ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Python for IoT Data Analytics

Data Visualization with Matplotlib

EIT4SEMM PHD Program, July 2023

Prof. Marco Di Felice

Department of Computer Science and Engineering, University of Bologna

marco.difelice3@unibo.it

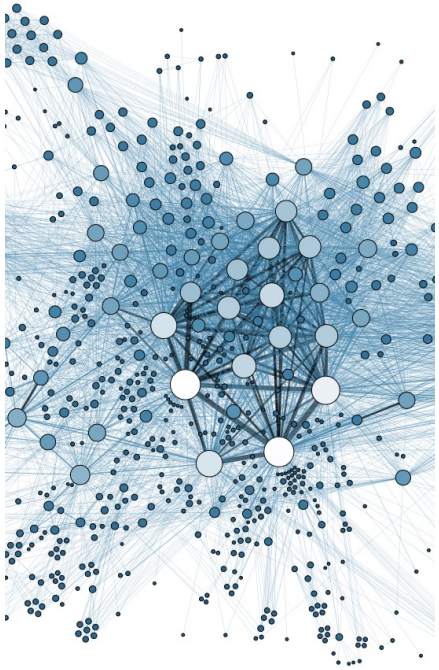
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

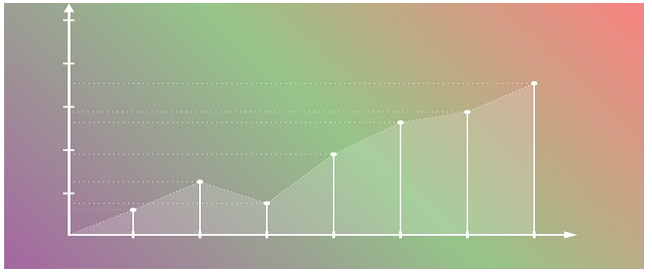
Data Visualization

- **Why data visualization**
- **Basic concepts and tools**
  - Some definitions: marks, channels, separability
  - Dealing with colors
  - Univariate attribute
  - Multivariate attributes
- **Plotting data with Python**
  - Matplotlib library
  - Seaborn library

Data Visualization

- **Data visualization** aims to communicate data clearly and effectively through geometrical representation
  - Effective tool to explore data
  - Help discovering data relationships otherwise not easily observable by looking at row data
  - Can be used to represent the data mining process and patterns/evaluation results obtained from a mining method
- Many different techniques for different kind of data





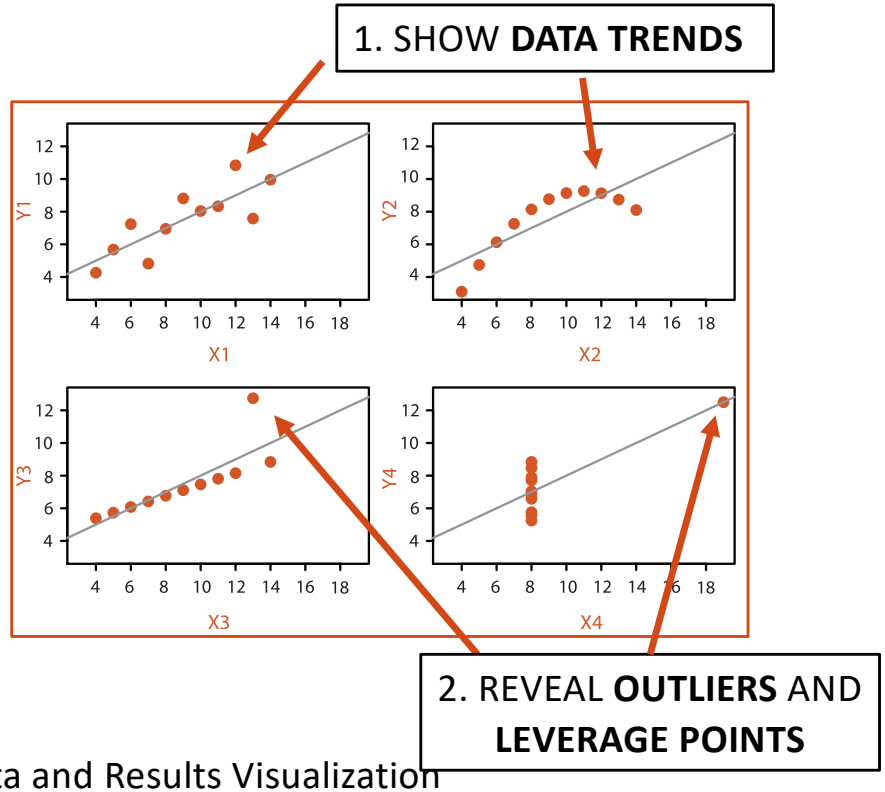
Data Visualization

	1		2		3		4	
	X	Y	X	Y	X	Y	X	Y
	10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
	8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
	13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
	9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
	11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
	14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
	6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
	4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
	12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
	7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
	5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89
Mean	9.0	7.5	9.0	7.5	9.0	7.5	9.0	7.5
Variance	10.0	3.75	10.0	3.75	10.0	3.75	10.0	3.75
Correlation	0.816		0.816		0.816		0.816	

Credits to: Daniele Loiacono, Data and Results Visualization

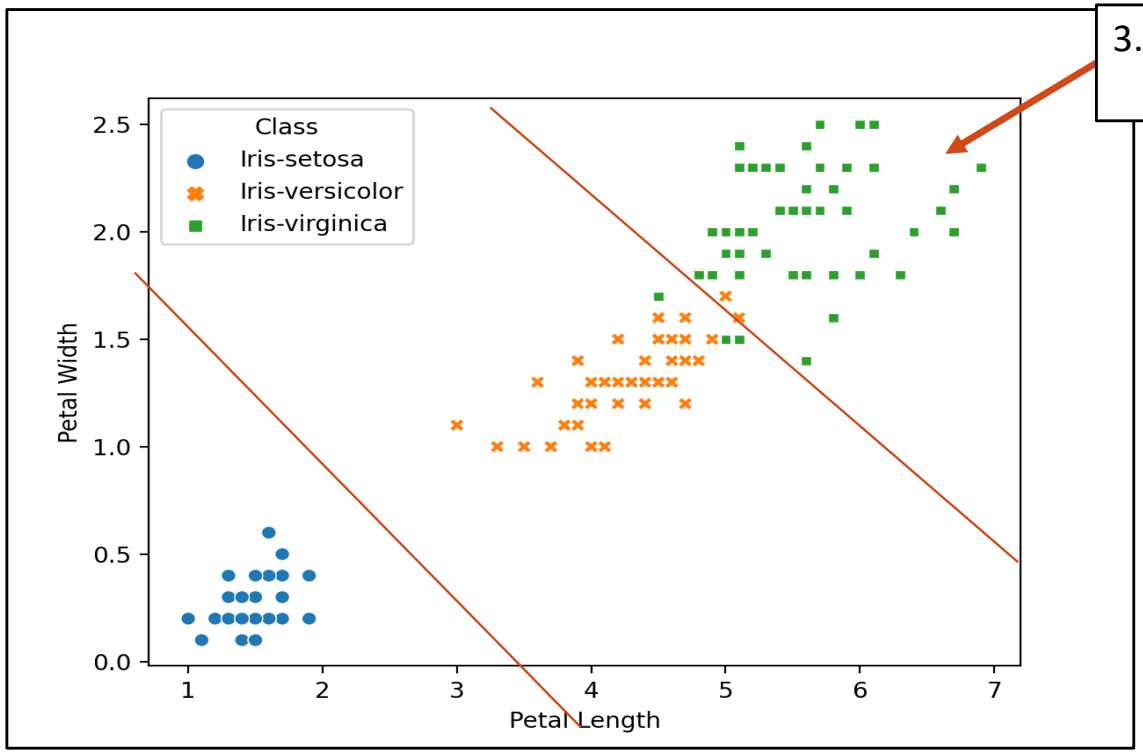
Data Visualization

	1		2		3		4	
	X	Y	X	Y	X	Y	X	Y
	10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
	8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
	13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
	9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
	11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
	14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
	6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
	4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
	12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
	7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
	5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89
Mean	9.0	7.5	9.0	7.5	9.0	7.5	9.0	7.5
Variance	10.0	3.75	10.0	3.75	10.0	3.75	10.0	3.75
Correlation	0.816		0.816		0.816		0.816	



Credits to: Daniele Loiacono, Data and Results Visualization

Data Visualization



3. PROVIDE HINTS ABOUT **MODELING TECHNIQUES TO APPLY**

Which conclusions can be derived from this chart?

Data Visualization

- **Why data visualization**
- **Basic concepts and tools**
  - *Some definitions: marks, channels, separability*
  - Dealing with colors
  - Univariate attribute
  - Multivariate attributes
- **Plotting data with Python**
  - Matplotlib library
  - Seaborn library

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Visualization: concepts

Attribute Types

Nominal/Categorical

hair_color	black, brown, blond
Occupation	teacher, dentist, programmer

Ordinal

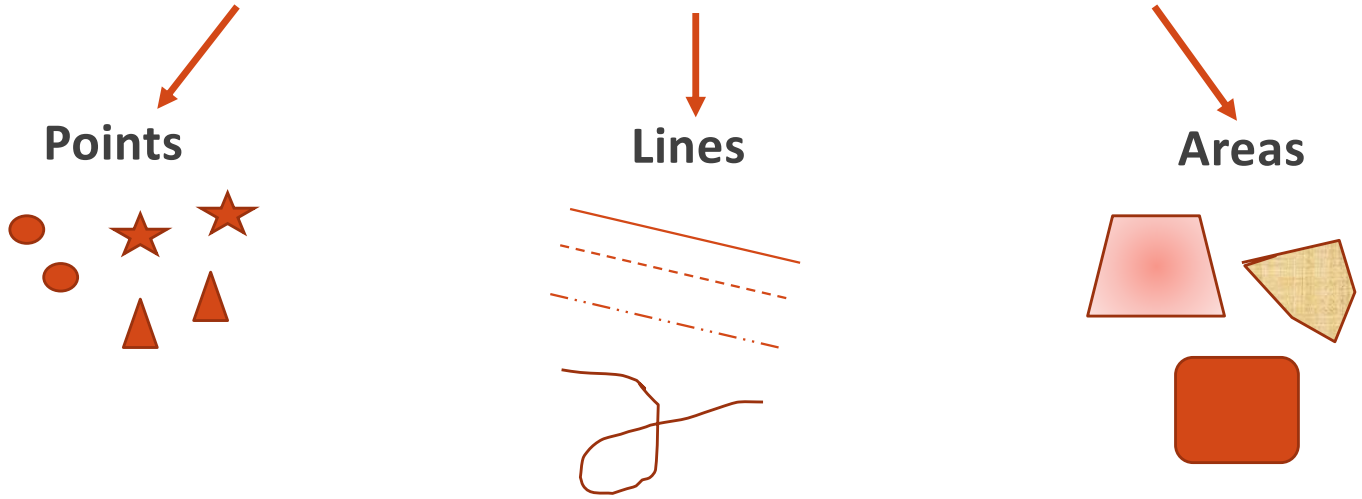
drink_size	Small, Medium, Large
professors rank	assistant, associate, and full

Numeric

e.g.. weight, height, latitude and longitude

Data Visualization

- Building blocks of visual encoding: **Marks** and **Channels**
- **Marks:** geometric elements used to depict the data samples

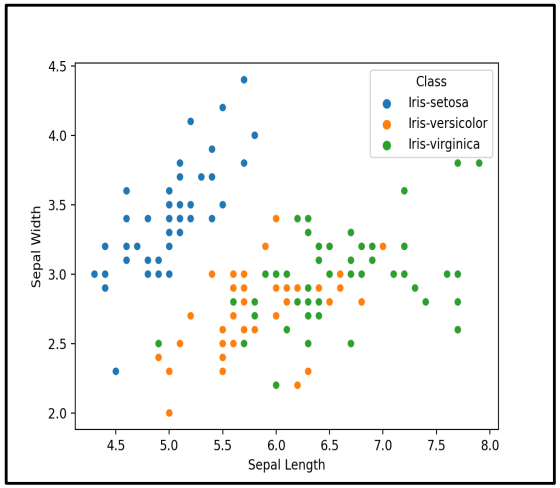


ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Data Visualization

- Building blocks of visual encoding: **Marks** and **Channels**

- **Channels:** control the appearance of a mark → Encode the attributes
  - Position
  - Hue, saturation, lightness
  - Shape
  - Tilt
  - Size
  - Orientation
  - etc



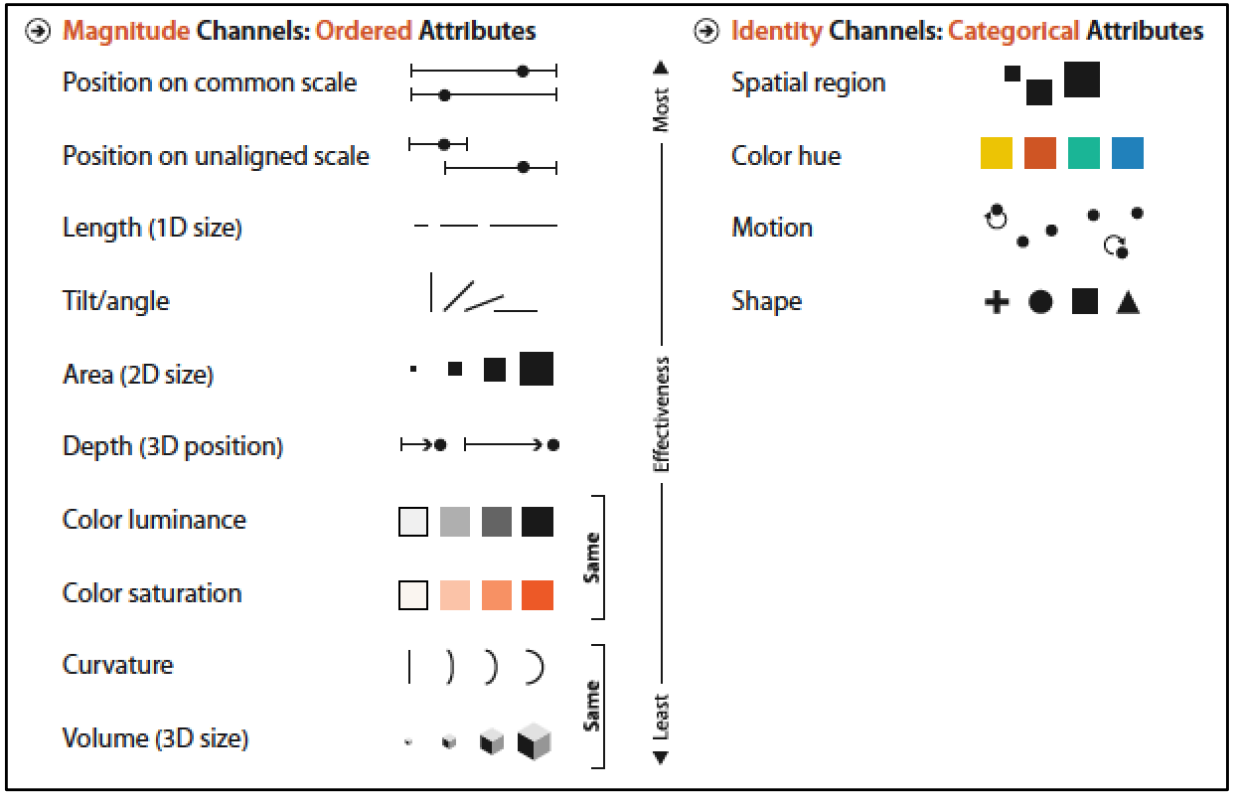
IRIS DATASET VISUALIZATION

- **Marks**
  Points
- **Channels**
  Position(X) → Sepal Length
  Position (Y) → Sepal Width
  Class → Color

Data Visualization

Channel types:

- **Magnitude channels**: enconde Ordinal/Numeric attributes
- **Identify channels**: encode Categorical attributes



Source: Munzner, "Visualization Analysis and Design"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Data Visualization

- **Why data visualization**
- **Basic concepts and tools**
  - Some definitions: marks, channels, separability
  - *Dealing with colors*
  - Univariate attribute
  - Multivariate attributes
- **Plotting data with Python**
  - Matplotlib library
  - Seaborn library

Data Visualization

- Our visual system can process signals into three opponent color channels
  - Red – Green
  - Blue – Yellow

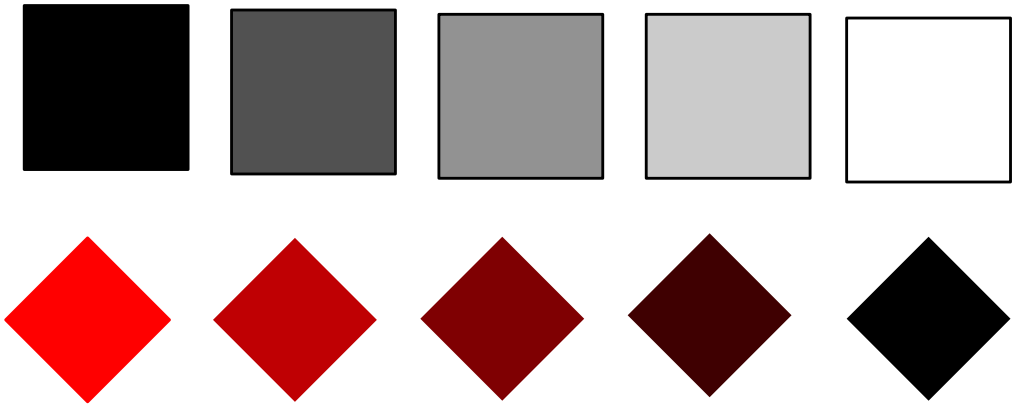
  } Chromaticity → Color
  - Black – White → Luminance → Edge Detection

- Three visual channels to encode data with colors:
  - **Luminance**: how bright is the marker
  - **Saturation**: how colorful is the marker
  - **Hue**: which color is used by the marker

Credits to: Daniele Loiacono, Data and Results Visualization

Data Visualization

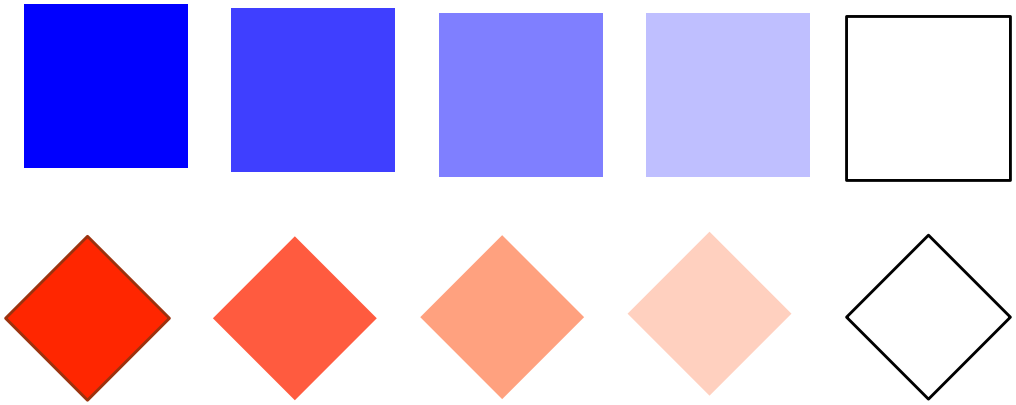
- **Luminance**
  - Magnitude channel, suitable to encode **ordinal** attributes
  - It allows to discriminate up to 5 different (non-contiguous) values



Credits to: Daniele Loiacono, Data and Results Visualization

Data Visualization

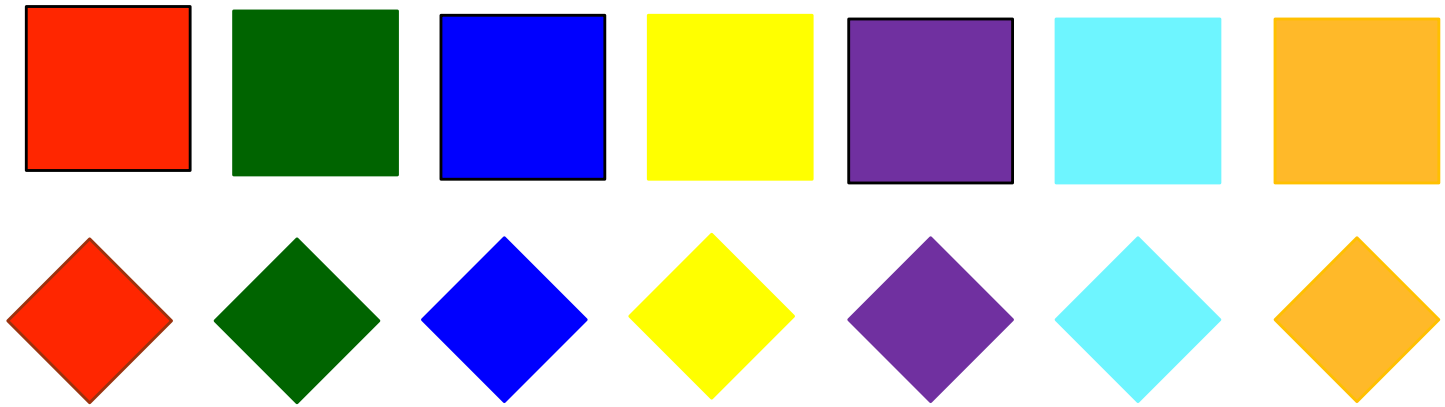
- **Saturation**
  - Magnitude channel, suitable to encode **ordinal** attributes
  - It allows to discriminate up to 3 different (non-contiguous) values (hence it is less effective than the Luminance channel).



Credits to: Daniele Loiacono, Data and Results Visualization

Data Visualization

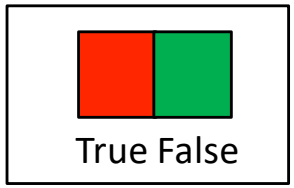
- **Hue**
  - Identity channel, suitable to encode **categorical** attributes
  - It allows to discriminate up to 7 different (non-contiguous) values



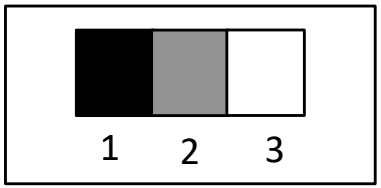
Credits to: Daniele Loiacono, Data and Results Visualization

Data Visualization

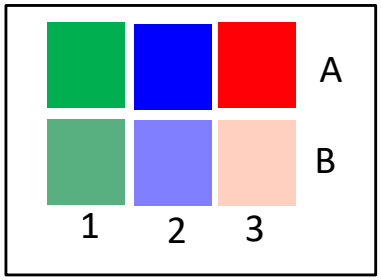
- **Colormap**
  - Mapping chosen to encode one (or two) attributes with color
  - Apply to both categorical and ordinal attributes



Univariate, categorical
(data encoding by Hue)



Univariate, ordinal
(data encoding by Luminance)



Multivariate, ordinal + categorical
(data encoding by Saturation + Hue)

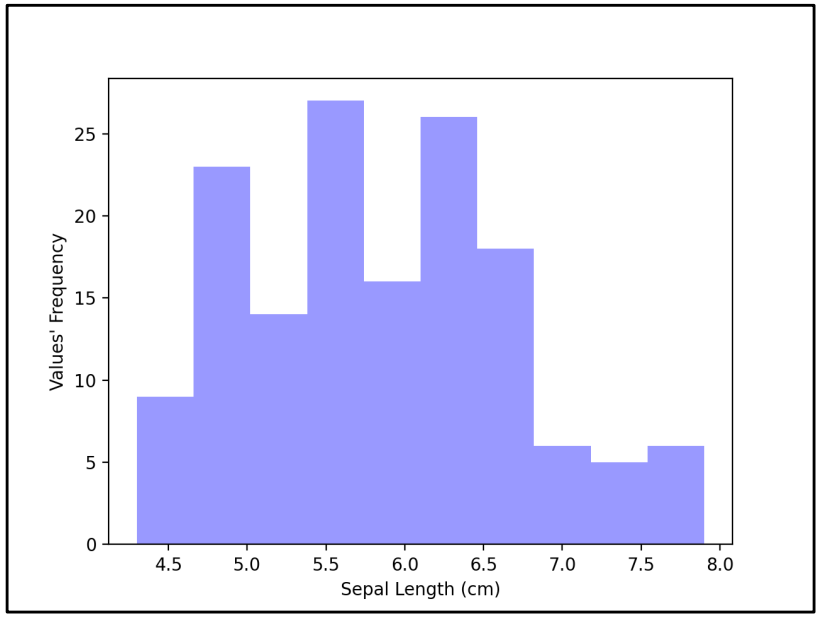
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Data Visualization

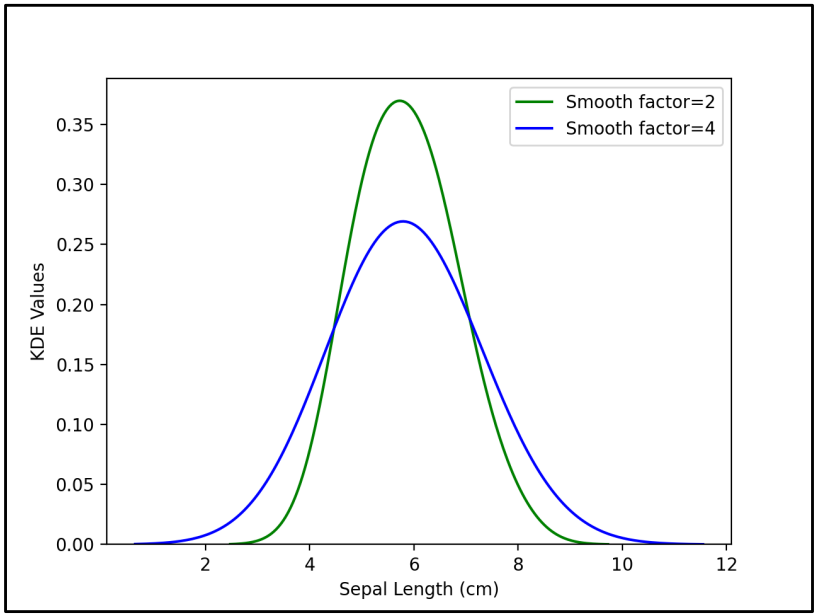
- **Why data visualization**
- **Basic concepts and tools**
  - Some definitions: marks, channels, separability
  - Dealing with colors
  - *Univariate attribute*
  - Multivariate attributes
- **Plotting data with Python**
  - Matplotlib library
  - Seaborn library

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Visualization: tools



HISTOGRAM



DENSITY PLOT

Visualization: tools

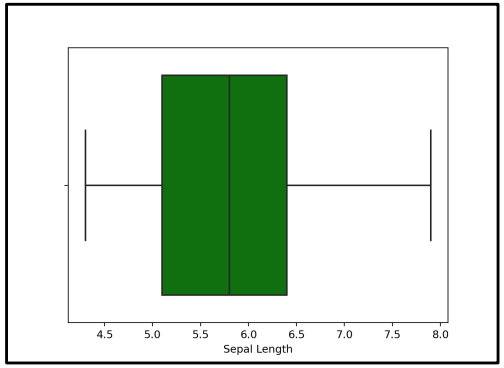
- **Boxplot**
  - Apply to

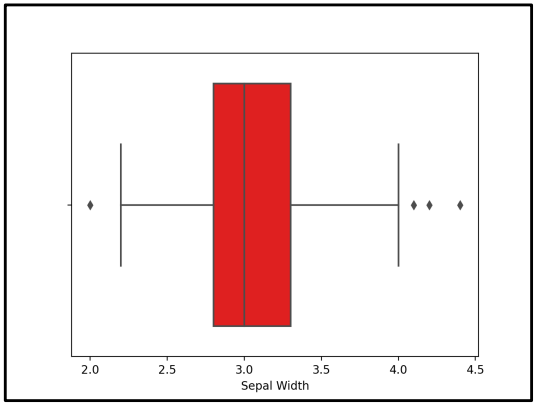
    1 Quantitative Attribute
  - What

    Statistical summary of a data distribution, it shows 5 variables within a single plot
  - How

    Median, first and third quartiles (top/bottom whishers), maximum and minimum values
  - Why

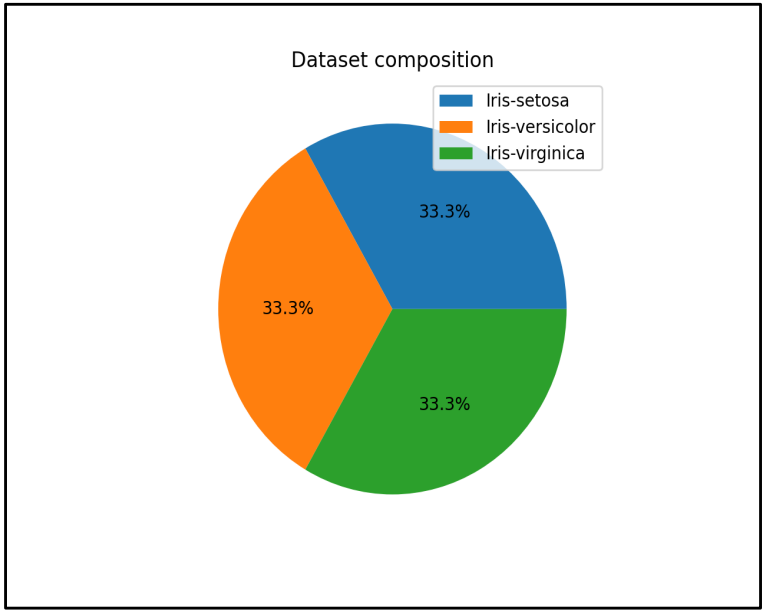
    Identify distribution and range of an attribute, have a quick look to its descriptive analysis



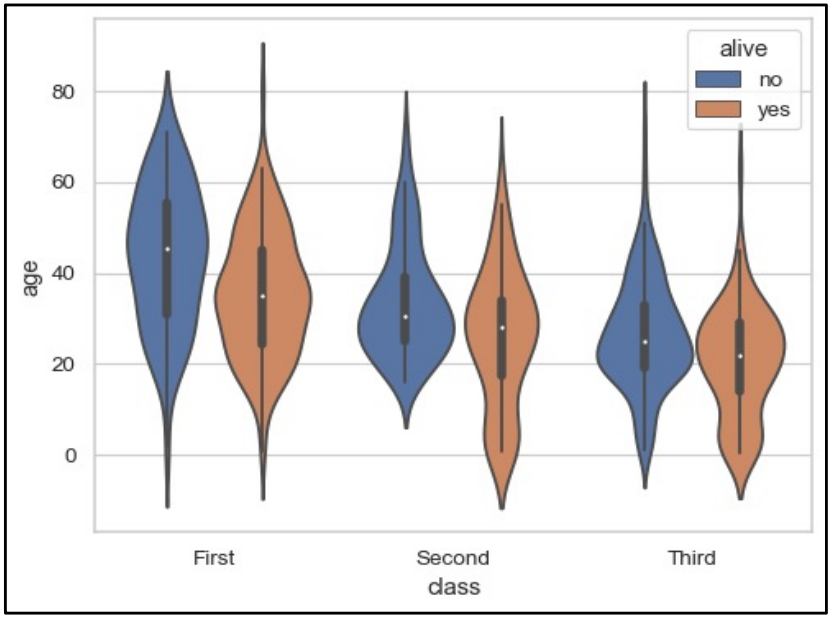


ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Visualization: tools



PIE CHART



VIOLIN CHART

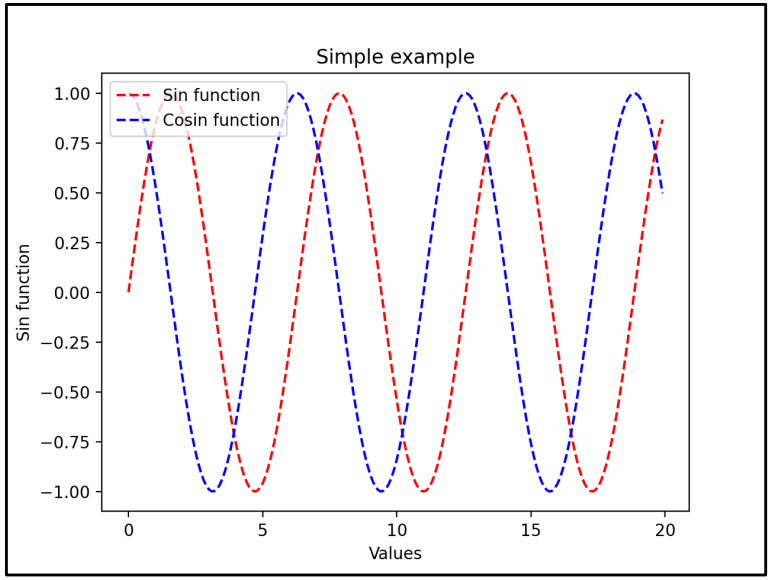
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Data Visualization

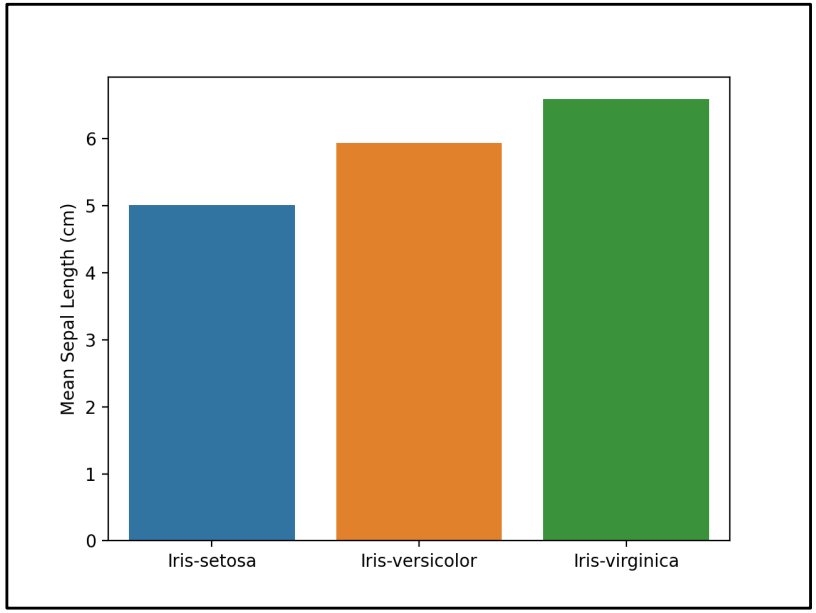
- **Why data visualization**
- **Basic concepts and tools**
  - Some definitions: marks, channels, separability
  - Dealing with colors
  - Univariate attribute
  - *Multivariate attributes*
- **Plotting data with Python**
  - Matplotlib library
  - Seaborn library

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Visualization: tools



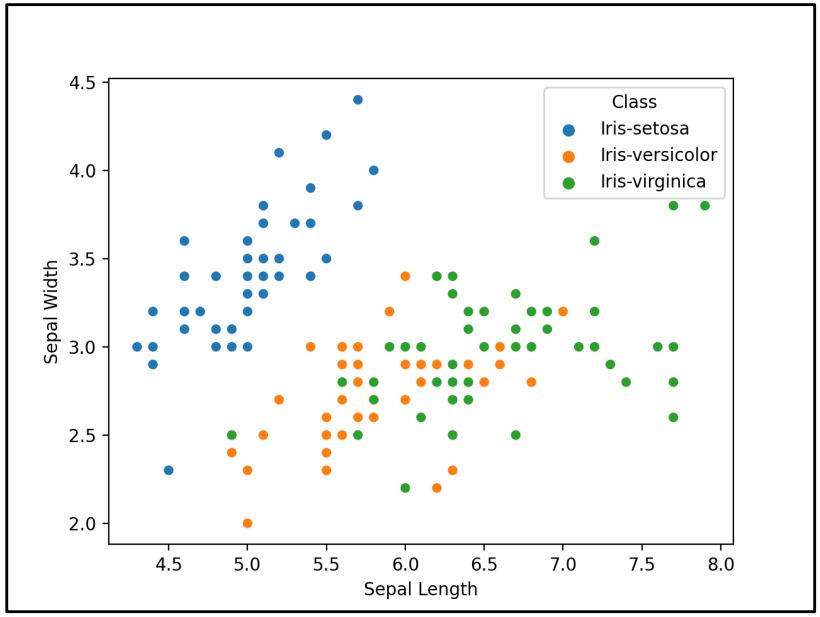
LINE PLOT



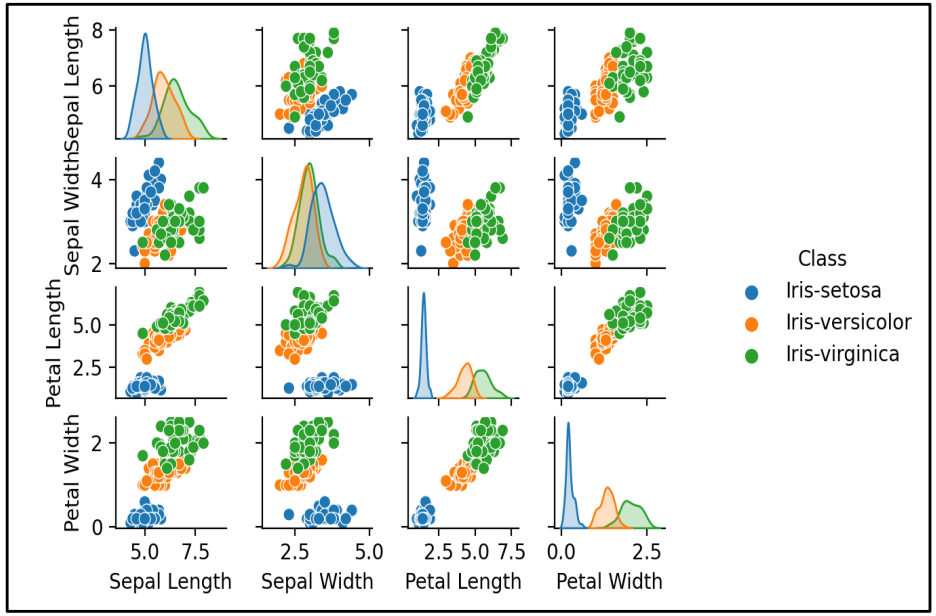
BAR CHART

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Visualization: tools



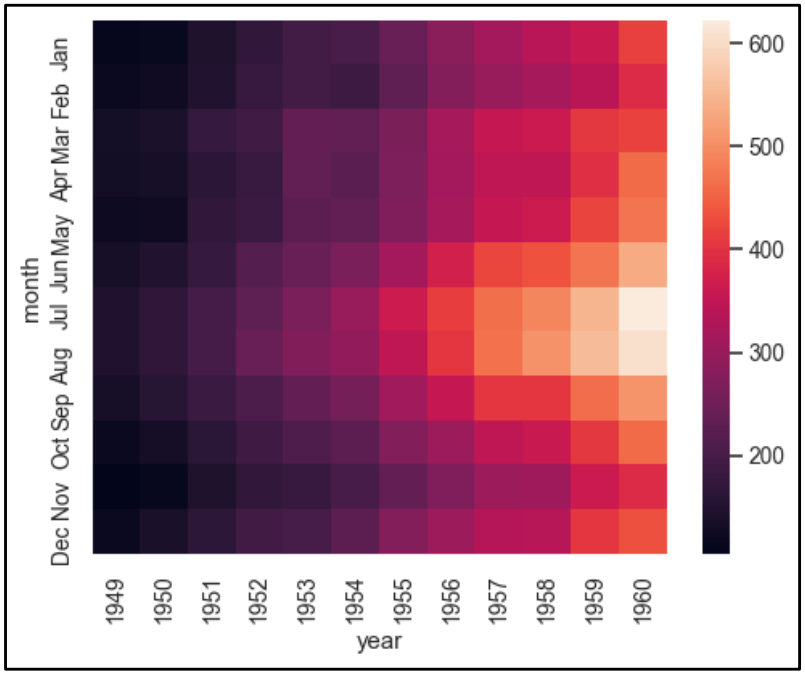
SCATTER PLOT



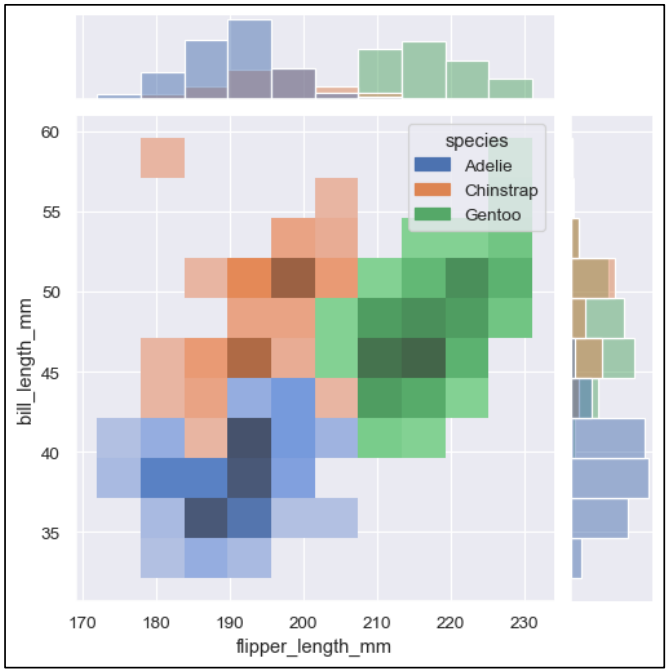
SCATTER PLOT MATRIX

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Visualization: tools



HEATMAP



HEATMAP+DISTPLOT

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Data Visualization

- **Why data visualization**
- **Basic concepts and tools**
  - Some definitions: marks, channels, separability
  - Dealing with colors
  - Univariate attribute
  - Multivariate attributes
- **Plotting data with Python**
  - Matplotlib library
  - Seaborn library

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Python 2D/3D plotting library**
  - Initially developed by J. Hunter in 2003
  - Object-oriented API (`pyplot`)
- **MATLAB-like interface**
  - plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc.
- **Official (introductory, intermediate, advanced) tutorials:**
  - https://matplotlib.org/3.1.1/tutorials/index.html


```
pip install -U matplotlib
```

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

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

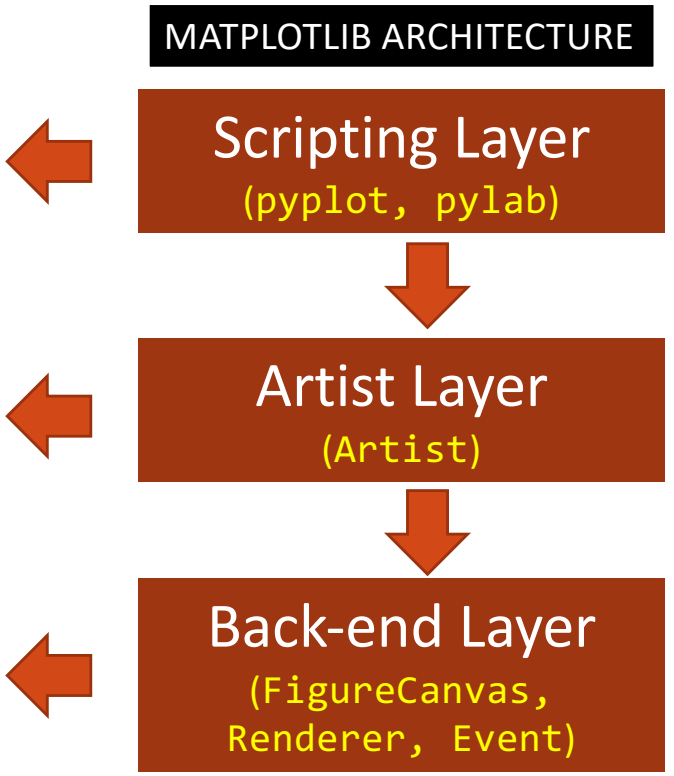
State-machine interface to the underlying Python libraries

- Object Oriented (OO) interface
- Procedural interface (discouraged)

Additional primitives for the fine-grained customization of each figure

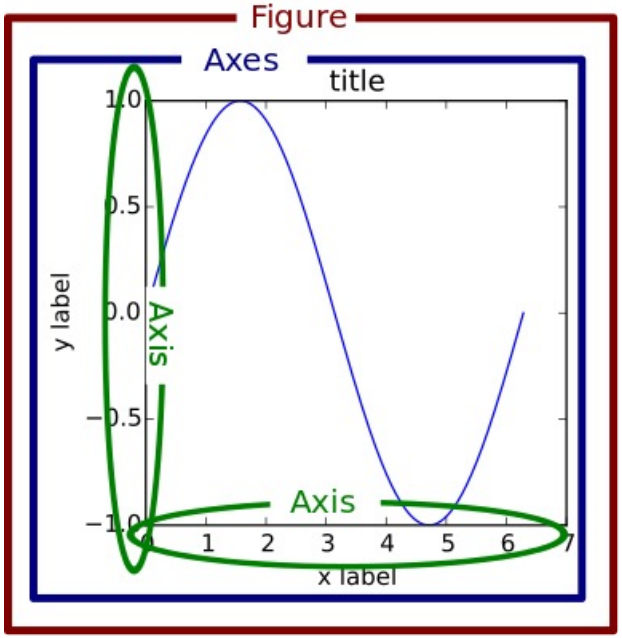
Hard work behind-the-scenes to **render** the figure and ro **export** it

- User interface rendering library (e.g. QT)
- Hardcopy backends to make files (e.g. PNG)

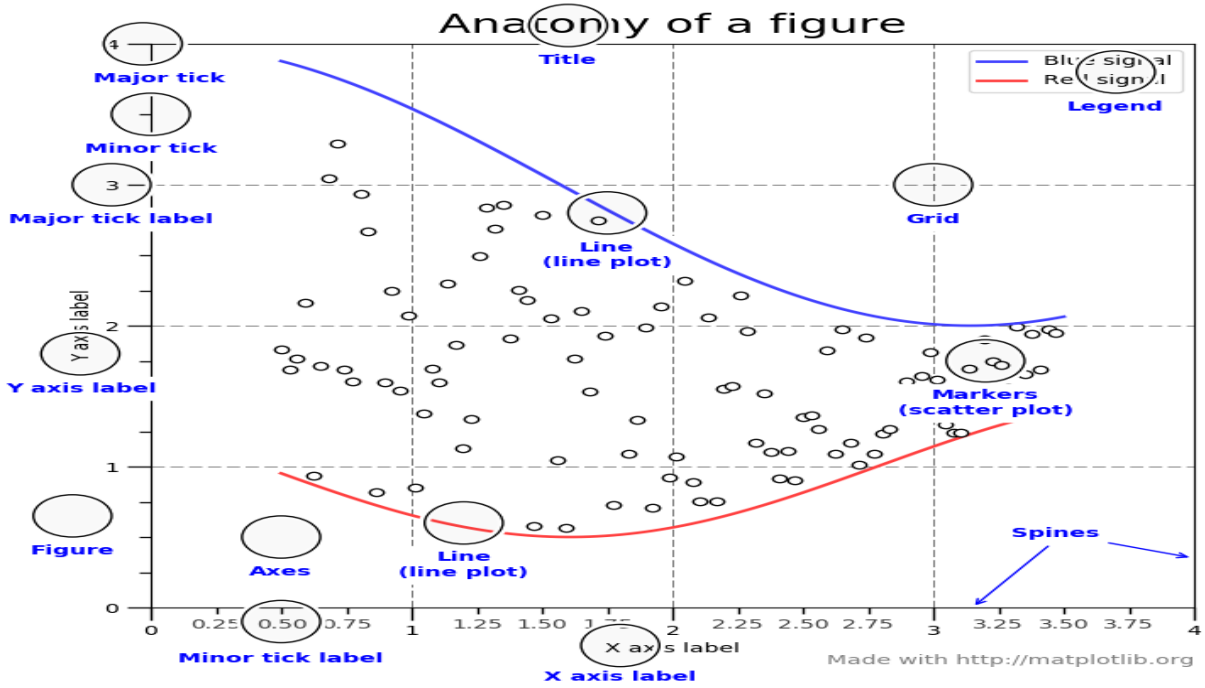


Matplotlib

- **Figure**
  - Overall window or page that everything is drawn on
  - Top-level component
- **Axes**
  - area on which data is plotted (e.g. via the `plot(…)` function)
  - A Figure contains at least one Axes, but it might contain more than one (e.g. subplots)
- **Axis**
  - Take care of setting the graph limits and generating the ticks (visual marks) and ticklabels (strings)



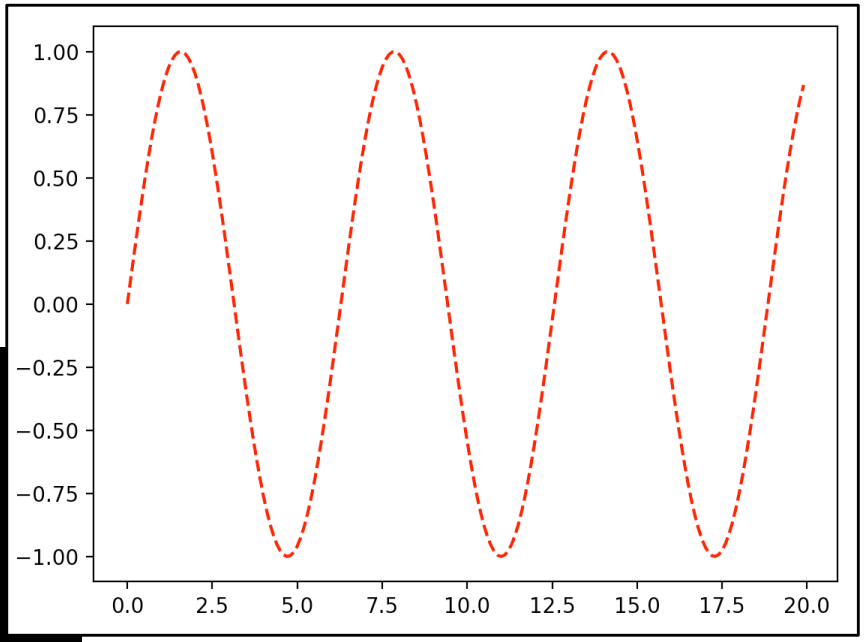
Matplotlib


Anatomy of a figure
Title
Major tick
Minor tick
Major tick label
Y axis label
Figure
Axes
Minor tick label
X axis label
Line
(line plot)
Grid
Legend
Markers
(scatter plot)
Spines
Made with http://matplotlib.org

Matplotlib

- **matplotlib.pyplot.plot()**
  - Plot one or multiple data series with linespoints
  - Argument: any list of data, including Numpy arrays
- **matplotlib.pyplot.show()**
  - Display the Figure

```
import matplotlib.pyplot as plt
data=np.arange(0.0, 20.0, 0.1)
plt.plot(data,np.sin(data),'r--')
plt.show()
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **DATA INPUT (1): Lists**

```
import matplotlib.pyplot as plt
x = [0,1,2,3,4,5,6,7,8,9]
y = [0,2,4,6,8,10,12,14,16,18]
plt.plot(x,y,'r--')
plt.show()
```

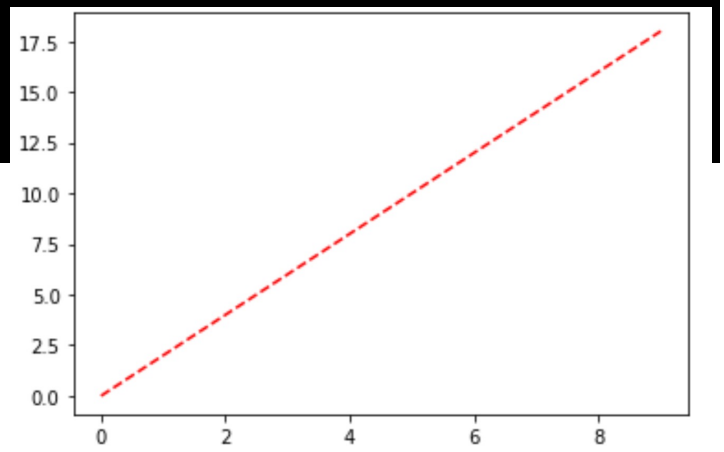
- **DATA INPUT (2): Numpy arrays**

```
import numpy as np
x = np.array([0,1,2,3,4,5,6,7,8,9])
y = np.array([0,2,4,6,8,10,12,14,16,18])
plt.plot(x,y,'r--')
plt.show()
```

Matplotlib

- **DATA INPUT (3): Dataframe Pandas**

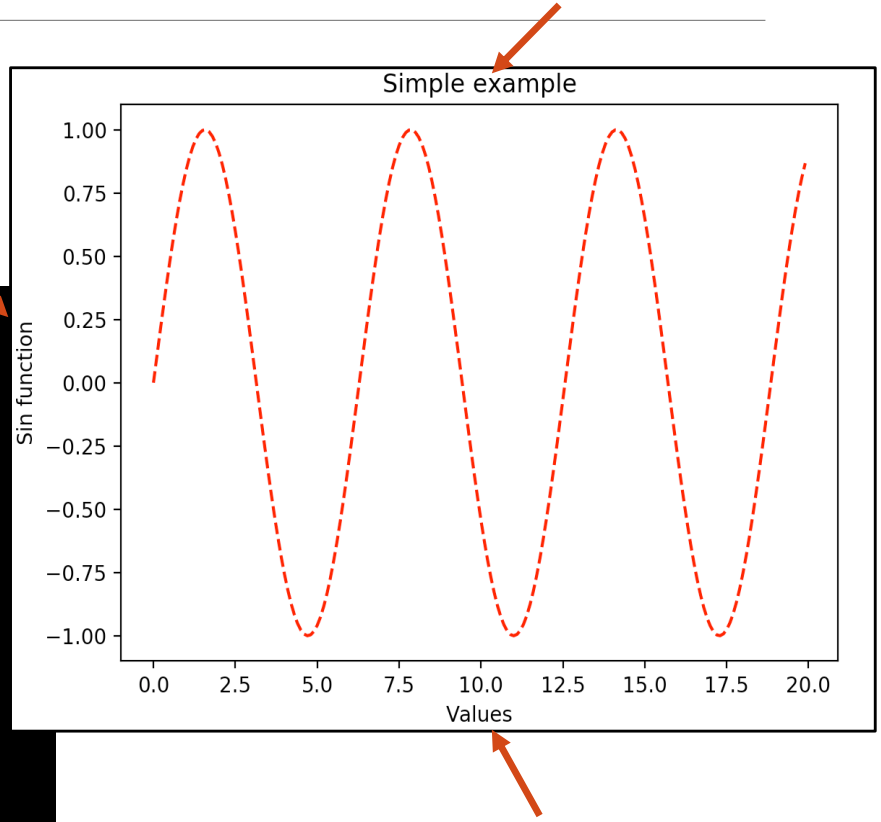
```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.DataFrame({"x":[0,1,2,3,4,5,6,7,8,9],
                   "y":[0,2,4,6,8,10,12,14,16,18] })
plt.plot(df["x"],df["y"],'r--')
plt.show()
```



Matplotlib

- **Customize the title and labels**
  - matplotlib.pyplot.xlabel(str)
  - matplotlib.pyplot.ylabel(str)
  - matplotlib.pyplot.title(str)

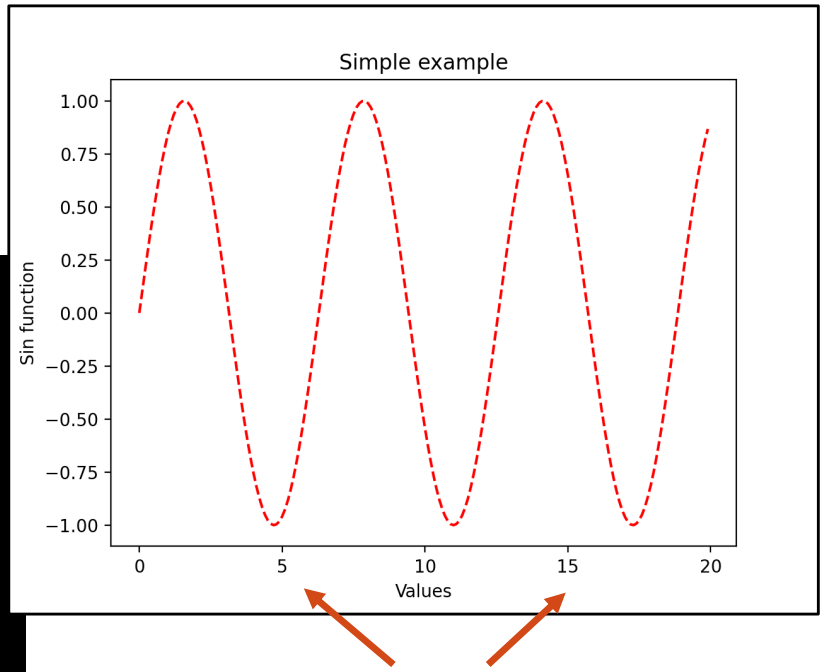
```
import matplotlib.pyplot as plt
data=np.arange(0.0, 20.0, 0.1)
plt.plot(data,np.sin(data),'r--')
plt.xlabel('Values')
plt.ylabel('Sin function')
plt.title('Simple example')
plt.show()
plt.savefig('example.png')
```



Matplotlib

- **Customize the tics on the x/y axis**
  - `matplotlib.pyplot.xtics(location, labels)`
  - `matplotlib.pyplot.ytics(location, labels)`

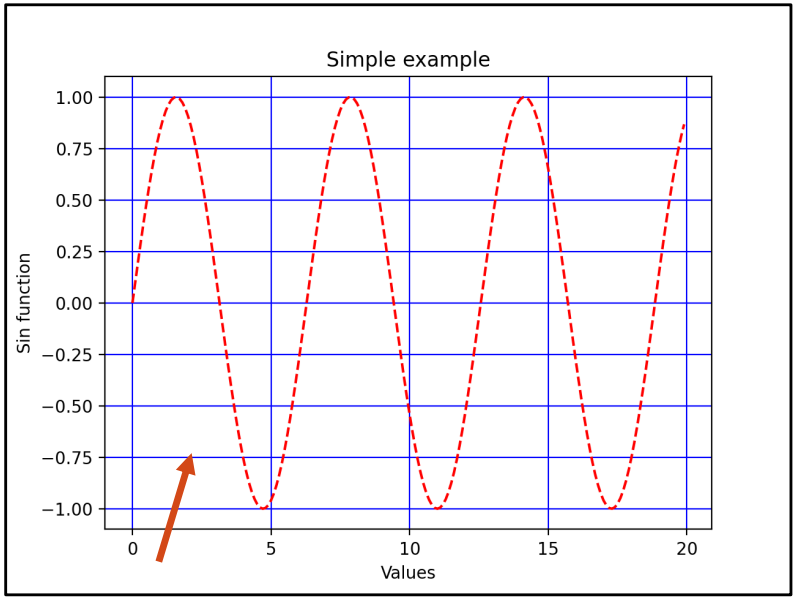
```
import matplotlib.pyplot as plt
data=np.arange(0.0, 20.0, 0.1)
plt.plot(data,np.sin(data),'r--')
plt.xlabel('Values')
plt.ylabel('Sin function')
plt.title('Simple example')
plt.xticks([0,5,10,15,20])
plt.show()
```



Matplotlib

- **Set the grid**
  - `matplotlib.pyplot.grid(color, linewidth)`

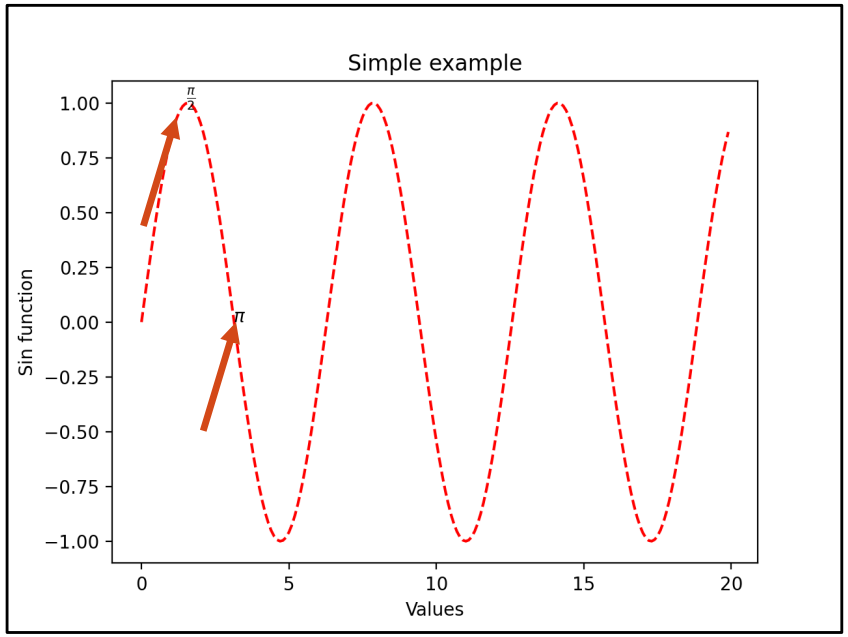
```
data = np.arange(0.0, 20.0, 0.1)
plt.plot(data, np.sin(data), 'r--')
plt.xlabel('Values')
plt.grid(color='b')
plt.ylabel('Sin function')
plt.title('Simple example')
plt.xticks([0, 5, 10, 15, 20])
plt.show()
```



Matplotlib

- **Add text at arbitratry positions**
  - `matplotlib.pyplot.text(x,y, str)`

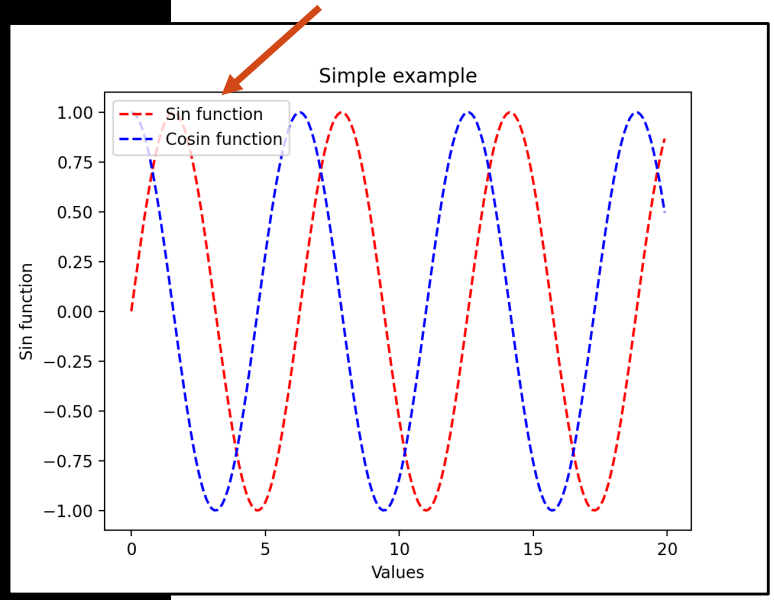
```
data = np.arange(0.0, 20.0, 0.1)
plt.plot(data, np.sin(data), 'r--')
plt.xlabel('Values')
plt.text(1.52,1,r'$\frac{\pi}{2}$')
plt.text(3.10, 0, r'$\pi$')
plt.ylabel('Sin function')
plt.title('Simple example')
plt.xticks([0, 5, 10, 15, 20])
plt.show()
```



Matplotlib

- **Add a legend**
  - `matplotlib.pyplot.legend(position)`

```
data = np.arange(0.0, 20.0, 0.1)

plt.plot(data, np.sin(data), 'r--', label='Sin function')

plt.plot(data, np.cos(data), 'b--', label='Cosin function')

plt.xlabel('Values')

plt.ylabel('Sin function')

plt.title('Simple example')

plt.xticks([0,5,10,15,20])

plt.legend(loc='upper left')

plt.show()
```



Matplotlib

- **Line plot**
  - `matplotlib.pyplot.plot(x,y, fmt,args)`
  - `fmt:` compact notation to define [color][marker][line]. Some examples:
    - `go--` → color=green, marker='o', linestyle='dashed'
    - `ro` → color=red, marker='o'
    - `b*-` → color=blue, marker='*', linestyle='solid'
  - Optional parameters (some examples):
    - linewidth: float number
    - antialiased: boolean value
    - label: object value
    - alpha: float value
    - ….

Matplotlib

Markers

character	description
`'.'`	point marker
`','`	pixel marker
`'o'`	circle marker
`'v'`	triangle_down marker
`'^'`	triangle_up marker
`'<'`	triangle_left marker
`'>'`	triangle_right marker
`'1'`	tri_down marker
`'2'`	tri_up marker
`'3'`	tri_left marker
`'4'`	tri_right marker
`'s'`	square marker
`'p'`	pentagon marker
`'*'`	star marker
`'h'`	hexagon1 marker
`'H'`	hexagon2 marker
`'+'`	plus marker
`'x'`	x marker
`'D'`	diamond marker
`'d'`	thin_diamond marker
`'|'`	vline marker
`'_'`	hline marker

Line Styles

character	description
`'-'`	solid line style
`'--'`	dashed line style
`'-.'`	dash-dot line style
`':'`	dotted line style

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

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Line plot**

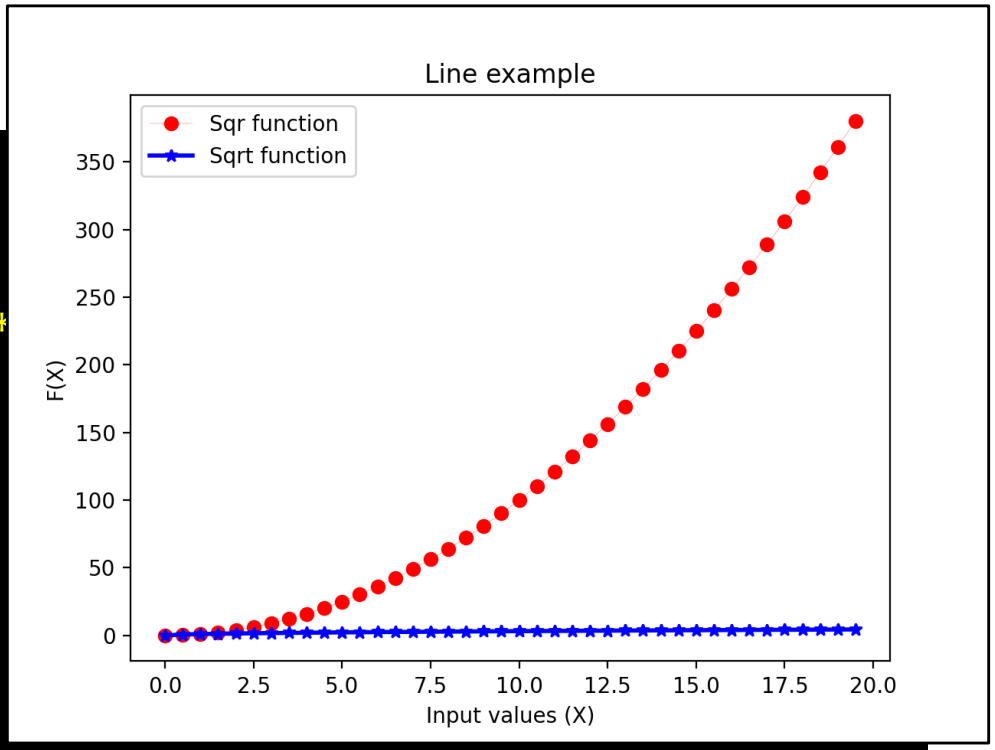
```
data = np.arange(0.0, 20.0, 0.5)
plt.plot(data, np.power(data,2), 'ro-', linewidth=0.1, label='Sqr function')
plt.plot(data, np.sqrt(data), 'b*-', linewidth=2, label='Sqrt function')
plt.xlabel('Input values (X)')
plt.ylabel('F(X)')
plt.title('Line example')
plt.legend(loc='upper left')
plt.show()
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Line plot**

```
data = np.arange(0.0, 20.0, 0.5)
plt.plot(data, np.power(data,2), 'r^', label='Sqr function')
plt.plot(data, np.sqrt(data), 'b*', label='Sqrt function')
plt.xlabel('Input values (X)')
plt.ylabel('F(X)')
plt.title('Line example')
plt.legend(loc='upper left')
plt.show()
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Histogram plot**
  - `matplotlib.pyplot.hist(x, bins, arguments)`
  - It approximates the probability density function (PDF) of the underlying variable
  - `x`: single or sequence of arrays
  - `bins`: if a single integer value, it defines the number of equal-width bins in the range. If is a sequence, it defines the bin edges, including the left edge of the first bin and the right edge of the last bin
  - Optional parameters. Some examples:
    - `rwidth`: The relative width of the bars
    - `color`: Color or sequence of colors, one per dataset.
  - Returns:
    - `<n, bins>`: values and edges of the bins

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Histogram plot**
  - matplotlib.pyplot.hist(x, bins, arguments)

```
data = np.random.normal(0, 0.1, 500)
result= plt.hist(data, 5)
print(result)
plt.xlabel('Input values (X)')
plt.ylabel('CDF')
plt.title('Histogram example')
plt.show()
```

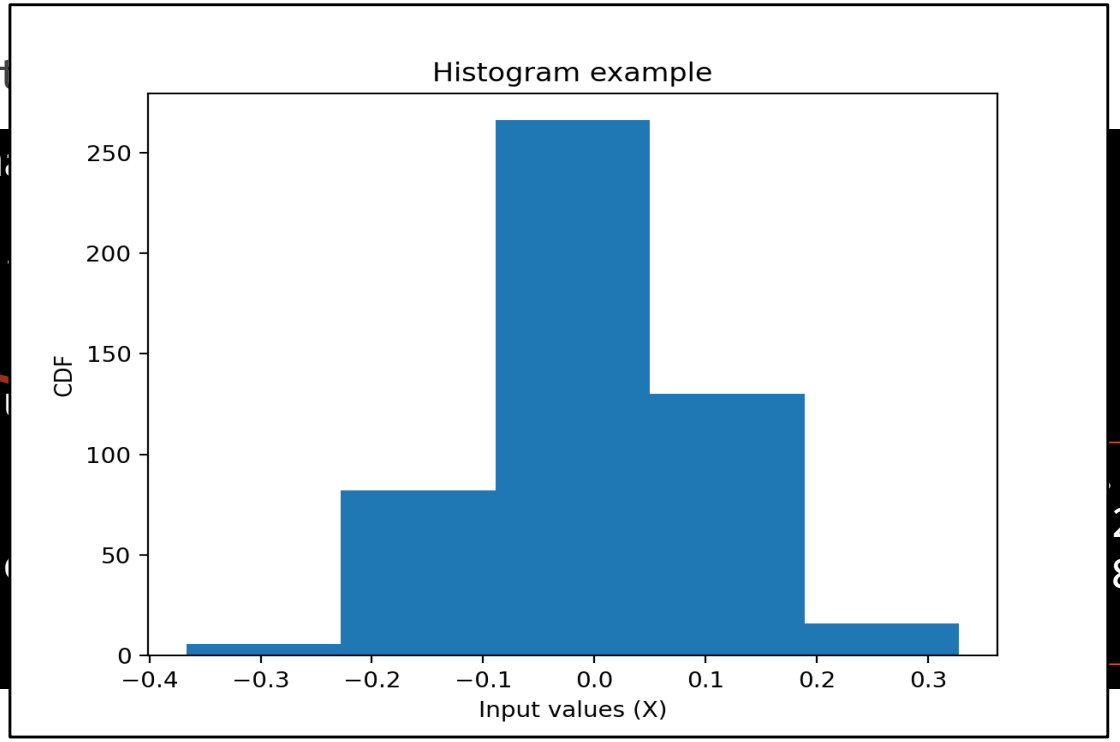
```
(array([  6.,  82., 266., 130.,
16.]), array([-0.36652272, -0.22773513,
-0.08894754,  0.04984004,  0.18862763,
0.32741522])
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Histogram plot**
  - matplotlib.pyplot.hist

```
data = np.random.norm
result= plt.hist(data
print(result)
plt.xlabel('Input val
plt.ylabel('CDF')
plt.title('Histogram
plt.show()
```




```
,
22773513,
8862763,
```

Matplotlib

- **Boxplot**
  - `matplotlib.pyplot.boxplot(x, arguments)`
  - `x`: Array or a sequence of vectors
  - Optional parameters. Some examples:
    - `sym`: The default symbol for flier points
    - `widths`: Sets the width of each box either with a scalar or a sequence
    - `whis`: reach of the whiskers to the beyond the first and third quartiles
    - `labels`: Labels for each dataset. Length must be compatible with dimensions of x
  - It returns a dictionary including the plot information in numeric format:
    - boxes, median, mean, whiskers, fliers

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

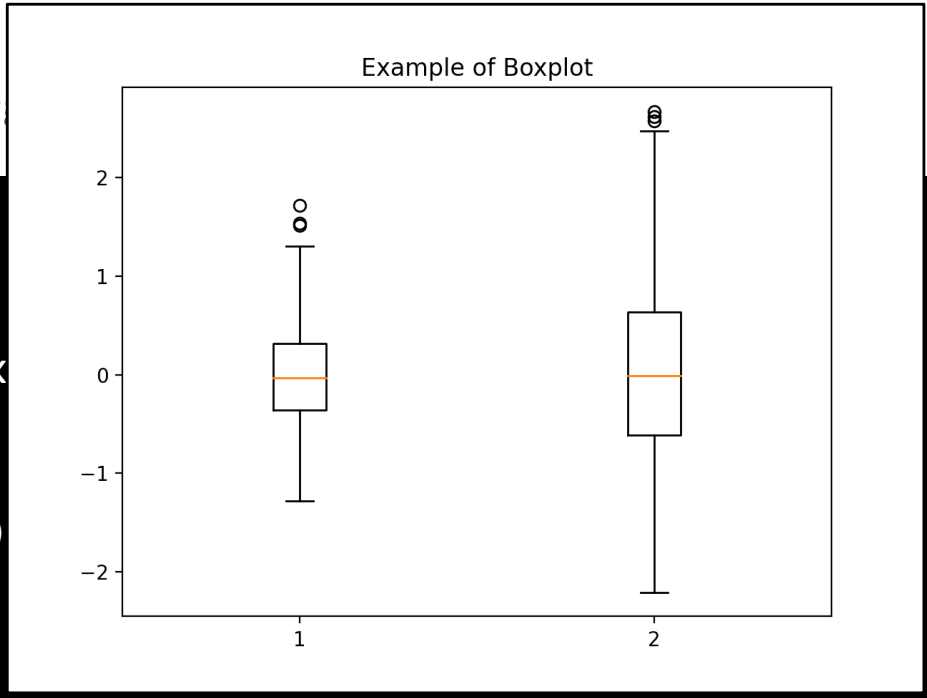
- **Boxplot**
  - `matplotlib.pyplot.boxplot(x, arguments)`

```
data0=np.random.normal(0, 0.5, size=(500,))
data1= np.random.normal(0, 1.0, size=(500,))
data=np.stack((data0,data1), axis=1)
plt.boxplot(data,  whis=1.5)
plt.title('Example of Boxplot')
plt.show()
```

Matplotlib

- **Boxplot**
  - matplotlib.pyplot.boxplot(x, ar

```
data0=np.random.normal(0, 0.5,
data1= np.random.normal(0, 1.0,
data=np.stack((data0,data1), ax
plt.boxplot(data,  whis=1.5)
plt.title('Example of Boxplot')
plt.show()
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Scatter plot**
  - `matplotlib.pyplot.scatter(x, y, arguments)`
  - `x,y`: data positions, array of shape (n,)
  - Optional parameters. Some examples:
    - `s`: marker size, flot of array of floats with shape (n,)
    - `c`: array-like or list of colors or color,
    - `marker`: marker style, default ‘o’
    - `alpha`: The alpha blending value, between 0 (transparent) and 1 (opaque).
    - `norm`: scale the color data `c` in the range 0 to 1

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

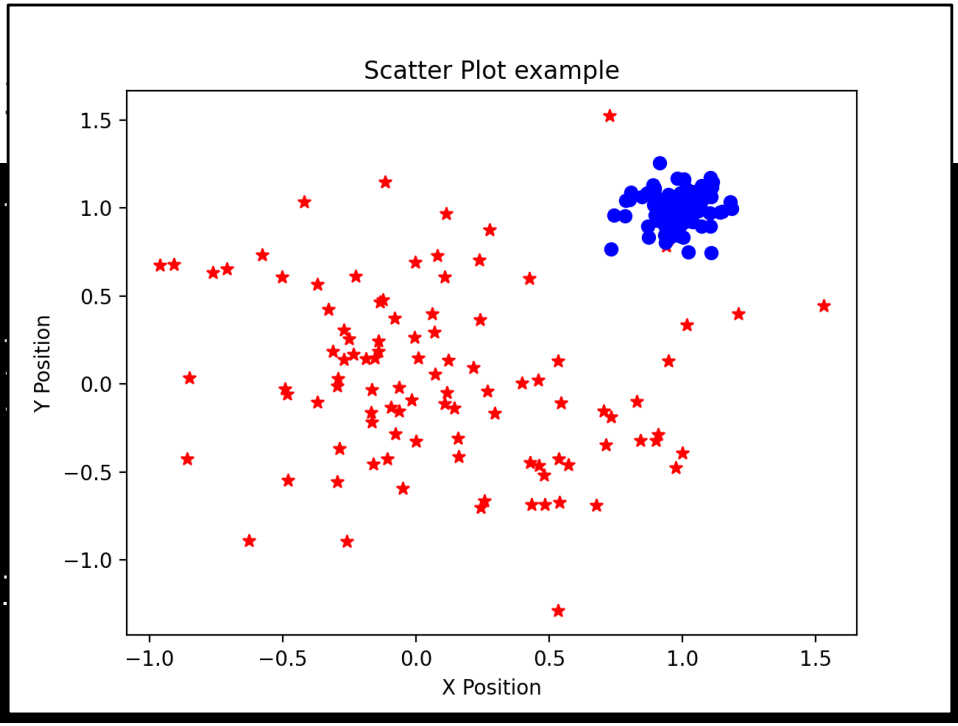
- **Scatter plot**
  - `matplotlib.pyplot.scatter(x, y, arguments)`

```
data0=np.random.normal(0, 0.5, size=(2,100))
data1 = np.random.normal(1.0, 0.1, size=(2, 100))
plt.scatter(data0[0],data0[1],c='r',marker='*')
plt.scatter(data1[0], data1[1], c='b', marker='o')
plt.xlabel('X Position')
plt.ylabel('Y Position')
plt.title('Scatter Plot example')
plt.show()
```

Matplotlib

- **Scatter plot**
  - matplotlib.pyplot.scatter(x,

```
data0=np.random.normal(0, 0.5
data1 = np.random.normal(1.0,
plt.scatter(data0[0],data0[1]
plt.scatter(data1[0], data1[1
plt.xlabel('X Position')
plt.ylabel('Y Position')
plt.title('Scatter Plot examp
plt.show()
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Bar plot**
  - `matplotlib.pyplot.bar(x, height, arguments)`
  - `x`: float or array of x coordinates of the bars
  - `height`: float or array of y coordinates of the bar
  - Optional parameters. Some examples:
    - `width`: float of array of float numbers
    - `bottom`: float of array of y coordinates of the bottom
    - `color`: color or array of colors of the bar faces
    - `edgecolor`: color or array of colors of the bar edges
    - `hatch`: filling pattern {'/', '\', '|', '-', '+', 'x', 'o', 'O', '.', '*'}

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Bar plot**
  - matplotlib.pyplot.bar(x, height, arguments)

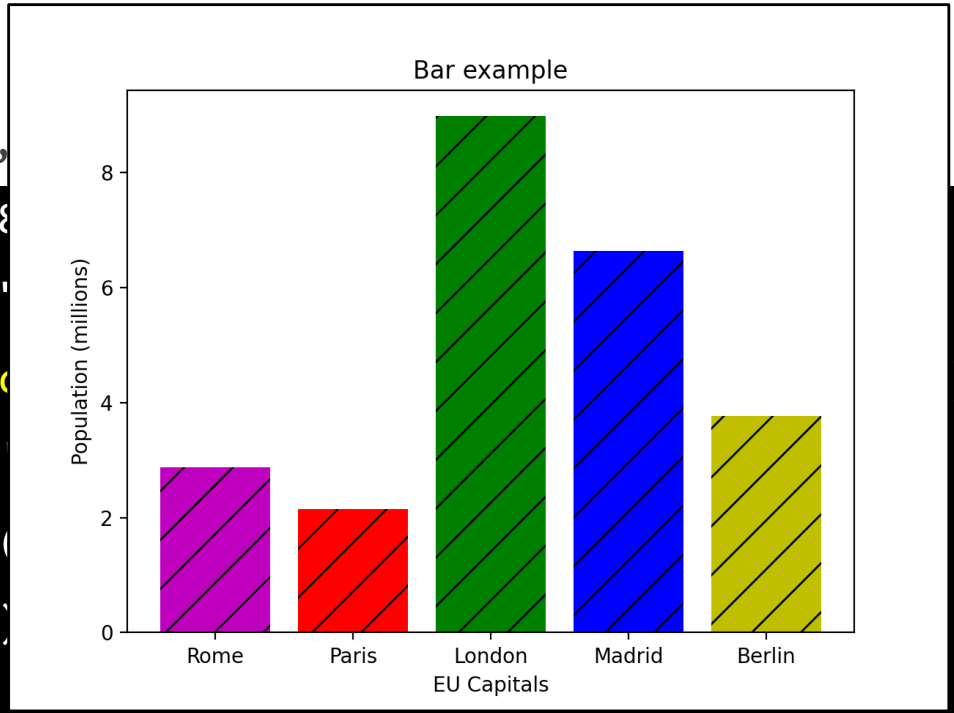
```
xvalue=[2.873,2.148,8.98,6.642,3.769]
yvalue=["Rome","Paris","London", "Madrid", "Berlin"]
plt.bar(yvalue,xvalue, color=['m','r','g','b','y'], hatch='/')
plt.xlabel('EU Capitals')
plt.ylabel('Population (millions)')
plt.title('Bar example')
plt.show()
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Bar plot**
  - matplotlib.pyplot.bar(x,

```
xvalue=[2.873,2.148,8.98
yvalue=["Rome","Paris",'
plt.bar(yvalue,xvalue, c
plt.xlabel('EU Capitals'
plt.ylabel('Population (
plt.title('Bar example')
plt.show()
```



Matplotlib

- **Bar plot (MULTI-BAR)**
  - matplotlib.pyplot.bar(x, height, arguments)

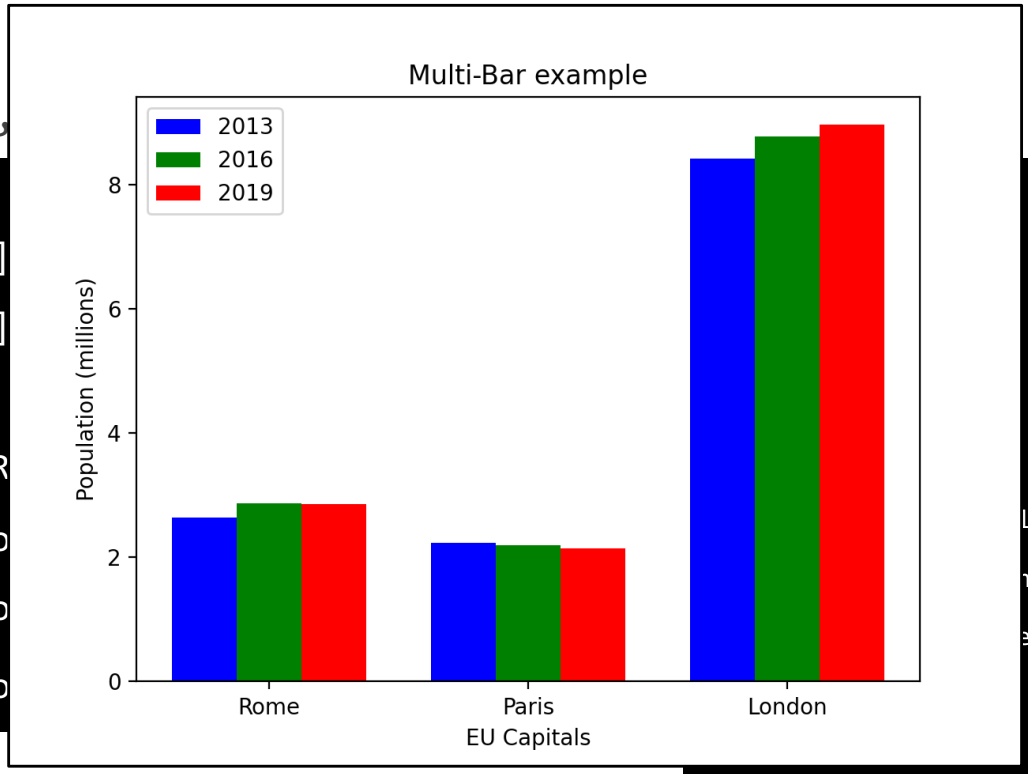
```
data = [[2.638, 2.23, 8.42],
            [2.864, 2.19, 8.77],
            [2.856, 2.14, 8.96]]
X = np.arange(3)
plt.xticks([0.25,1.25,2.25],['Rome','Paris','London'])
plt.bar(X + 0.00, data[0], color='b', width=0.25, label='2013')
plt.bar(X + 0.25, data[1], color='g', width=0.25, label='2016')
plt.bar(X + 0.50, data[2], color='r', width=0.25, label='2019')
plt.xlabel('EU Capitals')
plt.ylabel('Population (millions)')
plt.title('Multi-Bar example')
plt.legend()
plt.show()
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Bar plot (MULTI-BAR)**
  - matplotlib.pyplot.bar(x,

```
data = [[2.638, 2.23, 8.42],
        [2.864, 2.19, 8.77]
        [2.856, 2.14, 8.96]
X = np.arange(3)
plt.xticks([0.25,1.25,2.25],['R
plt.bar(X + 0.00, data[0], colo
plt.bar(X + 0.25, data[1], colo
plt.bar(X + 0.50, data[2], colo
```




```
ls')
 (millions)')
example')
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Pie Chart**
  - `matplotlib.pyplot.pie(x, arguments)`
  - `x`: The wedge sizes. The fractional area of each wedge is given by x/sum(x)
  - Optional parameters. Some examples:
    - `labels`: A sequence of strings providing the labels for each wedge
    - `colors`: A sequence of matplotlib color args through which the pie chart will cycle
    - `radius`: The radius of the pie
    - `startangle`: rotates the start of the pie chart counterclockwise from the x-axis
    - `shadow`: boolean, draw a shadow beneath the pie

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

- **Pie Chart**
  - `matplotlib.pyplot.pie(x, arguments)`

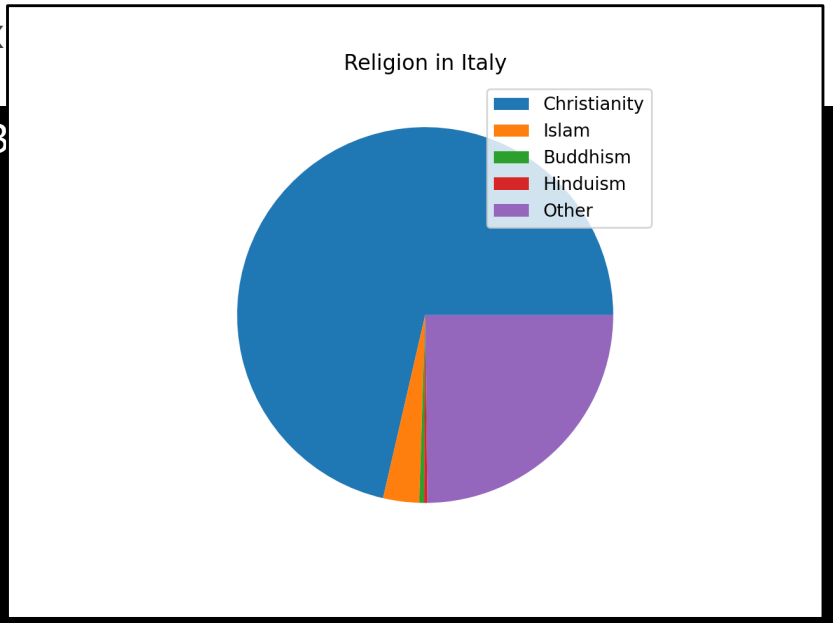
```
data1=np.array([71.4, 3.1, 0.4, 0.3, 24.8])
label=['Christianity', 'Islam', 'Buddhism', 'Hinduism', 'Other']
plt.pie(data1)
plt.legend(label)
plt.title('Religion in Italy')
plt.show()
```

Matplotlib

- **Pie Chart**
  - matplotlib.pyplot.pie(x

```
data1=np.array([71.4, 3
label=['Christianity',
plt.pie(data1)
plt.legend(label)
plt.title('Religion in
plt.show()
```

```
'Other']
```



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Data Visualization

- **Why data visualization**
- **Basic concepts and tools**
  - Some definitions: marks, channels, separability
  - Dealing with colors
  - Univariate attribute
  - Multivariate attributes
- **Plotting data with Python**
  - Matplotlib library
  - *Seaborn library*

Seaborn

- **Python 2D/3D plotting library**
  - Built on top of matplotlib
  - Object-oriented API

- **Integrated with Pandas library**
  - Same Matplotlib plot types (e.g. bar plots) + additional plot styles (e.g. heatmaps) and customization effects
- **Official (introductory, intermediate, advanced) tutorials:**
  - https://seaborn.pydata.org/tutorial.html

```
pip install -U seaborn
```

```
import seaborn as sns
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Seaborn

- **Histogram plot**
  - `sns.displot(x, bins, other_arguments)`
- **Box plot**
  - `sns.boxplot(df['age'], whis, other_arguments)`
- **Scatter plot**
  - `sns.scatterplot(x, y, hue, other_arguments)`
- **Line plot**
  - `sns.lineplot(x, y, , other_arguments)`

Seaborn

- **Density plot**
  - `seaborn.kdeplot(x,y,arguments)`
  - `x,y:` Variables that specify positions on the x and y axes.
  - Optional parameters. Some examples:
    - `kernel`: function that defines the kernel (default: Gaussian)
    - `bw`: smoothing factor
    - `cumulative`: If True, computes the compulative function
    - `clip`: do not evaluate the density outside of these limits.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Seaborn

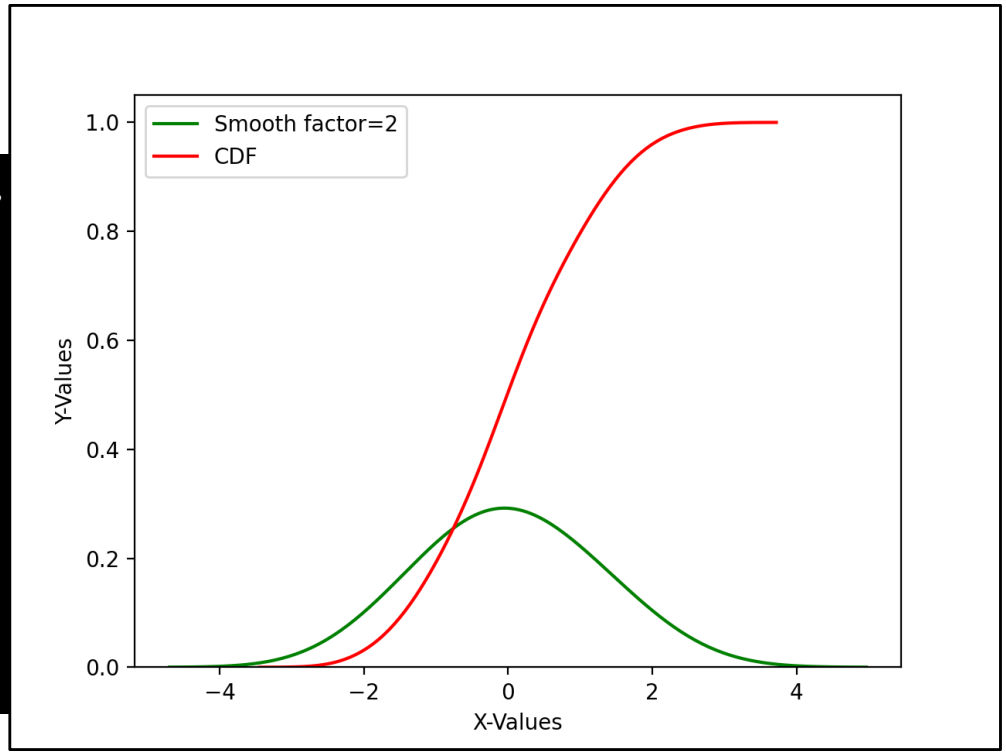
- **Density plot**
  - `seaborn.kdeplot(x,y,arguments)`

```
data0 = np.random.normal(0, 1.0, size=(100,))
sns.kdeplot(data0, color='g', bw_adjust=2, label='Smooth factor=2')
sns.kdeplot(data0, color='r', cumulative=True, label='CDF')
plt.legend()
plt.xlabel("X-Values")
plt.ylabel("Y-Values")
plt.show()
```

Seaborn

- **Density plot**
  - seaborn.kdeplot(x,y,arguments)

```
data0 = np.random.normal(0, 1.
sns.kdeplot(data0, color='g',
sns.kdeplot(data0, color='r',
plt.legend()
plt.xlabel("X-Values")
plt.ylabel("Y-Values")
plt.show()
```



Seaborn

Heatmap

- `seaborn.heatmap(data, arguments)`
- `data:` 2D dataset that can be coerced into an ndarray. If a Pandas DataFrame is provided, the index/column information will be used to label the columns and rows.
- Optional parameters. Some examples:
  - `cmap`: mapping from data values to color space
  - `vmin, vmax`: values to anchor the colormap, otherwise they are inferred from the data
  - `annot`: If True, write the data value in each cell
  - `cbar`: boolean, whether to show the colorbar
  - `linewidths`: Width of the lines that will divide each cell

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Matplotlib

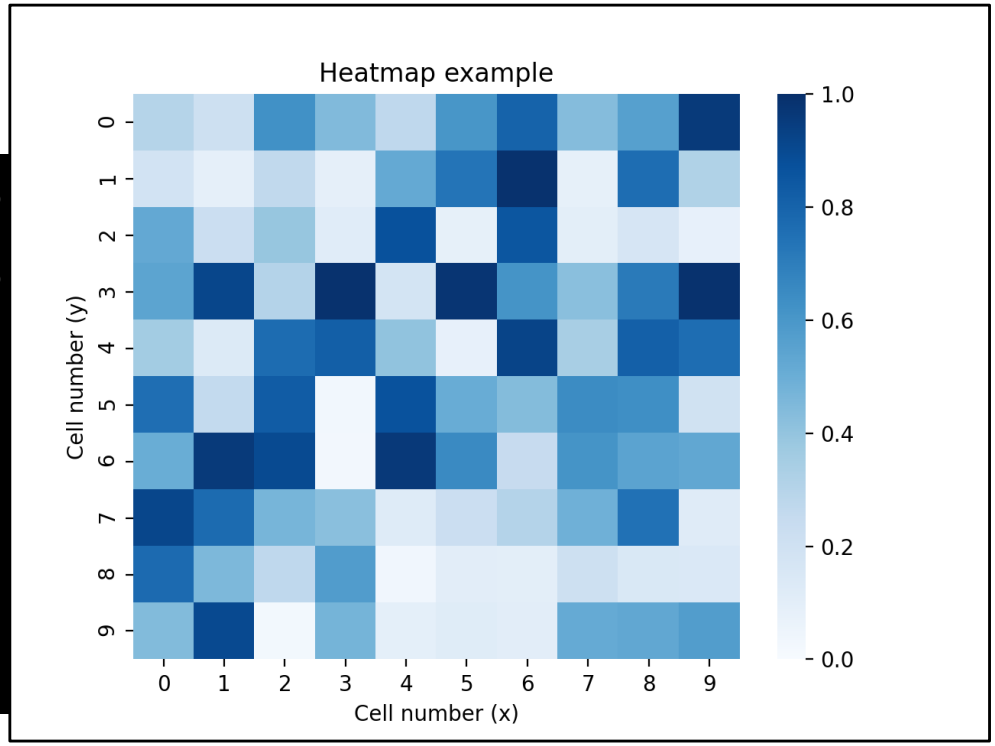
- **Heatmap**
  - seaborn.heatmap(map, arguments)

```
normal_data = np.random.rand(10, 10)
sns.heatmap(normal_data, vmin=0, vmax=1, cmap='Blues')
plt.xlabel('Cell number (x)')
plt.ylabel('Cell number (y)')
plt.title('Heatmap example')
plt.show()
```

Matplotlib

- **Heatmap**
  - seaborn.heatmap(map, arguments)

```
normal_data = np.random.rand(10,
sns.heatmap(normal_data, vmin=0,
plt.xlabel('Cell number (x)')
plt.ylabel('Cell number (y)')
plt.title('Heatmap example')
plt.show()
```



Seaborn

- **Scatter Plot Matrix**
  - `seaborn.pairplot(data, arguments)`
  - `data:` dataframe where each column is a variable and each row is an observation.
  - Optional parameters. Some examples:
    - `hue`: Variable in data to map plot aspects to different colors
    - `kind`: Kind of plot to make {‘scatter’, ‘kde’, ‘hist’, ‘reg’}
    - `height`: Height of each facet
    - `palette`: Set of color for the Hue variable

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Seaborn

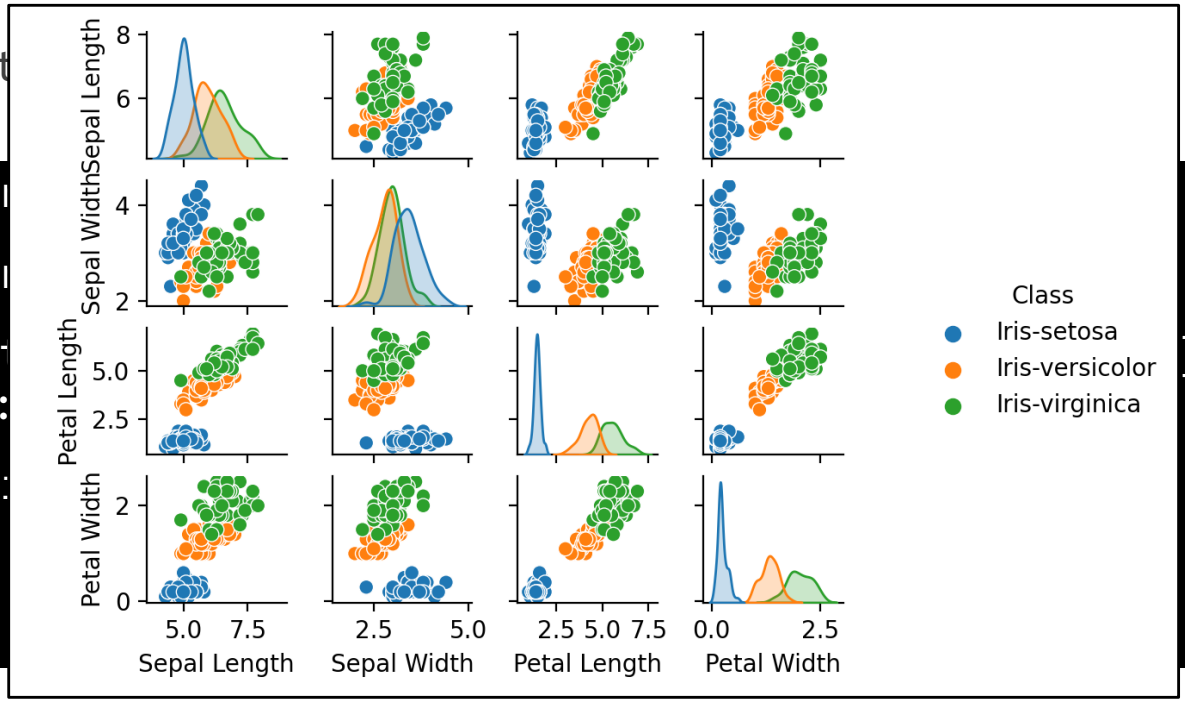
- **Scatter Plot Matrix**
  - `seaborn.pairplot(data, arguments)`

```
data = np.random.normal(0, 0.5, size=(100,3))

category=np.random.randint(1,4, size=(100,))

df=pd.DataFrame({'Attribute1': data[:, 0], 'Attribute2': data[:, 1],
'Attribute3': data[:, 2],'Class': category})

sns.pairplot(df, height=1, hue='Class')

plt.show()
```

Seaborn

- **Scatter Plot Matrix**
  - `seaborn.pairplot(dat`

```
data = np.random.nor

category=np.random.r

df=pd.DataFrame({'A
'Attribute3': data[:

sns.pairplot(df, he

plt.show()
```



Seaborn

Subplots with Seaborn

```
fig, axs = plt.subplots(2, figsize=(18, 10))

sns.kdeplot(df['age'], ax=axs[0], color='g', bw_adjust=2,
label='Smooth factor=2')

sns.kdeplot(df['age'], ax=axs[1], color='r', cumulative=True,
label='CDF')

plt.show()
```