Certification Training Courses for Graduates and Professionals

www.signitivetech.com

Seaborn: Statistical Data Visualization

Seaborn

- Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.

- Seaborn is a library for making attractive and informative statistical graphics in Python. It is built on top of matplotlib and tightly integrated with the PyData stack, including support for numpy and pandas data structures and statistical routines from scipy and statsmodels.

Seaborn

Some of the features that seaborn offers are:

- Several built-in themes for styling matplotlib graphics
- Tools for choosing color palettes to make beautiful plots that reveal patterns in your data
- Functions for visualizing univariate and bivariate distributions or for comparing them between subsets of data
- Tools that fit and visualize linear regression models for different kinds of independent and dependent variables
- Functions that visualize matrices of data and use clustering algorithms to discover structure in those matrices
- A function to plot statistical time series data with flexible estimation and representation of uncertainty around the estimate
- High-level abstractions for structuring grids of plots that let you easily build complex visualizations

Seaborn

First, import libraries to use seaborn:

- **import** matplotlib.pyplot **as** plt
- **import** seaborn **as** sns
- %matplotlib inline → To display plots inside the Jupyter Notebook

Seaborn

- Load dataset:

```
tips = pd.read_csv("../Datasets/tips.csv") → Load the dataset
tips.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

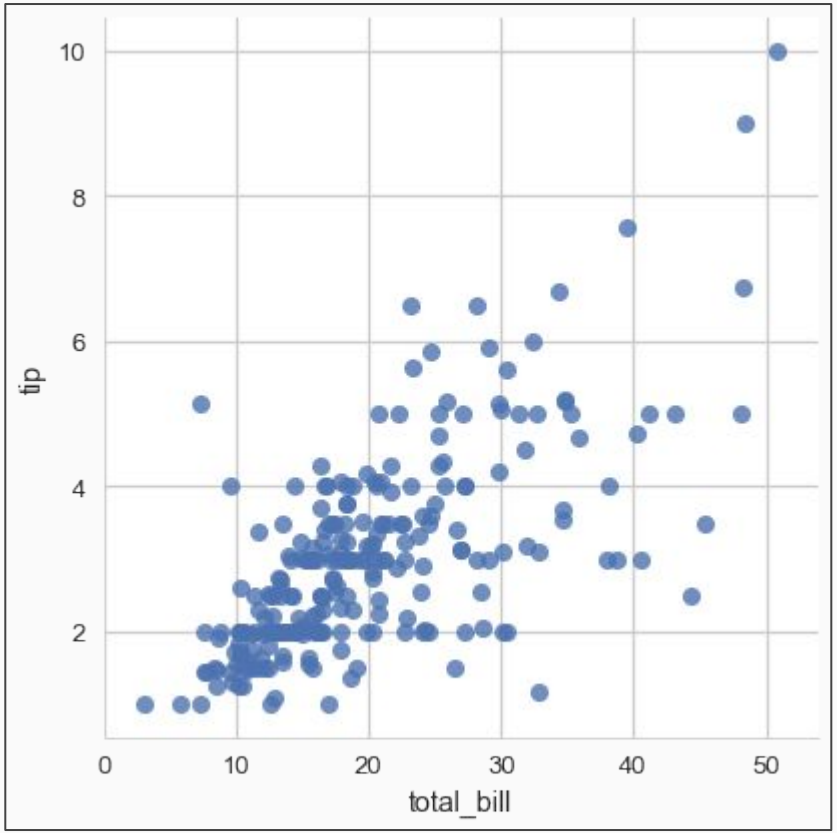
Scatter Plot:

```
sns.lmplot(x='total_bill', y='tip', data=tips, scatter=True, fit_reg=False, size=6)
```

lmplot() → Plot data
x, y → Input variables, column names in data
data → Dataframe where each column is a variable and each row is an observation
scatter → If True, draw a scatter plot
size → Height of each facet

Seaborn

Output:



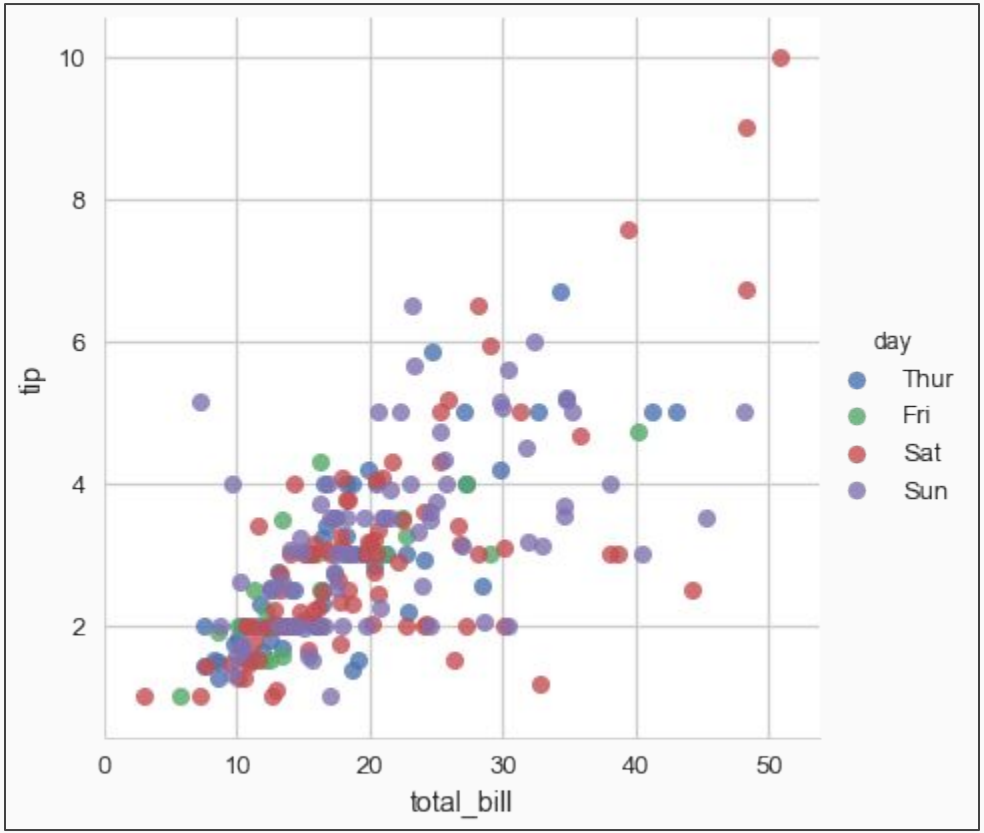
Scatter Plot:

```
sns.lmplot(x='total_bill', y='tip', hue='day', data=tips, scatter=True,
fit_reg=False, size=6)
```

hue → Variables that define subsets of the data i.e. third dimension of information

Seaborn

Output:



Scatter Plot:

```
sns.lmplot(x='total_bill', y='tip', hue='sex', data=tips, scatter=True,
fit_reg=False, size=6, palette='ocean', markers=['o', 'x'])
```

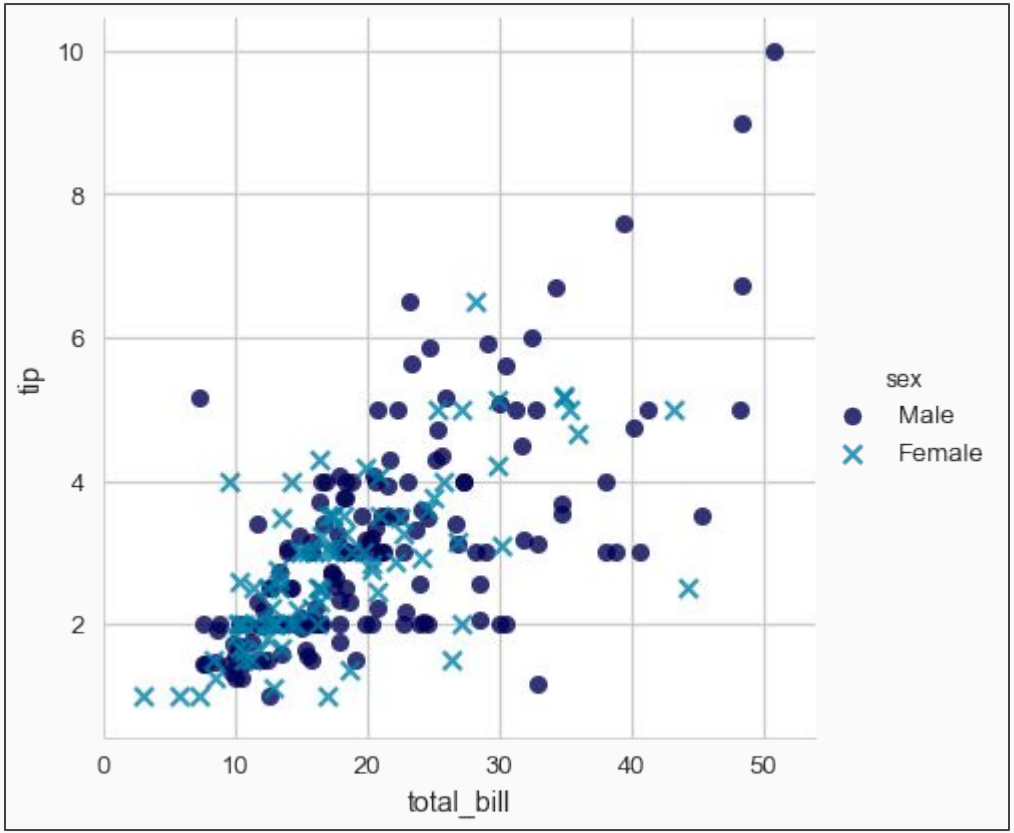
palette → Seaborn color palette
markers → Markers of the scatter plot

Possible values for Palette:

Accent, Blues, Greens, Greys, Oranges, Paired, Pastle1, Pastle2, Purples, Reds, Set1, Set2, Set3, Spectral, Vega10, Vega20, Wistia, autumn, binary, bone, cool, coolwarm, copper, cubehelix, flag, gist_earth, gist_gray, gist_heat, gist_rainbow, gray, hot, icefire, inferno, jet, magma, ocean, pink, plasma, prism, rainbow, rocket, seismic, spectral, spring, summer, tab10, tab20, terrain, winter

Seaborn

Output:



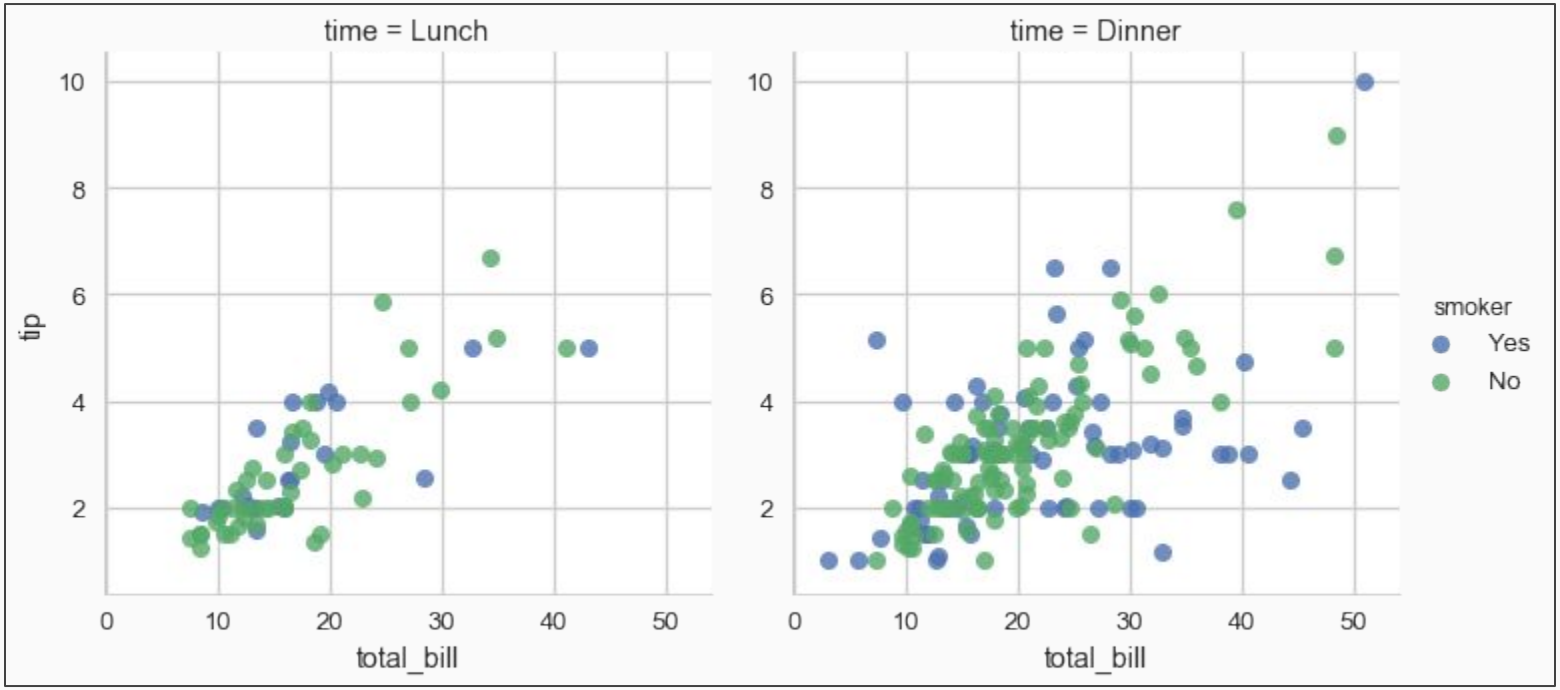
Scatter Plot:

```
sns.lmplot(x="total_bill", y="tip", hue="smoker", col="time", data=tips,
scatter=True, fit_reg=False)
```

col → Variables that define subsets of the data

Seaborn

Output:



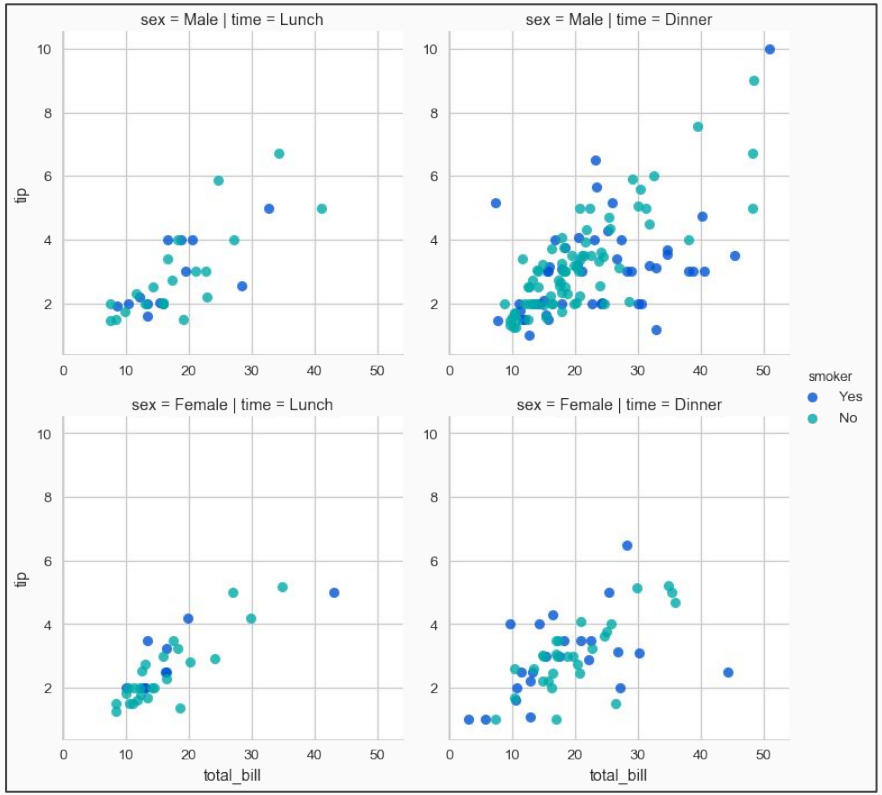
Scatter Plot:

```
sns.lmplot(x="total_bill", y="tip", hue="smoker", col="time", row="sex",
data=tips, scatter=True, fit_reg=False, palette='winter');
```

row → Variables that define subsets of the data

Seaborn

Output:



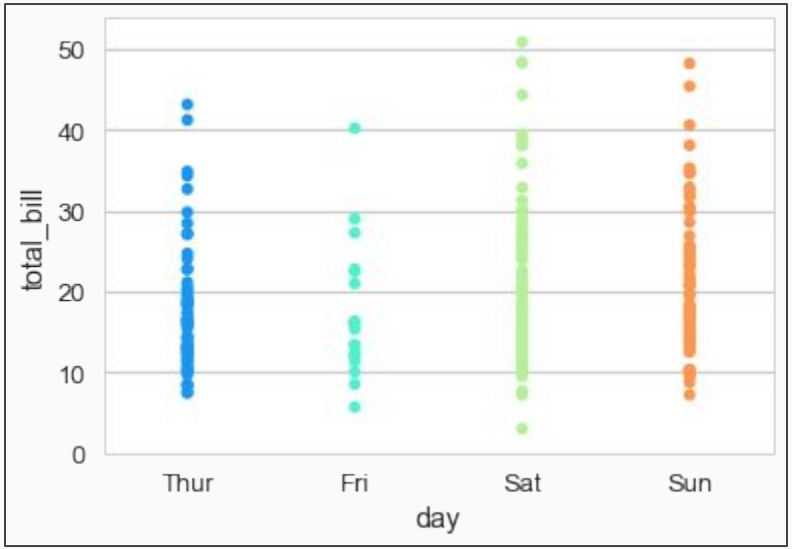
Strip Plot: Categorical scatter plots

- A simple way to show the the values of some quantitative variable across the levels of a categorical variable uses stripplot(), which generalizes a scatter plot to the case where one of the variables is categorical.

```
sns.stripplot(x='day', y='total_bill', data=tips, palette='rainbow', size=6)
```

stripplot() → Draw a scatter plot where one variable is categorical

Output:



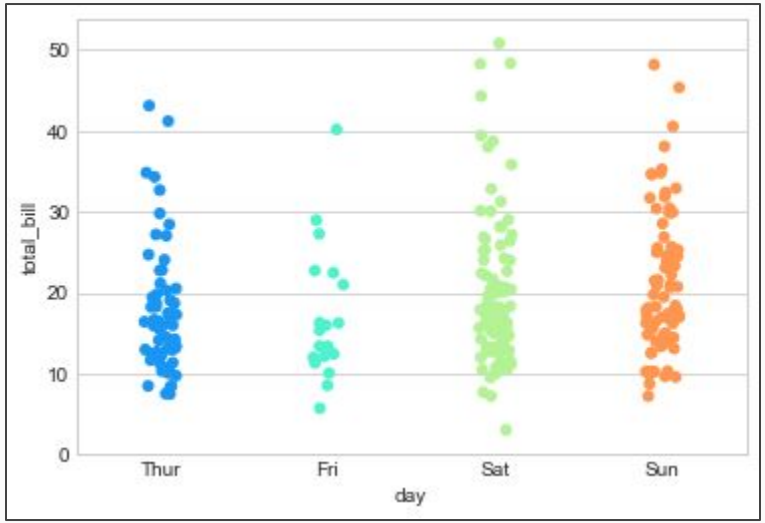
Strip Plot: Categorical scatter plots

- A simple way to show the the values of some quantitative variable across the levels of a categorical variable uses stripplot(), which generalizes a scatter plot to the case where one of the variables is categorical.
- In a strip plot, the scatter plot points will usually overlap. This makes it difficult to see the full distribution of data. One easy solution is to adjust the positions (only along the categorical axis) using some random “jitter”.

```
sns.stripplot(x="day", y="total_bill", data=tips, jitter=True)
```

stripplot() → Draw a scatter plot where one variable is categorical

Output:



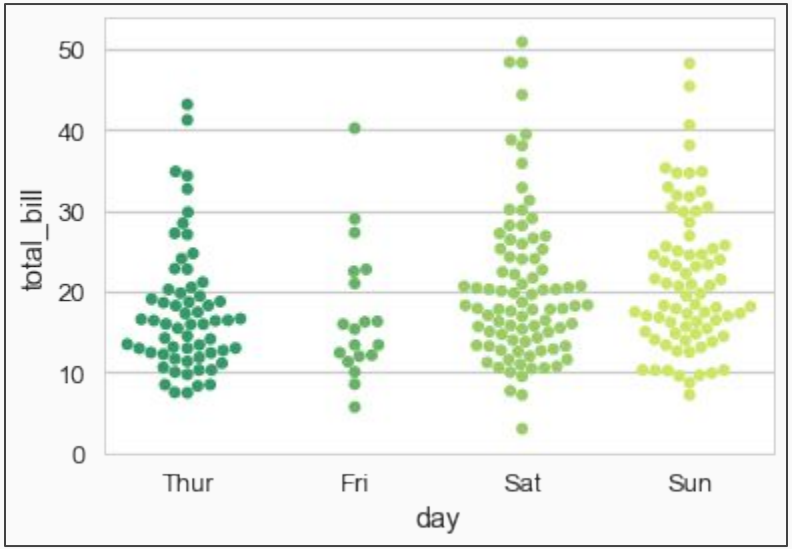
Swarm Plot:

- A different approach would be to use the function swarmplot(), which positions each scatterplot point on the categorical axis with an algorithm that avoids overlapping points.

```
sns.swarmplot(x='day', y='total_bill', data=tips, palette='spring', size=6)
```

swarmplot() → Draw a categorical scatter plot with non-overlapping points

Output:



Box Plot:

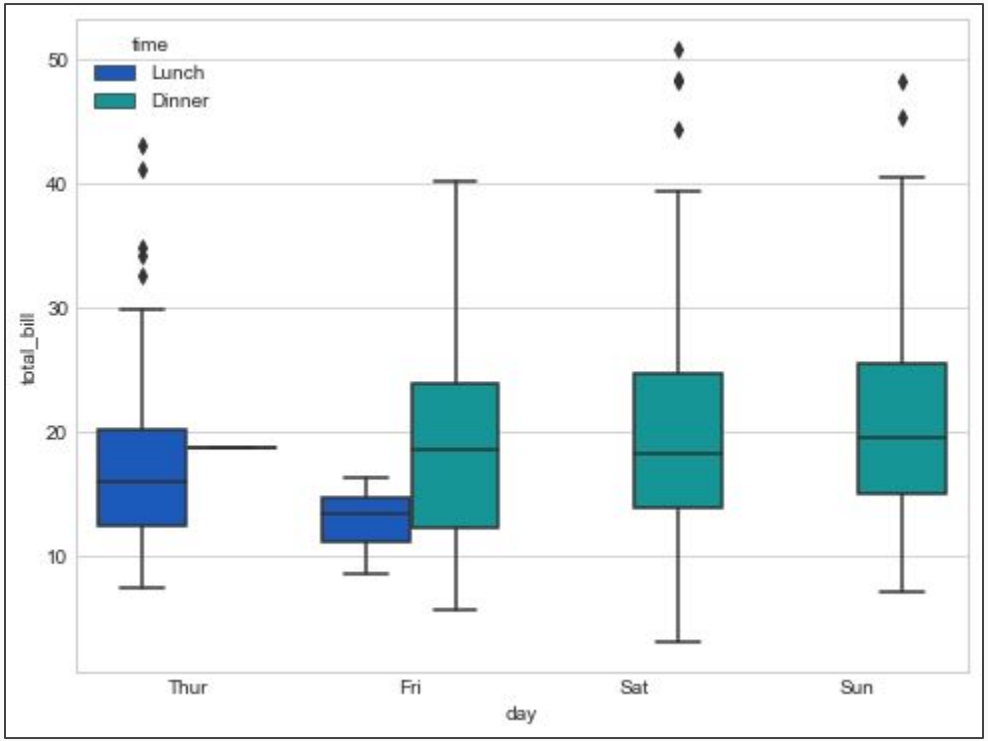
- This kind of plot shows the three quartile values of the distribution along with extreme values.
- The “whiskers” extend to points that lie within 1.5 IQRs of the lower and upper quartile, and then observations that fall outside this range are displayed independently.
- Importantly, this means that each value in the box plot corresponds to an actual observation in the data.

```
sns.boxplot(x='day', y='total_bill', hue='time', data=tips, palette='winter')
```

boxplot() → To show the distribution of the data

Seaborn

Output:



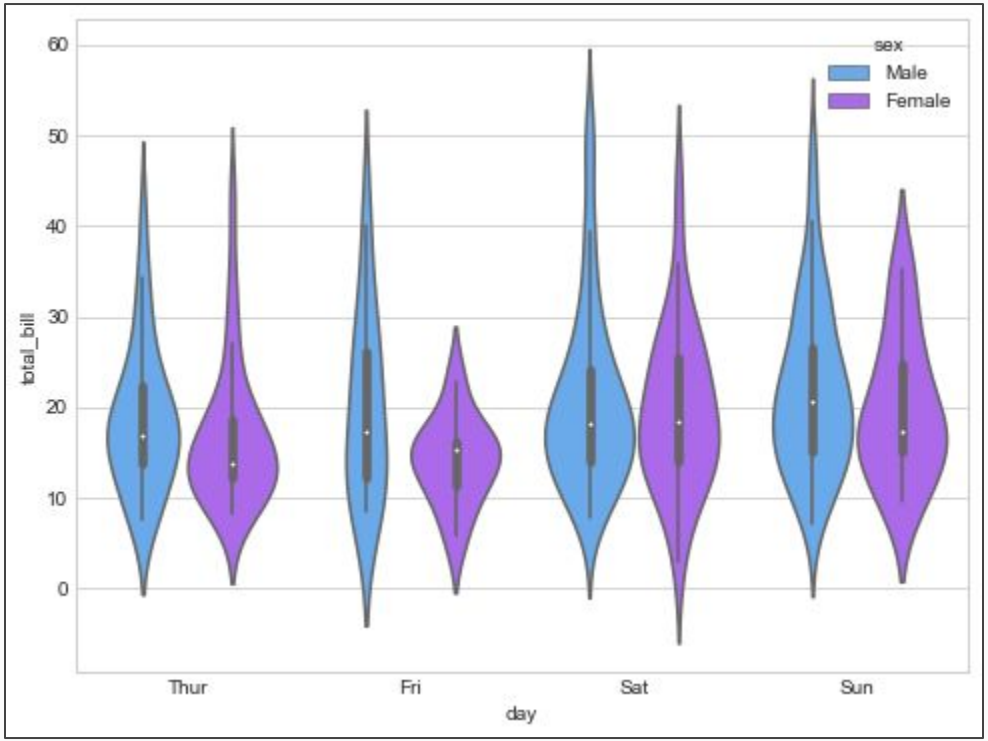
Violin Plot:

- A different approach is a violinplot(), which combines a boxplot with the kernel density estimation.

```
sns.violinplot(x='day', y='total_bill', hue='sex', data=tips, palette='cool')
```

boxplot() → To show the distribution of the data

Output:


sex
Male
Female
60
50
40
30
20
10
0
total_bill
Thur
Fri
Sat
Sun
day

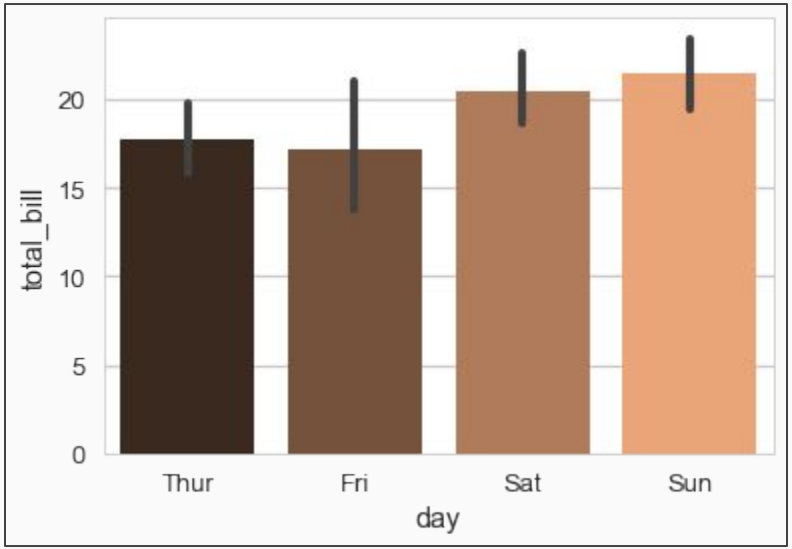
Bar Plot:

- The barplot() function operates on a full dataset and shows an arbitrary estimate, using the mean by default.
- When there are multiple observations in each category, it also uses bootstrapping to compute a confidence interval around the estimate and plots that using error bars.

```
sns.barplot(x='day', y='total_bill', data=tips, palette='copper')
```

barplot() → Draw a barplot

Output:



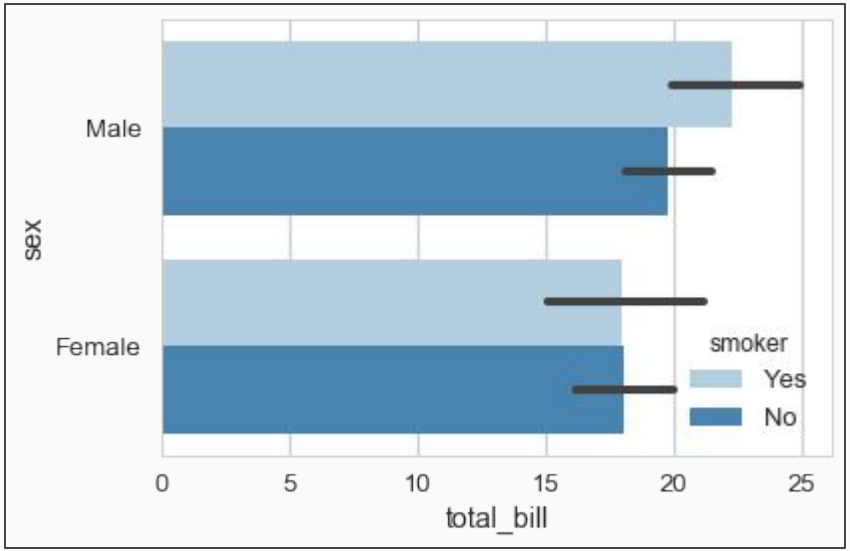
Bar Plot:

```
sns.barplot(x='total_bill', y='sex', hue='smoker', data=tips,
palette='Blues', orient='h')
```

Orient → Orientation of the plot

- h: horizontal
- v: vertical

Output:



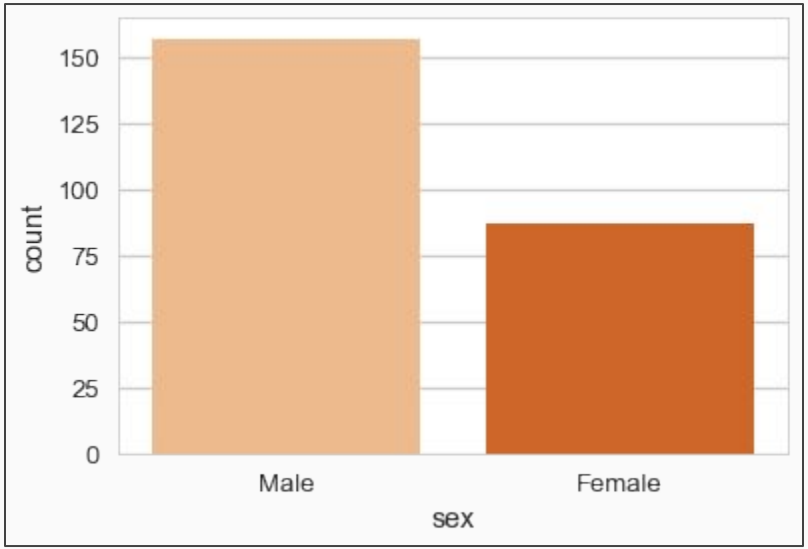
Count Plot:

- Count Plot is used to show the number of observations in each category rather than computing a statistic for a second variable.

```
sns.countplot(x='sex', data=tips, palette='Oranges')
```

countplot() → Show the counts of observations in each categorical bin using bars.

Output:

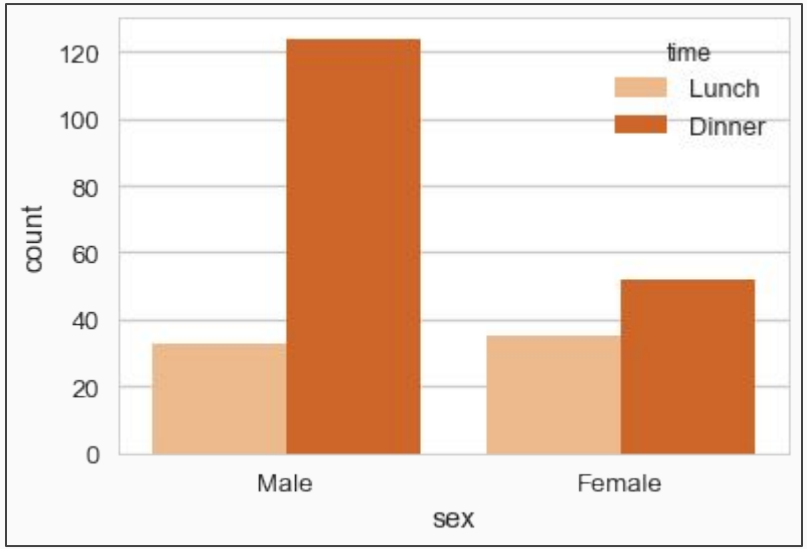


Count Plot:

```
sns.countplot(x='sex', hue='time', data=tips, palette='Oranges')
```

countplot() → Show the counts of observations in each categorical bin using bars.

Output:



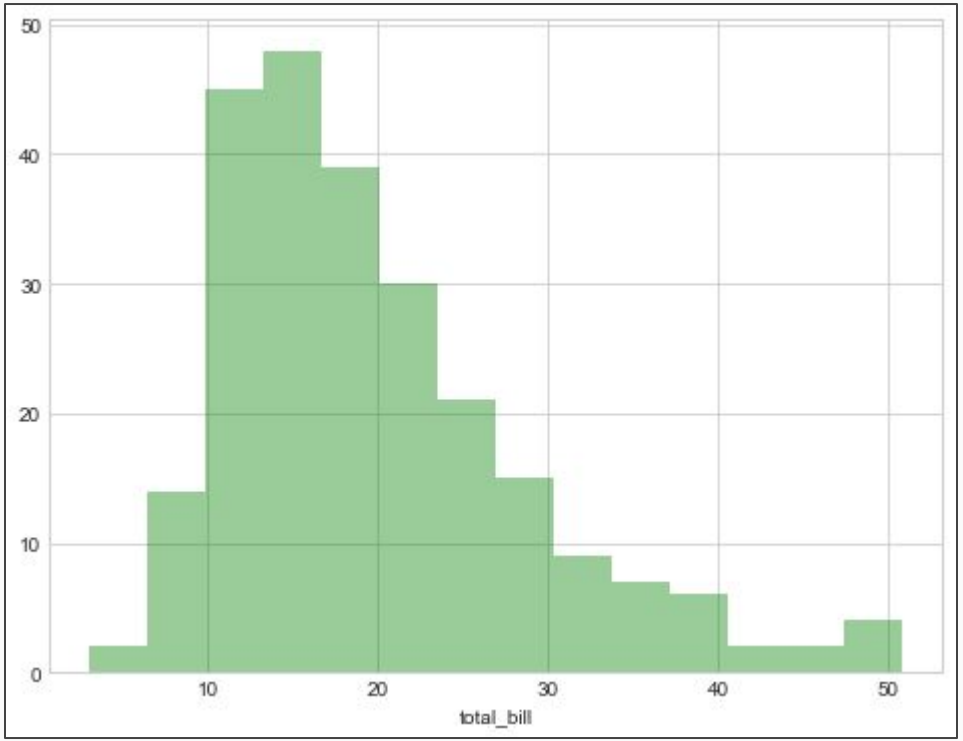
Histogram:

- A histogram represents the distribution of data by forming bins along the range of the data and then drawing bars to show the number of observations that fall in each bin.

```
sns.distplot(tips.total_bill, kde=False, color='g')
```

distplot() → Shows the distribution of data using bins or bars

Output:



Histogram:

- Kernel Density Estimation: It can be a useful tool for plotting the shape of a distribution

```
sns.distplot(tips.total_bill, hist=False, color='m')
```

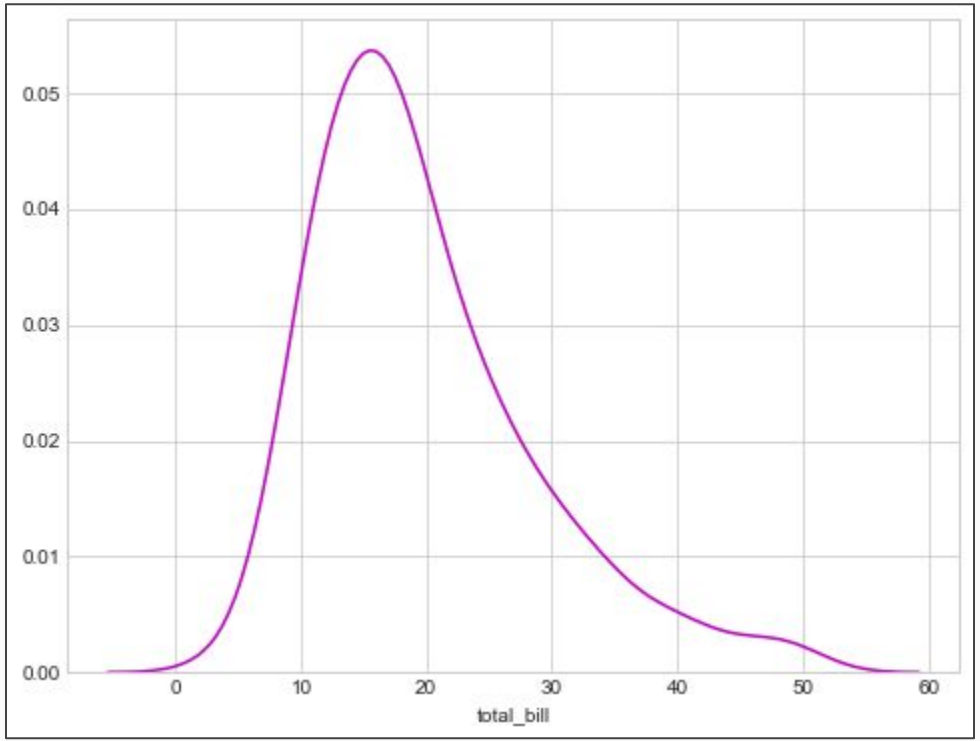
hist=false → This shows the KDE plot

Alternatively, kde plot can be shown by:

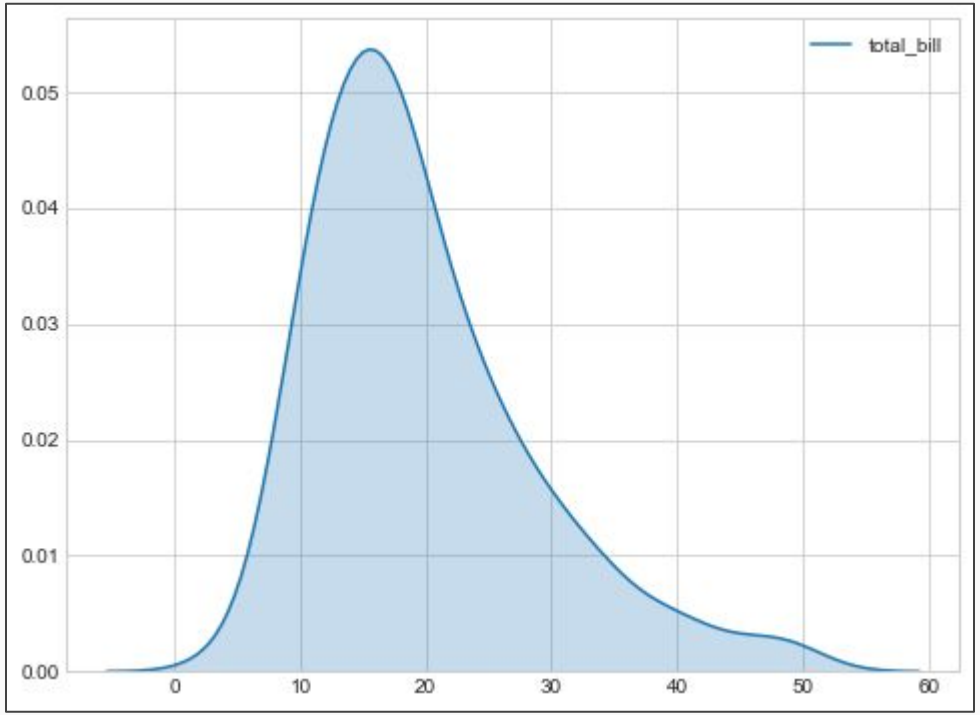
```
sns.kdeplot(tips.total_bill, shade=True)
```

kdeplot() → This shows the KDE plot

Output: Using distplot()



Output: Using kdeplot()



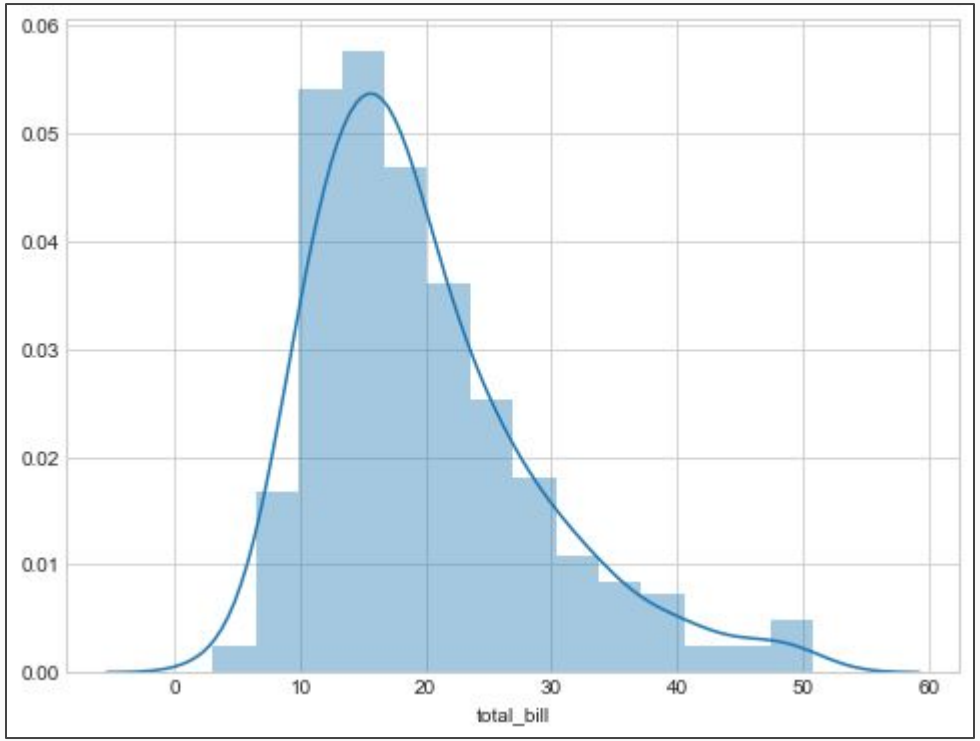
- Showing Histogram and Kernel Density Estimation together in a single figure is possible

sns.distplot(tips.total_bill)

Using just distplot() function, retrieves the histogram and kde plot together

Seaborn

Output:



Joint Plot: Plotting bivariate distributions:

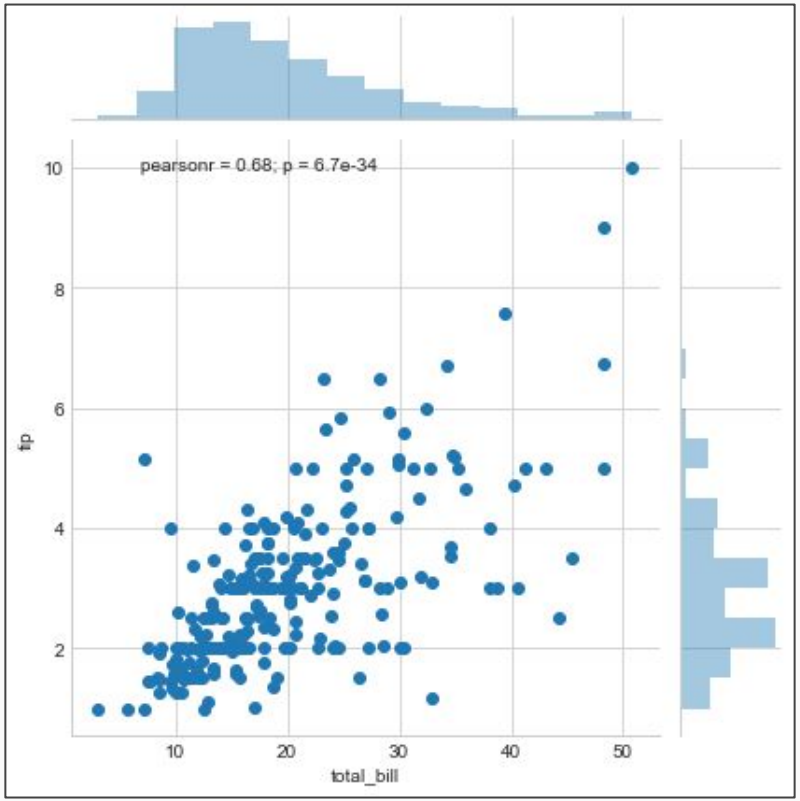
- The jointplot() function, which creates a multi-panel figure that shows both the bivariate (or joint) relationship between two variables along with the univariate (or marginal) distribution of each on separate axes i.e it shows the scatter plot and histograms together.

```
sns.jointplot(x='total_bill', y='tip', data=tips, size=6)
```

jointplot() → This shows the scatter plot and histogram together in the same figure i.e it used for plotting bivariate distributions

Seaborn

Output:



Visualizing pairwise relationships in a dataset:

Pairplot:

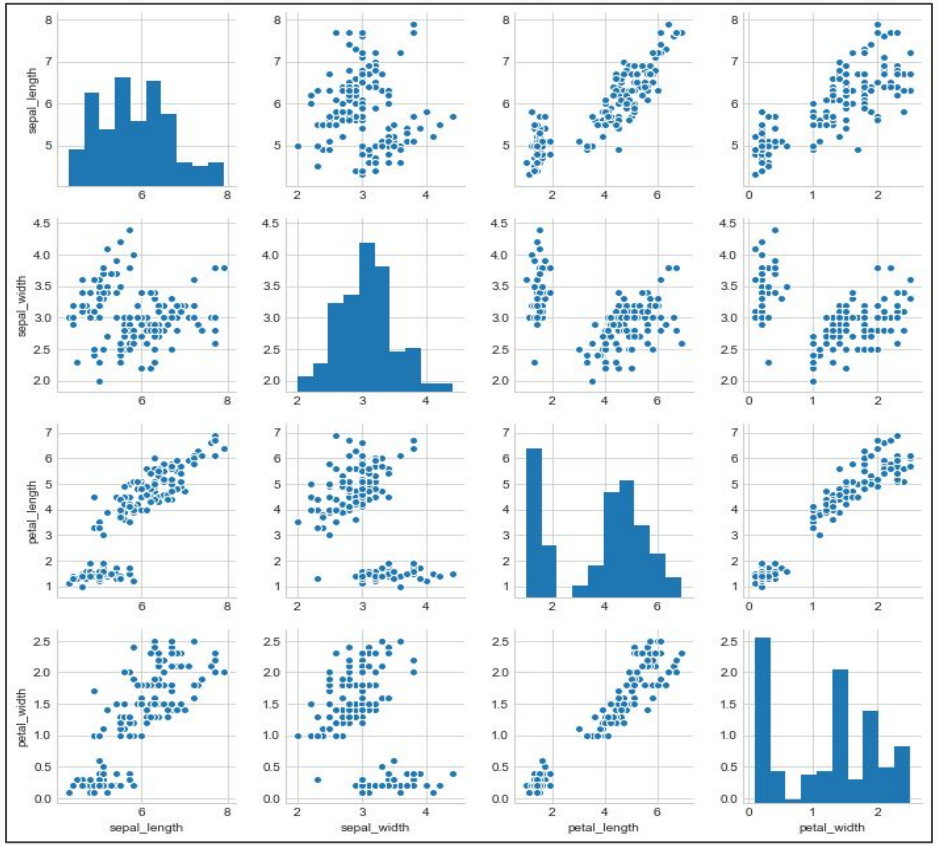
- To plot multiple pairwise bivariate distributions in a dataset, you can use the pairplot() function. This creates a matrix of axes and shows the relationship for each pair of columns in a DataFrame.
- By default, it also draws the univariate distribution of each variable on the diagonal Axes

```
iris = sns.load_dataset("iris")
sns.pairplot(iris, palette='ocean')
```

pairplot() → This shows the multiple pairwise bivariate distributions

Seaborn

Output:



Visualizing linear relationships:

- Functions to draw linear regression models:
- Two main functions in seaborn are used to visualize a linear relationship as determined through regression. These functions are **regplot()** and **lmplot()**

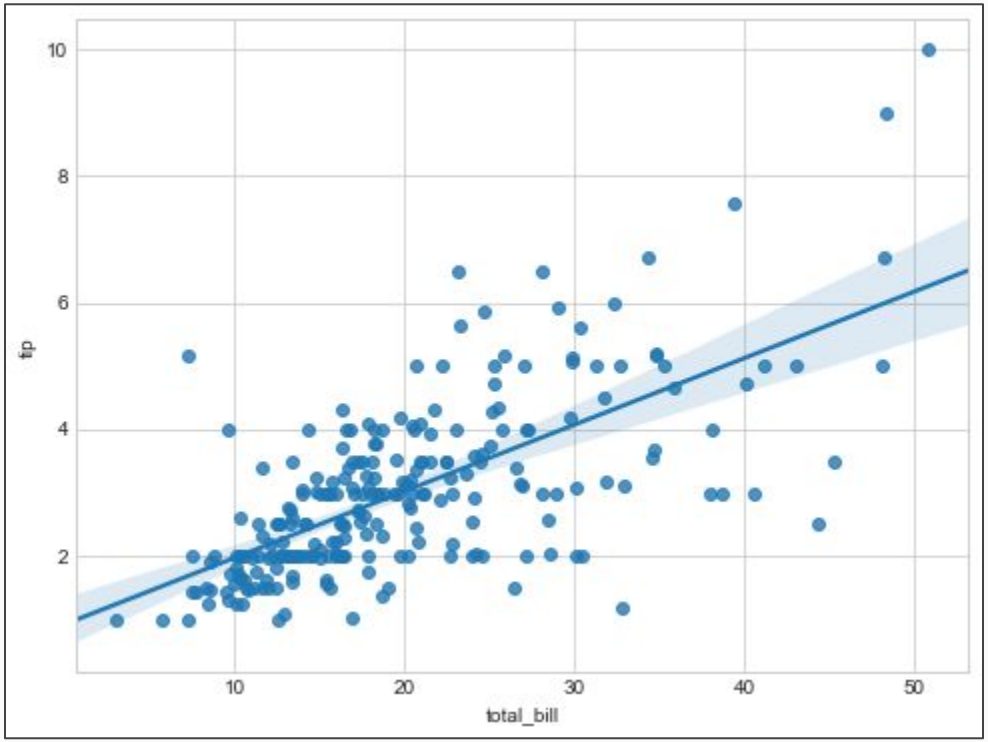
```
sns.regplot(x='total_bill', y='tip', data=tips)
```

regplot() → It is used to show the linear regression model

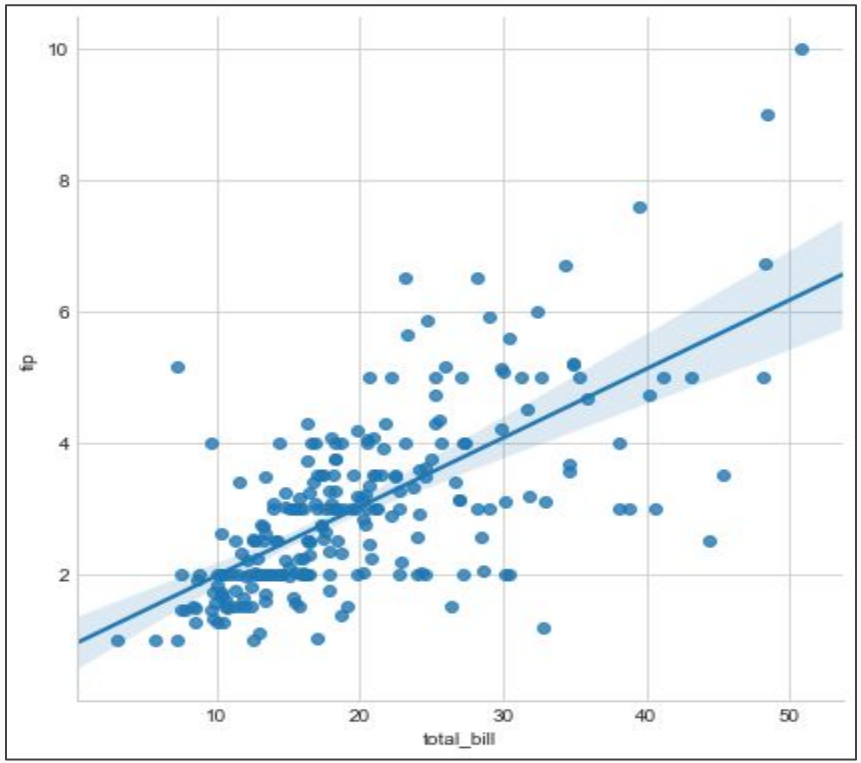
```
sns.lmplot(x='total_bill', y='tip', data=tips, size=6)
```

lmplot() → It is used to show the linear regression model

Output: Using regplot()



Output: Using lmplot()



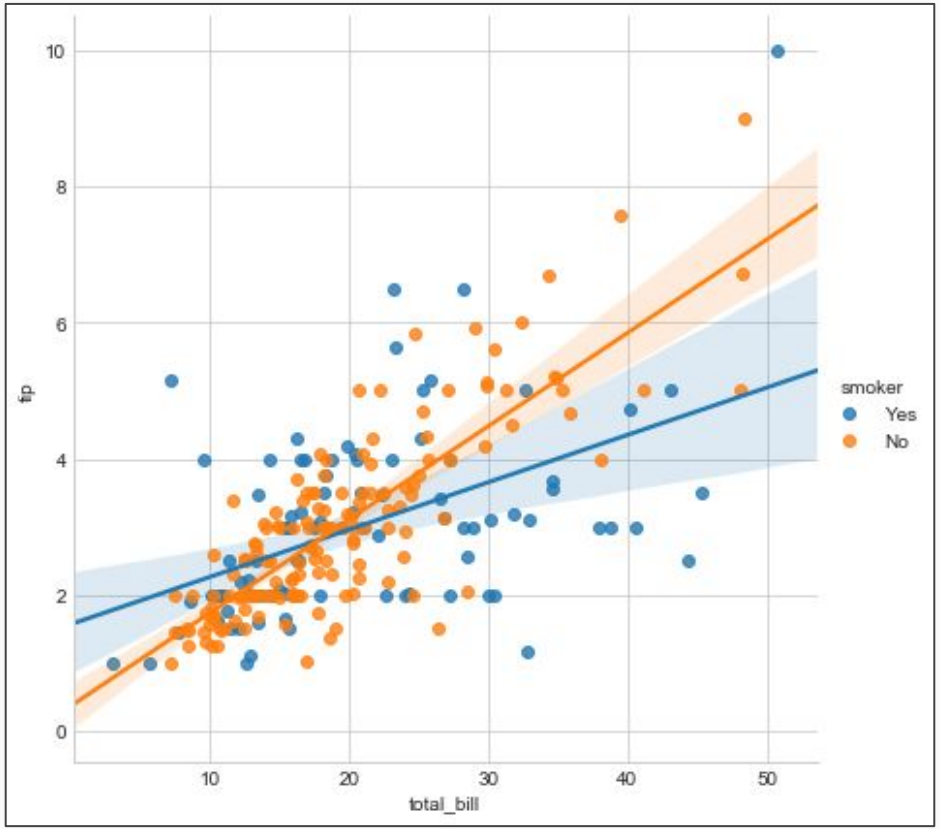
Visualizing linear relationships:

- Conditioning on other variables using hue parameter:

```
sns.lmplot(x='total_bill', y='tip', hue='smoker', data=tips, size=6)
```

Seaborn

Output:



Visualizing linear relationships:

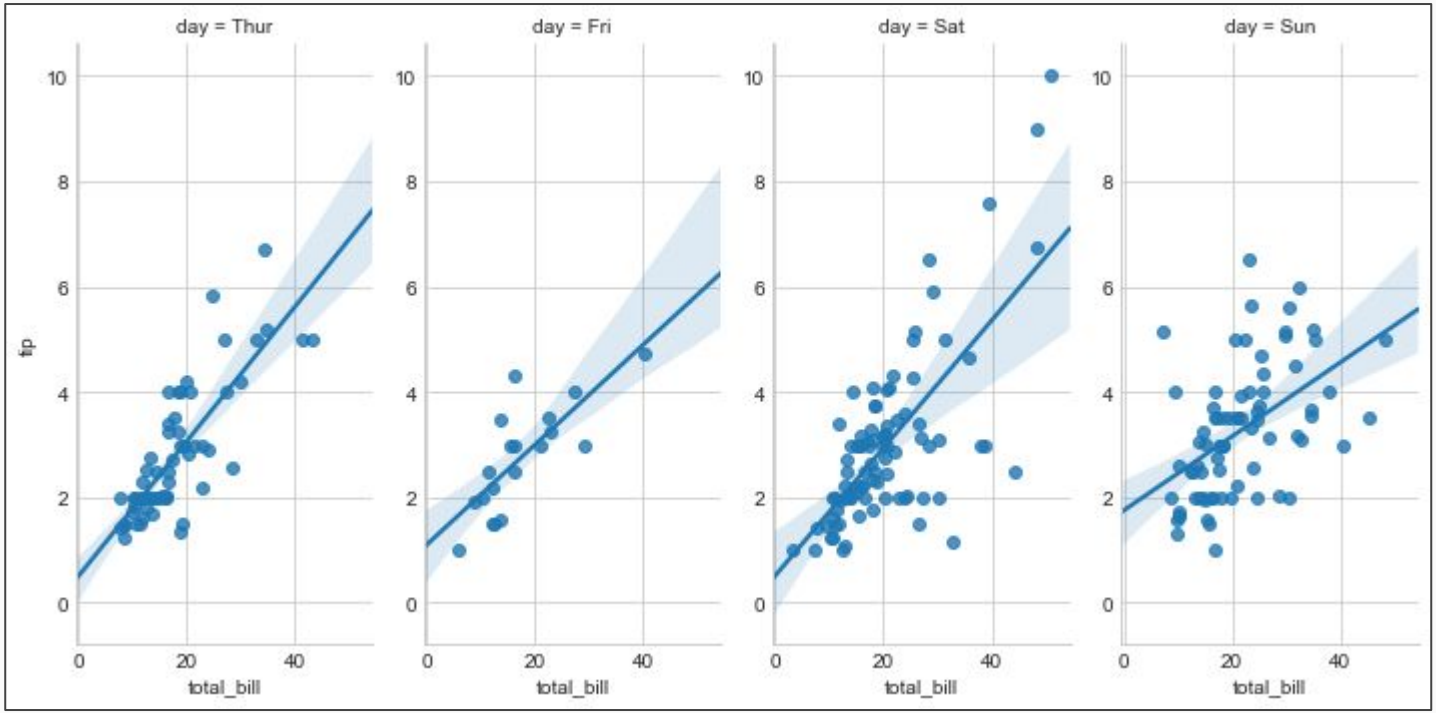
- Conditioning on other variables using hue parameter:
- Controlling the size and shape of the plot:

```
sns.lmplot(x='total_bill', y='tip', col='day', data=tips, aspect=.5)
```

aspect → Controls the size and shape of the plot

Seaborn

Output:



Visualizing linear relationships:

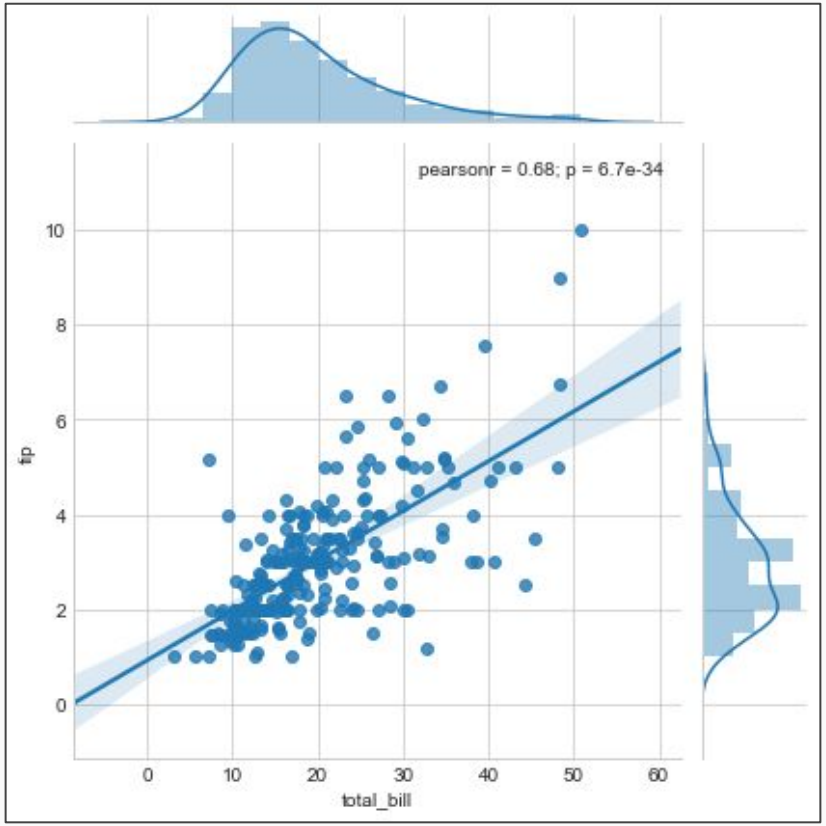
- Plotting a regression in other contexts:

```
sns.jointplot(x='total_bill', y='tip', data=tips, kind='reg')
```

jointplot() → Shows the scatter plot and histogram together
kind = 'reg' → This provides linear regression model which gives extra information of dataset

Seaborn

Output:



Heat Map:

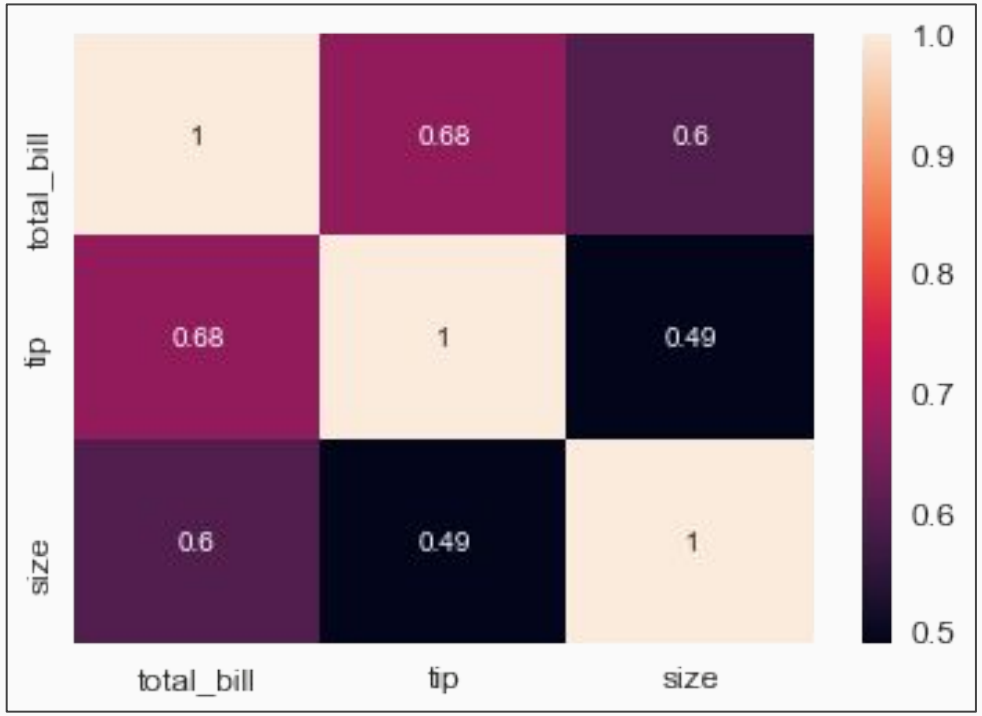
- A heatmap is a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colors. The heatmap() function allows the creation of annotated heatmaps.

```
sns.heatmap(tips.corr(), annot = True)
```

heatmap() → This method is used to plot rectangular data as a color-encoded matrix. Here annot = True indicates we want Annotated heatmap. Heatmap is used to create a color-encoded matrix to visualize correlation matrix or confusion matrix

Seaborn

Output:



Thanks!

Signitive Technologies

Sneh Nagar, Behind ICICI Bank,
Chhatrapati Square, Nagpur 15

Landmark:
Bharat Petrol Pump
Chhatrapati Square

Contact: 9011033776
www.signitivetech.com

“Keep Learning, Happy Learning”

Best Luck!

Have a Happy Future