Seaborn

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

Distribution Plots

Let's discuss some plots that allow us to visualize the distribution of a data set. These plots are:

- distplot
- jointplot
- pairplot
- rugplot
- kdeplot

Imports

In [ ]:
```
import seaborn as sns
%matplotlib inline
```

Data

Seaborn comes with built-in data sets!

In [6]:
```
tips = sns.load_dataset('tips')
```

In [7]:
```
tips.head()
```

Out[7]:

	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

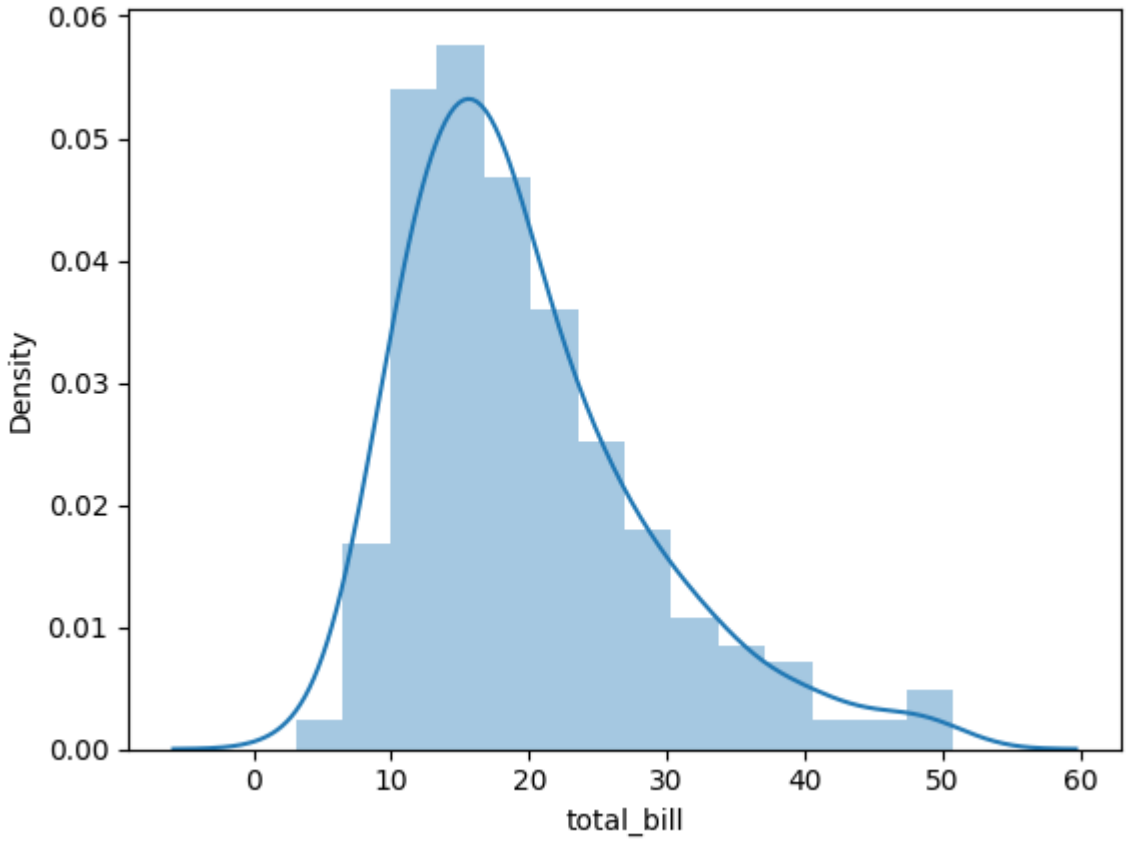
distplot

The distplot shows the distribution of a univariate set of observations.

In [17]:
```
sns.distplot(tips['total_bill'])
# Safe to ignore warnings
```

```
C:\Users\jonrey\AppData\Local\Temp\ipykernel_7464\1093071496.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.
github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(tips['total_bill'])
```

Out[17]:
```
<AxesSubplot:xlabel='total_bill', ylabel='Density'>
```

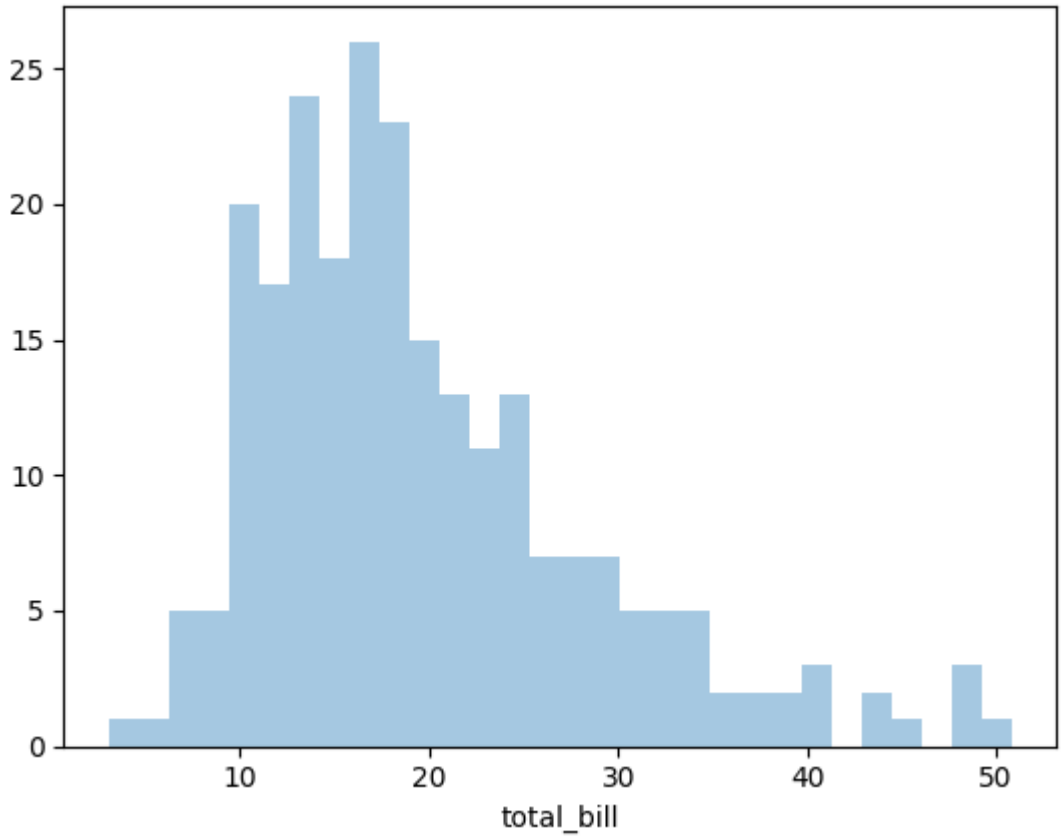


To remove the kde layer and just have the histogram use:

In [9]:
```
sns.distplot(tips['total_bill'],kde=False,bins=30)
```

```
C:\Users\jonrey\AppData\Local\Temp\ipykernel_7464\1274391954.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.
github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(tips['total_bill'],kde=False,bins=30)
```

Out[9]:
```
<AxesSubplot:xlabel='total_bill'>
```



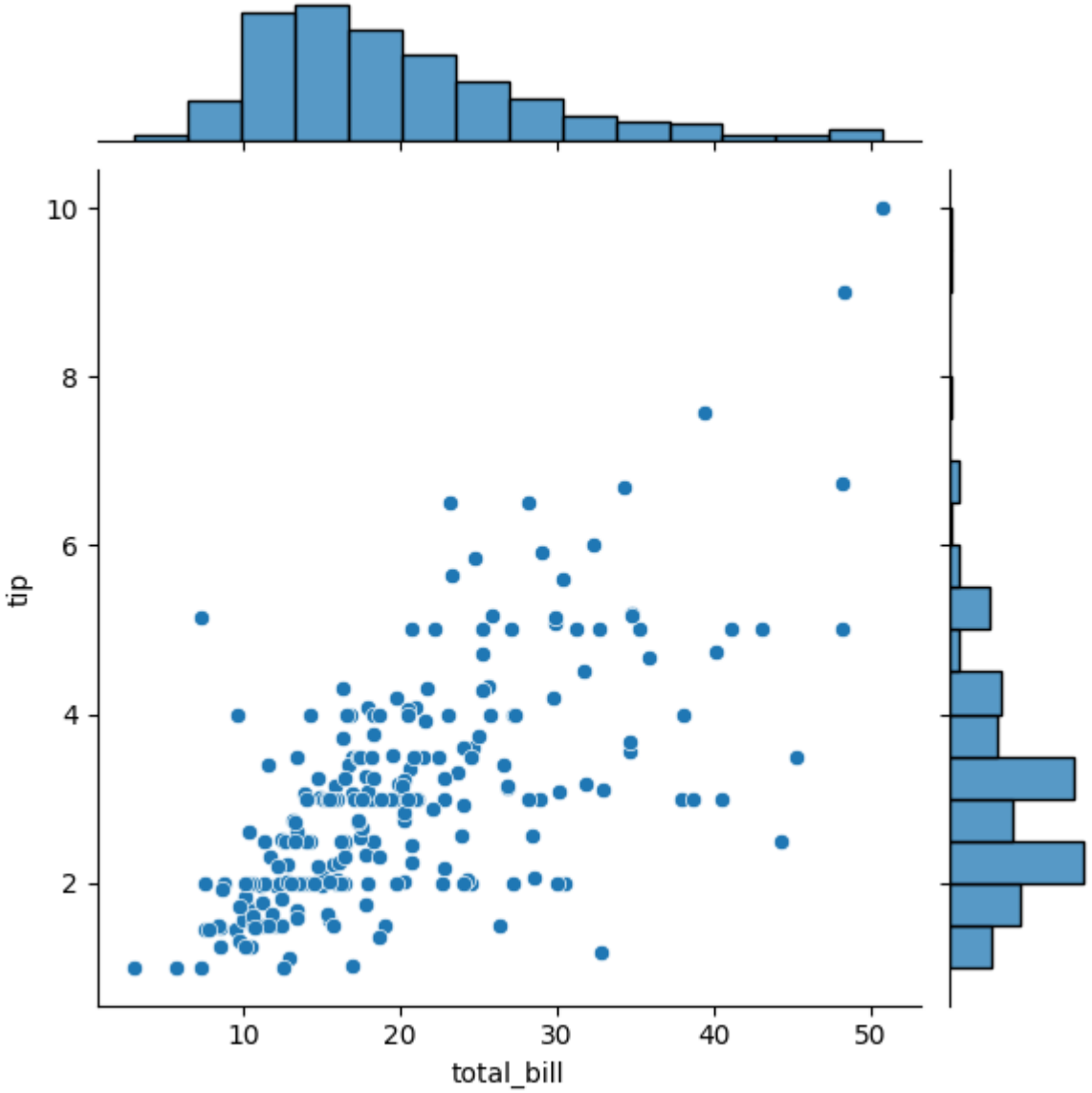
jointplot

jointplot() allows you to basically match up two distplots for bivariate data. With your choice of what **kind** parameter to compare with:

- “scatter”
- “reg”
- “resid”
- “kde”
- “hex”

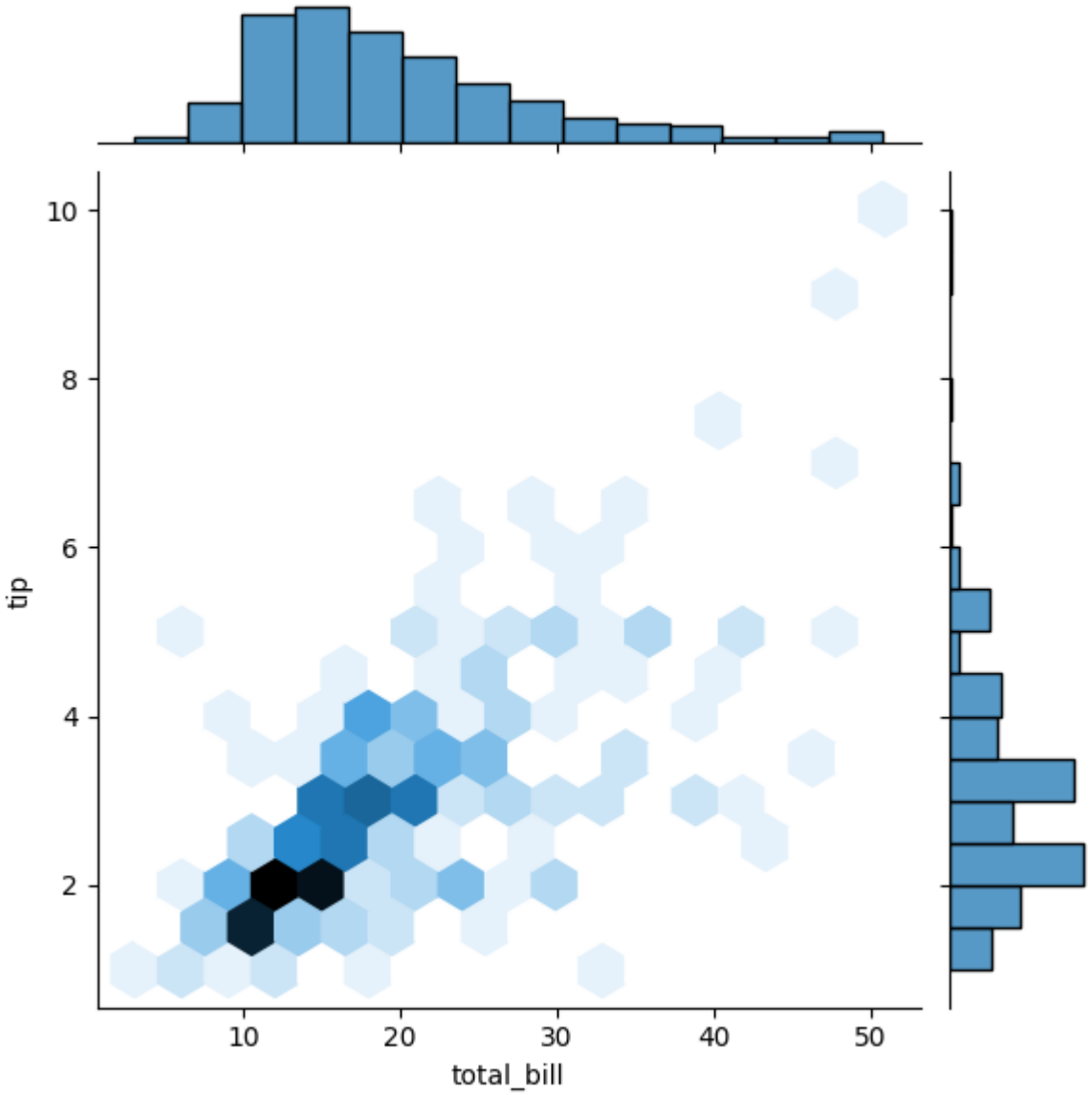
In [10]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='scatter')
```

Out[10]:
```
<seaborn.axisgrid.JointGrid at 0xf756bcef10>
```



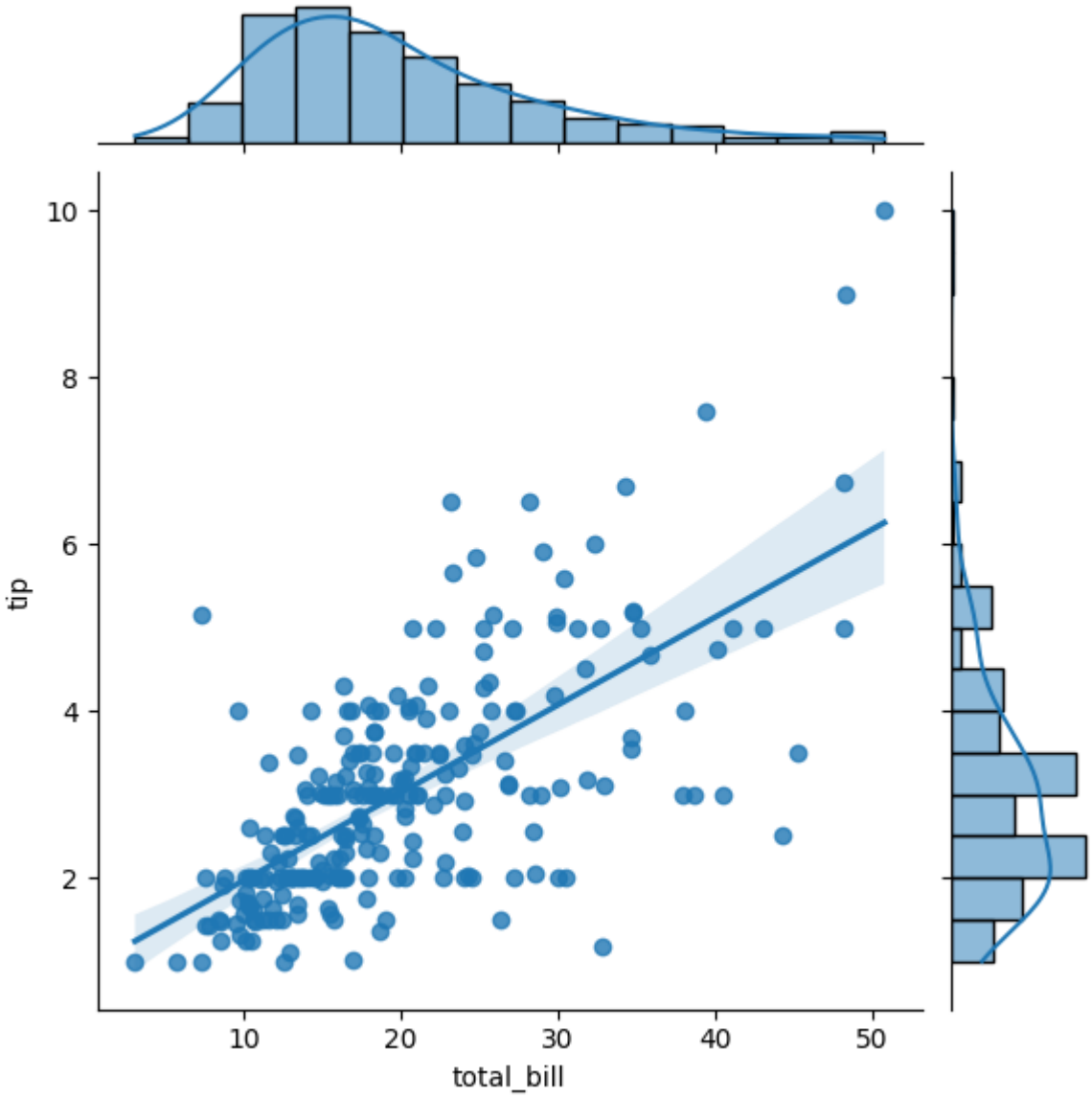
In [11]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='hex')
```

Out[11]:
```
<seaborn.axisgrid.JointGrid at 0xf757dc5070>
```



In [12]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='reg')
```

Out[12]:
```
<seaborn.axisgrid.JointGrid at 0xf7581dfd60>
```

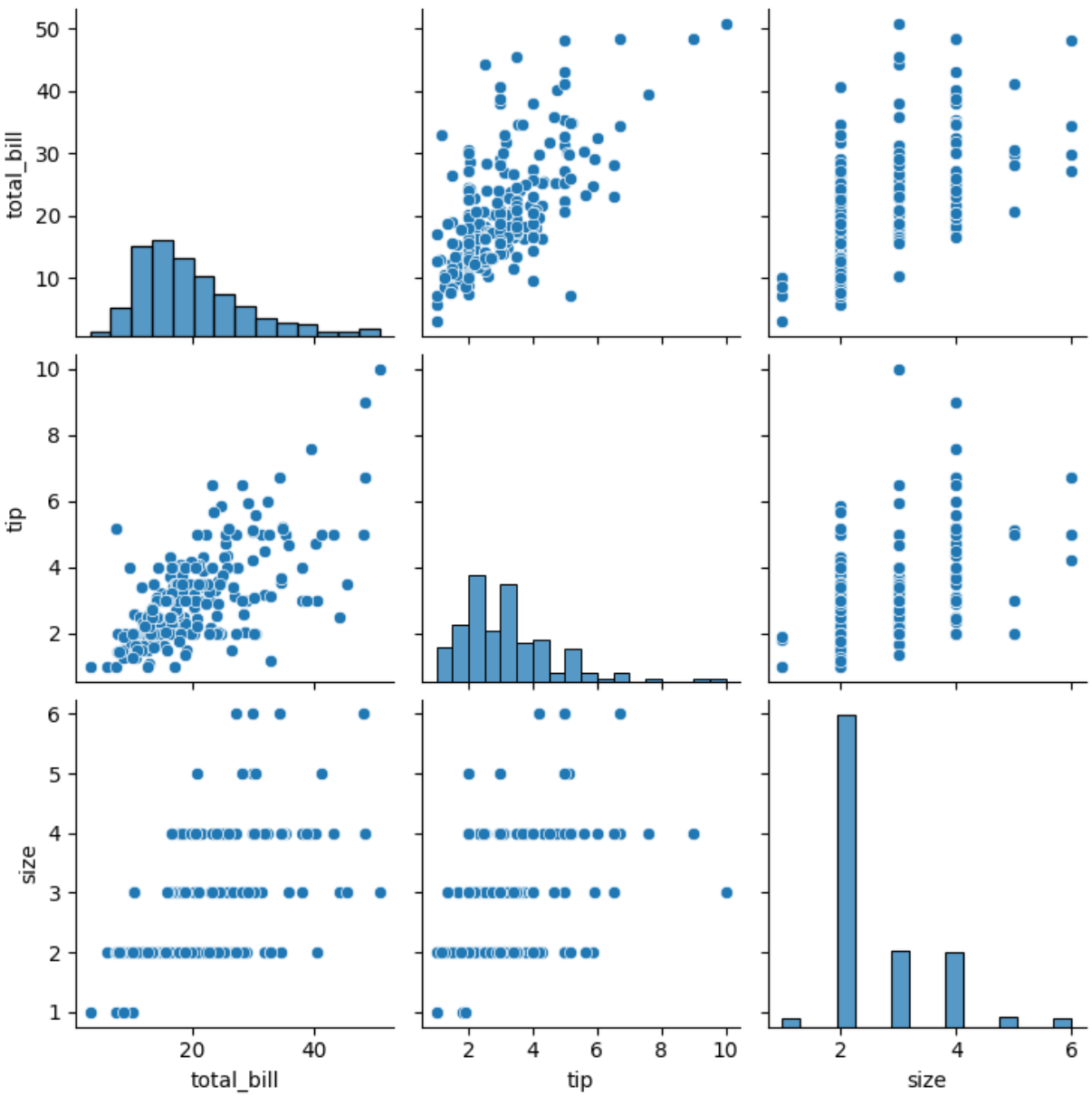


pairplot

pairplot will plot pairwise relationships across an entire dataframe (for the numerical columns) and supports a color hue argument (for categorical columns).

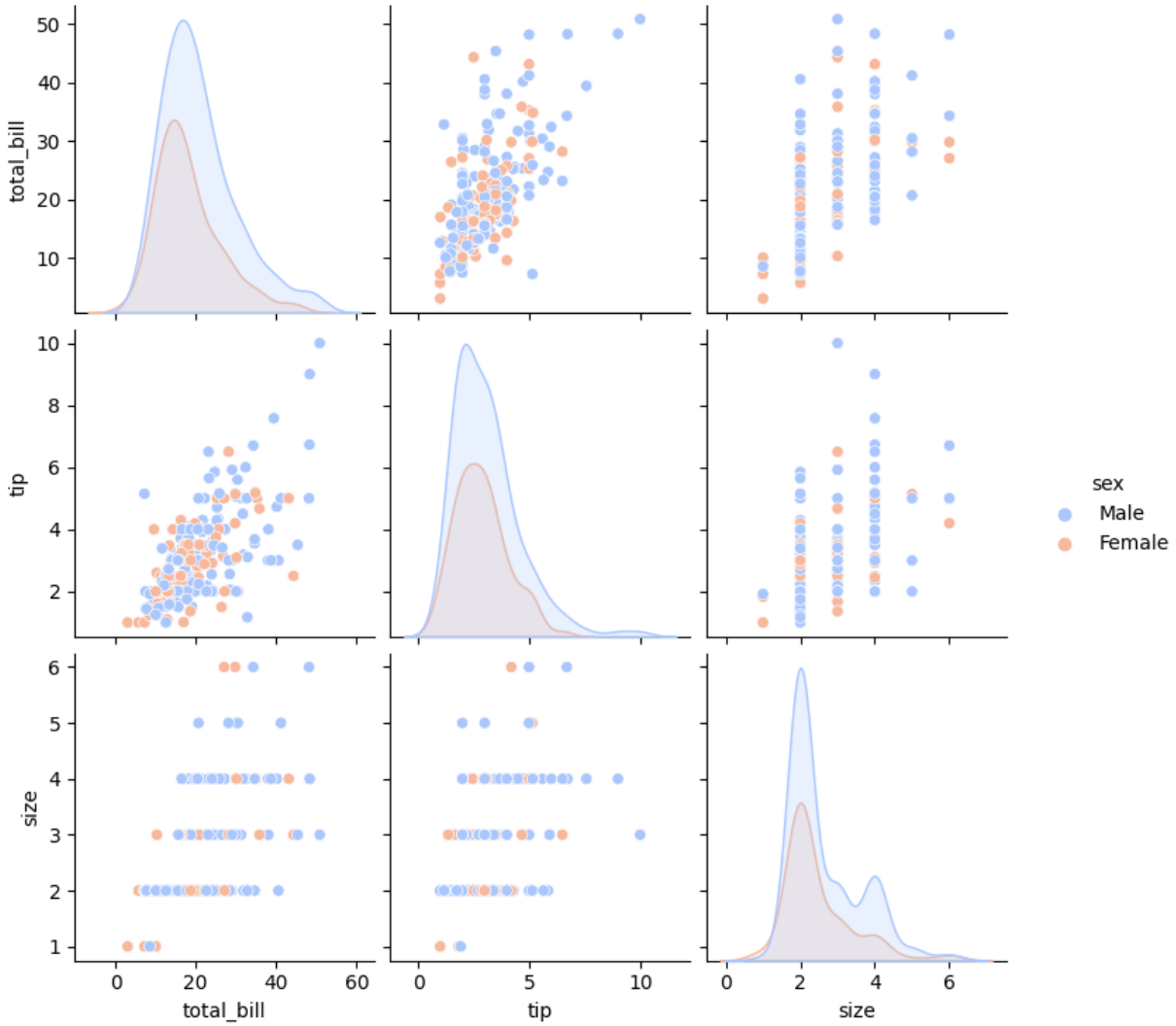
In [13]:
```
sns.pairplot(tips)
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0xf7581d1700>
```



In [14]:
```
sns.pairplot(tips,hue='sex',palette='coolwarm')
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0xf74b98dd30>
```

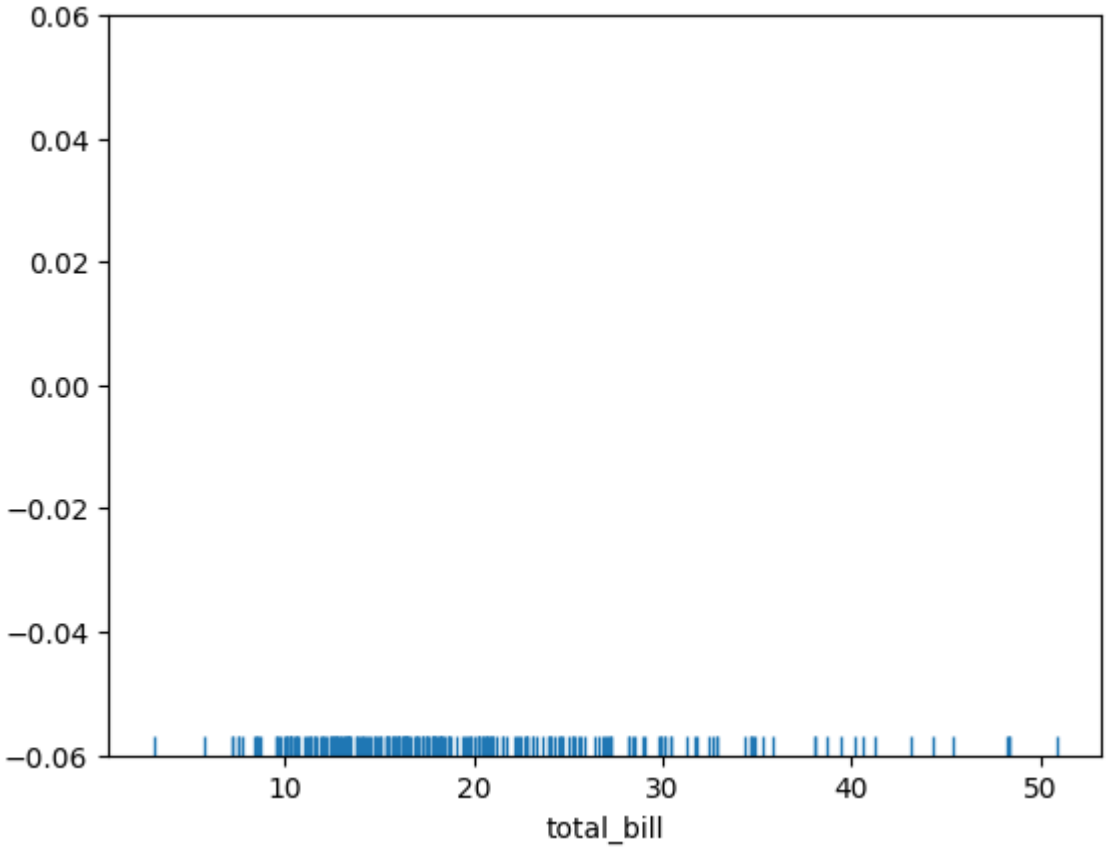


rugplot

rugplots are actually a very simple concept, they just draw a dash mark for every point on a univariate distribution. They are the building block of a KDE plot:

In [15]:
```
sns.rugplot(tips['total_bill'])
```

Out[15]:
```
<AxesSubplot:xlabel='total_bill'>
```

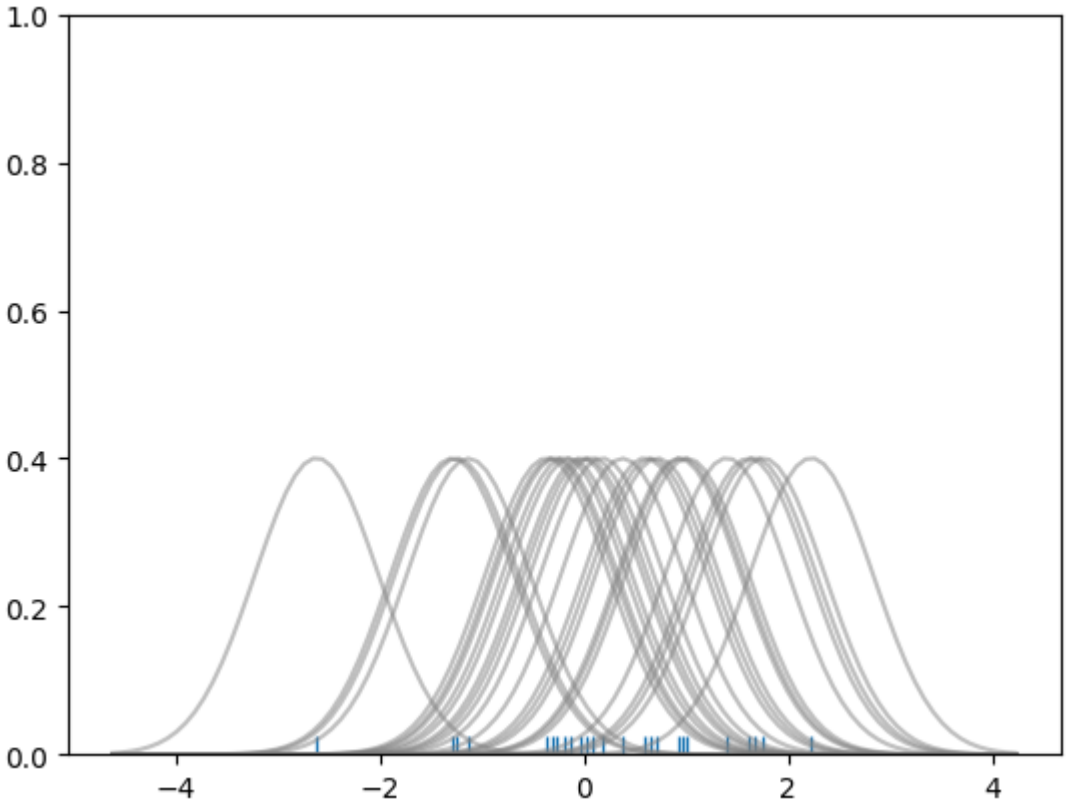


kdeplot

kdeplots are [Kernel Density Estimation plots (http://en.wikipedia.org/wiki/Kernel_density_estimation#Practical_estimation_of_the_bandwidth)](). These KDE plots replace every single observation with a Gaussian (Normal) distribution centered around that value. For example:

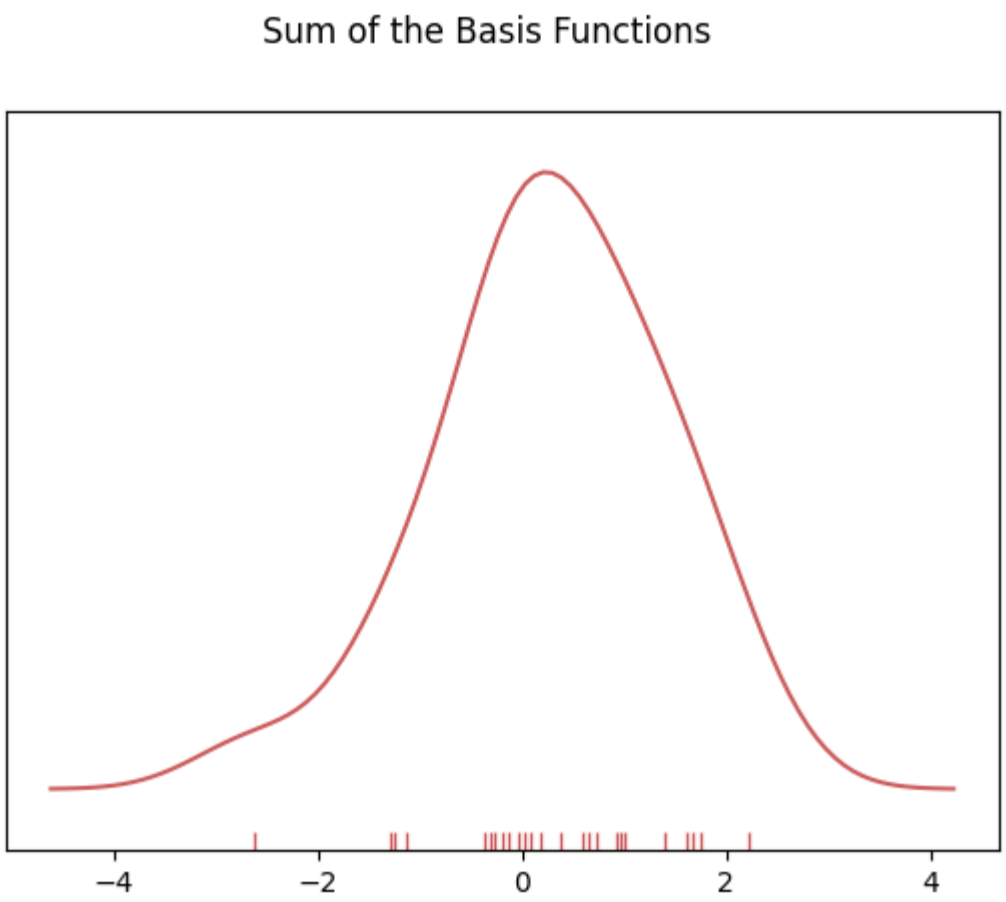
In [20]:
```
# Don't worry about understanding this code!
# It's just for the diagram below
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

#Create dataset
dataset = np.random.randn(25)

# Create another rugplot
sns.rugplot(dataset);

# Set up the x-axis for the plot
x_min = dataset.min() - 2
x_max = dataset.max() + 2

# 100 equally spaced points from x_min to x_max
x_axis = np.linspace(x_min,x_max,100)

# Set up the bandwidth, for info on this:
url = 'http://en.wikipedia.org/wiki/Kernel_density_estimation#Practical_estimatic

bandwidth = ((4*dataset.std()**5)/(3*len(dataset)))**.2


# Create an empty kernel list
kernel_list = []

# Plot each basis function
for data_point in dataset:

    # Create a kernel for each point and append to list
    kernel = stats.norm(data_point,bandwidth).pdf(x_axis)
    kernel_list.append(kernel)

    #Scale for plotting
    kernel = kernel / kernel.max()
    kernel = kernel * .4
    plt.plot(x_axis,kernel,color = 'grey',alpha=0.5)

plt.ylim(0,1)
```

Out[20]:
```
(0.0, 1.0)
```


1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

In [21]:
```
# To get the kde plot we can sum these basis functions.

# Plot the sum of the basis function
sum_of_kde = np.sum(kernel_list,axis=0)

# Plot figure
fig = plt.plot(x_axis,sum_of_kde,color='indianred')

# Add the initial rugplot
sns.rugplot(dataset,c = 'indianred')

# Get rid of y-tick marks
plt.yticks([])

# Set title
plt.suptitle("Sum of the Basis Functions")
```

Out[21]:
```
Text(0.5, 0.98, 'Sum of the Basis Functions')
```



So with our tips dataset:

In [ ]:
```
sns.kdeplot(tips['total_bill'])
sns.rugplot(tips['total_bill'])
```

In [ ]:
```
sns.kdeplot(tips['tip'])
sns.rugplot(tips['tip'])
```

Great Job!