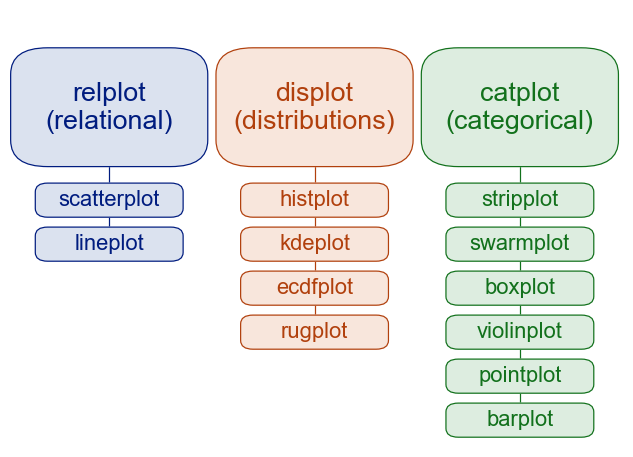
Seaborn is a library for making statistical graphics in Python. It builds on top of [matplotlib](https://matplotlib.org/) and integrates closely with [pandas](https://pandas.pydata.org/) data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

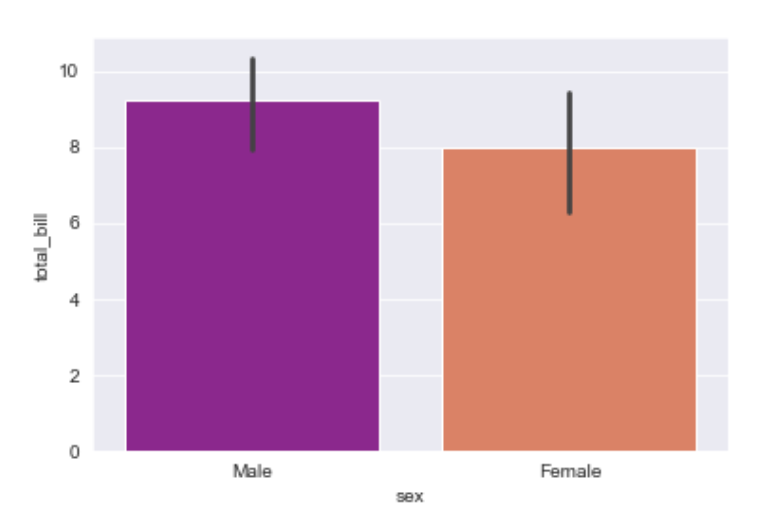


* **Relational plots:** This plot is used to understand the relation between two variables.
* A **barplot** is basically used to aggregate the categorical data according to some methods and by default its the mean. It can also be understood as a visualization of the group by action. To use this plot we choose a categorical column for the x axis and a numerical column for the y axis and we see that it creates a plot taking a mean per categorical column.

sns.barplot(x **=**'sex', y **=**'total\_bill', data **=** df,

            palette **=**'plasma', estimator **=** np.std)

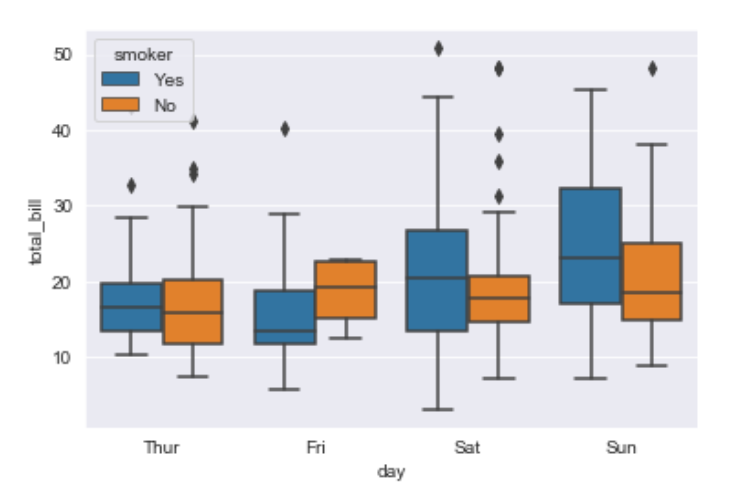
seaborn.**barplot**(*data=None*, *\**, *x=None*, *y=None*, *hue=None*, *order=None*, *hue\_order=None*, *estimator='mean'*, *errorbar=('ci'*, *95)*, *n\_boot=1000*, *seed=None*, *units=None*, *weights=None*, *orient=None*, *color=None*, *palette=None*, *saturation=0.75*, *fill=True*, *hue\_norm=None*, *width=0.8*, *dodge='auto'*, *gap=0*, *log\_scale=None*, *native\_scale=False*, *formatter=None*, *legend='auto'*, *capsize=0*, *err\_kws=None*, *ci=<deprecated>*, *errcolor=<deprecated>*, *errwidth=<deprecated>*, *ax=None*, *\*\*kwargs*)



**Boxplot**

A boxplot is sometimes known as the box and whisker plot.It shows the distribution of the quantitative data that represents the comparisons between variables. boxplot shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution i.e. the dots indicating the presence of outliers.

|  |
| --- |
| sns.boxplot(x **=**'day', y **=**'total\_bill', data **=** df, hue **=**'smoker') |

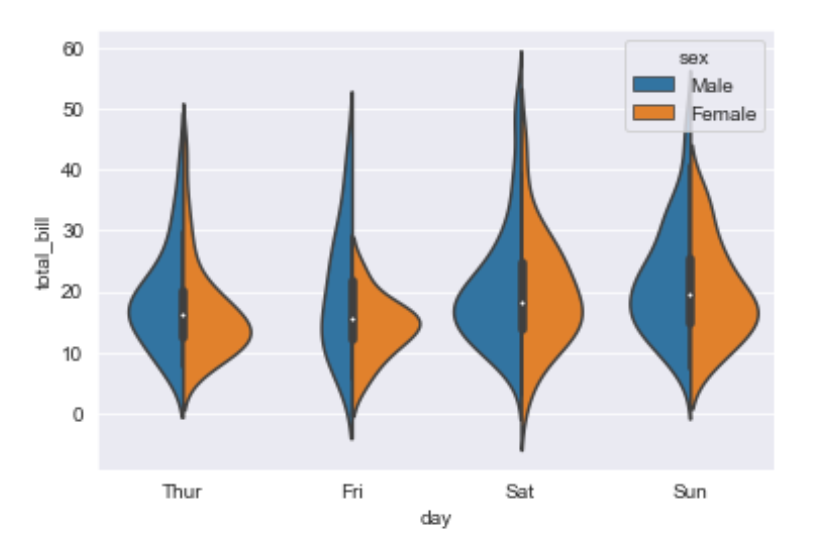


seaborn.boxplot(data=None, \*, x=None, y=None, hue=None, order=None, hue\_order=None, orient=None, color=None, palette=None, saturation=0.75, fill=True, dodge='auto', width=0.8, gap=0, whis=1.5, linecolor='auto', linewidth=None, fliersize=None, hue\_norm=None, native\_scale=False, log\_scale=None, formatter=None, legend='auto', ax=None, \*\*kwargs)

**Violinplot**

It is similar to the boxplot except that it provides a higher, more advanced visualization and uses the kernel density estimation to give a better description about the data distribution.

sns.violinplot(x ='day', y ='total\_bill', data = df, hue ='sex', split = True)



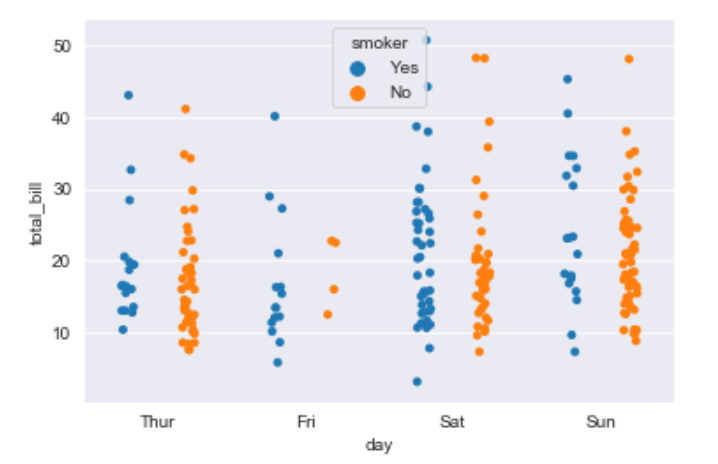
seaborn.**violinplot**(*data=None*, *\**, *x=None*, *y=None*, *hue=None*, *order=None*, *hue\_order=None*, *orient=None*, *color=None*, *palette=None*, *saturation=0.75*, *fill=True*, *inner='box'*, *split=False*, *width=0.8*, *dodge='auto'*, *gap=0*, *linewidth=None*, *linecolor='auto'*, *cut=2*, *gridsize=100*, *bw\_method='scott'*, *bw\_adjust=1*, *density\_norm='area'*, *common\_norm=False*, *hue\_norm=None*, *formatter=None*, *log\_scale=None*, *native\_scale=False*, *legend='auto'*, *scale=<deprecated>*, *scale\_hue=<deprecated>*, *bw=<deprecated>*, *inner\_kws=None*, *ax=None*, *\*\*kwargs*)

**stripplot**

It basically creates a scatter plot based on the category.

sns.stripplot(x ='day', y ='total\_bill', data = df,

jitter = True, hue ='smoker', dodge = True)

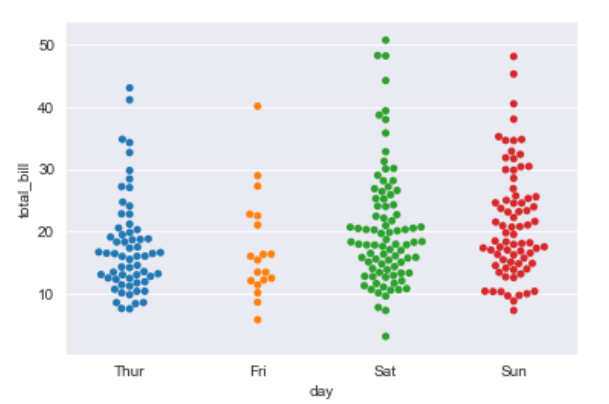


seaborn.**stripplot**(*data=None*, *\**, *x=None*, *y=None*, *hue=None*, *order=None*, *hue\_order=None*, *jitter=True*, *dodge=False*, *orient=None*, *color=None*, *palette=None*, *size=5*, *edgecolor=<default>*, *linewidth=0*, *hue\_norm=None*, *log\_scale=None*, *native\_scale=False*, *formatter=None*, *legend='auto'*, *ax=None*, *\*\*kwargs*)

**Swarmplot**

It is very similar to the stripplot except the fact that the points are adjusted so that they do not overlap.Some people also like combining the idea of a violin plot and a stripplot to form this plot. One drawback to using swarmplot is that sometimes they dont scale well to really large numbers and takes a lot of computation to arrange them. So in case we want to visualize a swarmplot properly we can plot it on top of a violinplot.

sns.swarmplot(x ='day', y ='total\_bill', data = df)

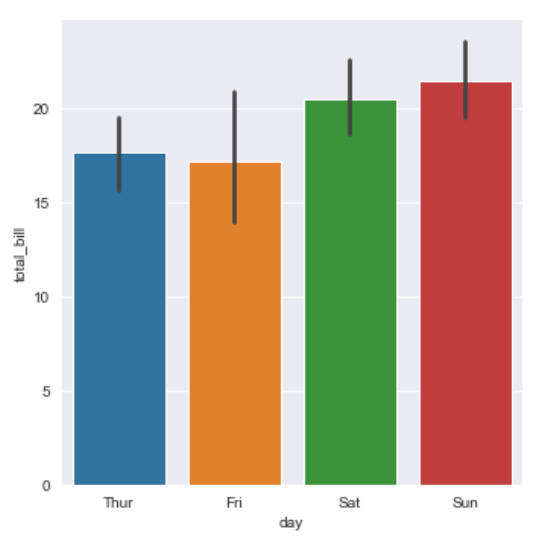


seaborn.**swarmplot**(*data=None*, *\**, *x=None*, *y=None*, *hue=None*, *order=None*, *hue\_order=None*, *dodge=False*, *orient=None*, *color=None*, *palette=None*, *size=5*, *edgecolor=None*, *linewidth=0*, *hue\_norm=None*, *log\_scale=None*, *native\_scale=False*, *formatter=None*, *legend='auto'*, *warn\_thresh=0.05*, *ax=None*, *\*\*kwargs*)

**Factorplot**

It is the most general of all these plots and provides a parameter called kind to choose the kind of plot we want thus saving us from the trouble of writing these plots separately. The kind parameter can be bar, violin, swarm etc.

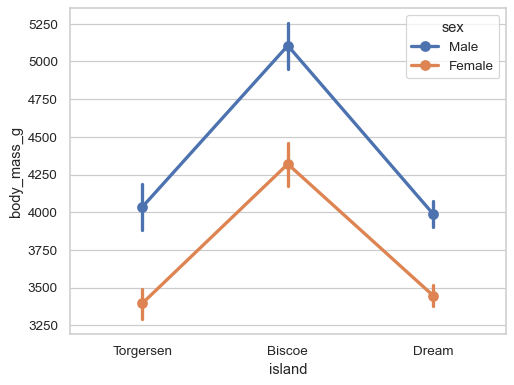
sns.factorplot(x ='day', y ='total\_bill', data = df, kind ='bar')



Point plots:

Point plots can be more useful than bar plots for focusing comparisons between different levels of one or more categorical variables.

sns.pointplot(data=penguins, x="island", y="body\_mass\_g", hue="sex")

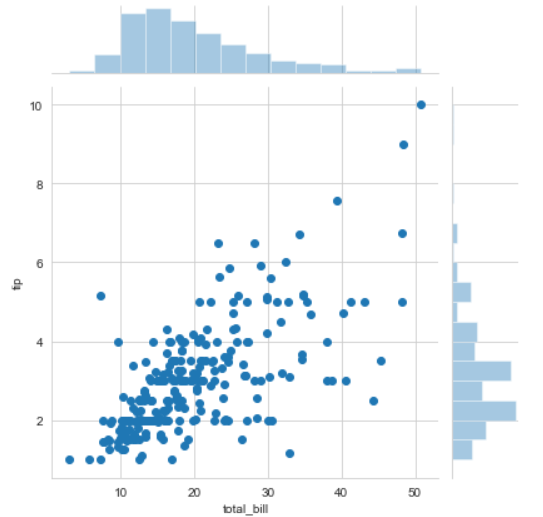


seaborn.**pointplot**(*data=None*, *\**, *x=None*, *y=None*, *hue=None*, *order=None*, *hue\_order=None*, *estimator='mean'*, *errorbar=('ci'*, *95)*, *n\_boot=1000*, *seed=None*, *units=None*, *weights=None*, *color=None*, *palette=None*, *hue\_norm=None*, *markers=<default>*, *linestyles=<default>*, *dodge=False*, *log\_scale=None*, *native\_scale=False*, *orient=None*, *capsize=0*, *formatter=None*, *legend='auto'*, *err\_kws=None*, *ci=<deprecated>*, *errwidth=<deprecated>*, *join=<deprecated>*, *scale=<deprecated>*, *ax=None*, *\*\*kwargs*)

**Joinplot**

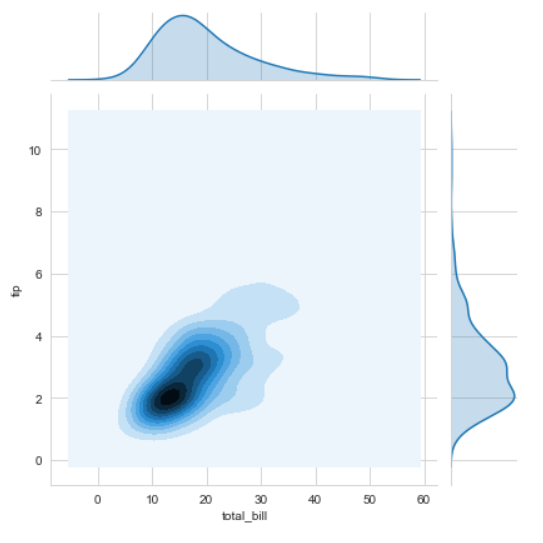
It is used to draw a plot of two variables with bivariate and univariate graphs. It basically combines two different plots.

sns.jointplot(x ='total\_bill', y ='tip', data = df)



sns.jointplot(x ='total\_bill', y ='tip', data = df, kind ='kde')

# KDE shows the density where the points match up the most

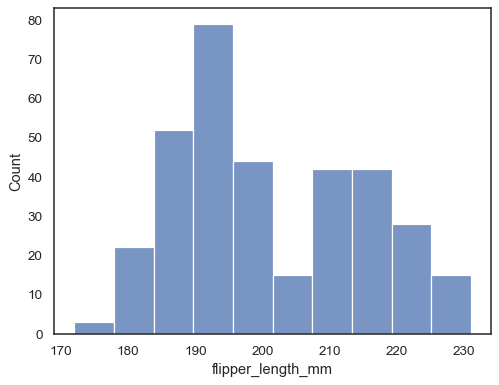


Hist plot:

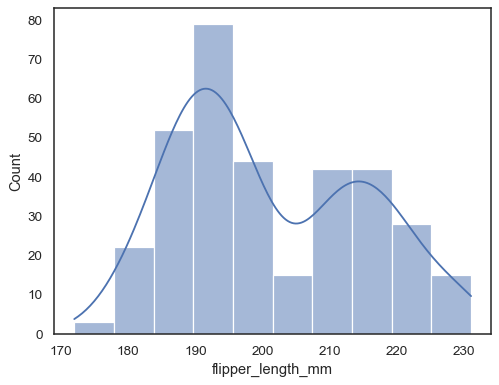
A histogram is a classic visualization tool that represents the distribution of one or more variables by counting the number of observations that fall within discrete bins.

penguins = sns.load\_dataset("penguins")

sns.histplot(data=penguins, x="flipper\_length\_mm")



sns.histplot(data=penguins, x="flipper\_length\_mm", kde=True)

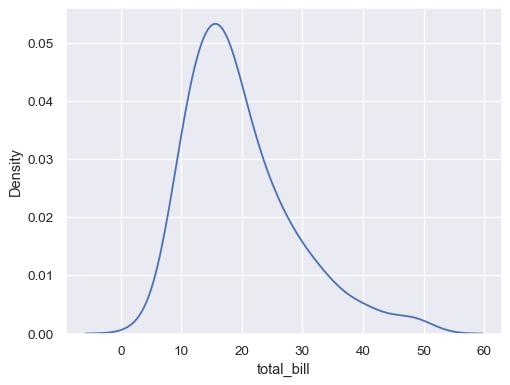


Kde plot:

A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.

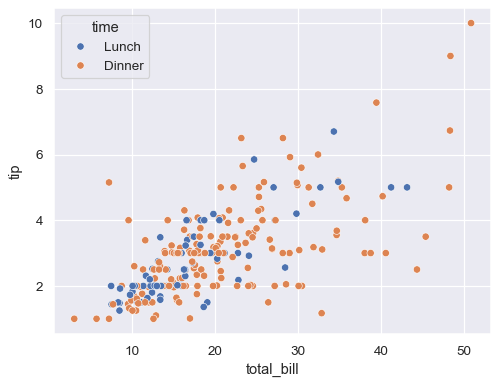
tips = sns.load\_dataset("tips")

sns.kdeplot(data=tips, x="total\_bill")



Scatter plot:

sns.scatterplot(data=tips, x="total\_bill", y="tip", hue="time")



Line plot:

may\_flights = flights.query("month == 'May'")

sns.lineplot(data=may\_flights, x="year", y="passengers")

