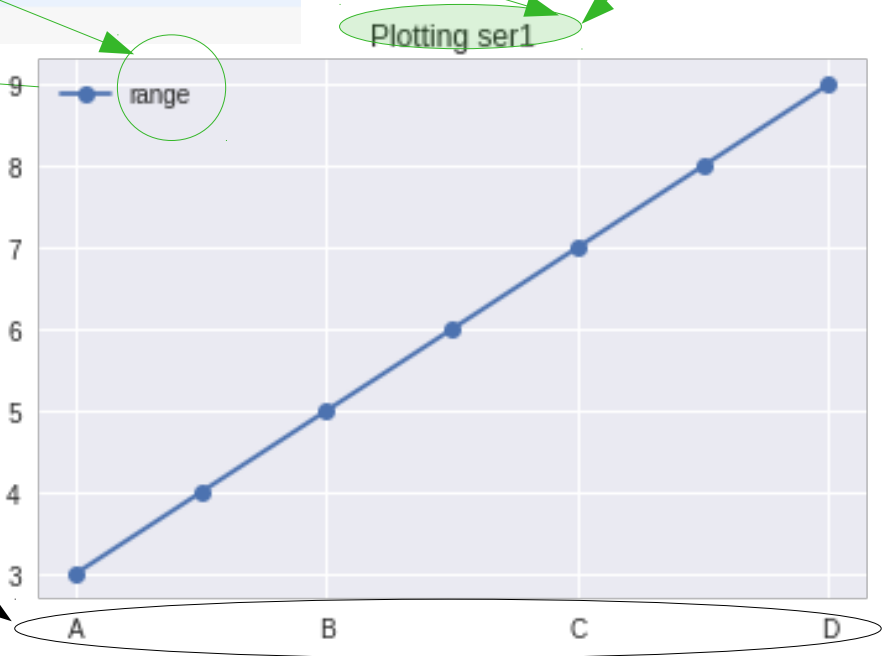
Mark the x axis at the positions: 0, 2 4 and 6 and labeled A, B, C and D

The title

ser1

```
a    3
b    4
c    5
d    6
e    7
f    8
g    9
dtype: int64
```

The xticklabes: **4**
xticks ==> **4**
labels

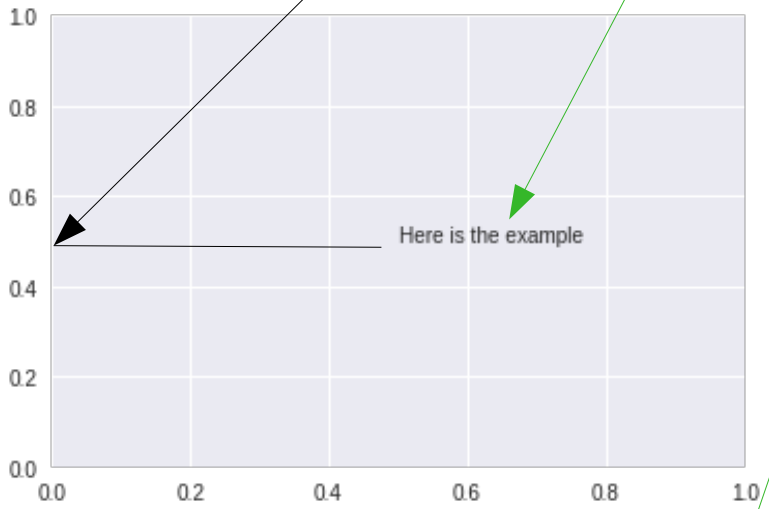


[By Amina Delali]

1- matplotlib

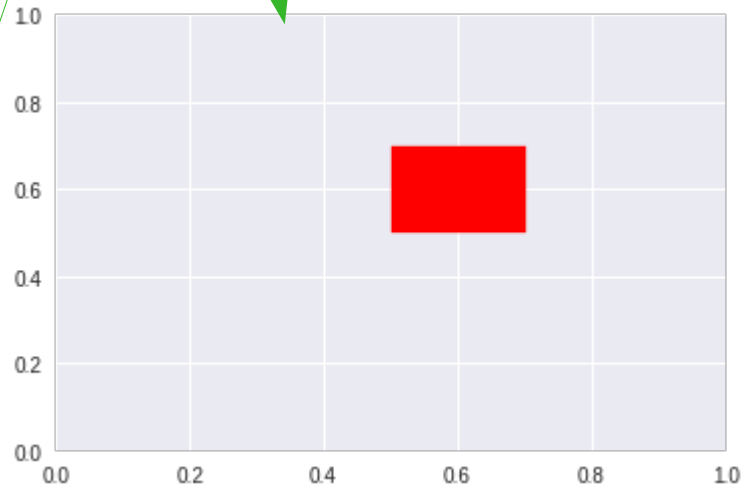
Annotations

```
# adding a text in a plot at a certain position
plt.text(0.5,0.5,"Here is the example")
```



Creating the rectangle

```
# adding a figure in  a plot at a certain position
rec= plt.Rectangle((0.5, 0.5), 0.2, 0.2, color='r')
axis = plt.figure().add_subplot(1,1,1)
axis.add_patch(rec)
```



Adding the rectangle to the plot

[By Amina Delali]

1- matplotlib

Annotations

```
1
2  axis = plt.figure().add_subplot(1,1,1)
3  # add a text anotations+ arrow :
4  # the arrow will be added if th xytext position
5  # is different from xy position
6  axis.annotate("My red annotation", xy=(2,5),
7  xytext=(2,5-2),
8  arrowprops=dict(facecolor='red', width=3),horizontalalignment='right')
9  # add an other text anotations+ arrow :
10 # the y position of the text is higher than xy annotation
11 # so by default the arrow will be drawn under the text
12 axis.annotate("My green annotation", xy=(4,7),
13 xytext=(4,7+2),
14 arrowprops=dict(facecolor='green', width=3),horizontalalignment="left")
15
16 axis.plot(ser1)
```

The arrow is drawn at the **left** of the text

The **y** position of the **text (3)** is under the **y** position of the **annotation (5)** so by defalut the arrow is drawn at the botom of the plot

[By Amina Delali]

File handling and configuration

```
# to save a plot to a file just use
# savefig

plt.plot(ser1)
plt.savefig("myFigure.png")
```

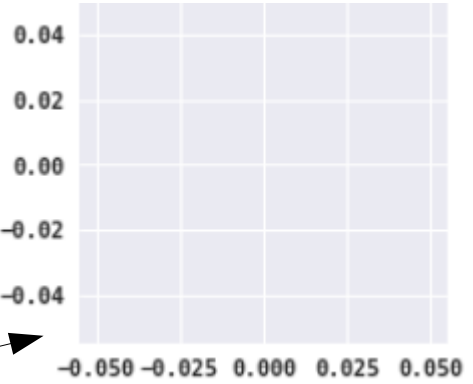
```
# list the content of the current
# directory to see if the file is
# created
%ls
```

```
myFigure.png  sample_data/
```

You can specify an **svg** file as well

- You can **customize** the **default** options of matplotlib plots, you can use the **rc** method

```
# set the default size of a figure
#  to 3 x 3
plt.rc('figure', figsize=(3, 3))
# a dict of font options
font_options = {'family' : "Courier New, monospace",
'weight' : "bold",'size':2.5}
# customizing the default font
plt.rc('font', **font_options)
plt.plot()
```

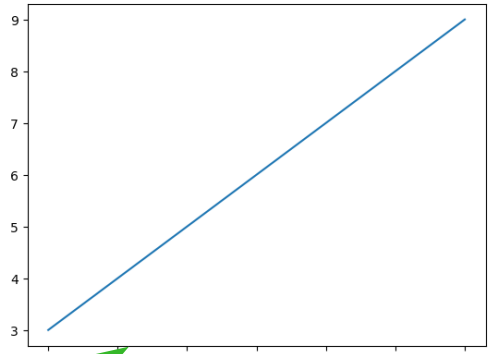


Real size

Line plots with pandas

ser1

```
a    3
b    4
c    5
d    6
e    7
f    8
g    9
dtype: int64
```

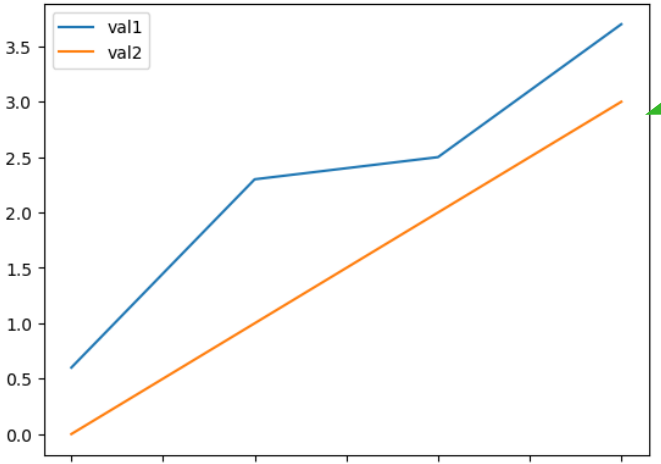



```
1 ser1.plot()
```

df1

	val1	val2	val3
1990	0.6	0	20
1991	2.3	1	21
1992	2.5	2	22
1993	3.7	3	23

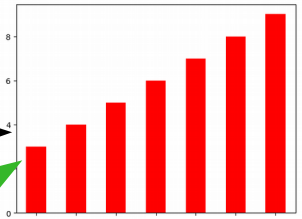
Plotting only these **2** columns



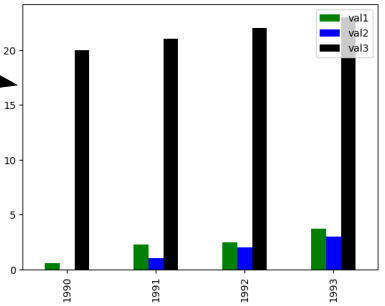

```
df1.loc[:,["val1","val2"]].plot()
```

Bar plots with pandas

```
1 # bar plot with the same series ser1 and dataframe df1
2 ser1.plot.bar(color="r")
3 df1.plot.bar(color=["g","b","k"])
```

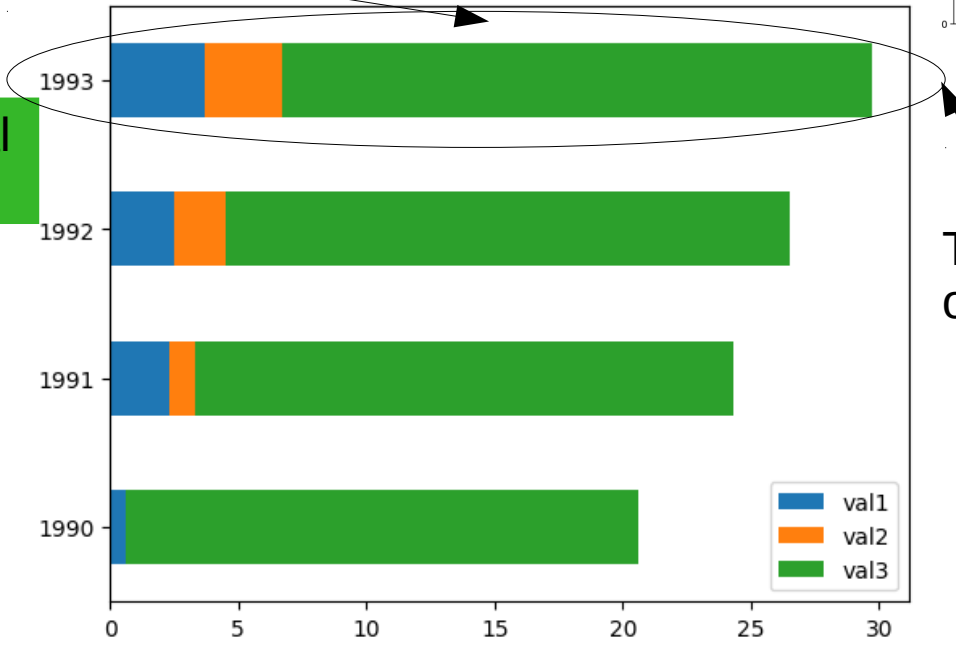


To **each value** correspond a **bar**. In df1 plot, the bars are grouped by **column**




```
1 # plotting staked and horizontal bar plot
2 # for df1
3 df1.plot.barh(stacked=True)
```

For horizontal bars



The **3 v**alues in one bar

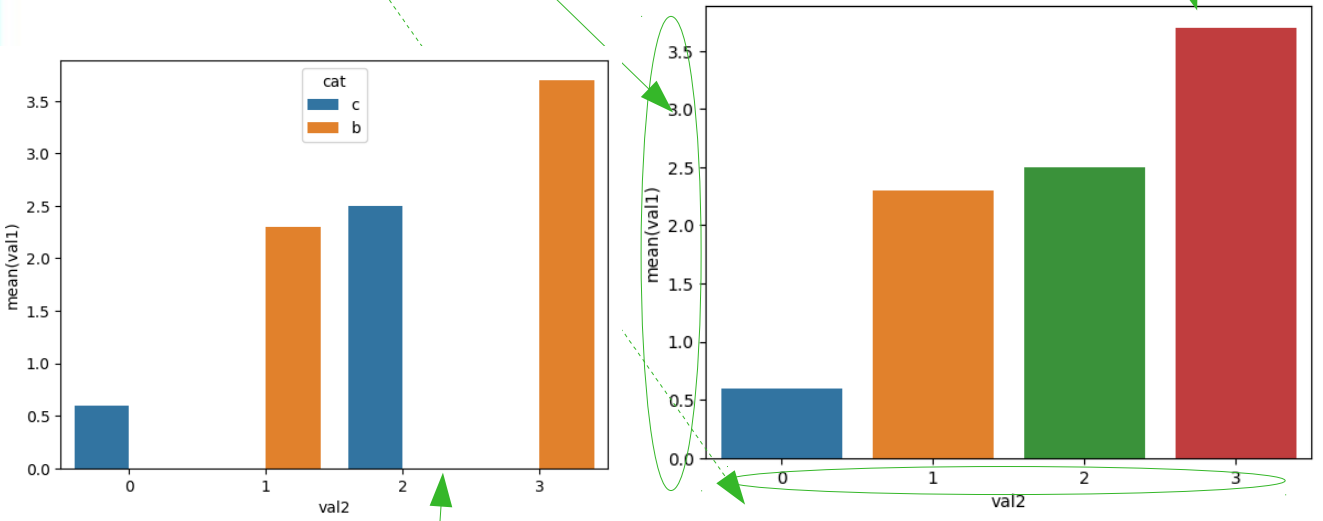
2- Plotting with pandas and seaborn

Bar plots with seaborn

```
# importing seaborn
import seaborn as sns
```

Importing seaborn library

```
sns.barplot(x="val2",y="val1",data=df1,orientation="vertical")
```




```
sns.barplot(x="val2",y="val1",hue="cat",data=df1,orientation="vertical")
```

An other column **cat** added to **df1** used to separate data in categories. Since there is no val2 duplicates, so for each val2 there is only one bar

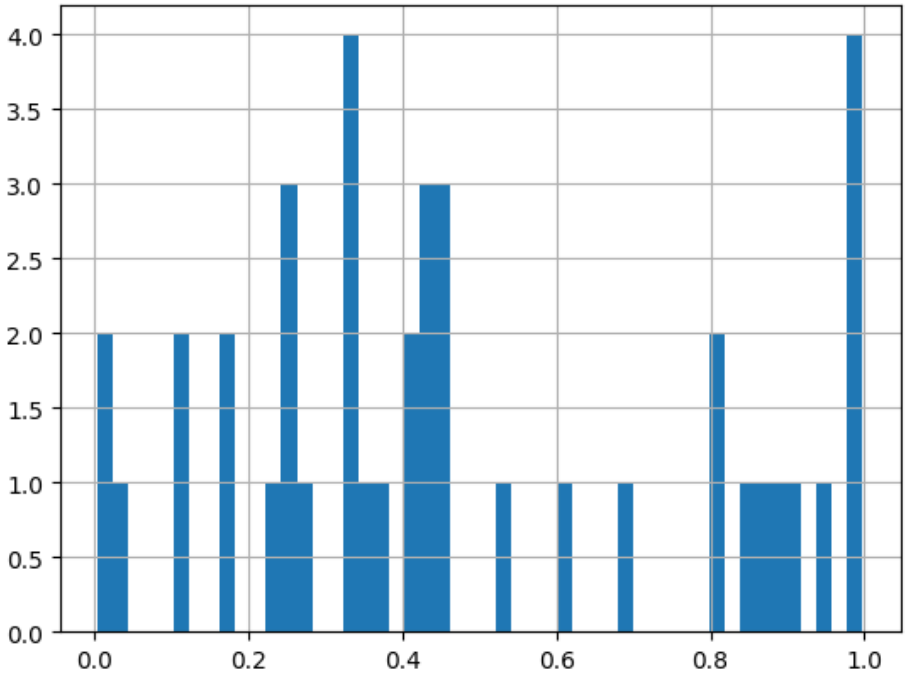
[By Amina Delali]

2- Plotting with pandas and seaborn

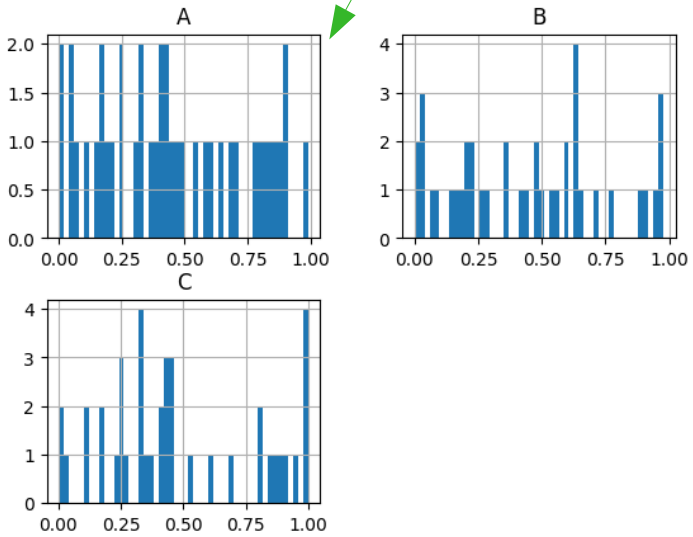
Histograms

```
# creating a dataframe with random values
df3 = DF(np.random.rand(40,3),columns=list("ABC"))
```

```
# plotting the histogram corresponding to the number
# of values of column "C" in each interval (bin) from 50 intervals
# having all the same length
df3.loc[:,"C"].hist(bins=50)
```




```
# plotting the histograms of all columns
df3.hist(bins=50)
```



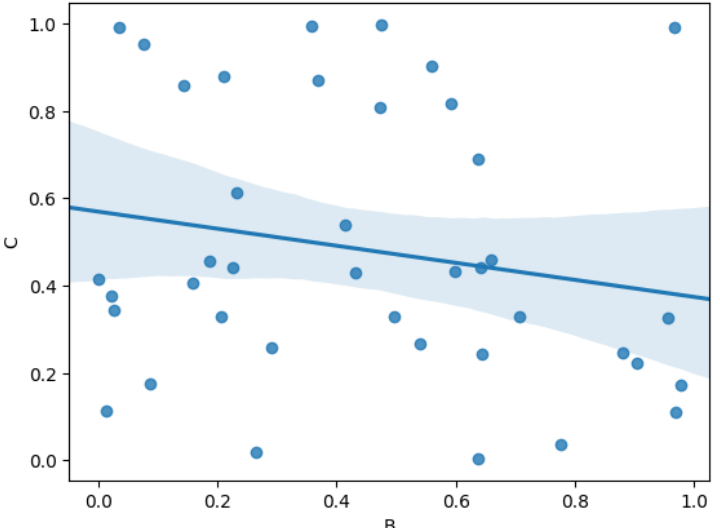
2- Plotting with pandas and seaborn

Points (scatter) plot with seaborn

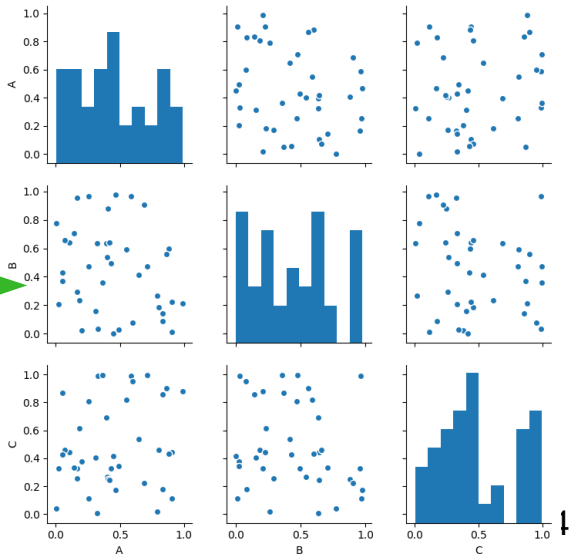
```
sns.regplot("B","C",data=df3)
```

Will plot the data **points** and a **linear regression** model fit

The line that tries to fit to the data




```
# plotting all the columns by pairs
# the diagonal represents the corresponding
# histogram for each column
sns.pairplot(df3)
```



[By Amina Delali]

2- Plotting with pandas and seaborn

Categorical Data

```
1 # ploting the values as bars regarding
2 # two groupings : col3 values,and col4 values
3 sns.factorplot(x="col1",y="col2",hue="col3",col="col4",data=df4,kind="bar")
```

df4

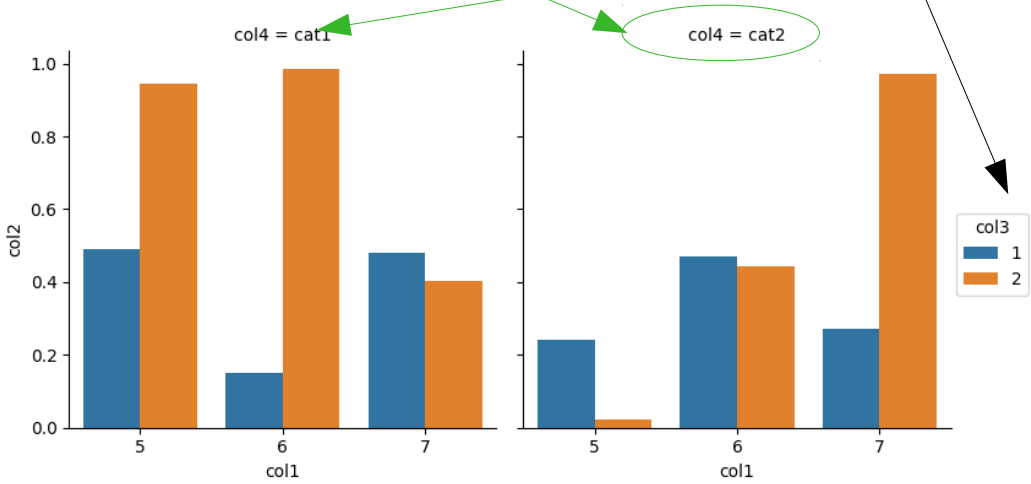
	col1	col2	col3	col4
0	5	0.489210	1	cat1
1	6	0.151948	1	cat1
2	7	0.478811	1	cat1
3	5	0.945796	2	cat1
4	6	0.985511	2	cat1
5	7	0.402356	2	cat1
6	5	0.240515	1	cat2
7	6	0.471802	1	cat2
8	7	0.271782	1	cat2
9	5	0.023132	2	cat2
10	6	0.445108	2	cat2
11	7	0.972034	2	cat2

Category 1, dimension **1** **(hue)**

Category 1, dimension 2 **(col)**

Category 2, dimension **1** **(hue)**

Category 2, dimension 2 **(col)**



[By Amina Delali]

3- Interactive and dynamic graphics

Bokeh

```
# we need to install first bokeh library
!pip install bokeh
```

To install **bokeh** library

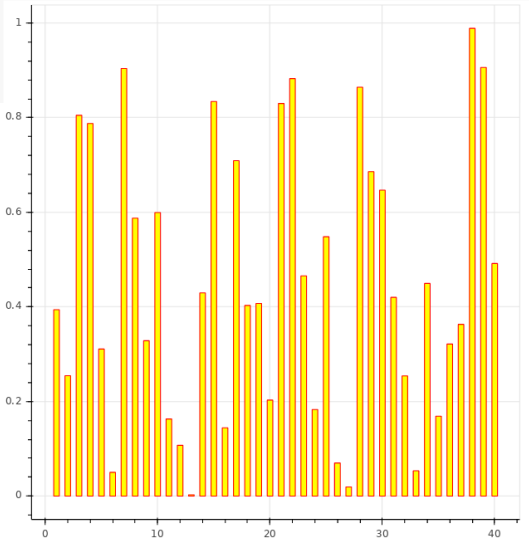
```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
```

```
# the figure contatining the plot
myFig3= figure()
# the plot: bars fir df3 "A" column values
myFig3.vbar(range(1,41),0.5, df3.A,fill_color="#ffff00",line_color="#ff0000")
# necessary instruction to show the plot in this notebook
output_notebook()
# showing the figure
show(myFig3)
```

y values

Hexadecimal colors code

X values



[By Amina Delali]

3- Interactive and dynamic graphics

plotly

```
import plotly.plotly as py
from plotly.offline import iplot
import plotly.graph_objs as go
```

```
# code from  https://colab.research.google.com/notebooks/charts.ipynb#scrollTo=WWbPMtDkO4xg
def enable_plotly_in_cell():
  import IPython
  from plotly.offline import init_notebook_mode
  display(IPython.core.display.HTML('''
        <script src="/static/components/requirejs/require.js"></script>
  '''))
  init_notebook_mode(connected=False)
```

You have to **define** this function and **call** it in each cell containing a plot

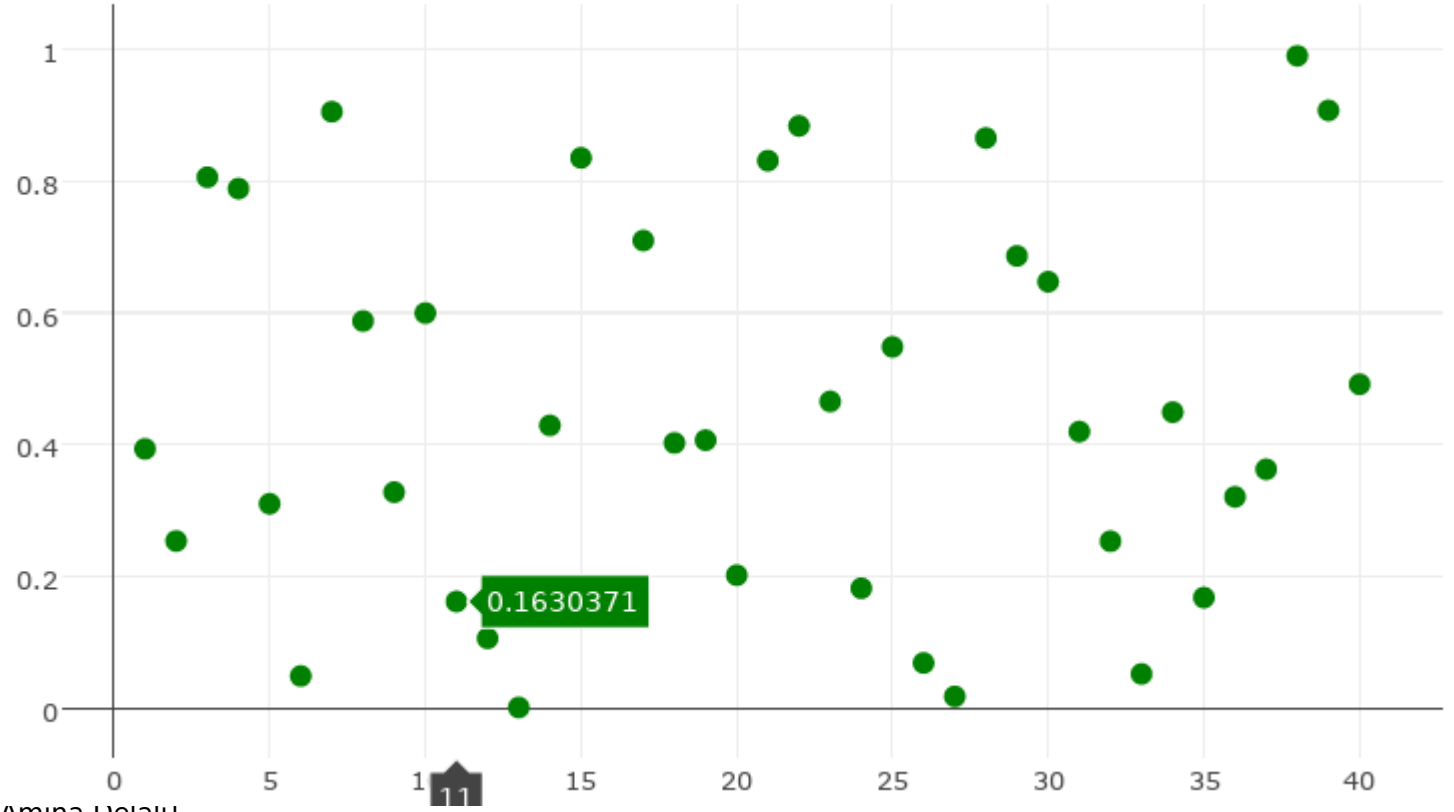
```
# to enable plotting in the notebook
enable_plotly_in_cell()
# the markers used
myMark=Marker(color="green", size=10)
# x values
myX=list(np.arange(1,41))
# y values
myY= df3.A
thePlot=go.Scatter(x=myX,y=myY, mode='markers', marker=myMark)
iplot([thePlot])
```

[By Amina Delali]

3- interactive
And dynamic graphics

Basic Plotting

The generated scatter plot



References

- Wes McKinney. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. O’Reilly Media, Inc, 2018.

Thank you!

FOR ALL YOUR TIME