DATA VISUALIZATION IN PYTHON

Jagriti Goswami

Overview

- Visualization Libraries And Modules
- Version Overview
- Visualization Plot Types

- **Bar Chart**
- **Horizontal Bar Chart**
- **Stacked Bar Chart**
- **Grouped Bar Chart**
- **Line Chart**
- **Stacked Area Graph**
- **Pie Chart**
- **Histogram**
- **Density Plot**
- **Density Plot with Histogram**
- **Scatter Plot**
- **Boxplot**
- **Violin Plot**
- **Heatmap**
- **Time Series**

Visualization Libraries And Modules

Libraries and Modules	Description	Doc Link	Installation Link	Import Statement
Pandas	pandas is an open source software library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.	Link	Installation	import pandas as pd
NumPy	NumPy is the fundamental package for scientific computing with Python providing multidimensional array object, various derived objects such as masked arrays and matrices.	Link	Installation	import numpy as np
Matplotlib	Matplotlib is a Python 2D plotting library.	Link	Installation	import matplotlib
matplotlib.pyplot	matplotlib.pyplot is a state-based interface to matplotlib. It provides a MATLAB-like plotting.	Link		import matplotlib.pyplot as plt
Seaborn	Seaborn is a python data visualization library based on matplotlib.	Link	Installation	import seaborn as sns
Plotly	Plotly python open source graphing library makes interactive graphs online	Link	Installation	import plotly.express as px import plotly.graph_objects as go

Version Overview

Version	What's New in Each Version?
2015.09	• Basic visualization plots using matplotlib, seaborn, and pandas
2017.06	• Visualization plots using seaborn • Updates some graphs and charts with new functions and data
2018.03	• Data visualization with Plotly • 3D Visualization plots • Updates some graphs and charts with new functions and data

Version 2015.09

Visualization Plots Using Matplotlib, Seaborn, Pandas

Bar Chart

- **Bar Chart** Shows comparisons between different categories, different parts of a whole.
- Shows relationship between a numerical variable and a discrete variable.

Variations of Bar Chart :

- **Vertical Bar Chart or Column Chart :** Best used to visualize relationship or comparisons with chronological data.
- **Horizontal Bar Chart :** Useful when category labels are long.
- **Stacked Bar Chart :** Used to compare different parts of a whole using discrete or continuous variables.
- **Grouped Bar Chart :** Used to compare multiple data series in a given category.

Vertical Bar Chart (Column Chart)

Functions for plotting bar chart:

1. ***fig, ax = plt.subplots(nrows=no_of_row, ncols=no_of_col, figsize=(x , y))*** **:** Creates an figure and a set of subplots and sets figure size. Returns a single Axes object or an array of axes objects if more than one subplot are created. For details click here.

2. ***matplotlib.pyplot.bar(x, height, width, bottom=None, align='center', data=None, **kwargs)***

Make a Bar Chart. For more details, click here.

Parameters:	**x :**	Sequence of scalars; the bars are positioned at x with the given alignment.
	height, width :	Scalar or sequence of scalar or array like; the dimensions of bar are set by these parameters.
	bottom :	Scalar or array like; the vertical baseline is bottom (default 0).
	align :	Alignment of the bars to the x coordinates; {'center', 'edge'}, default('center').
****kwargs**	**color :**	Scalar or array-like; the color of the bar faces. For, e.g., we have set blue color as color='b' in the given example.
	alpha :	Float or None; set the alpha transparency of the patch (a 2D artist with a face color or an edge color). For, e.g., we have set alpha=0.6 in the given example.

3. ***matplotlib.pyplot.xticks(ticks=None, labels=None, **kwargs)*** **:**

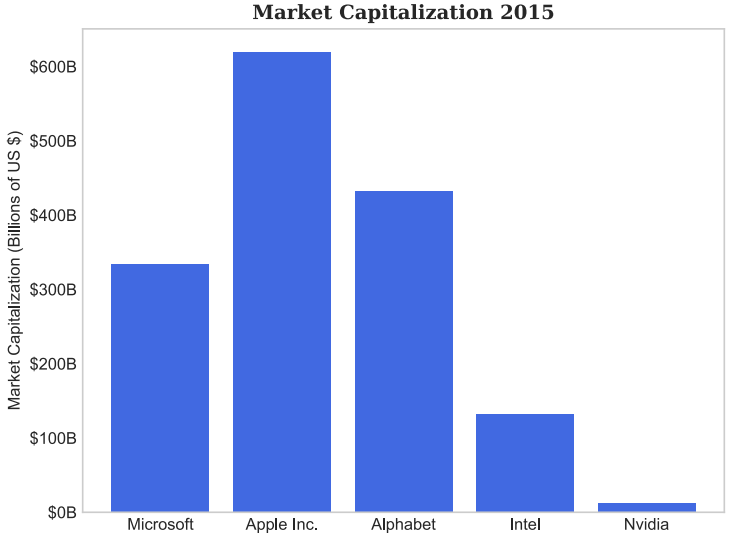
Get or set the current tick locations and labels of the x-axis. For more details, click here.

Vertical Bar Chart : Example

4. *matplotlib.ticker.FuncFormatter(func)* : Use a user defined function for label formatting. It takes two inputs (a tick value x and a position pos), and returns a string containing the corresponding tick label. For details click here.

Example : Displays Market-Cap of different technology industries. Data were collected from Yahoo Finance . See full code on github-barchart.

```
# Prepare data
organization = ('Microsoft', 'Apple Inc.', 'Alphabet', 'Intel', 'Nvidia')
y = np.arange(len(organization))
market_cap = [334.39, 619.76, 432.15, 132.06, 11.69]


# Plot Bar Chart
def billions(x, pos):
    """The two args are the value and tick position"""
    return '$%1.0fB' % (x*1)


formatter = FuncFormatter(billions)

fig, ax = plt.subplots()
ax.yaxis.set_major_formatter(formatter)
plt.bar(y, height=market_cap, align='center', color='royalblue')
plt.xticks(y, organization)
ax.tick_params(width=0)
plt.ylim(ymin=0)
plt.ylabel('Market Capitalization (Billions of US $)')
plt.title('Market Capitalization 2015', font_param)

plt.grid(False)
plt.tight_layout()

# Save and show figure
plt.savefig('bar_chart.pdf')
plt.show()
```



Horizontal Bar Chart

Functions for plotting horizontal bar chart :

1. ***matplotlib.axes.Axes.barh(self, y, width, height=0.8, left=None, align='center', **kwargs)***

Make a horizontal bar. For more details click here.

Parameters:	**y :**	Scalar or array like; y coordinates of the bars.
	Width, height :	Scalar or array like; sequence of scalars; the dimensions of bar are set by these parameters.
	left :	Sequence of scalars; the x coordinates of the left sides of the bars (default 0).
	align :	Alignment of the bars to the y coordinates; {'center', 'edge'}, default('center').
****kwargs**	**color :**	Scalar or array-like; the color of the bar faces. For details click here.
	alpha :	Float or None; set the alpha transparency of the patch (a 2D artist with a face color or an edge color).

2. ***matplotlib.ticker.FuncFormatter(func)*** **:** Uses user defined function for label formatting. For details click here.

3. ***ax.set_yticks(self, ticks, minor=False)*** **:** Set the y ticks with list of ticks. It the parameter minor is False sets major ticks, if True sets major ticks. Default is False.

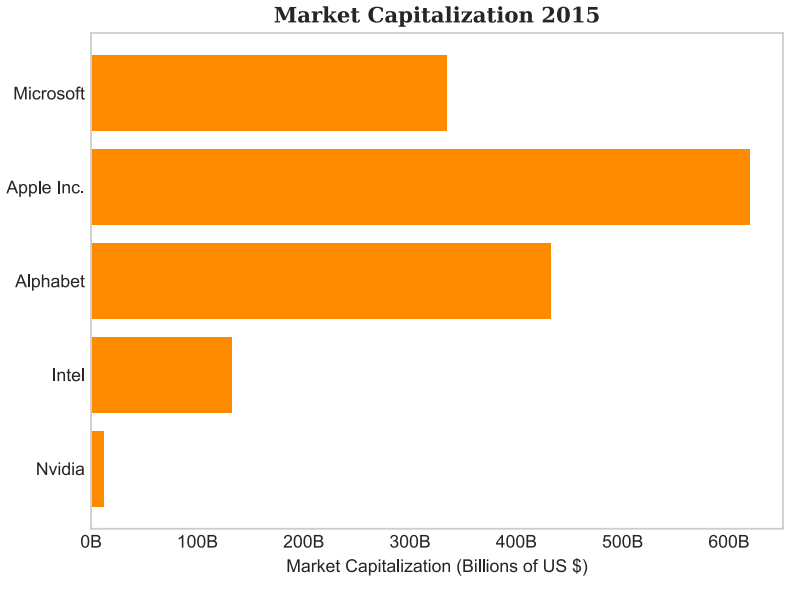
4. ***ax.set_yticklabels(self, labels, fontdict=None, minor=False, **kwargs)*** **:** Sets the y-tick labels with list of strings labels. For details click here.

5. ***ax.invert_yaxis(self)*** **:** Inverts the y-axis.

Horizontal Bar Chart : Example

Example : Displays Market-Cap of different technology industries. Data were collected from Yahoo Finance . See full code on github-horizontalbar.

```
# Prepare data
organization = ('Microsoft', 'Apple Inc.', 'Alphabet', 'Intel', 'Nvidia')
y = np.arange(len(organization))
market_cap = [334.39, 619.76, 432.15, 132.06, 11.69]


# Plot horizontal bar chart
def billions(x, pos):
    """The two args are the value and tick position"""
    return '%1.0fB' % (x*1)


formatter = FuncFormatter(billions)

fig, ax = plt.subplots()
ax.xaxis.set_major_formatter(formatter)
ax.barh(y, width=market_cap, align='center', color='darkorange')
ax.set_yticks(y)
ax.set_yticklabels(organization)
ax.invert_yaxis()  # labels read top-to-bottom
ax.tick_params(width=0)
plt.xlabel('Market Capitalization (Billions of US $)')
plt.title('Market Capitalization 2015', font_param)
plt.grid(False)
plt.tight_layout()
```



Stacked Bar Chart

Functions for plotting stacked bar chart :

1. *matplotlib.pyplot.bar(x, height, width, bottom=None, align='center', data=None, **kwargs)*

Make a Stacked Bar Chart. For more details, click here.

Parameters:	**x :**	Sequence of scalars; the bars are positioned at x with the given alignment.
	height, width :	Scalar or sequence of scalar or array like; the dimensions of bar are set by these parameters.
	bottom :	Scalar or array like; the vertical baseline is bottom (default 0). In the given example, we have set bottom=revenue to plot stacked bar chart.
	align :	Alignment of the bars to the x coordinates; {'center', 'edge'}, default('center').

2. *matplotlib.pyplot.xticks(ticks=None, labels=None, **kwargs)* :

Get or set the current tick locations and labels of the x-axis. For more details, click here.

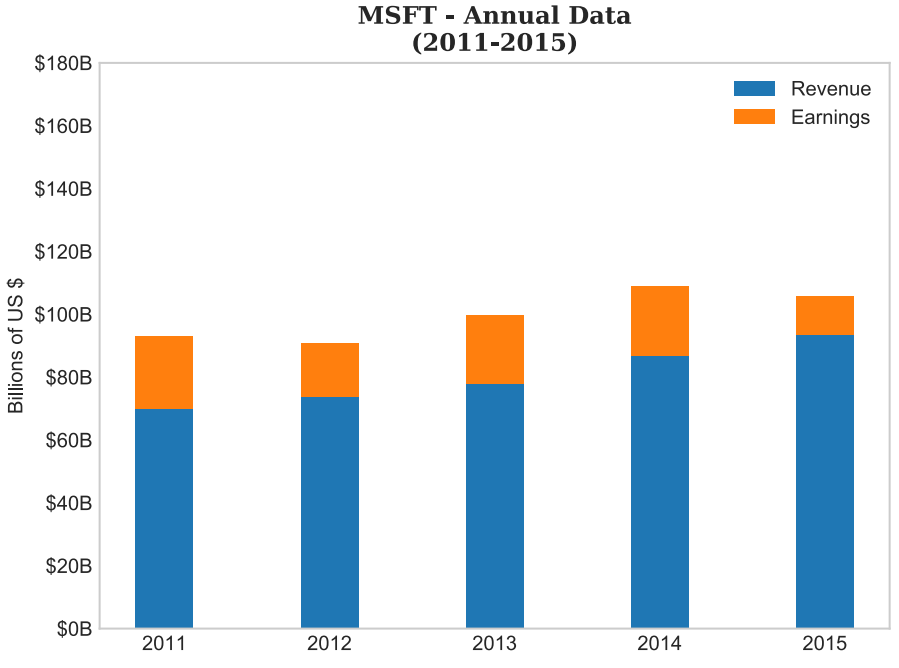
3. *ax.yaxis.set_major_formatter(formatter)* : Provides Configurable tick locating and formatting.

4. *matplotlib.ticker.FuncFormatter(func)* : Use user defined function for label formatting. For details click here.

Stacked Bar Chart : Example

Example : Comparison between Microsoft's Revenue and Earnings (in billions) for the year 2010-2015. Data were collected from Yahoo Finance. See full code on github-stackedbar.

```
# Plot stacked bar chart for revenue vs earnings
def billions(x, pos):
    """The two args are the value and tick positic
    return '$%1.0fB' % (x*1)


formatter = FuncFormatter(billions)
fig, ax = plt.subplots()
p1 = plt.bar(x, revenue, width)
p2 = plt.bar(x, earnings, width, bottom=revenue)

ax.yaxis.set_major_formatter(formatter)
plt.title('MSFT - Annual Data\n(2011-2015)', font_
plt.xticks(x, year)
plt.ylim(0, 180)
plt.ylabel('Billions of US $')
plt.legend((p1[0], p2[0]), ('Revenue', 'Earnings'),
plt.grid(False)
plt.tight_layout()
```



Grouped Bar Chart

Functions for plotting grouped bar chart :

1. ***matplotlib.pyplot.bar(x, height, width, bottom=None, align='center', data=None, **kwargs)***

Make a Stacked Bar Chart. For more details, click here.

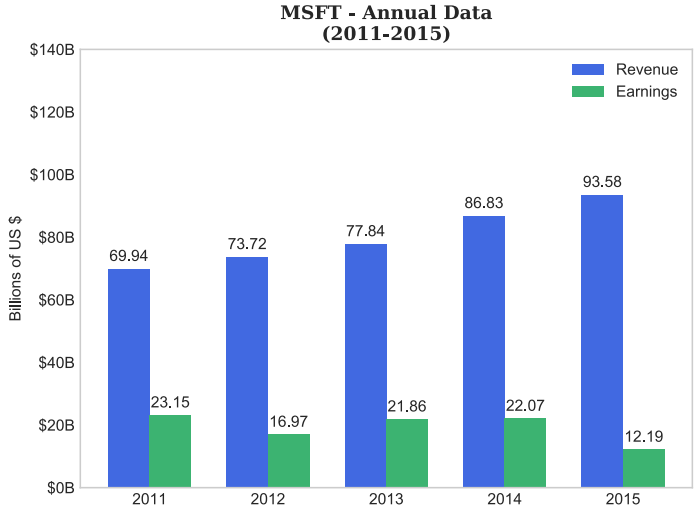
Parameters:	**x :**	Sequence of scalars; the bars are positioned at x with the given alignment. In the given example, for bar1, we have set (x = x - width/2) and for bar2 (x = x + width/2) to plot a grouped bar.
	height, width :	Scalar or sequence of scalar or array like; the dimensions of bar are set by these parameters.
	bottom :	Scalar or array like; the vertical baseline is bottom (default 0).
	align :	Alignment of the bars to the x coordinates; {'center', 'edge'}, default('center').

2. ***matplotlib.pyplot.xticks(ticks=None, labels=None, **kwargs)*** **:** Get or set the current tick locations and labels of the x-axis. For more details, click here.
3. **A*xes.yaxis.set_major_formatter(formatter)*** **:** Provides Configurable tick locating and formatting.
4. ***matplotlib.ticker.FuncFormatter(func)*** **:** Use user defined function for label formatting. For details click here.
5. ***autolabel(bars)*** **:** Attach a text label above each bar, displaying its height.

Grouped Bar Chart: Example

Example : Comparison between Microsoft's Revenue and Earnings (in billions) for the year 2010-2015. Data were collected from Yahoo Finance. See full code on github-groupedbar.

```
# Plot Grouped bar between Revenue and Earnings
def billions(x, pos):
    """The two args are the value and tick position"""
    return '$%1.0fB' % (x*1)


formatter = FuncFormatter(billions)
fig, ax = plt.subplots()
bar1 = ax.bar(x - width/2, revenue, width, color='royalblue', label='Revenue')
bar2 = ax.bar(x + width/2, earnings, width, color='mediumseagreen', label='Earnings')


def autolabel(bars):
    """Attach a text label above each bar in *bars*, displaying its height."""
    for rect in bars:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points", ha='center', va='bottom')


autolabel(bar1)
autolabel(bar2)
```



Line Graph

- **Line Chart** displays time-series relationships with continuous data.

Functions for plotting line graph :

1. ***Axes.plot(self, *args, scalex=True, scaley=True, data=None, **kwargs)*** **:** Plot y versus x as line and/or markers.

For details, click here.

Call signatures: **Plot([x], y, [fmt], *, data=None, **kwargs)**

Parameters:	**x, y :**	array-like or scalar; the coordinates of the points or line nodes are given by x, y.
	fmt :	str, optional; a format string, e.g., 'ro' for red circles.
	data :	indexable object, optional; An object with label data.
	****kwargs**	Used to specify properties like line label (for auto legend), line width, antialiasing, marker face color.

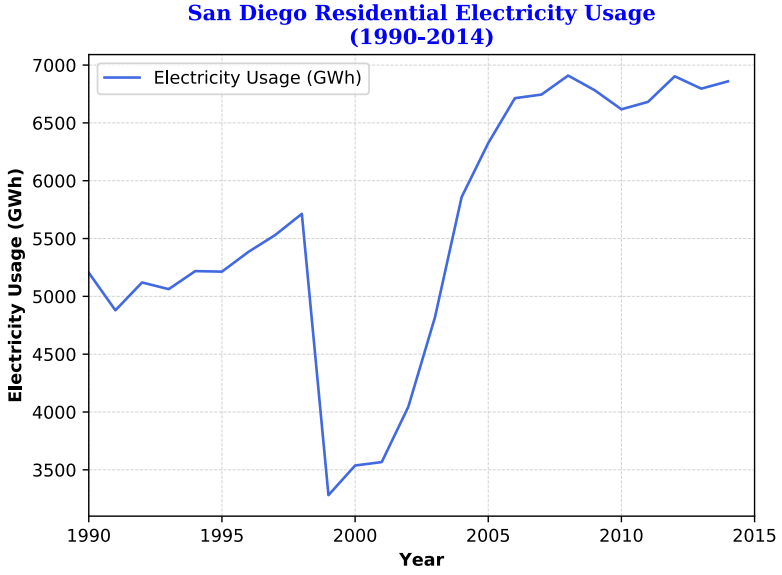
2. ***Axes.tick_params(axis='x', direction='out', length=3, width=0.5, labelrotation=75.00)*****:** Decorate the appearance of ticks, ticklabels, and gridlines. For more details, click here.

3. ***Axes.grid(color='lightgray', linestyle='--', linewidth='0.5')*****:** Provides configurable grid lines. For details, click here.

Line Graph: Example

Example : Line graph of residential electricity usage of San Diego (1990-2014). Source: California Electricity Consumption Database. Data were collected from data.ca.gov. All Usage Expressed in Millions of kWh (GWh). See full code on github-linegraph.

```
# Plot line graph of San Diego residential electricity usage
x = sd_electricity['Year']
y = sd_electricity['Usage']
label = 'Electricity Usage (GWh)'
fig, ax = plt.subplots()
ax.plot(x, y, label=label, c='royalblue')

ax.set_xlabel('Year', fontsize=10, fontweight='semibold')
ax.set_ylabel('Electricity Usage (GWh)', fontsize=10,
              fontweight='semibold')
ax.set_title('San Diego Residential Electricity Usage\n(1990-2014
             font_param, c='b')
ax.grid(color='lightgray', linestyle='--', linewidth='0.5')
plt.xlim(1990, 2015)
plt.legend()
plt.tight_layout()
```



Stacked Area Graph

- **Stacked Area Graph** Displays the contribution of each data series to a cumulative total over time.

Functions for plotting stacked area graph :

1. ***Axes.stackplot(axes, x, *args, labels=(), colors=None, baseline='zero', data=None, **kwargs) :***

Draw a stacked area plot. For details, click here.

Parameters:

x, y :	**x** : 1d array of dimension N; **y**: 2d array (MxN) or sequence of 1d arrays(each dimension 1xN)
baseline :	Calculate the baseline; {'zero', 'sym', 'wiggle', 'weighted_wiggle'}.
labels :	Length N sequence of strings. Labels to assign to each data series.
colors :	Length N sequence of colors; a list or tuple of colors; used to color the stacked areas.

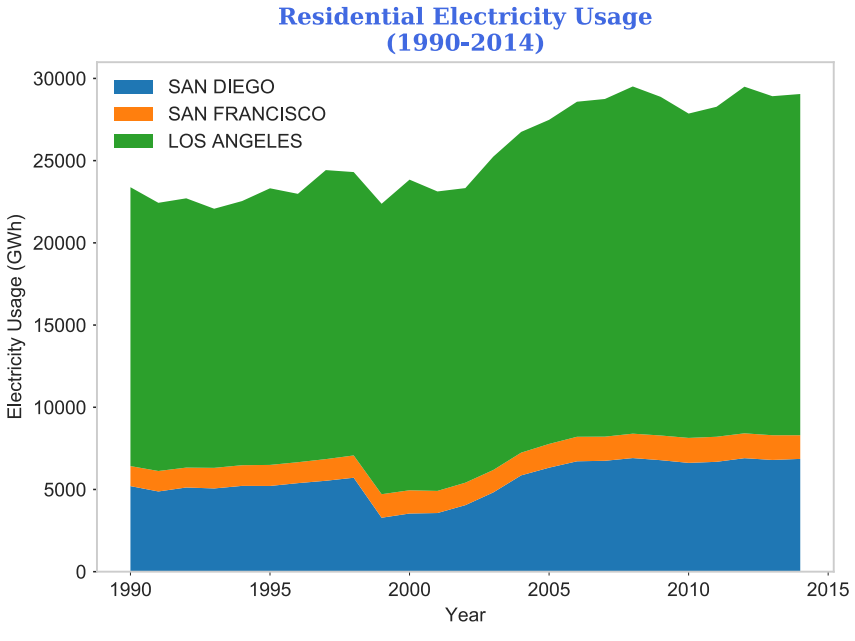
2. ***Axes.tick_params(axis='x', direction='out', length=3, width=0.5, labelrotation=75.00)*****:**

Decorate the appearance of ticks, ticklabels, and gridlines. For more details, click here.

Stacked Area Graph: Example

Example : Stacked area graph of residential electricity usage of California County (1990-2014). Source: California Electricity Consumption Database. Data were collected from data.ca.gov. All Usage Expressed in Millions of kWh (GWh). See full code on github-areagraph.

```
# Plot Stacked Area Graph
x = sd_electricity_usage['Year']
y = [sd_electricity_usage['Usage'],
     sf_electricity_usage['Usage'],
     la_electricity_usage['Usage']]
labels = ['SAN DIEGO', 'SAN FRANCISCO', 'LOS ANGELES']

fig, ax = plt.subplots()
ax.stackplot(x, y, labels=labels)
ax.legend(loc='upper left')

ax.set_xlabel('Year', fontsize=10)
ax.set_ylabel('Electricity Usage (GWh)', fontsize=10)
ax.set_title('Residential Electricity Usage\n(1990-2014)
             font_param, c='royalblue')

ax.tick_params(axis='x', direction='out', length=2,
               width=0.5, )
ax.tick_params(axis='y', direction='out', length=2)
```



Pie Chart

- **Pie Chart** is a circular statistical graphic, which is divided into slices to show part-to-whole relationships with continuous or discrete data.
- It is best used with a small data.

Functions for plotting Pie Chart :

1. ***Axes.pie(self, x, autopct=None, textprops=None, *args)*** **:** Make a Pie chart of array x. For details, click here.

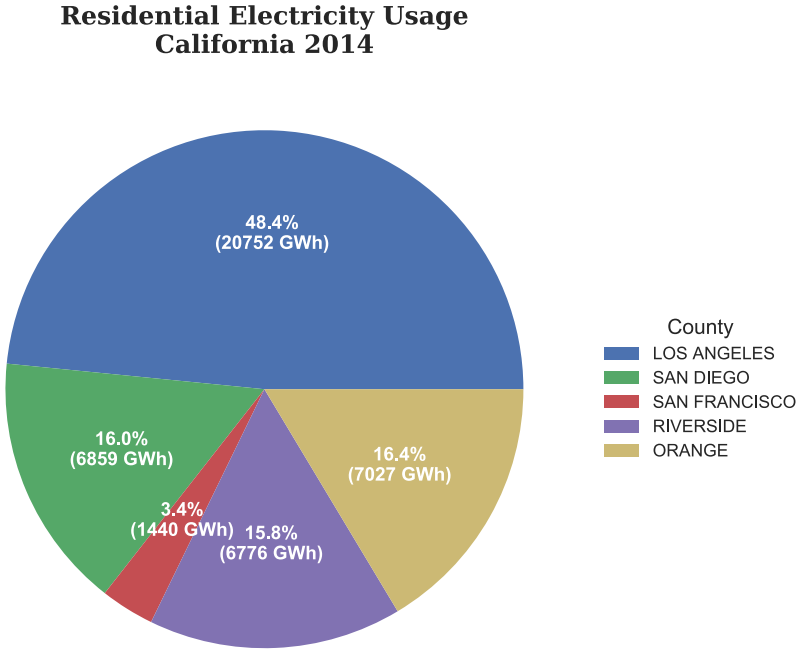
Parameters:	**x :**	array-like; the wedge sizes.
	autopct :	None (default), string, or function, optional. If not None, is a string or function used to label the wedges with their numeric value.
	textprops :	dict, optional; default: None. Dict of arguments to pass to the text objects.
Other parameters		explode, labels, colors, pctdistance, shadow, labeldistance, radius, counterclock, wedgeprop, etc.

2. ***plt.setps(obj, *args, **kwargs)*** **:** set the property on an artist object. For more details, click here.

Pie Chart : Example

Example : Pie Chart of total residential electricity usage of California Counties (1990-2015). Source: California Electricity Consumption Database. Data were collected from data.ca.gov. All Usage Expressed in Millions of kWh (GWh). See full code on github-piechart.

```
x = residential_electricity_by_county['2014']
# Plot Pie Chart
fig, ax = plt.subplots(figsize=(10, 5), subplot_kw=dict(aspect="equal"))


def func(pct, allvals):
    absolute = int(pct/100.*np.sum(allvals))
    return "{:.1f}%\n({:d} GWh)".format(pct, absolute)


auto_pct=lambda pct: func(pct, residential_electricity_by_county['2014'])
text_props=dict(color="w")

wedges, texts, autotexts = ax.pie(x, autopct= auto_pct,
                                  textprops=text_props)

ax.legend(wedges, residential_electricity_by_county['County'],
          title="County",
          fontsize="small",
          loc="center left",
          bbox_to_anchor=(1, 0, 0.5, 1))

plt.setp(autotexts, size=9, weight="bold")

ax.set_title('Residential Electricity Usage\nCalifornia 2014', font_param)
plt.tight_layout()
```



Histogram

- **Histogram** displays the underlying frequency distribution of a set of continuous data (univariate data).

Functions for plotting Histogram:

1. A*xes.hist(self, x, bins=None, range=None, density=None, histtype='bar', *args, **kwargs)* :

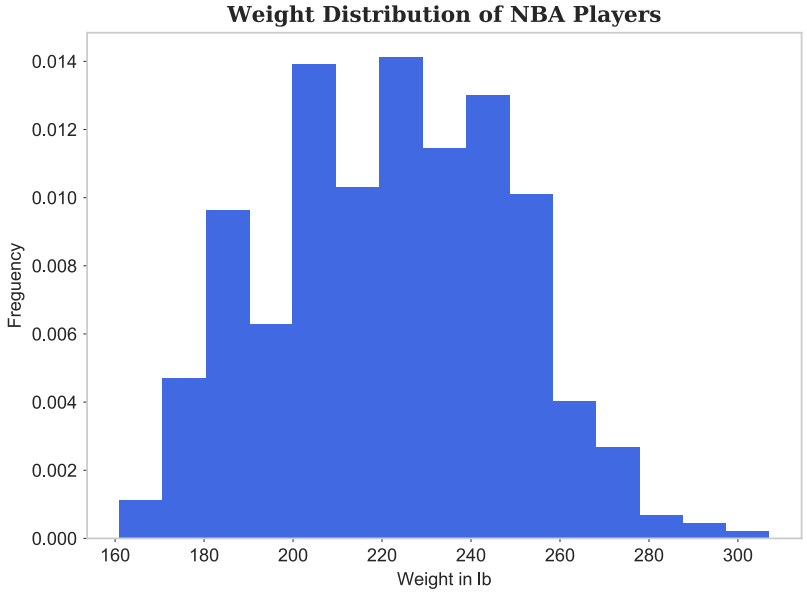
Plot a histogram. For details, click here.

Parameters:	**x :**	(n,) array or sequence of (n,) arrays. Input values, this takes either a single array or a sequence of arrays which are not required to be of same length.
	bins :	int or sequence of str, optional. If an integer is given, bins + 1 bin edges are calculated and returned. If bins is a sequence, gives bin edges, including left edge of first bin and right edge of last bin.
	density :	Bool, optional; if True, the area under histogram will sum to 1.
Other parameters:		weights, cumulative, bottom, histtype, align, orientation, rwidth, log, label, stack, normed, data, **kwargs. Details here.

Histogram : Example

Example : Histogram of NBA player's weight. See full code on github-histogram.

```
# Load nba data from file into pandas DataFrame
nba = pd.read_csv('processed_nba.csv',)
x = nba['Weight']
bins = 15

# Plot histogram
fig, ax = plt.subplots()
ax.hist(x, bins, density=1, color='royalblue')

ax.set_title('Weight Distribution of NBA Players', font
ax.set_xlabel('Weight in lb')
ax.set_ylabel('Freguency')
ax.tick_params(axis='both', direction='out',
               length=2, width=0.5, )
plt.grid(False)
fig.tight_layout()
```



Density Plot

- **Density Plot** displays the univariate distribution of data.
- Uses a kernel density estimate to show the probability density function (PDF) of the variable.

Functions for plotting density plot:

1. *seaborn.kdeplot(data, data2=None, shade=False, vertical=False, kernel='gau', ...)* :

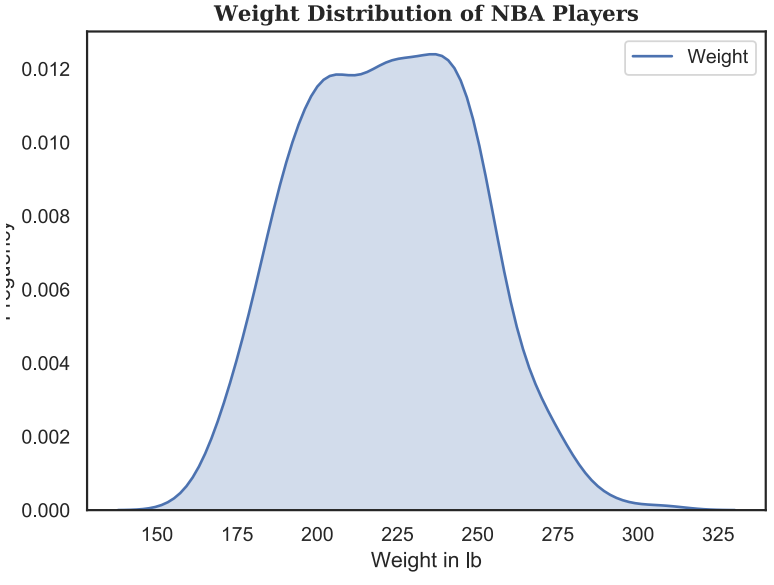
Fit and plot a univariate or bivariate kernel density estimate. For details, click here.

Parameters:	**data :**	1d array-like; input data.
	data2 :	1d array-like, optional. Second input data; if present a bivariate kde will be estimated.
	shade :	Bool, optional; if True, shade in the area under the kde curve.
	vertical :	Bool, optional; if True, density is on x-axis.
	kernel :	{'gau' | 'cos' | 'biw' | 'epa' | 'tri' | 'triw'}, optional. Code for shape of kernel to fit with. Bivariate kde can only use Gaussian kernel.

Density Plot : Example

Example : Density plot of NBA player's weight using seaborn kdeplot() function. See full code on github-seaborn-kdeplot.

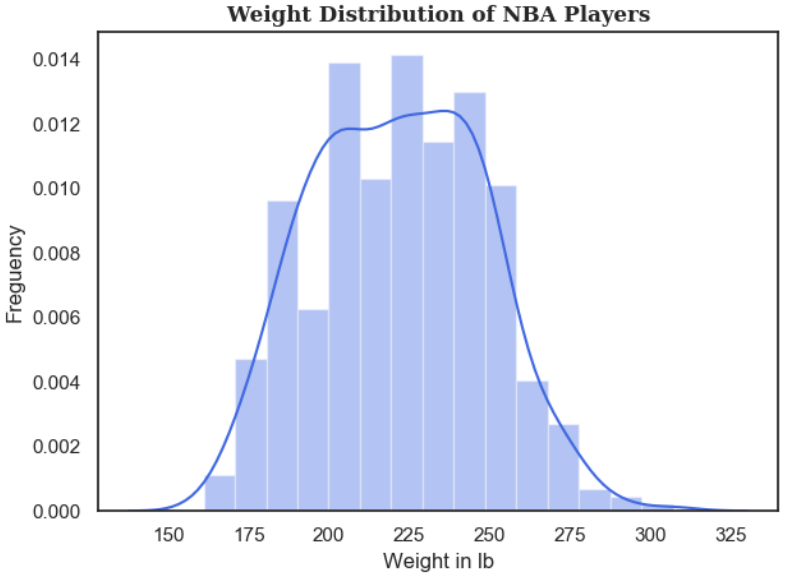
```
sns.set(style="white", palette="deep", color_codes=True)

# Load nba data from file into pandas DataFrame
nba = pd.read_csv('processed_nba.csv')
x = nba['Weight']

kde = sns.kdeplot(x, cbar=True, shade=True)
kde.set_title('Weight Distribution of NBA Players',
              font_param)
kde.set_xlabel('Weight in lb')
kde.set_ylabel('Freguency')

plt.tight_layout()

# Save and show figure
fig = kde.get_figure()
fig.savefig('hdr_nba_weigth.pdf')
plt.show()
```



Density Plot with Histogram : Example

Example : NBA player's weight using seaborn distplot() function. See full code on github-distplot.

```
# Load nba data from file into pandas DataFrame
nba = pd.read_csv('processed_nba.csv')
a = nba['Weight']


# Plot histogram and kernel density estimation
hd = sns.distplot(a, bins=15, color='royalblue')
hd.set_title('Weight Distribution of NBA Players', font_pa
hd.set_xlabel('Weight in lb')
hd.set_ylabel('Freguency')

plt.tight_layout()

# Save and show figure
fig = hd.get_figure()
fig.savefig('hd_nba_weigth.png')
plt.show()
```



Scatter Plot

- **Scatter Plot** shows correlation between two sets of data (bivariate data).
- Scatter plots are best used for large dataset.

Functions for plotting Scatter plot:

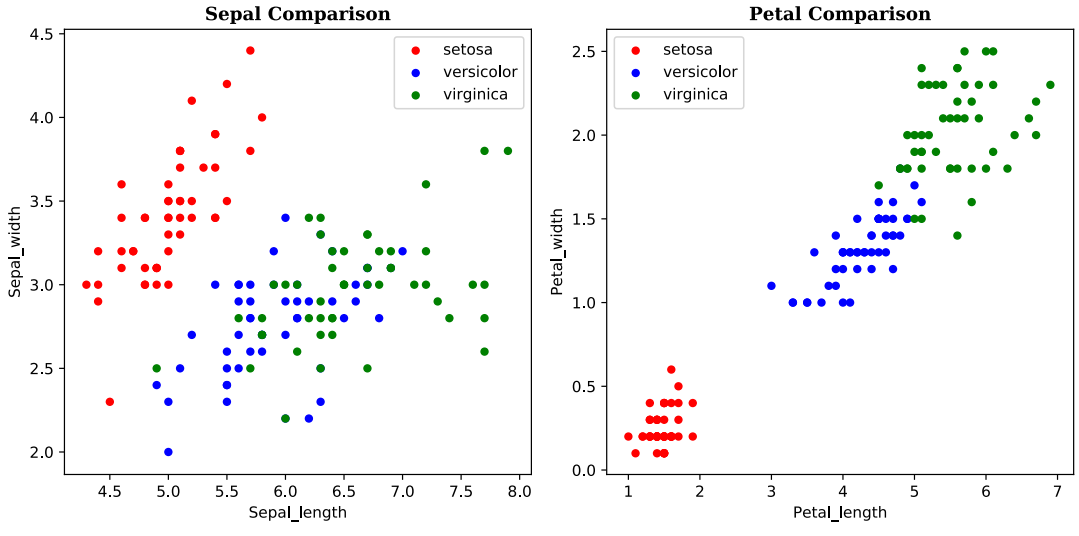
1. ***DataFrame.plot.scatter(self, x, y, s=None, c=None, **kwargs)*** **:** Create a scatter plot with varying marker point size and color. Returns matplotlib.axes.Axes or numpy.ndarray of them. For details, click here.

Parameters:	**x :**	int or str; the column name or column position to be used as horizontal coordinates for each point.
	y :	int or str; the column name or column position to be used as vertical coordinates for each point.
	s :	Scalar or array-like, optional; the size of each point.
	c :	Scalar or array-like, optional; the color of each point.
	***kwargs :**	Keyword arguments to pass on to DataFrame.plot().

Scatter Plot : Example

Example : Scatter plot using iris dataset. In the given example, the scatter plot shows petal and sepal distribution for each species. Data were collected from uci-machine-learning-repository. See full code on github-scatter.

```
# For each Species ,let's check what is petal and sepal distribution
fig, ax = plt.subplots(1,2,figsize=(10, 5))
setosa.plot(x="Sepal_length", y="Sepal_width",
            kind="scatter", ax=ax[0],
            label='setosa', color='r')
versicolor.plot(x="Sepal_length", y="Sepal_width",
                kind="scatter",ax=ax[0],
                label='versicolor', color='b')
virginica.plot(x="Sepal_length", y="Sepal_width",
               kind="scatter", ax=ax[0],
               label='virginica', color='g')

setosa.plot(x="Petal_length", y="Petal_width",
            kind="scatter",ax=ax[1],
            label='setosa',color='r')
versicolor.plot(x="Petal_length", y="Petal_width",
                kind="scatter", ax=ax[1],
                label='versicolor', color='b')
virginica.plot(x="Petal_length", y="Petal_width",
               kind="scatter", ax=ax[1],
               label='virginica', color='g')

ax[0].set_title('Sepal Comparison', font_param)
ax[1].set_title('Petal Comparison', font_param)
ax[0].legend()
ax[1].legend()
```



Boxplot

- **Box Plot** displays the distribution of data through their quartiles (minimum, first quartile(Q1), median, third quartile(Q3), and maximum).
- Displays outliers with their values.

Functions for plotting Boxplot:

1. *Axes.boxplot(self, x, notch=None, patch_artist=None, labels=None, flierprops=None, …)* :

Makes a box and whisker plot for each column of x or each vector in sequence x. Returns matplotlib.axes.Axes or numpy.ndarray of them. For details, click here.

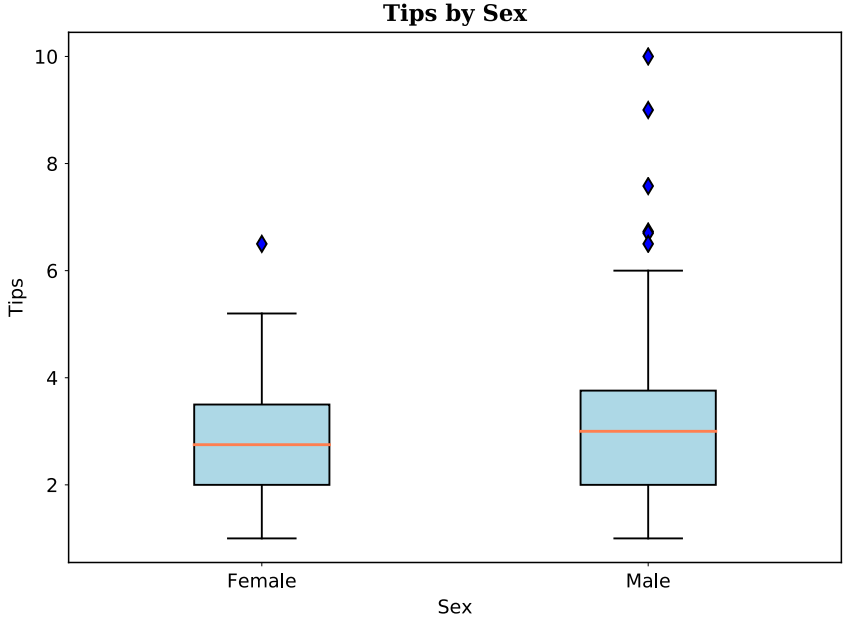
Parameters:	**x :**	Array or a sequence of vectors; the input data.
	notch :	bool, optional (False). If True, will produce a notched box plot. Otherwise, a rectangular boxplot is produced.
	labels :	Sequence, optional. Labels for each dataset. Length must be compatible with dimensions of x.
	patch_artist :	bool, optional (False). If True, produces boxes with the Line2D artist. Otherwise, boxes and drawn with patch artist.
	flierprops :	dict, optional (None). Specifies the style of the fliers.

Boxplot : Example

Example : Boxplot using Tips data. In the given example, the boxplot shows Tips by Sex. Data were collected from GitHub.

See full code on github-boxplot.

```
# Plot data
x = [tips[tips['sex'] == 'Female']['tip'],
           tips[tips['sex'] == 'Male']['tip']]
label = ['Female', 'Male']
box_props = dict(facecolor="lightblue")
median_props = dict(color="coral", lw=1.5)
flier_props = dict(markerfacecolor='b', marker='d')

fig1, ax = plt.subplots()
ax.boxplot(x, labels=label, flierprops=flier_props,
           widths=0.35, patch_artist=True,
           boxprops=box_props,
           medianprops=median_props)


ax.set_xlabel('Sex', fontsize=10)
ax.set_ylabel('Tips', fontsize=10)
ax.tick_params(axis='both', direction='out',
               length=2, width=0.5, )
ax.set_title('Tips by Sex', font_param)
```



Violin Plot

- **Violin Plot** plots numeric data with a rotated kernel density plot on each side.
- Shows the probability density of the data at different values, usually smoothed by a kernel density estimator.

Functions for plotting Boxplot:

1. *seaborn.violinplot(x=None, y=None, hue=None, data=None, order=None, hue_order=None…)* :

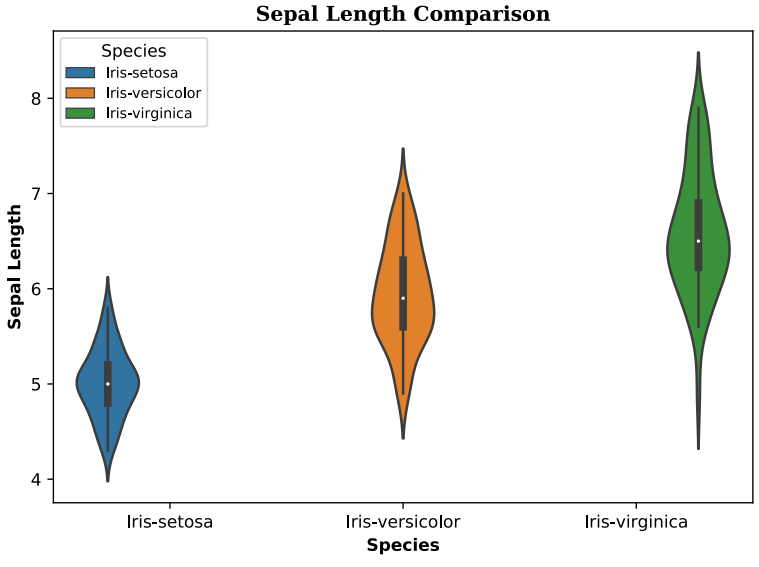
Draws a combination of boxplot and kernel density estimate. For details, click here.

Parameters:	**x, y, hue :**	Names of variables in data or vector data; optional.
	data :	DataFrame, array, or list of arrays; optional. Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise, it is expected to be long-form.
	order, hue_order :	Lists of strings; optional. Order to plot the categorical levels in, otherwise the levels are inferred from the data objects.

Violin Plot : Example

Example : Violin plot using iris dataset. In the given example, the violin plot shows sepal length comparison for each species. Data were collected from uci-machine-learning-repository. See full code on github-seaborn-violinplot.

```
# Load Data
col_names = ['Sepal_length', 'Sepal_width', 'Petal_length', 'Petal_width', 'Species']
iris = pd.read_csv('iris.csv', names=col_names)

x = iris['Species']
y = iris['Sepal_length']
hue = iris['Species']
data = iris

# Draw violinplot to show sepal length comparison
sepal_length = sns.violinplot(x, y, data=data, hue=hue)
sepal_length.set_title("Sepal Length Comparison", font_param)
sepal_length.set_xlabel("Species", size=10, fontweight='semibold')
sepal_length.set_ylabel("Sepal Length", size=10, fontweight='semibold')


plt.tight_layout()
fig = sepal_length.get_figure()
fig.savefig('iris_sepal_comparison.pdf')
plt.show()
```



Heatmap

- **Heatmap** represents data where individual values contained in a matrix are represented as colors.

Functions for plotting Boxplot:

1. ***heatmap(data, row_labes, col_labels, ax=None, cbar_kw={}, cbarlabel="", **kwargs)*** **:**

Create a heatmap from a numpy array and two lists of labels . For details, click here.

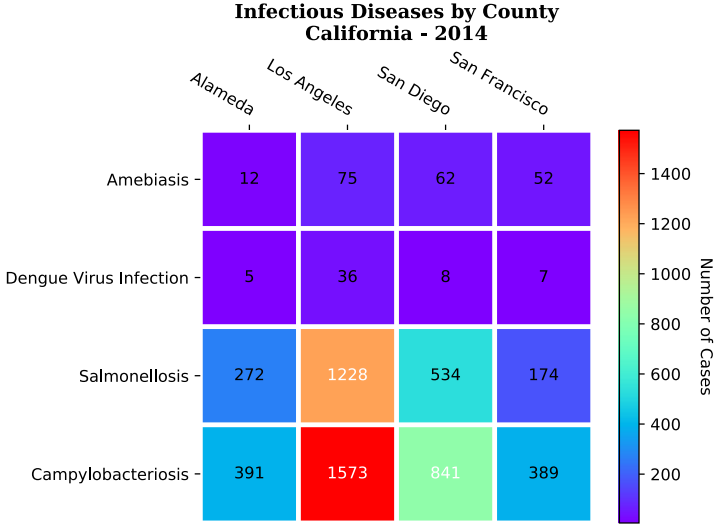
Parameters:	**data :**	A 2D numpy array of shape (N, M).
	row_labels :	A list or array of length N with the labels for the rows.
	col_labels :	A list or array of length M with the labels for the columns.
	ax :	A 'matplotlib.axes.Axes' instance to which the heatmap is plotted; optional.
	cbar_kw :	A dictionary with arguments to matplotlib.Figure.colorbar'; optional.
	cbarlabel :	The label for the colorbar; optional.

Heatmap : Example

Example : Shows heatmap of infectious disease by California county (2014). Data were collected from data-chhs-ca-gov.

See full code on github-heatmap.

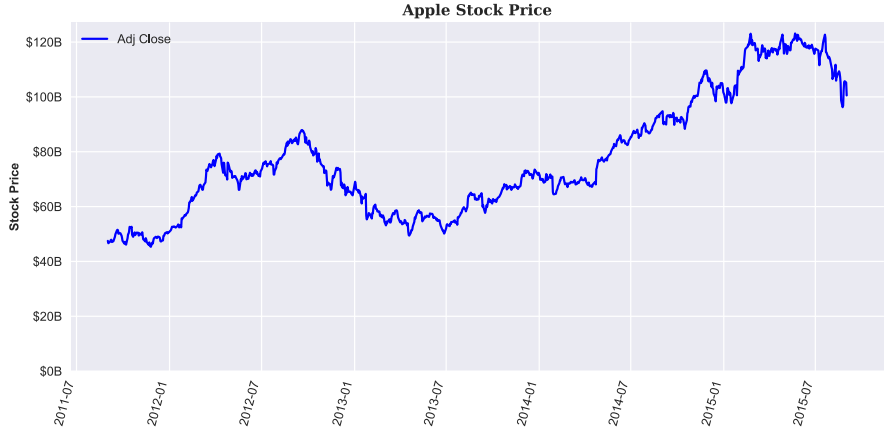
```
# Plot data
data = cases
row_labels = disease
col_labels = county
fig, ax = plt.subplots()

im, cbar = Heatmap.heatmap(data, row_labels, col_labels, ax=ax,
                           cmap="rainbow", cbarlabel="Number of Cases")
texts = Heatmap.annotate_heatmap(im, valfmt="{x:.0f}")

ax.set_title("Infectious Diseases by County\nCalifornia - 2014",
plt.tight_layout()
plt.savefig('heatmap_disease.pdf')
```



Time Series

- **Time Series** displays series of data points indexed in time order.
- **Example :** Draws time series using Apple stock price data. Data were collected from yahoo-finance. See full code on github-timeseries.

```
# Generate the plot
x = apple['Date']
y = apple['Adj Close']
label = 'Adj Close'


def billions(x, pos):
    """The two args are the value and tick position"""
    return '$%1.0fB' % (x*1)


formatter = FuncFormatter(billions)

fig, ax = plt.subplots(figsize=(10, 5))
ax.yaxis.set_major_formatter(formatter)
ax.plot(x, y, label=label, color='b')
plt.xticks(horizontalalignment='right', rotation=75)
plt.ylabel('Stock Price', size=10, fontweight='semibold')
plt.title('Apple Stock Price', font_param)
plt.ylim(ymin=0)
plt.legend(loc='upper left', fancybox=True)
plt.tight_layout()
```



Summary

Plot Types	API Call
Bar Chart, Stacked Bar Chart, Grouped Bar Chart	matplotlib.pyplot.bar(x, height, width, bottom=None, align='center', data=None, …)
Horizontal Bar Chart	matplotlib.axes.Axes.barh(self, y, width, height=0.8, left=None, align='center', …)
Line Chart	matplotlib.axes.Axes.plot(self, *args, scalex=True, scaley=True, data=None, **kwargs)
Stacked Area Graph	matplotlib.axes.Axes.stackplot(axes, x, *args, labels=(), colors=None, baseline='zero',…)
Pie Chart	matplotlib.axes.Axes.pie(self, x, autopct=None, textprops=None, *args)
Histogram	matplotlib.axes.Axes.hist(self, x, bins=None, range=None, density=None, histtype='bar', …)
Density Plot	seaborn.kdeplot(data, data2=None, shade=False, vertical=False, kernel='gau', …)
Histogram with Kde Plot	seaborn.distplot(a, bins=None, hist=True, kde=True, rug=False, hist_kws=None, …)
Scatter Plot	DataFrame.plot.scatter(self, x, y, s=None, c=None, **kwargs)
Box Plot	matplotlib.axes.Axes.boxplot(self, x, notch=None, patch_artist=None, labels=None, …)
Violin Plot	seaborn.violinplot(x=None, y=None, hue=None, data=None, order=None, hue_order=None…)
Heatmap	heatmap(data, row_labes, col_labels, ax=None, cbar_kw={}, cbarlabel="", **kwargs)

Version 2017.06

Visualization Plots Using Seaborn

Visualization Plots Using Seaborn

❖ Visualization Plot Types

- **Bar Plot**
- **Strip Plot**
- **Swarm Plot**
- **Boxplot**
- **Boxen Plot**
- **Density Plot with filled area**
- **Histogram**
- **Histogram and Rug Plot**
- **Density Plot with Rug Plot**
- **Histogram, Density and Rug Plot**
- **lmplot**
- **Regplot**
- **2D Kde Plot**
- **Joint Plot**
- **Pair Plot**

Barplot using catplot()

- ***seaborn.catplot(x=None, y=None, hue=None, data=None, kind='bar', …)*** **:**

  Figure-level interface for drawing categorical plots onto a FacetGrid. This function provides access to several axes-level functions that show the relationship between a numerical and one or more categorical variables using one of several visual representations. For more details, click here.

- The *kind* parameter in function catplot() selects the underlying functions to plot:

  Categorical scatterplots: kind='strip' for stripplot(); kind='swarm' for swarmplot()

  Categorical distribution plots: kind='box' for boxplot(); kind='violin' for violinplot(); kind='boxen' for boxenplot()

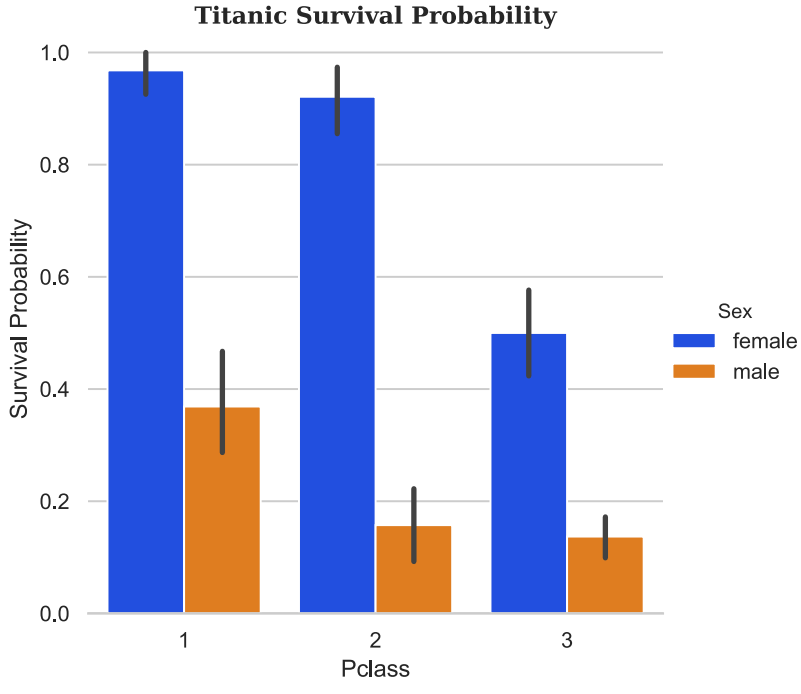
  Categorical estimate plots: kind='point' for pointplot(); kind='bar' for barplot(); kind='count' for countplot

Parameters:	**x, y, hue:**	Name of variables in data; inputs for plotting long-form data.
	data :	DataFrame; long-form (tidy) dataset.
	kind :	String, optional; the kind of plot to draw (bar, strip, swarm, box, violin, boxen).

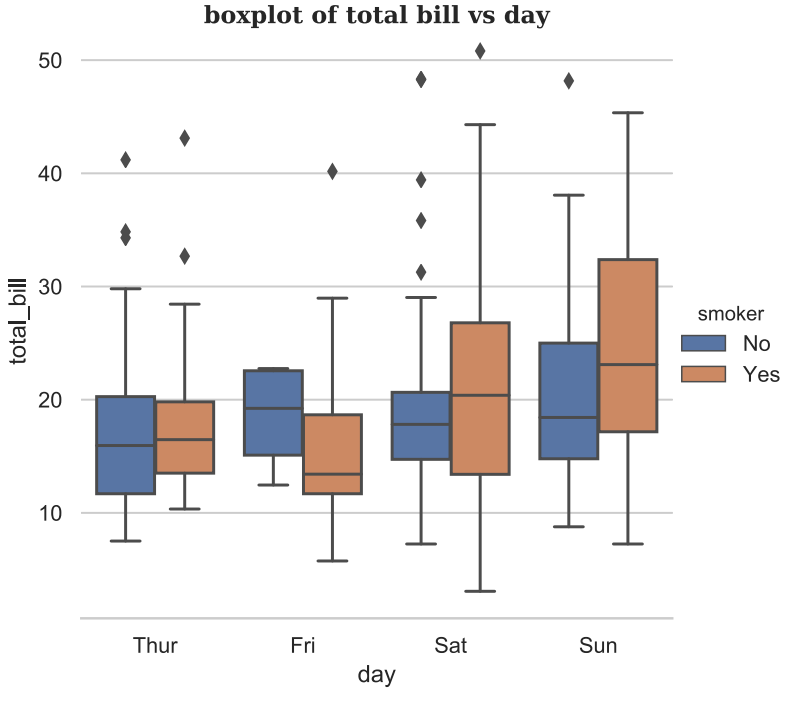
Barplot using catplot(): Example

Example : Displays Titanic survival probability for class and sex. See full code on [github-seaborn-catplot](#).

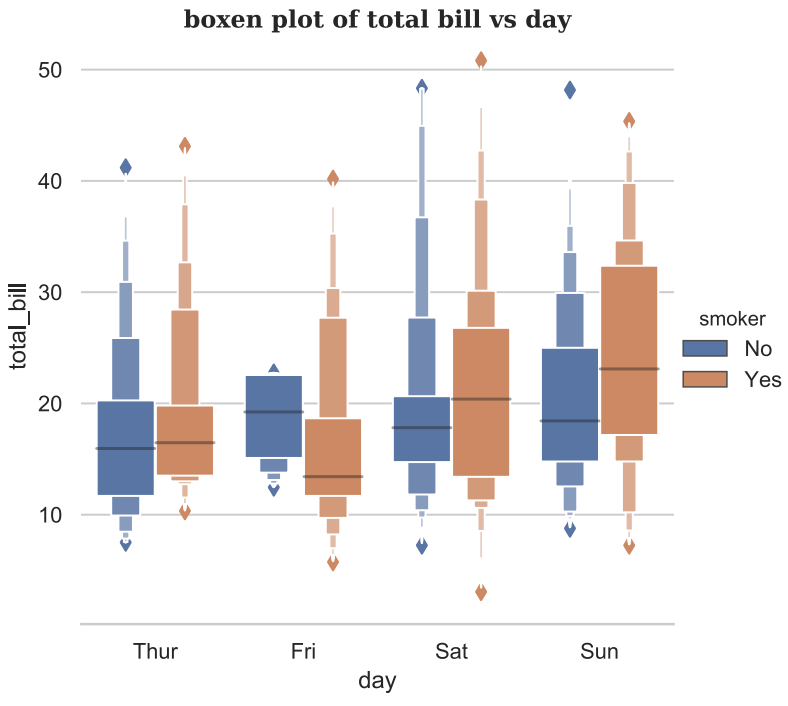
```
# Draw a barplot to show survival for class and sex
x = "Pclass"
y = "Survived"
hue = "Sex"
data = titanic

g = sns.catplot(x, y, hue, data,
                height=5, kind="bar",
                aspect=1, palette="bright",
                hue_order=['female', 'male'])
g.despine(left=True)
g.set_ylabels("Survival Probability")
plt.title('Titanic Survival Probability',
          font_param, pad=0.4)

g.savefig('bar_titanic.pdf')
plt.show()
```



Boxplot and Boxen plot using catplot()

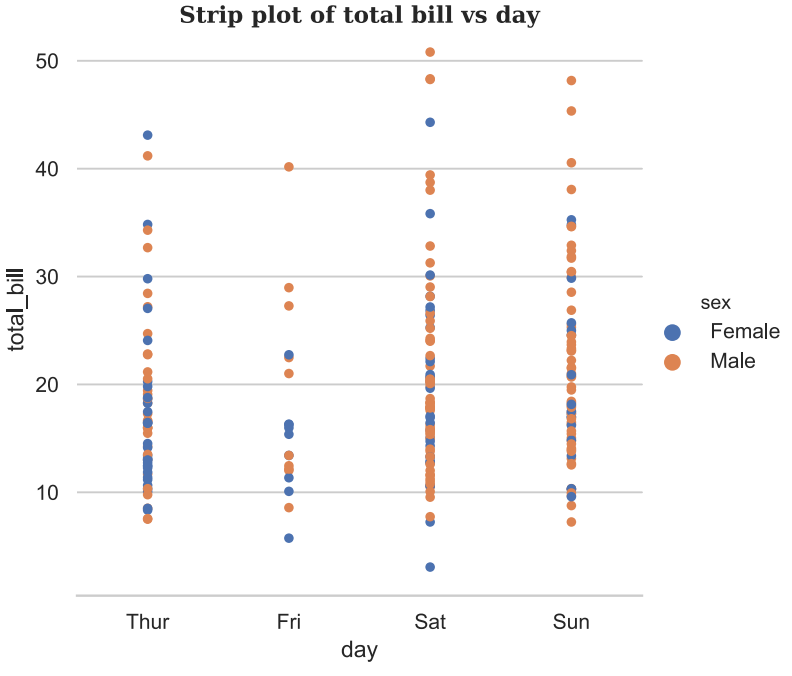


kind='box'

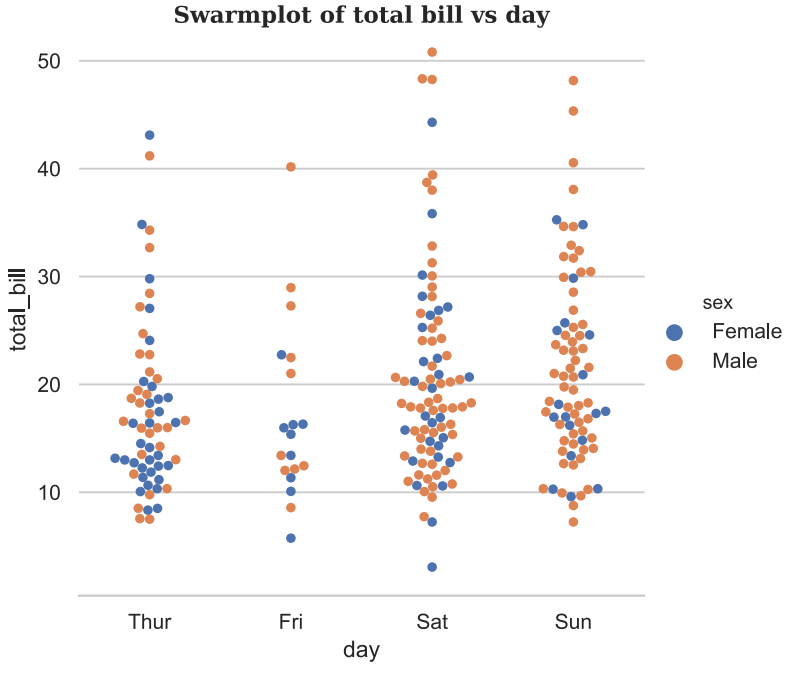


kind='boxen'

Strip plot and Swarm plot using catplot()



kind='strip'



kind='swarm'

Histogram, Density Plot, Rug Plot using distplot()

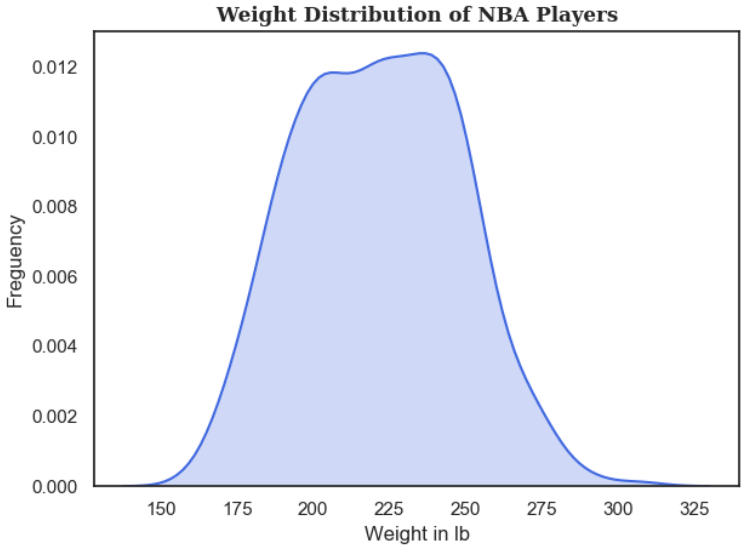
- ***seaborn.distplot(a, bins=None, hist=True, kde=True, rug=False, hist_kws=None, kde_kws, rug_kws=None, …)*** **:** Combines the matplotlib hist() function, with the seaborn kdeplot() and rugplot() functions and plot the estimate PDF over the data. For details, click here.

Parameters:	**a :**	Series, 1d-array, or list.
	bins :	Argument for matplotlib hist(), or None, optional; specification for hist bins.
	hist :	Bool, optional; whether to plot a (normed) histogram
	kde :	Bool, optional; whether to plot a Gaussian kernel density estimate.
	rug :	Bool, optional; whether to plot a rugplot on the support axis.
{hist, kde, rug, fit}_kws :		Dictionaries, optional; keyword arguments for underlying plotting functions.

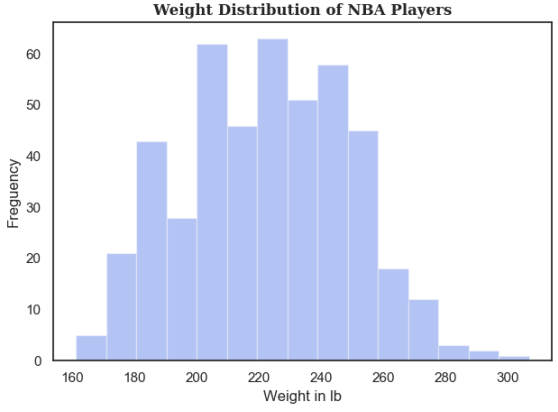
Density Plot using distplot()

Example : Density plot of NBA player's weight using seaborn distplot() function. See full code on [github-seaborn-distplot](#).

```
# Load nba data from file into pandas DataFrame
nba = pd.read_csv('processed_nba.csv')
a = nba['Weight']


# plot kernel density with filled area
kde = sns.distplot(a, hist=False, color='royalblue',
                   kde_kws={"shade": True})
kde.set_title('Weight Distribution of NBA Players', font_param)
kde.set_xlabel('Weight in lb')
kde.set_ylabel('Freguency')

plt.tight_layout()

# Save and show figure
fig = kde.get_figure()
fig.savefig('kde_filled_nba_weigth.png')
plt.show()
```

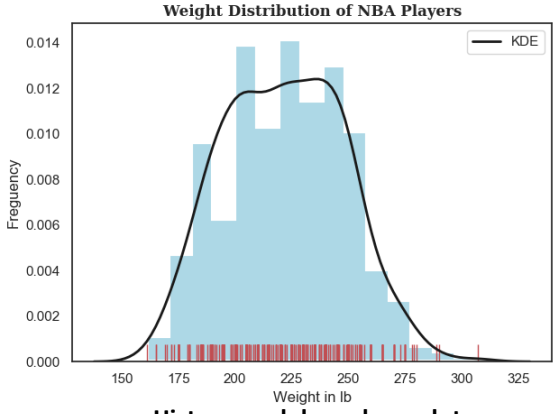


distplot(): Example



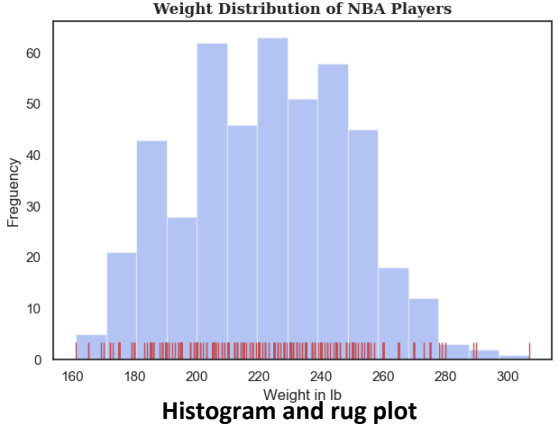
Histogram

sns.distplot(a, bins=15, kde=False,…)



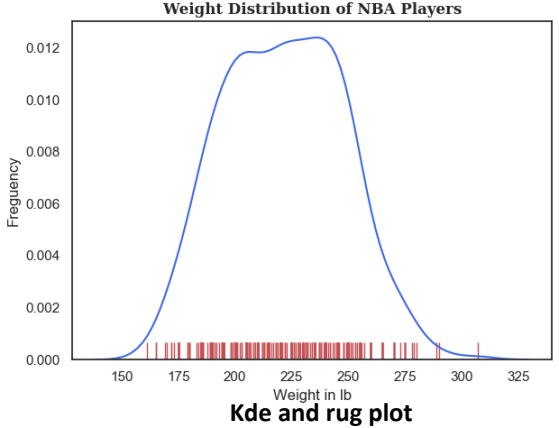
Histogram, kde and rug plot

sns.distplot(a, rug=True,…)



Histogram and rug plot

sns.distplot(a, bins=15, kde=False, rug=True…)



Kde and rug plot

sns.distplot(a, bins=15, hist=False, rug=True…)

Implot()

- ***seaborn.lmplot(x, y, data, hue=None, col=None, …)*** **:**

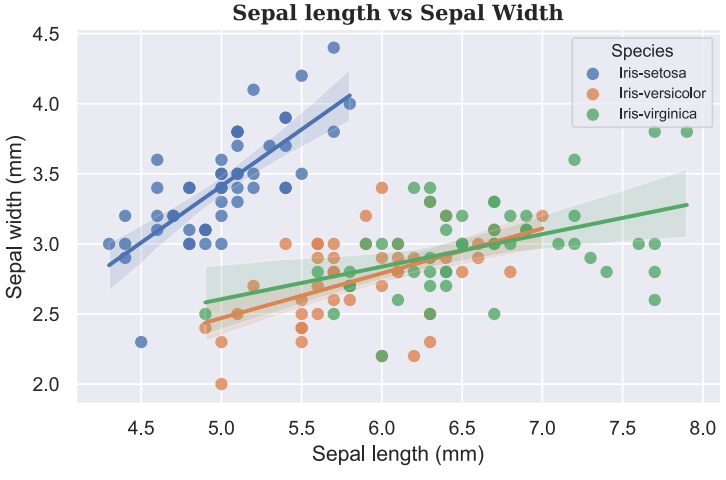
  Plots data and regression model fits across a FacetGrid. Combines regplot() and FacetGrid. For details, click here.

Parameters:	**x, y :**	Strings, optional
	data :	DataFrame; Tidy (long-form) dataframe where each column is a variable and each row is an observation.
	hue, col, row :	Strings; variables that define subsets of the data.

Implot(): Example

Example : Scatter plot with regression line showing sepal length and sepal width comparison of iris species. See full code on [github-seaborn-lmplot](#).

```
# Draw lmplot of iris sepal comparison
x = 'Sepal_length'
y = 'Sepal_width'
hue = 'Species'
data = iris

# Plot scatter plot and also fit a regression line
scatter = sns.lmplot(x, y, data, hue=hue,
                     truncate=True, height=4,
                     aspect=1.5, legend=False)
scatter.set_axis_labels("Sepal length (mm)", "Sepal width (mm)")
ax = plt.gca()
ax.set_title('Sepal length vs Sepal Width', font_param)

ax.legend(loc='upper right', fontsize='x-small',
          title='Species', title_fontsize=10,
          fancybox=True)
plt.tight_layout()

# Save and show figure
scatter.savefig('lmplot_iris.pdf')
plt.show()
```



regplot()

- ***seaborn.regplot(x, y, data=None, x_estimator=None, x_bins=None, fit_reg=True, …)* :**

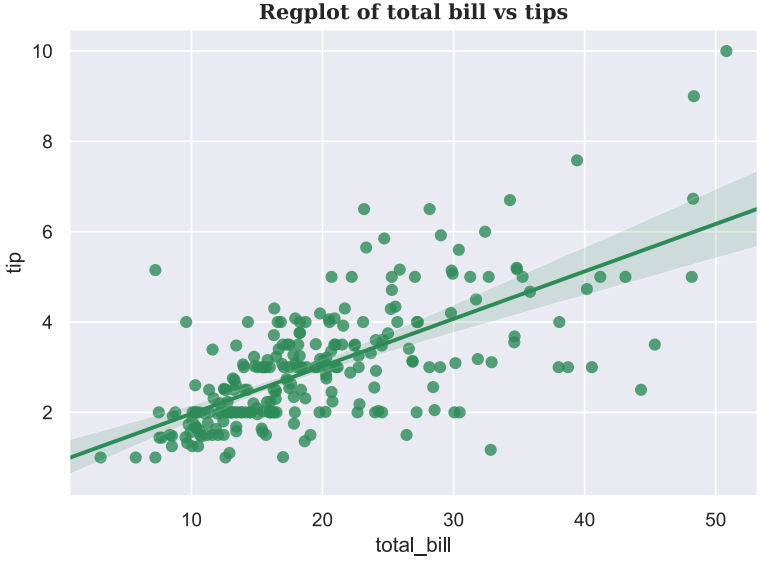
  Plot data and a linear regression model fit. For details, click here.

Parameters:	**x, y :**	String, series, or vector array; input variables.
	data :	DataFrame; Tidy (long-form) dataframe where each column is a variable and each row is an observation.
	x_estimator :	Called that maps vector -> scalar, optional.
	x_bins :	int or vector, optional.
	fit_reg :	Bool, optional. If True, estimate and plot a regression model relating the x and y variables.

regplot(): Example

Example : Displays regplot of total bill vs tips using tips dataset. See full code on [github-seaborn-regplot](#).

```
# Plot 2D kernel density plot
x = "total_bill"
y = "tip"
data = tips

# Plot scatter plot and also fit a regression line
scatter = sns.regplot(x, y, data=data, color="seagreen")
ax = plt.gca()
ax.set_title('Regplot of total bill vs tips', font_param)

plt.tight_layout()

# Save and show figure
plt.savefig('regplot_totalbill_tip.pdf')
plt.show()
```



2D Density Plot using kdeplot()

- ***seaborn.kdeplot(data, data2=None, shade=False, vertical=False, kernel='gau', ...)*** **:**

  Fit and plot a univariate or bivariate kernel density estimate. For details, click here.

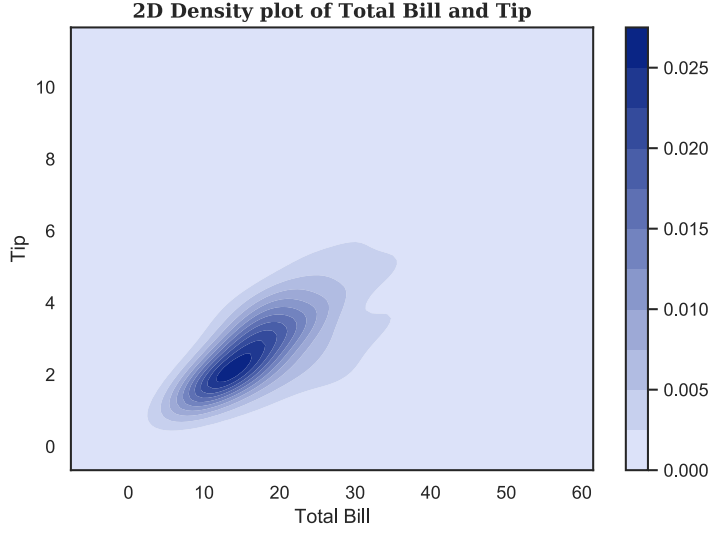
Parameters:

data :	1d array-like; input data.
data2 :	1d array-like, optional. Second input data; if present a bivariate kde will be estimated.
shade :	Bool, optional; if True, shade in the area under the kde curve.
vertical :	Bool, optional; if True, density is on x-axis.
kernel :	{'gau' | 'cos' | 'biw' | 'epa' | 'tri' | 'triw'}, optional. Code for shape of kernel to fit with. Bivariate kde can only use Gaussian kernel.

2D kdeplot(): Example

Example : 2D Density plot of total bill and tip from tips dataset using seaborn kdeplot() function. See full code on [github-seaborn-2dkdeplot](#).

```
sns.set(style="white", palette="dark", color_codes=True)

# Load tips data from file into pandas DataFrame
tips = pd.read_csv('tips.csv')

# Plot 2D kernel density plot
x = tips["total_bill"]
y = tips["tip"]
data = tips

# If cbar=True, draw a bivariate KDE plot, add a colorbar
# shade=True, shade in the area under the KDE curve
kde = sns.kdeplot(x, y, cbar=True, shade=True)
kde.set_title('2D Density plot of Total Bill and Tip',
              font_param)
kde.set_xlabel('Total Bill')
kde.set_ylabel('Tip')
```



jointplot()

- ***seaborn.jointplot(x, y, data=None, kind='scatter', stat_func=None, color=None, …)* :**
  Draw a plot of two variables with bivariate and univariate graphs. For details, click here.

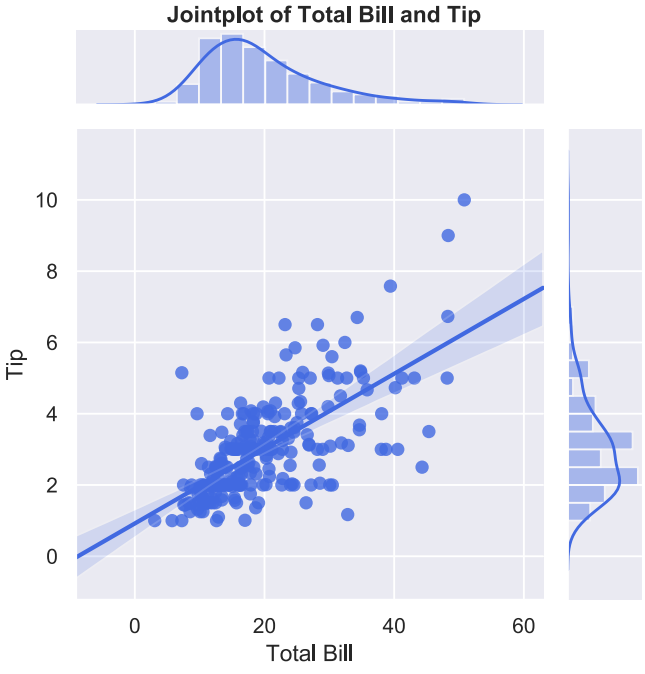
Parameters:

x, y :	Strings or vectors. Data or names of variables in data.
data :	DataFrame, optional. DataFrame when x and y are variable names.
kind :	{"scatter" | "reg" | "resid" | "kde" | "hex"}, optional. Kind of plot to draw.
stat_func :	Callable or None, optional. Deprecated.
color :	Matplotlib color, optional.

jointplot(): Example

Example : Show regplot with a univariate plot on each axis using jointplot() using tips dataset. See full code on github-seaborn-jointplot.

```
# Regplot with a univariate plot on each axis using jointplot()
x = tips["total_bill"]
y = tips["tip"]
data = tips

scatter = sns.jointplot(x, y, data=data, kind='reg',
                        height=5, ratio=5, space=4,
                        color='royalblue')
scatter.set_axis_labels("Total Bill", "Tip")

# add a title, set font size, and move the text above the total axes
scatter.fig.suptitle("Jointplot of Total Bill and Tip",
                     fontsize=12, fontweight='semibold', y=1)
plt.tight_layout()
scatter.savefig("jointplot_totalbill_tip.pdf")
plt.show()
```



pairplot()

- ***seaborn.pairplot(data, hue=None, hue_order=None, palette=None, vars=None, …)* :**

  Draw a plot of two variables with bivariate and univariate graphs. For details, click here.

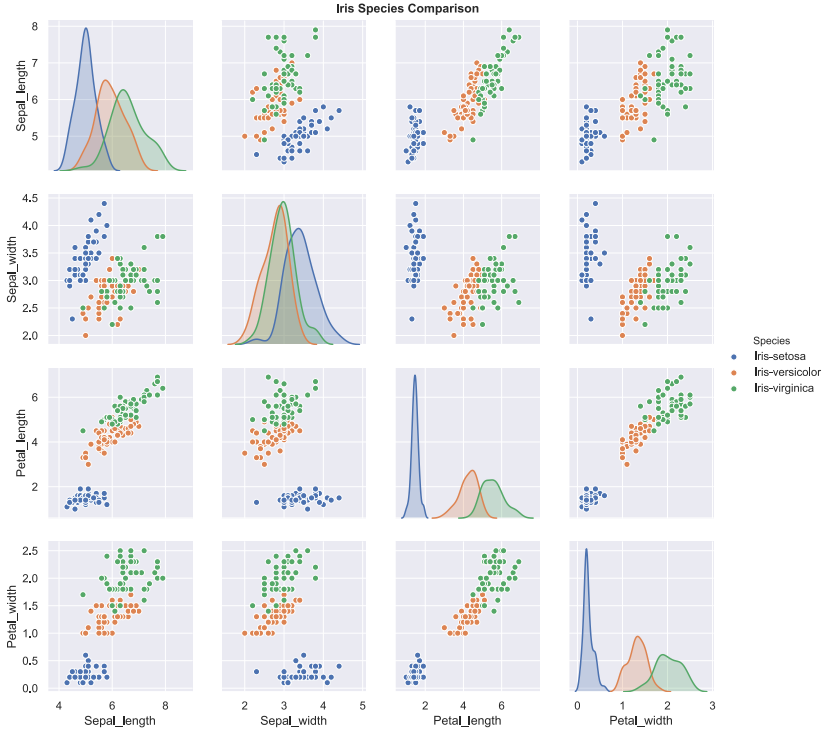
Parameters:

data :	DataFrame; Tidy (long-form) dataframe where each column is a variable and each row is an observation.
hue :	String (variable name), optional. Variable in data to map plot aspects to different colors.
hue_order :	List of strings. Order for the levels of the hue variable in the palette.
palette :	dict or seaborn color palette. Set of colors for mapping the hue variable. If a dict, keys should be values in the hue variable.
vars :	List of variable names, optional. Variables within data to use, otherwise use every column with a numeric datatype.

pairplot(): Example

Example : Displays pairplot using iris dataset. See full code on github-seaborn-pairplot.

```
sns.set(color_codes=True)

# Load tips data from file into pandas DataFrame
# Load Data
col_names = ['Sepal_length', 'Sepal_width', 'Petal_length',
             'Petal_width', 'Species']
iris = pd.read_csv('iris.csv', names=col_names)

# Draw pairwise plot
data = iris
hue = 'Species'
pair_plot= sns.pairplot(data, hue=hue)
pair_plot.fig.suptitle("Iris Species Comparison",
                       fontsize=12, fontweight='semibold',
                       y=1)

pair_plot.savefig("pairplot_iris.pdf")
plt.show()
```



Summary

Plot Types		API Call
Categorical data Visualization:	Bar plot Strip plot Swarm plot Count plot Boxplot Boxen plot Violin plot	seaborn.catplot(x=None, y=None, hue=None, data=None, kind='bar', …)
Distribution of data:	Histogram Density plot Rug plot	seaborn.distplot(a, bins=None, hist=True, kde=True, rug=False, hist_kws=None, kde_kws, rug_kws=None, …)
	Density plot	seaborn.kdeplot(data, data2=None, shade=False, vertical=False, kernel='gau', …)
	Pairplot	seaborn.pairplot(data, hue=None, hue_order=None, palette=None, vars=None, …)
	Jointplot	seaborn.jointplot(x, y, data=None, kind='scatter', stat_func=None, color=None, …)
Linear relationship	Scatter plot with regression line	seaborn.regplot(x, y, data=None, x_estimator=None, x_bins=None, fit_reg=True, …)
		seaborn.lmplot(x, y, data, hue=None, col=None, …)

Version 2018.03

Visualization Plots Using Plotly

Visualization Plots Using Plotly

- Plotly modules:
  - **Plotly.plotly** contains the functions that will communicate with the Plotly server.
  - **Plotly.graph_objects** contains the functions that will generate graph objects.
- Objects that define a plot in plotly:
  - Data : It contains all the traces to be plot. A **trace** object is the name of the given data and specifications of which that data are to be plotted.
  - Layout : The **layout** object defines the look of the plot, and plot features which are unrelated to the data.
  - Figure : The **figure** object **go.Figure** creates the final object to be plotted. It creates a dictionary-like object that contains both data and the layout object.
- Visualization Plot Types
  - **Time Series**
  - **Candlestick Chart**
  - **OHLC Chart**
  - **Waterfall Chart**
  - **Annotated Heatmap**
  - **Polar Chart**
  - **Radar Chart**
  - **3D Scatter Plot**

Time Series

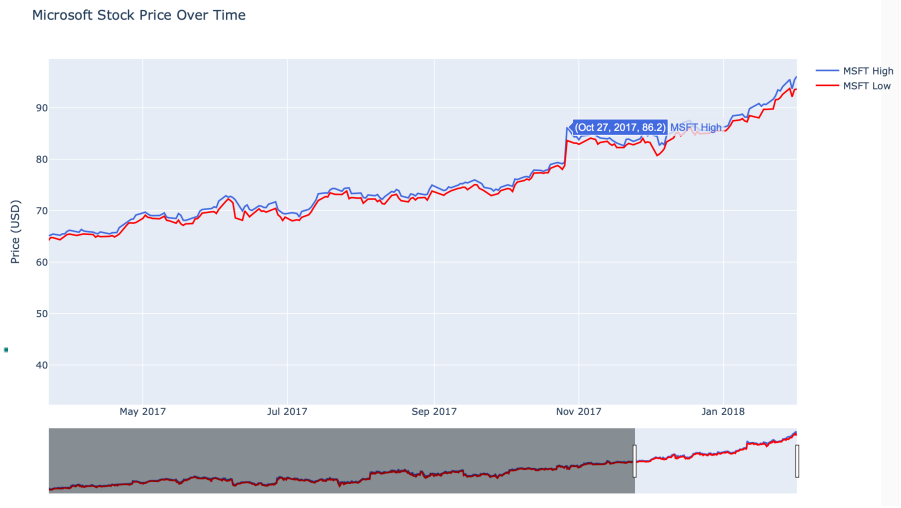
- **Time Series** can be represented using either plotly.express functions (px.line, px.scatter) or plotly.graph_objects charts objects (go.Scatter).
- Plotly auto-sets the axis type to a date format when the corresponding data are either ISO-formatted date strings or if they are a date pandas column or datetime NumPy array.
- ***plotly.graph_objects.Scatter(x, y, name, line_color, opacity, …)*:** The scatter trace type encompasses line charts, scatter charts, text charts, and bubble charts. The data visualization as scatter point or lines is set in 'x' and 'y'. Text appears either on the chart or on the hover only via 'text'. Bubble charts are achieved by setting 'marker.size' and/or 'marker.color' to numerical arrays. For details, click here.

Arguments or Attributes:	**x:**	Parent: data[type=scatter]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the x coordinates.
	y :	Parent: data[type=scatter]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the y coordinates.
	name :	Parent: data[type=scatter]; Type: string. Sets the trace name. The trace name appears as the legend item and on hover.
	line_color :	Parent: data[type=scatter].line; Type: color. Sets the line color.
	opacity :	Parent: data[type=scatter]; Type: number between or equal to 0 and 1. default: 1. Sets the opacity of the trace.

Time Series : Example

- **Example :** Draws time series using Microsoft stock price data over the time. Data were collected from yahoo-finance. See full code on github-plotly-timeseries.

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df.Date, y=df['High'],
                         name="MSFT High",
                         line_color='royalblue'))

fig.add_trace(go.Scatter(x=df.Date, y=df['Low'],
                         name="MSFT Low",
                         line_color='red'))

fig.update_layout(title_text='Microsoft Stock Price Over
                  yaxis_title='Price (USD)',
                  xaxis_rangeslider_visible=True)
```



- Click here to see figure

Candlestick Chart

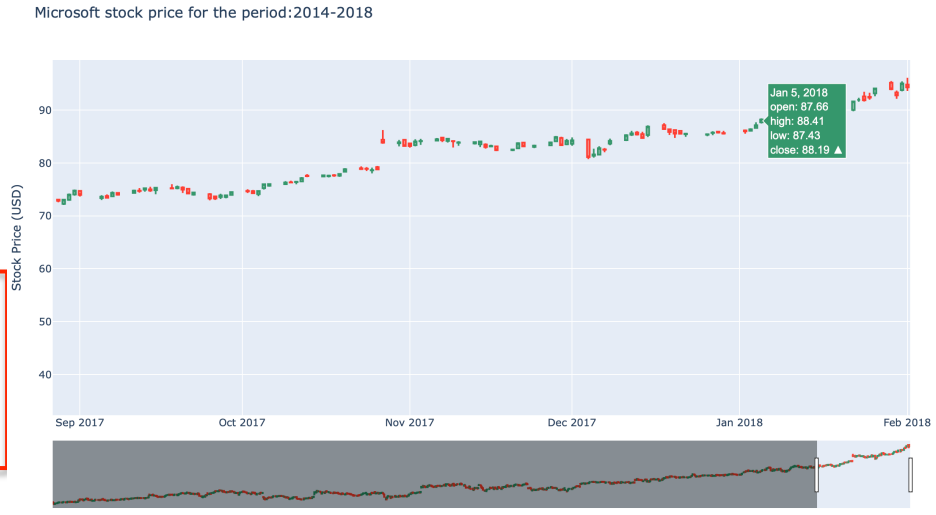
- The **Candlestick chart** is a financial chart showing open, high, low, and close for a given x coordinate (time).
- Boxes show the spread between the open and close values and the lines show the spread between the low and high values.
- Sample points where close value is higher (lower) than the open value are called increasing (decreasing). Increasing candles are drawn in green and decreasing candles are drawn in red.
- ***plotly.graph_objects.Candlestick(x, open, high, low, close,…)*:** A plotly.graph_objects.Candlestick trace is a graph object in the figure's data list. For details, click here.

Arguments or Attributes:	**x:**	Parent: data[type=candlestick]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the x coordinates. If absent, linear coordinate will be generated.
	open :	Parent: data[type=candlestick]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the open values.
	high :	Parent: data[type=candlestick]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the high values.
	low :	Parent: data[type=candlestick]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the low values.
	close :	Parent: data[type=candlestick]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the close values.

Candlestick Chart: Example

- **Example :** Draws candlestick chart using Microsoft stock price data over the time. Data were collected from yahoo-finance. See full code on github-plotly-candlestick.

```
import plotly
import plotly.graph_objects as go
import pandas as pd


msft_stock = pd.read_csv('MSFT2014-18.csv')
print(msft_stock.head())

fig = go.Figure(data=[go.Candlestick(x=msft_stock['Date'],
                open=msft_stock['Open'],
                high=msft_stock['High'],
                low=msft_stock['Low'],
                close=msft_stock['Close'])])

fig.update_layout(
    title='Microsoft stock price for the period:2014-2018',
    yaxis_title='Stock Price (USD)',
)
```



- Click here to see figure

OHLC Chart

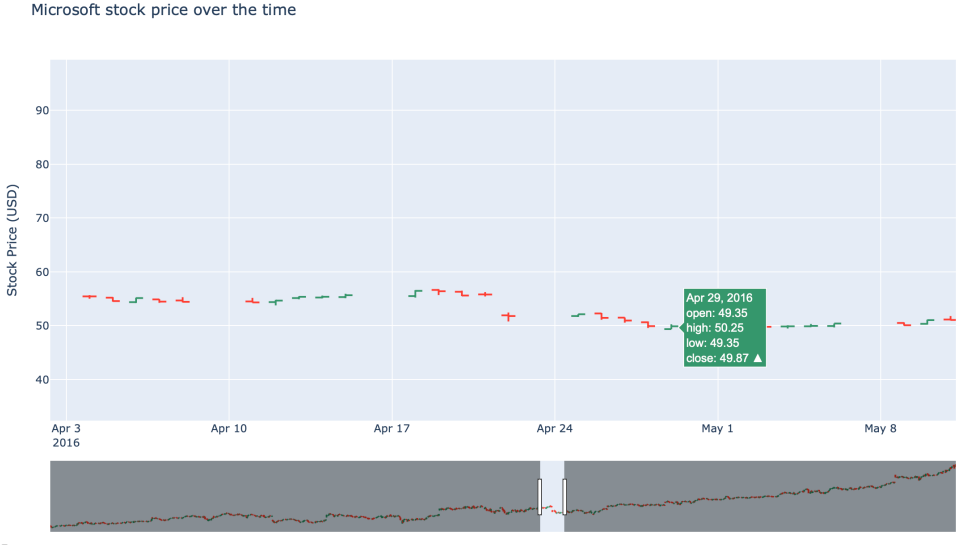
- The **OHLC chart** is a financial chart showing open, high, low, and close for a given x coordinate (time).
- The tip of the lines represent the 'low' and 'high' values and the horizontal segments represent the 'open' and 'close' values.
- Sample points where the close value is higher (lower) than the open value are called increasing (decreasing). Increasing candles are drawn in green and decreasing candles are drawn in red.
- ***plotly.graph_objects.Ohlc(x, open, high, low, close,…)*:** A plotly.graph_objects.Candlestick trace is a graph object in the figure's data list. For details, click here.

Arguments or Attributes:	**x:**	Parent: data[type=ohlc]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the x coordinates. If absent, linear coordinate will be generated.
	open :	Parent: data[type=ohlc]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the open values.
	high :	Parent: data[type=ohlc]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the high values.
	low :	Parent: data[type=ohlc]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the low values.
	close :	Parent: data[type=ohlc]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the close values.

OHLC Chart: Example

- **Example :** Draws OHLC chart using Microsoft stock price data over the time. Data were collected from yahoo-finance. See full code on github-plotly-ohlc.

```
import plotly.graph_objects as go
import pandas as pd
import plotly

df = pd.read_csv('MSFT2014-18.csv')

fig = go.Figure(data=go.Ohlc(x=df['Date'],
                             open=df['Open'],
                             high=df['High'],
                             low=df['Low'],
                             close=df['Close']))

fig.update_layout(
    title='Microsoft stock price over the time',
    yaxis_title='Stock Price (USD)',
)
```



- Click here to see figure

Waterfall Chart

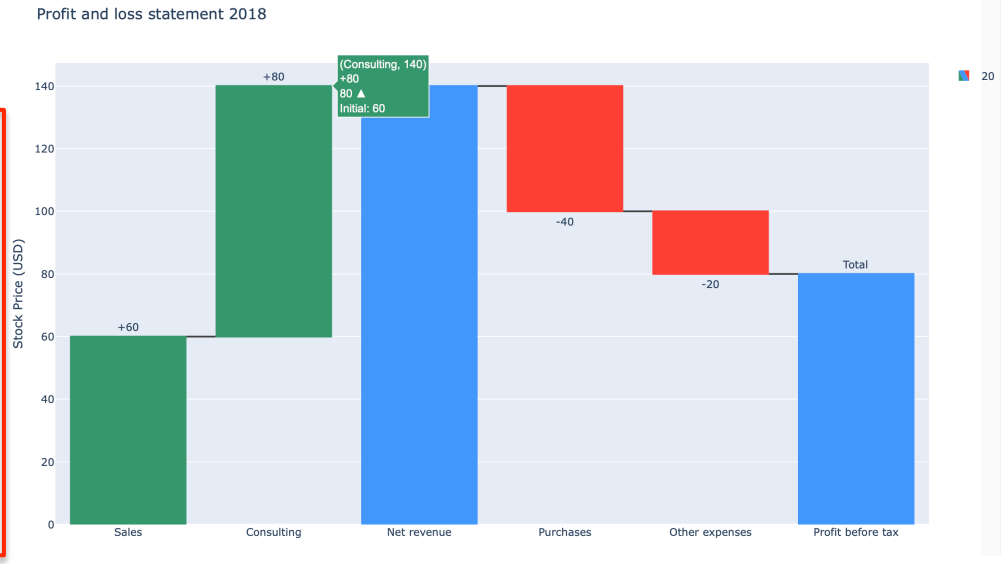
- The **Waterfall chart** displays the contribution of various elements (either positive or negative) in a bar chart.
- The data visualized by the span of the bars is set in ‘y’ if ‘orientation’ is set to the “v” (the default) and the labels are set in ‘x’.
- ***plotly.graph_objects.Waterfall(name, orientation, measure, x, textposition, text, y, connector, …)*:** A plotly.graph_objects.Candlestick trace is a graph object in the figure’s data list. For details, click here.

Arguments or Attributes:	**name:**	Parent: data[type=waterfall]; Type: string. Sets the trace name. The trace name appears as the legend item and on hover.
	orientation:	Parent: data[type=waterfall]; Type: enumerated, one of (“v” | “h”). Sets the orientation of the bars.
	measure:	Parent: data[type=waterfall]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. An array containing types of values. By default, values are considered as ‘relative’. Also ‘total’ is used to compute sums and ‘absolute’ is used to reset the computed total.
	x:	Parent: data[type=waterfall]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the x coordinates.
	y:	Parent: data[type=waterfall]; Type: list, numpy array, or Pandas series of numbers, strings, or datetimes. Sets the y coordinates.

Waterfall Chart: Example

- **Example :** Displays waterfall chart showing profit and loss statement. Data were collected from yahoo-finance. See full code on github-plotly-waterfall.

```
import plotly.graph_objects as go
import plotly

fig = go.Figure(go.Waterfall(
    name="20", orientation="v",
    measure=["relative", "relative", "total",
             "relative", "relative", "total"],
    x=["Sales", "Consulting", "Net revenue",
       "Purchases", "Other expenses",
       "Profit before tax"],
    textposition="outside",
    text=["+60", "+80", "", "-40", "-20", "Total"],
    y=[60, 80, 0, -40, -20, 0],
    connector={"line":{"color":"rgb(63, 63, 63)"}},
))

fig.update_layout(
        title="Profit and loss statement 2018",
        yaxis_title='Stock Price (USD)',
        showlegend=True
)
```



- Click here to see figure

Annotated Heatmap

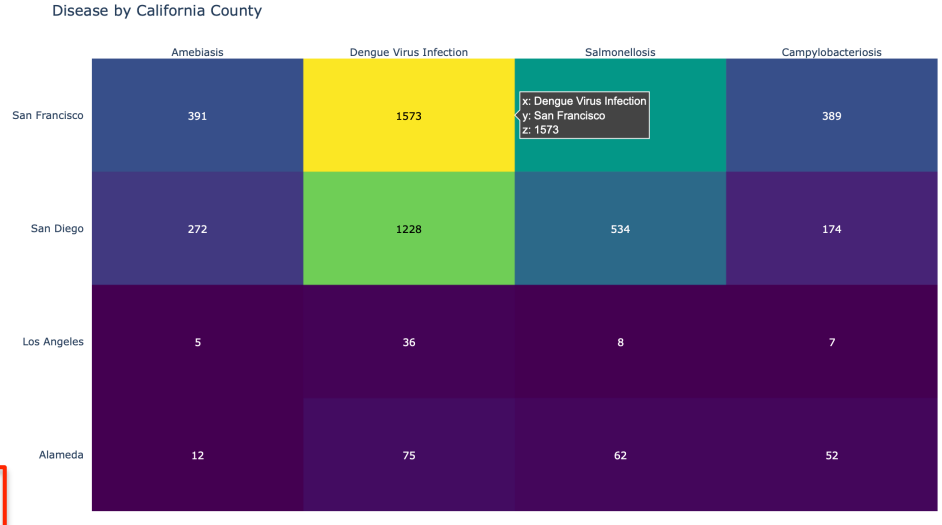
- **Plotly.figure_factory module is used to draw Annotated heatmap**
- ***ff.create_annotated_heatmap(z, x=None, y=None, annotation_text=None, colorscale='Plasma', font_colors=None, …)*:** BETA function that creates annotated heatmap. For details, click here.

Arguments or Attributes:	**z :**	List[list] | ndarray; z matrix to create heatmap.
	x :	List; x axis labels.
	y :	List; y axis labels.
	annotation_text :	List[list] | ndarray; text strings for annotations. Should have same dimensions as the z matrix.
	colorscale :	List | str; heatmap colorscale.
	font_colors :	List; list of two color strings.

Annotated Heatmap: Example

- **Example :** Draws annotated heatmap using California diseases data. See full code on [github-plotly-heatmap](#).

```
import plotly.figure_factory as ff
import numpy as np
import plotly

disease = ['Amebiasis', 'Dengue Virus Infection',
           'Salmonellosis', 'Campylobacteriosis']
county = ['Alameda', 'Los Angeles',
          'San Diego', 'San Francisco']
cases = np.array([[12, 75, 62, 52],
                  [5, 36, 8, 7],
                  [272, 1228, 534, 174],
                  [391, 1573, 841, 389]])

fig = ff.create_annotated_heatmap(z=cases, x=disease,
                                  y=county,
                                  annotation_text=cases,
                                  colorscale='Viridis')

fig.update_layout(title='Disease by California County')
```



- Click here to see figure

Polar Chart

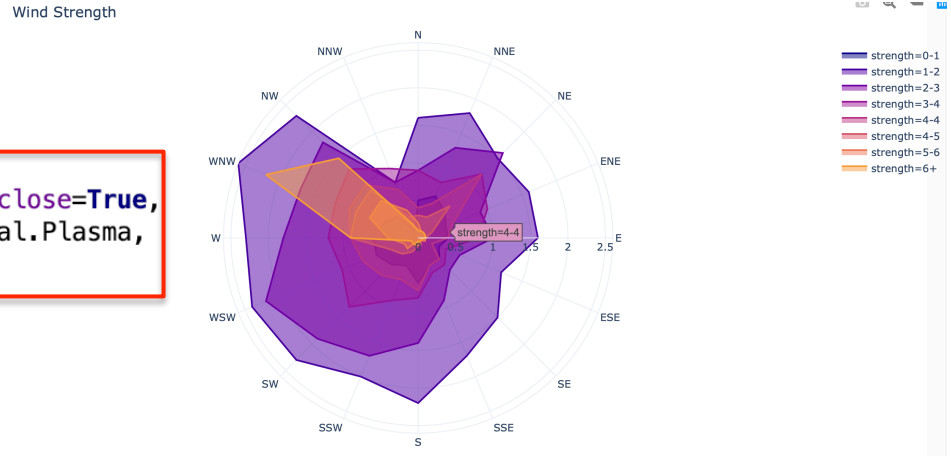
- **Polar chart** represents data along radial and angular axes. With plotly.express, polar data can be displayed as scatter markers with px.scatter_polar, and as lines with px.line_polar.
- ***plotly.express.line_polar (data_frame=None, r=None, theta=None, color=None, line_close=False, color_discrete_sequence=None,…)*** **:** In a polar line plot, each row of data_frame is represented as vertex of a polyline mark in polar coordinates. For details, click here.

Parameters:	**data_frame:**	DataFrame or array-like or dict; this argument needs to be passed for column names (and not keyword names) to be used. Array-like and dict are transformed internally to a pandas DataFrame.
	r :	Str or int or Series or array-like; values from this column or array-like are used to position marks along the radial axis in polar coordinates.
	theta :	Str or int or Series or array-like; values from this column or array-like are used to position marks along the angular axis in polar coordinates.
	color :	Str or int or Series or array-like; values from this column or array-like are used to assign color to marks.
	line_close :	Boolean (default False); if True, an extra line segment is drawn between the first and last point.
	color_discrete_sequence	List of str; strings should define valid CSS-colors.

Polar Chart: Example

➢ **Example :** Displays Polar chart using function px.line_polar(). See full code on [github-plotly-polarchart](#).

```
import plotly.express as px
import plotly

wind = px.data.wind()
fig = px.line_polar(wind, r="frequency",
                    theta="direction", color="strength", line_close=True,
                    color_discrete_sequence=px.colors.sequential.Plasma,
                    template="plotly_white",)

fig.update_layout(
  title="Wind Strength"
)

fig.update_traces(fill='toself')

plotly.offline.plot(fig, filename='polarplot_px.html')
fig.show()
```



- Click [here](#) to see figure

Radar Chart

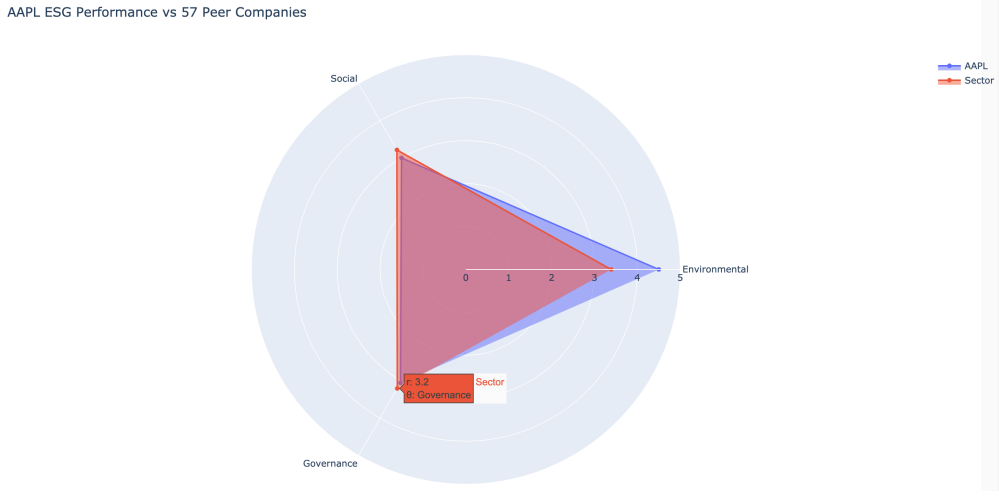
- **Radar Chart** displays multivariate data in the form of a two dimensional chart of quantitative variables represented on axes originating from the center.
- For a Radar Chart, use a polar chart with categorical angular variables, with px.line_polar for data available as a tidy DataFrame, or with go.Scatterpolar in the general case.
- ***go.scatterpolar(r, theta, fill, name, …)* :** The scatterpolar trace type encompasses line charts, scatter charts, text charts, and bubble charts in polar coordinates. For details, click here.

Parameters:	**r :**	Parent: data[type=scatterpolar]; type: number or categorical string. Default 0. Builds a linear space of r coordinates.
	theta :	Parent: data[type=scatterpolar]; type: list, numpy array, or pandas series of numbers, strings, or datetimes. Sets the angular coordinates.
	fill :	Parent: data[type=scatterpolar]; type: enumerated, one of (“none” | “toself” | “tonext”). Default : “none”. Sets the area to fill with a solid color.
	name :	Parent: data[type=scatterpolar]; type: string. Sets the trace name.

Radar Chart: Example

- **Example :** Displays Radar chart using function px.Scatterpolar(). Data were collected from yahoo-finance. See full code on github-plotly-radarchart.

```
esg_performance = ['Environmental','Social','Governance']

fig = go.Figure()

fig.add_trace(go.Scatterpolar(
      r=[4.5, 3, 3.05],
      theta=esg_performance,
      fill='toself',
      name='AAPL'
))
fig.add_trace(go.Scatterpolar(
      r=[3.39, 3.215, 3.2],
      theta=esg_performance,
      fill='toself',
      name='Sector'
))

fig.update_layout(
  polar=dict(
    radialaxis=dict(
      visible=True,
      range=[0, 5]
    )),
  title="AAPL ESG Performance vs 57 Peer Companies",
  showlegend=True
)
```



- Click here to see figure

3D Scatter Plot

- ***plotly.express.scatter_3d(data_frame=None, x=None, y=None, z=None, color=None, symbol=None, size=None, …)*** **:**

In a 3D scatter plot, each row of data_frame is represented by a symbol mark in 3D space. For details, click here.

Parameters:

data_frame: DataFrame or array-like or dict; this argument needs to be passed for column names (and not keyword names) to be used. Array-like and dict are transformed internally to a pandas DataFrame.

x : Str or int or Series or array-like; values from this column or array-like are used to position marks along the x axis in coordinates.

y : Str or int or Series or array-like; values from this column or array-like are used to position marks along the y axis in coordinates.

z : Str or int or Series or array-like; values from this column or array-like are used to position marks along the z axis in coordinates.

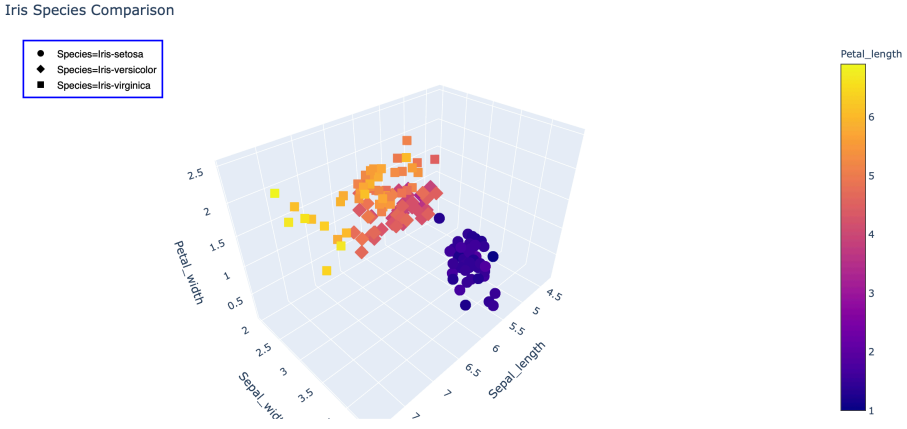
color : Str or int or Series or array-like; values from this column or array-like are used to assign color to marks.

symbol : Str or int or Series or array-like; values from this column or array-like are used to assign symbols to marks.

3D Scatter Plot: Example

➢ **Example :** Displays 3D scatter plot using iris data. See full code on [github-plotly-3dscatter](#).

```
col_names = ['Sepal_length', 'Sepal_width',
             'Petal_length', 'Petal_width', 'Species']
iris = pd.read_csv('iris.csv', names=col_names)

fig = px.scatter_3d(iris, x='Sepal_length',
                    y='Sepal_width', z='Petal_width',
                    color='Petal_length', symbol='Species',
                    color_continuous_scale=px.colors.sequential.Plasma)


# tight layout
fig.update_layout(title_text='Iris Species Comparison',
    legend=go.layout.Legend(
        x=0,
        y=1,
        traceorder="normal",
        font=dict(
            family="sans-serif",
            size=12,
            color="black"
        ),
    bgcolor="white",
    bordercolor="Blue",
    borderwidth=2))
```



- Click here to see figure

Summary

Plot Types	API Call
Time Series	plotly.graph_objects.Scatter(x, y, name, line_color, opacity, …)
Candlestick Chart	plotly.graph_objects.Candlestick(x, open, high, low, close,…)
OHLC Chart	plotly.graph_objects.Ohlc(x, open, high, low, close,…)
Waterfall Chart	plotly.graph_objects.Waterfall(name, orientation, measure, x, textposition, text, y, connector, …)
Annotated Heatmap	ff.create_annotated_heatmap(z, x=None, y=None, annotation_text=None, colorscale='Plasma', font_colors=None, …)
Polar Chart	plotly.express.line_polar (data_frame=None, r=None, theta=None, color=None, line_close=False, color_discrete_sequence=None,…)
Radar Chart	go.scatterpolar(r, theta, fill, name, …)
3D Scatter Plot	plotly.express.scatter_3d(data_frame=None, x=None, y=None, z=None, color=None, symbol=None, size=None, …)