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 [1]:

```
import seaborn as sns
%matplotlib inline
```

In [ ]:

```

```

Data

Seaborn comes with built-in data sets!

In [2]:

```
tips = sns.load_dataset('tips')
```

In [4]:

```
1 tips
```

Out[4]:

	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
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2
10	10.27	1.71	Male	No	Sun	Dinner	2
11	35.26	5.00	Female	No	Sun	Dinner	4
12	15.42	1.57	Male	No	Sun	Dinner	2
13	18.43	3.00	Male	No	Sun	Dinner	4
14	14.83	3.02	Female	No	Sun	Dinner	2
15	21.58	3.92	Male	No	Sun	Dinner	2
16	10.33	1.67	Female	No	Sun	Dinner	3
17	16.29	3.71	Male	No	Sun	Dinner	3
18	16.97	3.50	Female	No	Sun	Dinner	3
19	20.65	3.35	Male	No	Sat	Dinner	3
20	17.92	4.08	Male	No	Sat	Dinner	2
21	20.29	2.75	Female	No	Sat	Dinner	2
22	15.77	2.23	Female	No	Sat	Dinner	2
23	39.42	7.58	Male	No	Sat	Dinner	4
24	19.82	3.18	Male	No	Sat	Dinner	2
25	17.81	2.34	Male	No	Sat	Dinner	4
26	13.37	2.00	Male	No	Sat	Dinner	2
27	12.69	2.00	Male	No	Sat	Dinner	2
28	21.70	4.30	Male	No	Sat	Dinner	2
29	19.65	3.00	Female	No	Sat	Dinner	2
...	...	...	...	...	...	...	...
214	28.17	6.50	Female	Yes	Sat	Dinner	3
215	12.90	1.10	Female	Yes	Sat	Dinner	2
216	28.15	3.00	Male	Yes	Sat	Dinner	5

	total_bill	tip	sex	smoker	day	time	size
217	11.59	1.50	Male	Yes	Sat	Dinner	2
218	7.74	1.44	Male	Yes	Sat	Dinner	2
219	30.14	3.09	Female	Yes	Sat	Dinner	4
220	12.16	2.20	Male	Yes	Fri	Lunch	2
221	13.42	3.48	Female	Yes	Fri	Lunch	2
222	8.58	1.92	Male	Yes	Fri	Lunch	1
223	15.98	3.00	Female	No	Fri	Lunch	3
224	13.42	1.58	Male	Yes	Fri	Lunch	2
225	16.27	2.50	Female	Yes	Fri	Lunch	2
226	10.09	2.00	Female	Yes	Fri	Lunch	2
227	20.45	3.00	Male	No	Sat	Dinner	4
228	13.28	2.72	Male	No	Sat	Dinner	2
229	22.12	2.88	Female	Yes	Sat	Dinner	2
230	24.01	2.00	Male	Yes	Sat	Dinner	4
231	15.69	3.00	Male	Yes	Sat	Dinner	3
232	11.61	3.39	Male	No	Sat	Dinner	2
233	10.77	1.47	Male	No	Sat	Dinner	2
234	15.53	3.00	Male	Yes	Sat	Dinner	2
235	10.07	1.25	Male	No	Sat	Dinner	2
236	12.60	1.00	Male	Yes	Sat	Dinner	2
237	32.83	1.17	Male	Yes	Sat	Dinner	2
238	35.83	4.67	Female	No	Sat	Dinner	3
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [6]:

```
1 tips.head(10)
```

Out[6]:

	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
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2

In [9]:

```
1 tips.tail(10)
```

Out[9]:

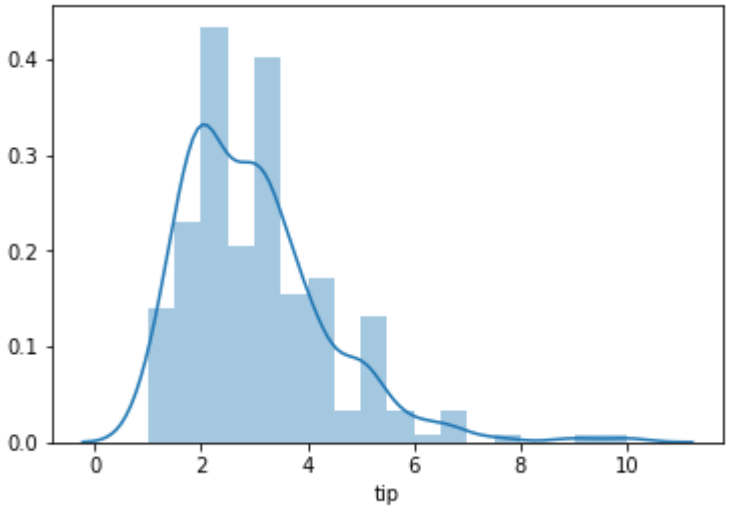
	total_bill	tip	sex	smoker	day	time	size
234	15.53	3.00	Male	Yes	Sat	Dinner	2
235	10.07	1.25	Male	No	Sat	Dinner	2
236	12.60	1.00	Male	Yes	Sat	Dinner	2
237	32.83	1.17	Male	Yes	Sat	Dinner	2
238	35.83	4.67	Female	No	Sat	Dinner	3
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

In [11]:

```
1 sns.distplot(tips['tip'])
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66f5d62b70>
```

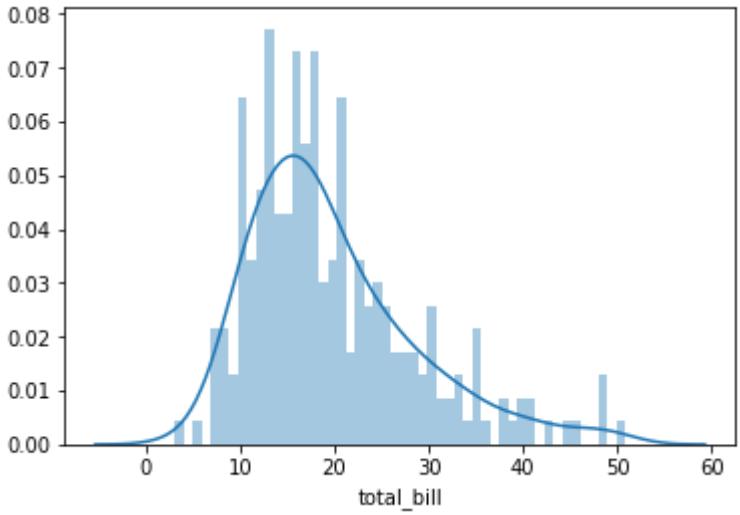


In [15]:

```
1 sns.distplot(tips['total_bill'],bins=50)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66f563ada0>
```



distplot

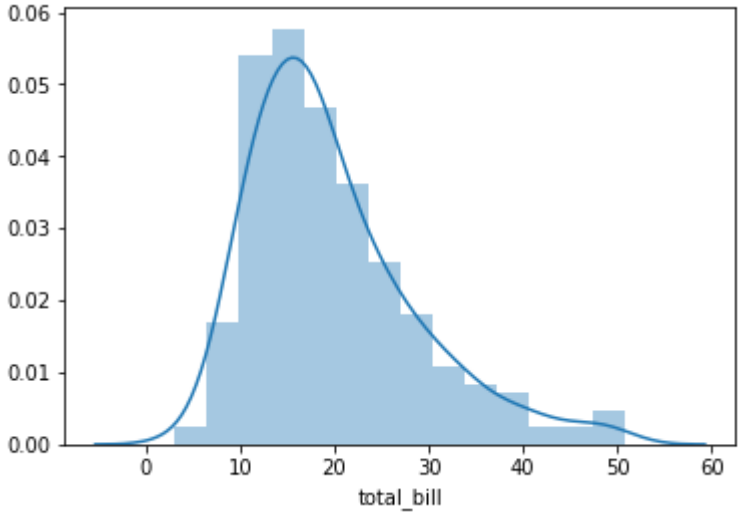
The distplot shows the distribution of a univariate set of observations.

In [18]:

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

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66f558e748>
```

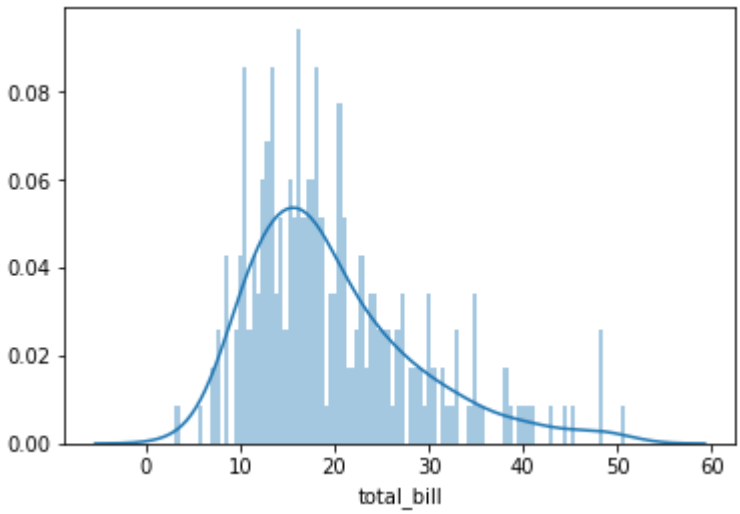


In [20]:

```
sns.distplot(tips['total_bill'],bins=100)

```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66f5368dd8>
```



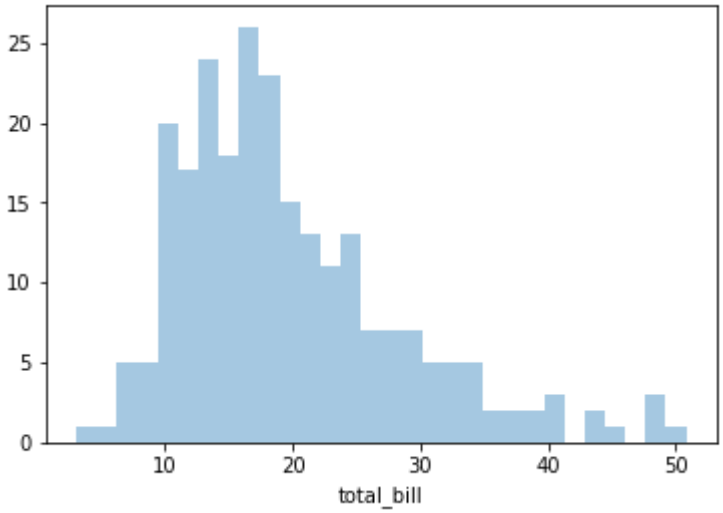
To remove the kde layer and just have the histogram use:

In [16]:

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

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66f558ed30>
```



In [23]:

```
1 tips.head()
```

Out[23]:

	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

In [ ]:

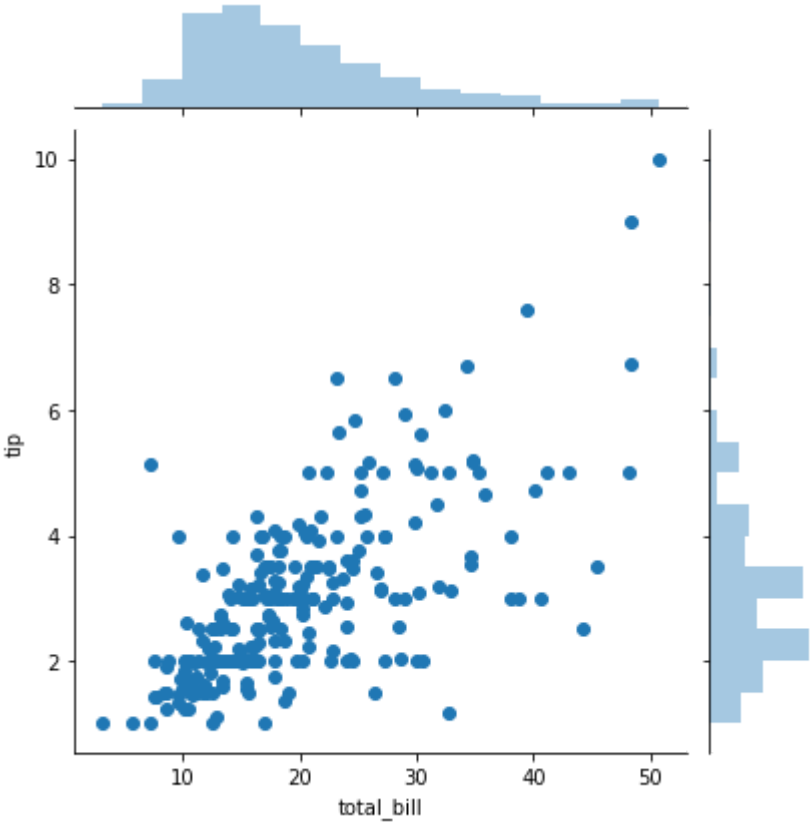
```
1
```

In [24]:

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

Out[24]:

```
<seaborn.axisgrid.JointGrid at 0x7f66f4b2c5c0>
```



jointplot

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

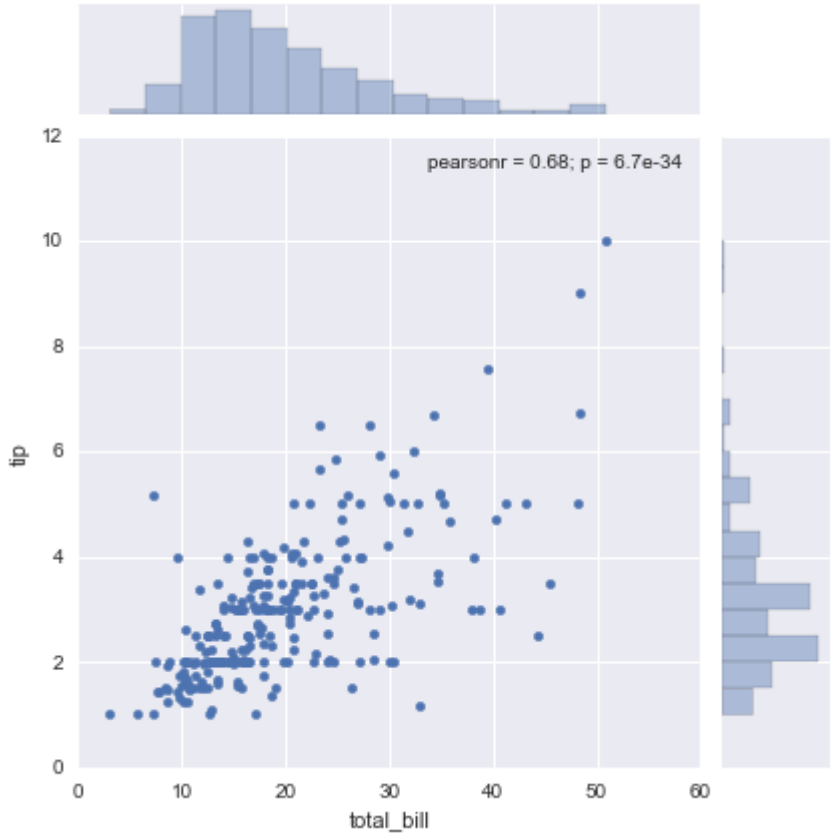
- “scatter”
- “reg”
- “resid”
- “kde”
- “hex”

In [12]:

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

Out[12]:

```
<seaborn.axisgrid.JointGrid at 0x11cfb28d0>
```

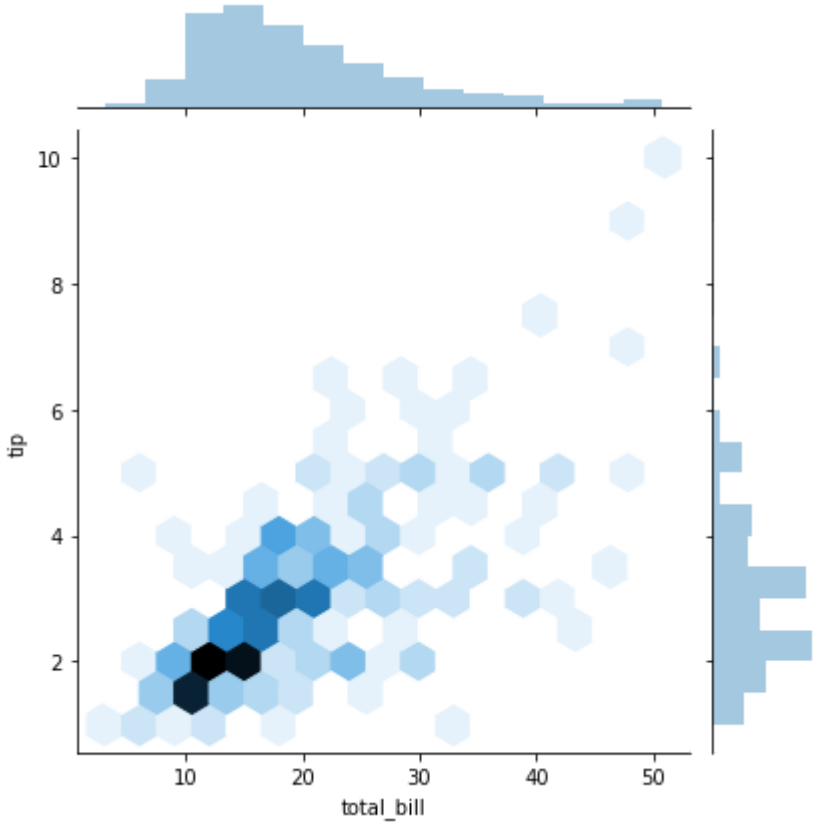


In [25]:

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

Out[25]:

```
<seaborn.axisgrid.JointGrid at 0x7f66f4bd4e80>
```

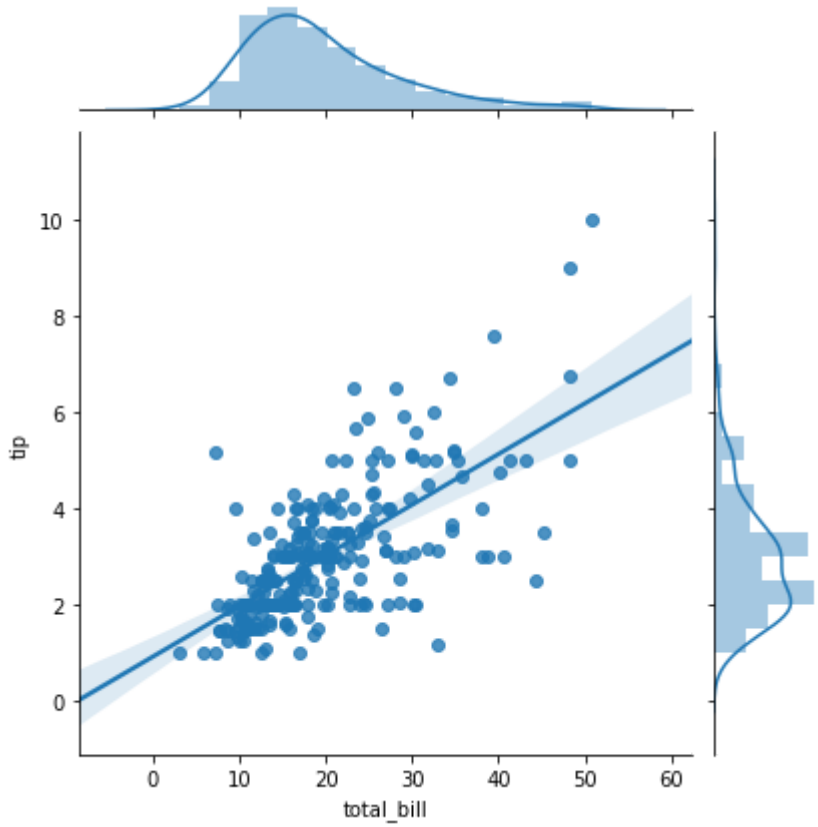


In [26]:

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

Out[26]:

```
<seaborn.axisgrid.JointGrid at 0x7f66f4fdee48>
```



pairplot

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

In [31]:

```
1 tips.head()
```

Out[31]:

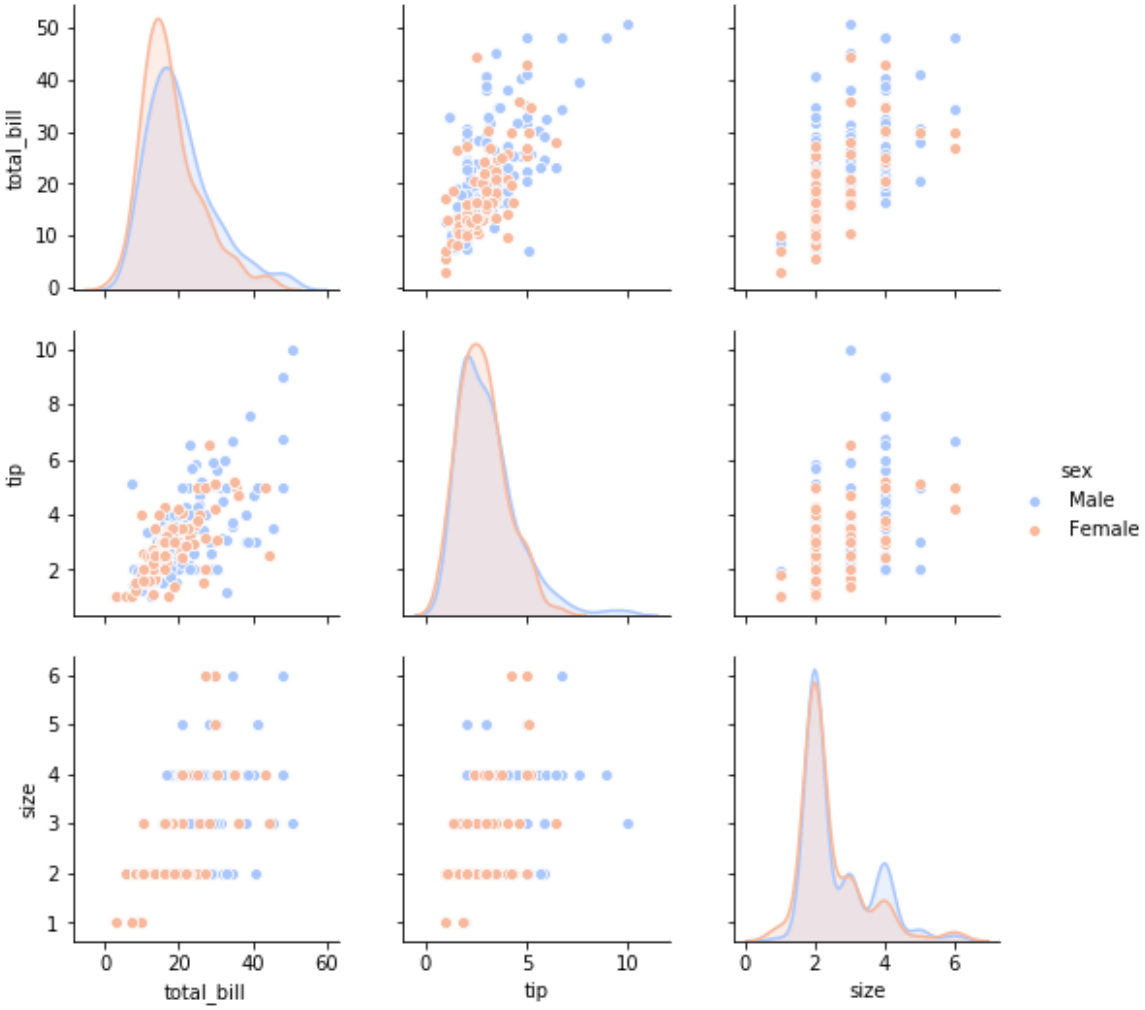
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [30]:

```
1 sns.pairplot(tips,hue='sex',palette='coolwarm')
```

Out[30]:

```
<seaborn.axisgrid.PairGrid at 0x7f66f293b908>
```



In [ ]:

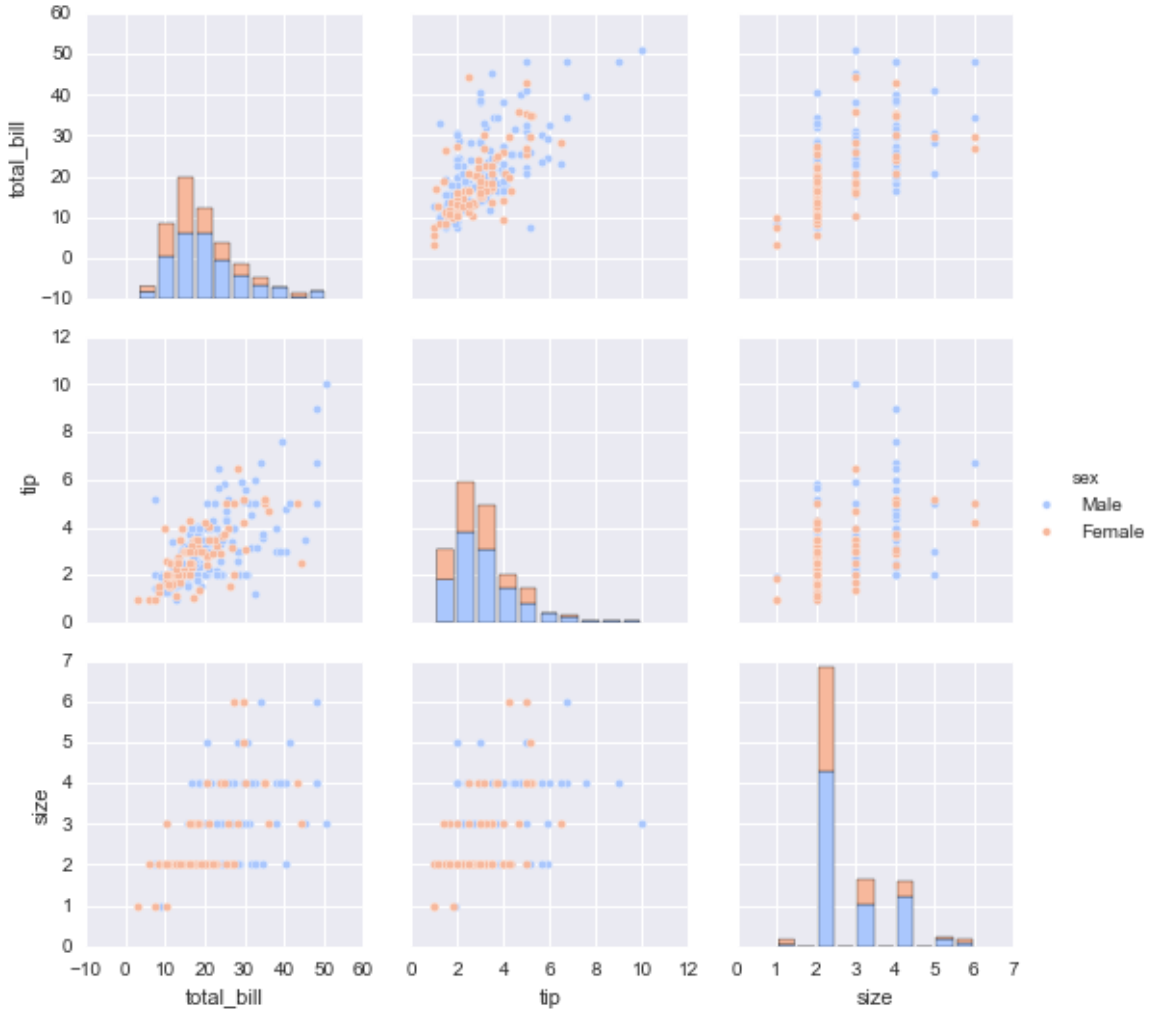
```
1
```

In [21]:

```
1 sns.pairplot(tips,hue='sex',palette='coolwarm')
```

Out[21]:

```
<seaborn.axisgrid.PairGrid at 0x11ff7a828>
```



rugplot

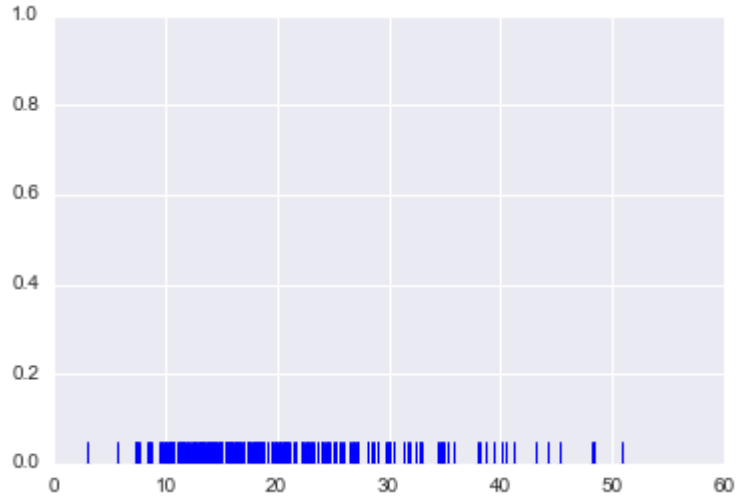
rugplots are actually a very simple concept, they just draw a dash mark for every point on a univariate distribution. They are the building block of a KDE plot:

In [22]:

```
1 sns.rugplot(tips['total_bill'])
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1207c8b70>
```



kdeplot

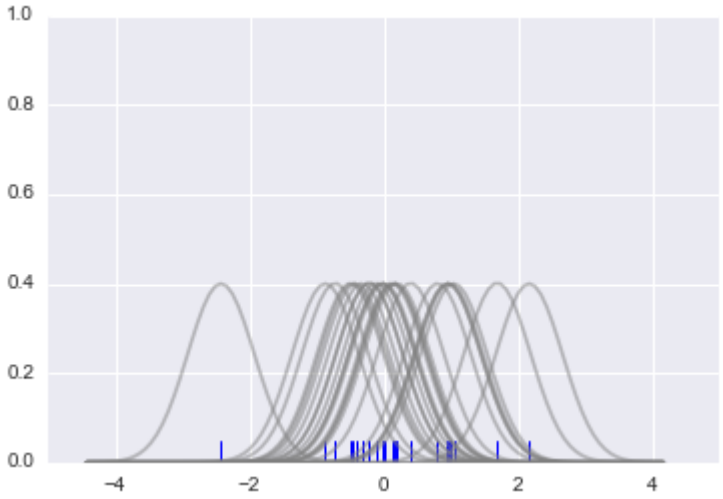
kdeplots are Kernel Density Estimation plots (http://en.wikipedia.org/wiki/Kernel_density_estimation#Practical_estimation_of_the_bandwidth). These KDE plots replace every single observation with a Gaussian (Normal) distribution centered around that value. For example:

In [35]:

```
# 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_estimat

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[35]:

```
(0, 1)
```

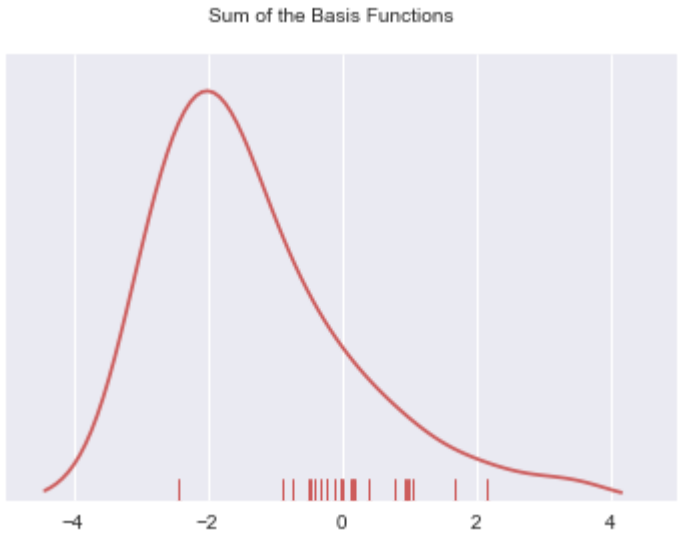


In [37]:

```
# 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[37]:

```
<matplotlib.text.Text at 0x121c41da0>
```



So with our tips dataset:

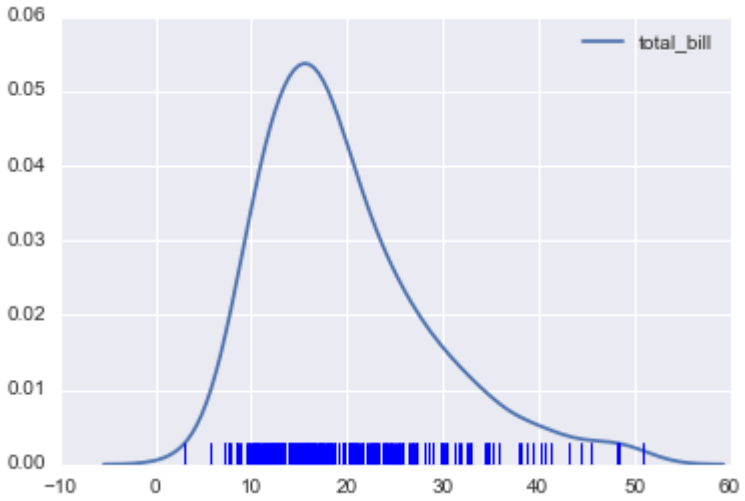
In [41]:

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

```
/Users/marci/anaconda/lib/python3.5/site-packages/statsmodels/nonpara
metric/kdetools.py:20: VisibleDeprecationWarning: using a non-integer
number instead of an integer will result in an error in the future
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x121b82c50>
```



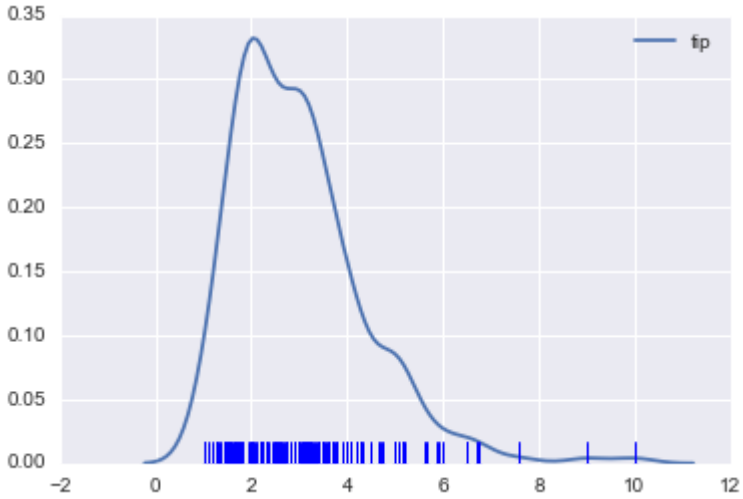
In [42]:

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

```
/Users/marci/anaconda/lib/python3.5/site-packages/statsmodels/nonpara
metric/kdetools.py:20: VisibleDeprecationWarning: using a non-integer
number instead of an integer will result in an error in the future
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12252cfd0>
```



Great Job!