



Data manipulation: Plotting and Visualization

AAA-Python Edition



school of a

Plan

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



school of ai

1- matplotlib

Basic Plotting

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

Importing pyplot module
from matplotlib (as plt)

```
%matplotlib inline
```

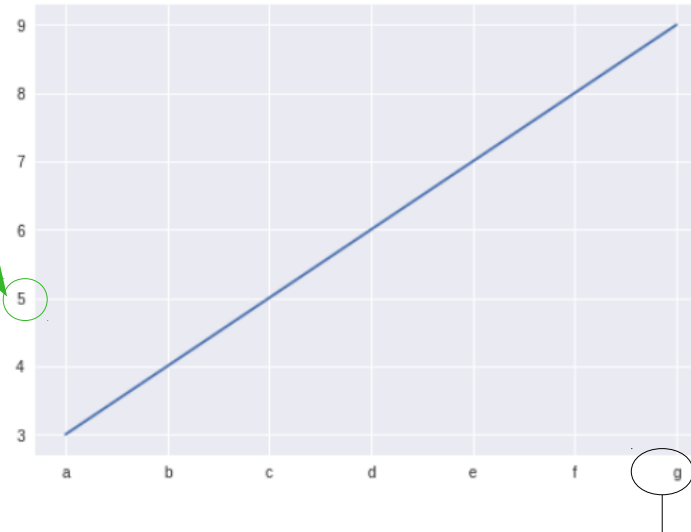
You have to use this command, to
be able to see all the plots
(embed an image after each plot)

ser1

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

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

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



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



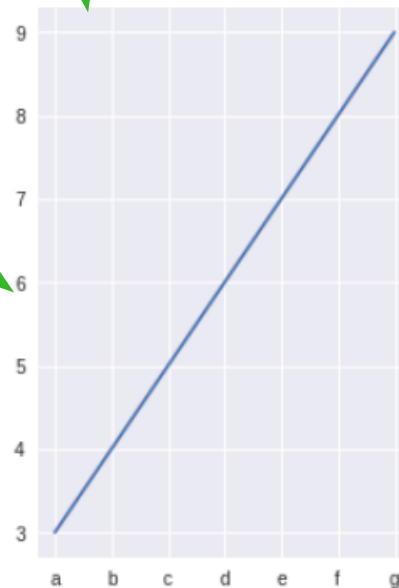
school of ai

1- matplotlib

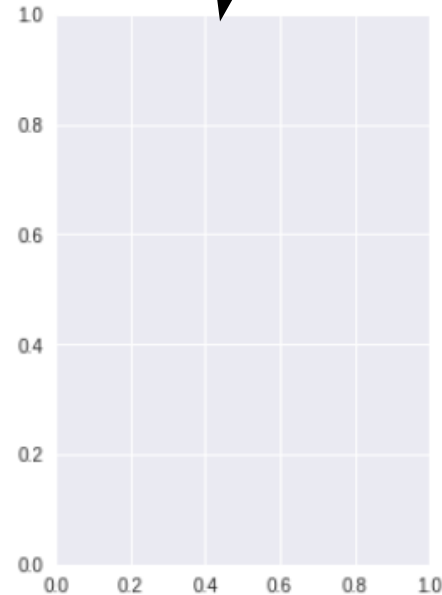
Basic Plotting

```
1 # another 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 # dividing 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)



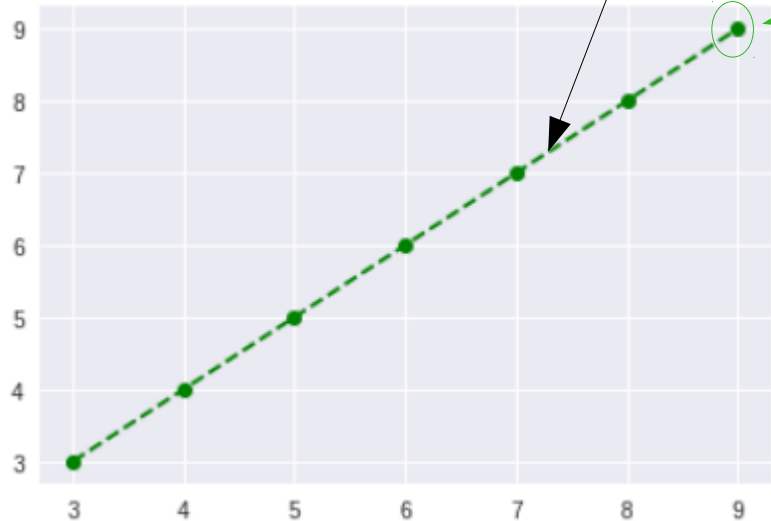


school of ai

1- matplotlib

Customizing plots

```
1  
2 # you can specify : the color, the line_style  
3 # and the marker type of the plot  
4 plt.plot(x, y, 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
'x'	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker

(From google colab help)

Some line styles and
markers



school of a

1- matplotlib

Customizing plots

```
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 their names
6 axis1.set_xticklabels(list("ABCD"))
7 # adding a title to the plot
8 axis1.set_title("Plotting ser1")
9 # plotting axis1 with 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

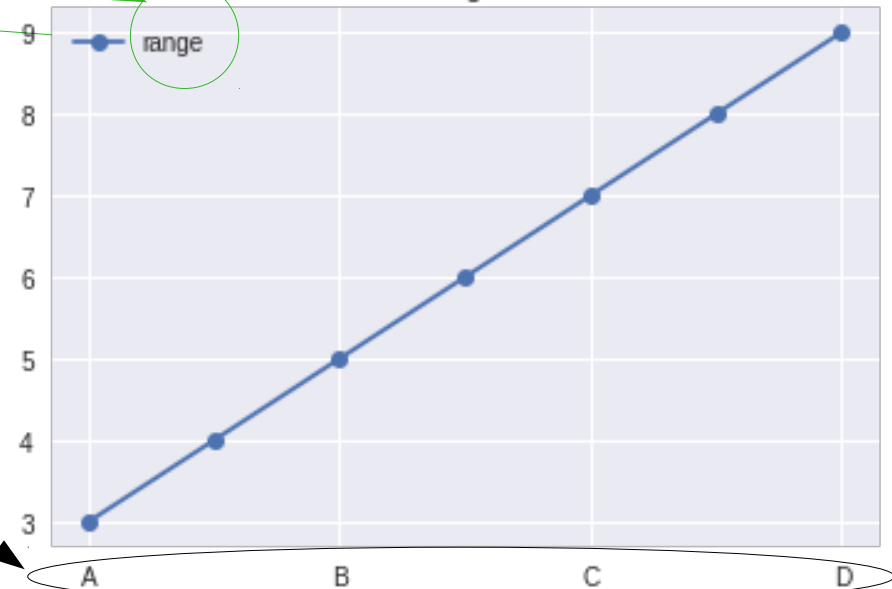
The title

Plotting ser1

ser1

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

The xticklabels: 4
xticks ==> 4
labels



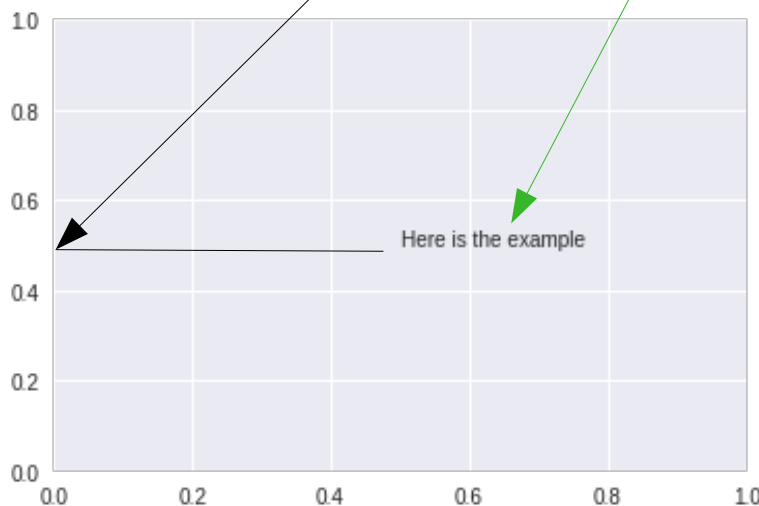


school of a

1- matplotlib

Annotations

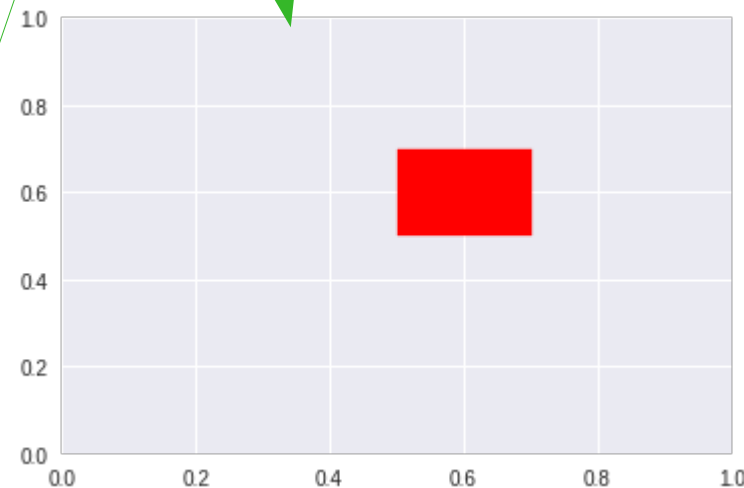
- 1 # adding a text in a plot at a certain position
- 2 plt.text(0.5, 0.5, "Here is the example")



Adding the
rectangle to the
plot

Creating the
rectangle

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



Annotations

```
1
2 axis = plt.figure().add_subplot(1,1,1)
3 # add a text annotations+ arrow :
4 # the arrow will be added if the 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 another text annotations+ 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
default the arrow is
drawn at the bottom
of the plot



[By Amina Delali]



school of a

1- matplotlib



school of ants

1- matplotlib

File handling and configuration

- ```
1 # to save a plot to a file just use
2 # savefig
3
4 plt.plot(ser1)
5 plt.savefig("myFigure.png")
```

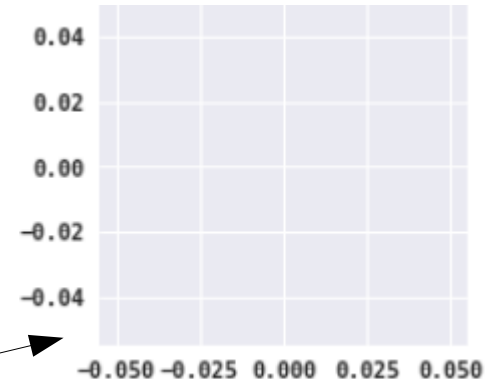
You can specify an  
**svg** file as well

```
1 # list the content of the current
2 # directory to see if the file is
3 # created
4 %ls
```

myFigure.png sample\_data/

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

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



Real size



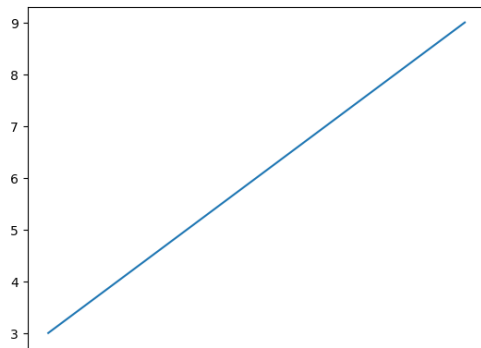
school of ai

## 2- Plotting with pandas and seaborn

### Line plots with pandas

ser1

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

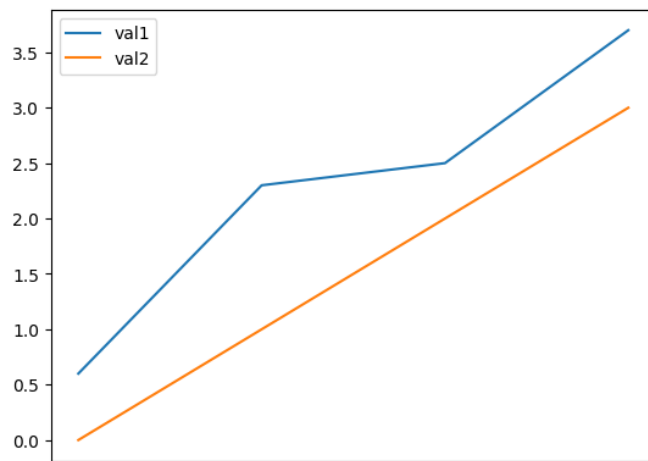


```
1 ser1.plot()
```

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()
```

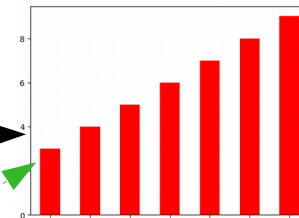


school of a

## 2- Plotting with pandas and seaborn

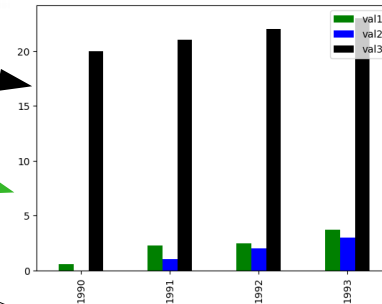
### Bar plots with pandas

```
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"])
```

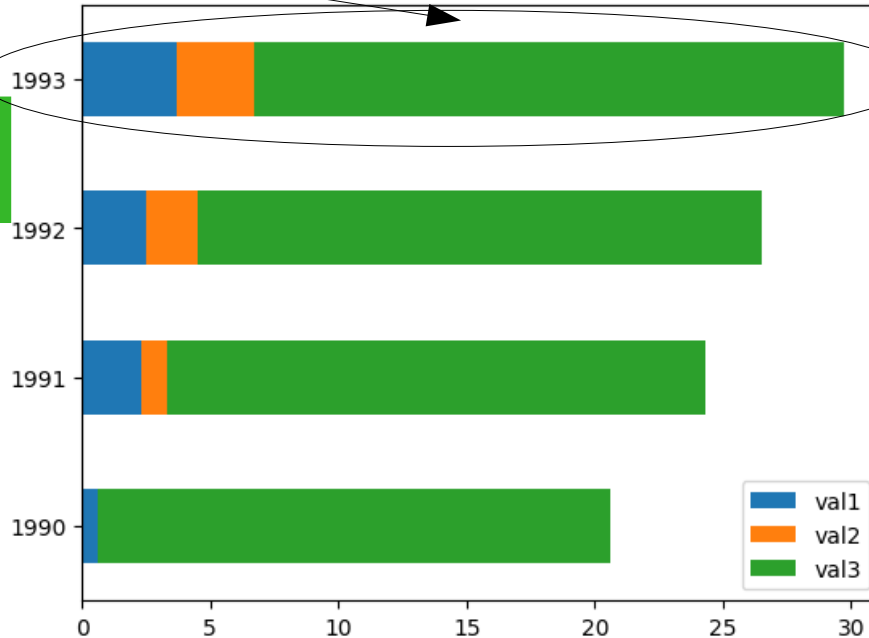


To each value correspond a bar. In df1 plot, the bars are grouped by column

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



For horizontal bars



The 3 values in one bar



school of a

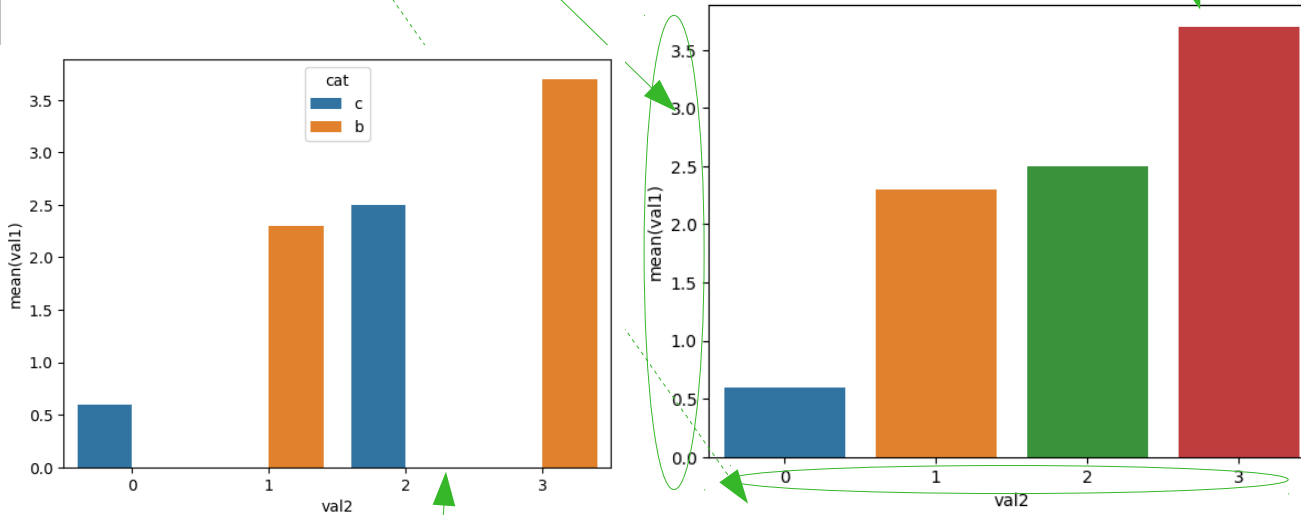
## 2- Plotting with pandas and seaborn

### Bar plots with seaborn

```
1 # importing seaborn
2 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



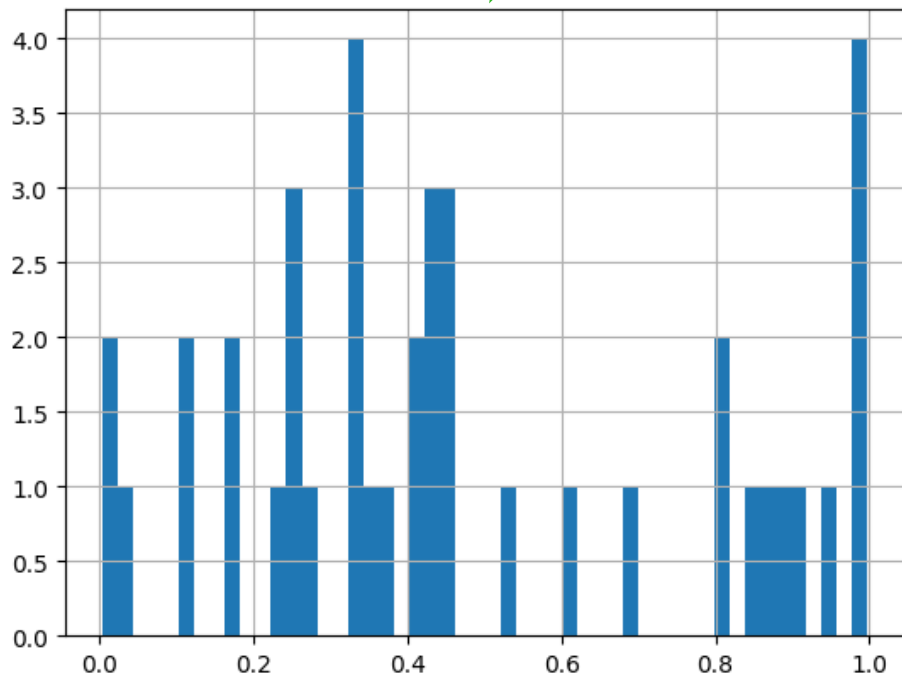
school of ai

## 2- Plotting with pandas and seaborn

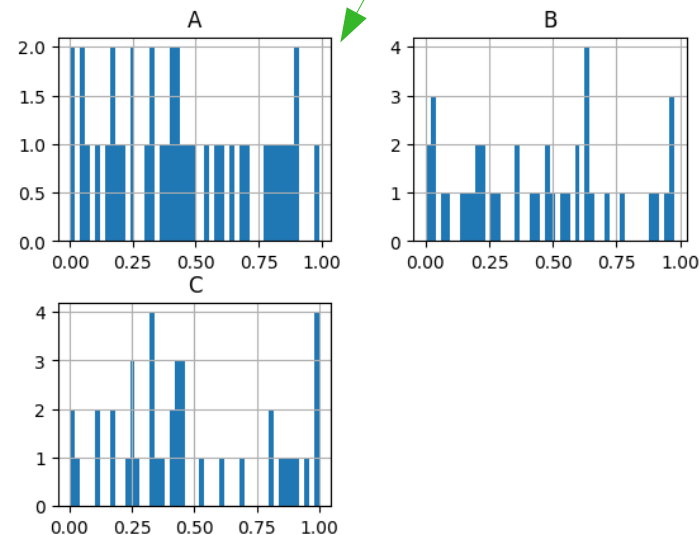
### Histograms

```
1 # creating a dataframe with random values
2 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)
```



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





school of ai

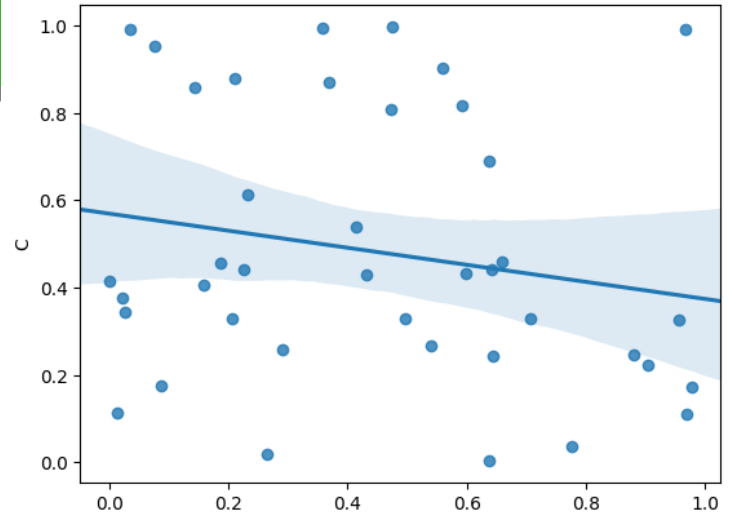
## 2- Plotting with pandas and seaborn

### Points (scatter) plot with seaborn

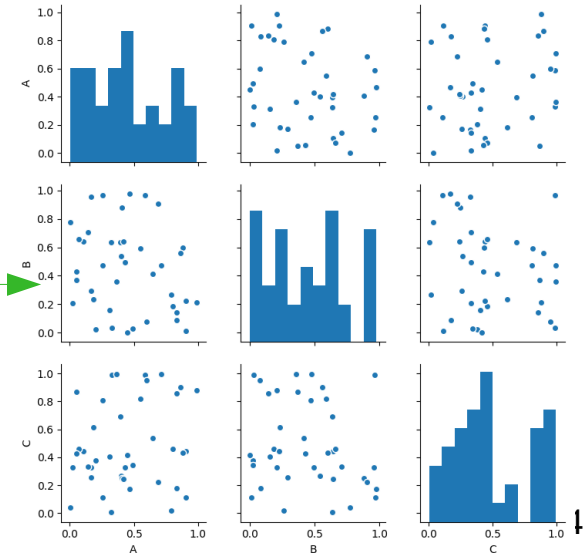
```
sns.regplot("B", "C", data=df3)
```

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

The line that tries to fit to the data



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





school of a

## 2- Plotting with pandas and seaborn

### Categorical Data

df4

```
1 # plotting 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")
```

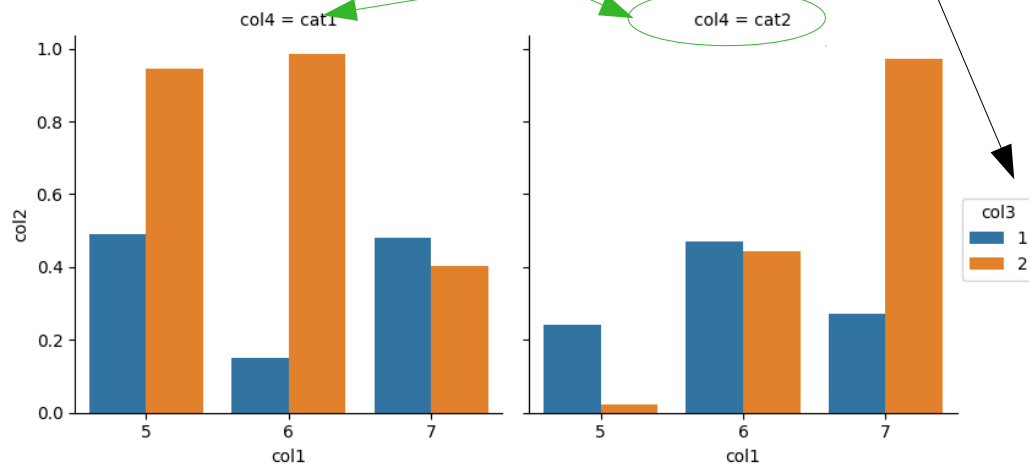
|    | col1 | col2     | col3 | col4 |
|----|------|----------|------|------|
| 0  | 5    | 0.489210 | 1    | cat1 |
| 1  | 6    | 0.151948 | 1    | cat1 |
| 2  | 7    | 0.478811 | 1    | cat1 |
| 3  | 5    | 0.945796 | 2    | cat1 |
| 4  | 6    | 0.985511 | 2    | cat1 |
| 5  | 7    | 0.402356 | 2    | 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]



school of ai

### 3- Interactive and dynamic graphics

## Bokeh

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

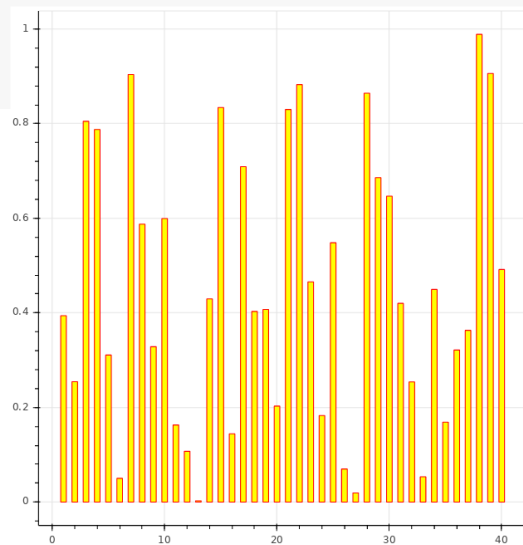
To install **bokeh** library

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

```
1 # the figure containing the plot
2 myFig3= figure()
3 # the plot: bars for df3 "A" column values
4 myFig3.vbar(range(1,41), 0.5, df3.A, fill_color="#ffff00", line_color="#ff0000")
5 # necessary instruction to show the plot in this notebook
6 output_notebook()
7 # showing the figure
8 show(myFig3)
```

y values

X values



Hexadecimal  
colors code





school of ai

### 3- Interactive and dynamic graphics

#### plotly

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

```
1 # code from https://colab.research.google.com/notebooks/charts.ipynb#scrollTo=wWbPMTDk04xg
2 def enable_plotly_in_cell():
3 import IPython
4 from plotly.offline import init_notebook_mode
5 display(IPython.core.display.HTML('''
6 <script src="/static/components/requirejs/require.js"></script>
7 '''))
8 init_notebook_mode(connected=False)
```

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

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

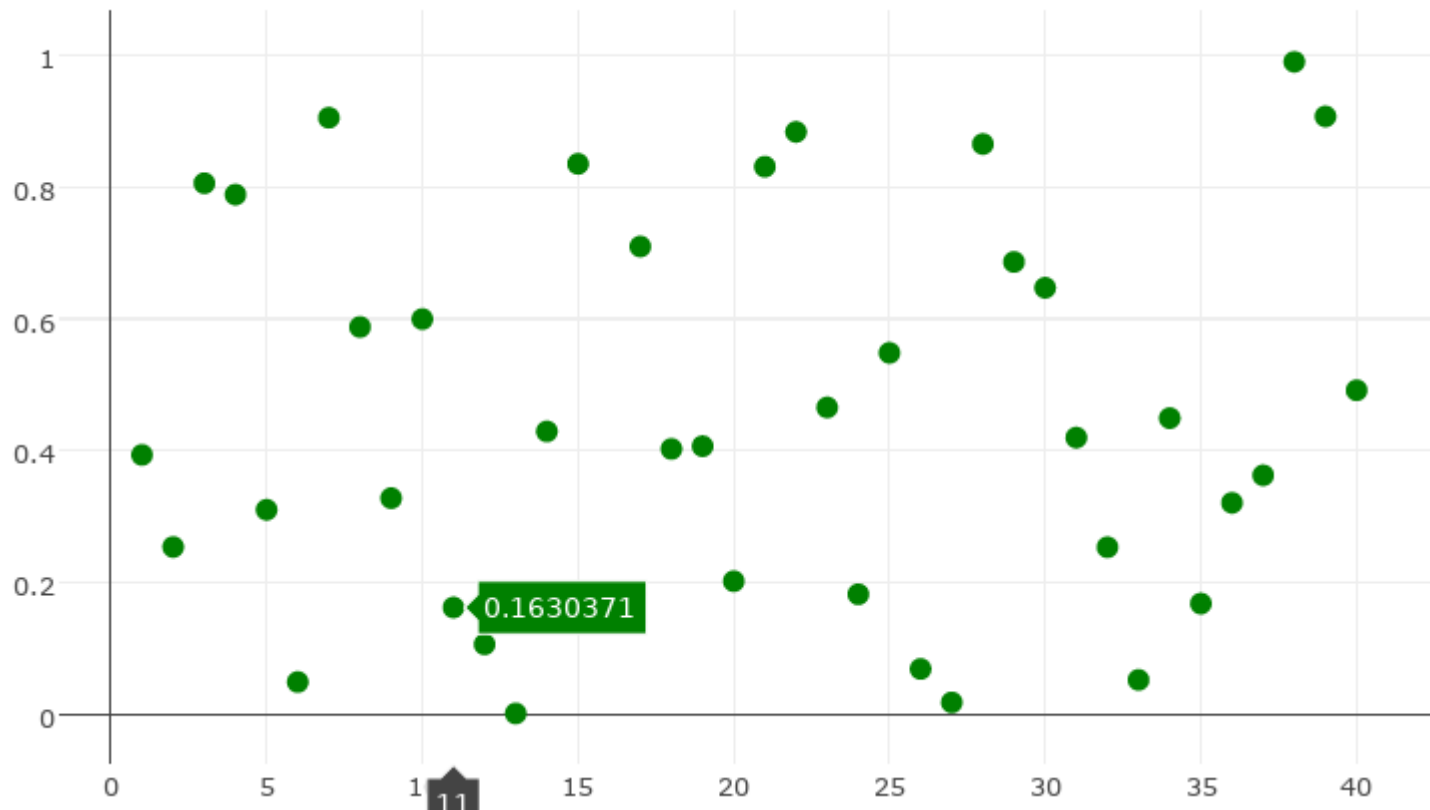


school of a

### 3- Interactive and dynamic graphics

## Basic Plotting

The generated scatter plot



[By Amina Delali]



school of a

# References

- Bokeh. bokeh.plotting. On-line at <https://bokeh.pydata.org/en/latest/docs/reference/plotting.html>. Accessed on 31-10-2018.
- Google Colab. Charts in colaboratory. On-line at <https://colab.research.google.com/notebooks/charts.ipynb#scrollTo=6YVIGlxQ54fH>. Accessed on 31-10-2018.
- Google Colab doc.
- Wes McKinney. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. O'Reilly Media, Inc, 2018.
- Jake VanderPlas. Python data science handbook: essential tools for working with data. O'Reilly Media, Inc, 2017.



# Thank you!

FOR ALL YOUR TIME