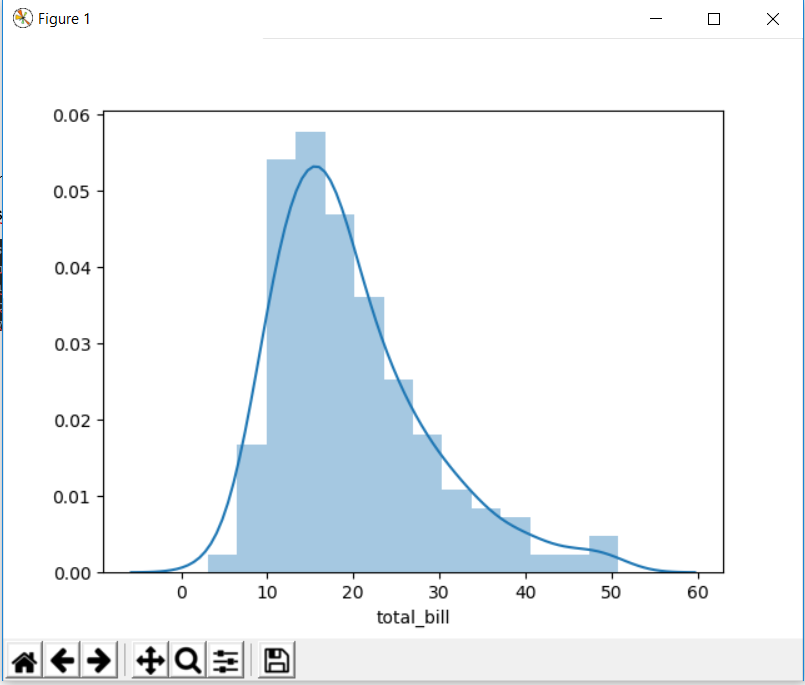
# Distribution Plots

Seaborn comes ready with some sample datasets and one of them is “tips”

## distplot

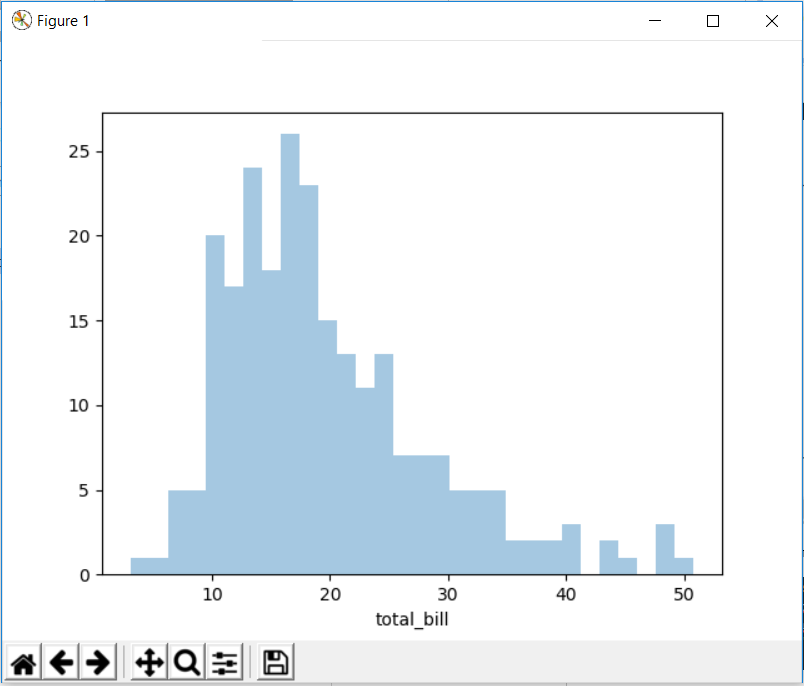
The distplot shows the distribution of a univariate set of observations. You can see below seaborn plots don’t show up on their own. You need to use matplotlib’s .show() method to display the plots.

import seaborn as sns  
import matplotlib.pyplot as plt  
tips = sns.load\_dataset("tips")  
sns.distplot(tips["total\_bill"])  
plt.show()



We can also remove kde line which is kernel density estimation and also specify the number of bins.

import seaborn as sns  
import matplotlib.pyplot as plt  
tips = sns.load\_dataset("tips")  
sns.distplot(tips["total\_bill"],kde=False,bins=30)  
plt.show()

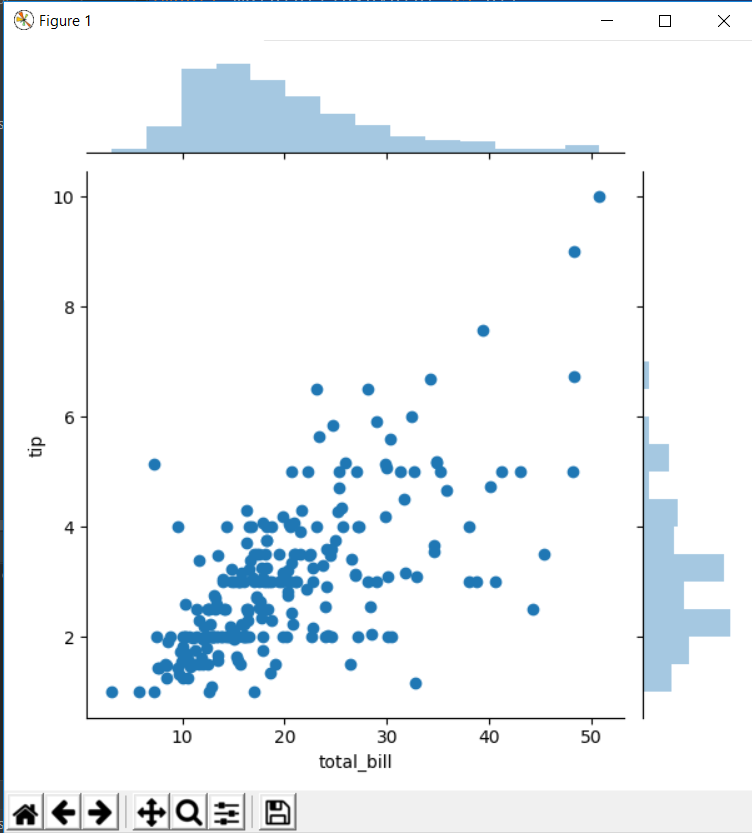


But keep in mind that bins should be specified keeping in mind the value range otherwise very confusing histograms can come.

## jointplot

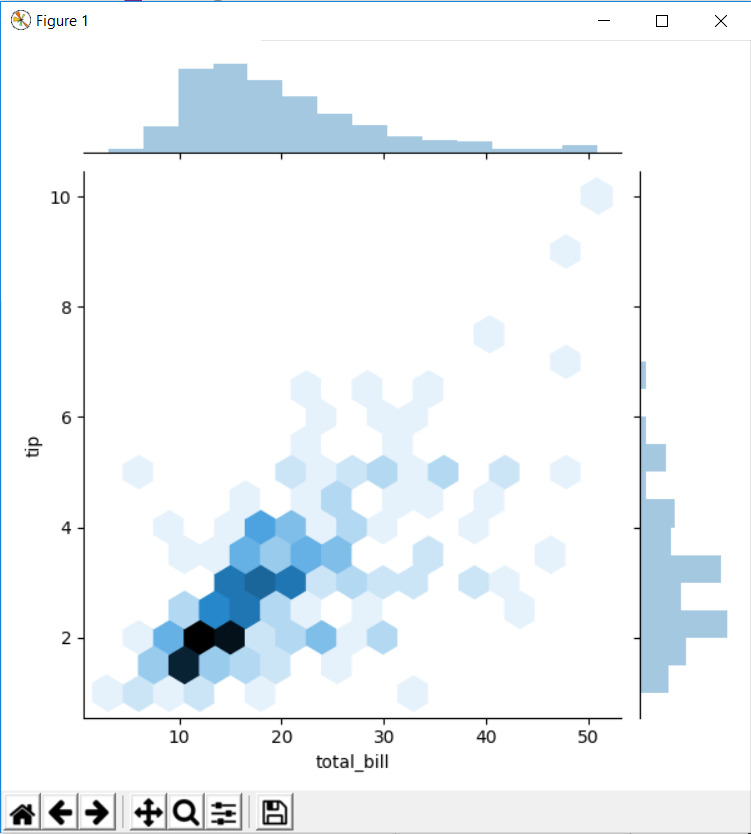
jointplot() allows you to basically match up two distplots for bivariate data. With your choice of what **kind** parameter to compare with:

* “scatter”
* “reg”
* “resid”
* “kde”
* “hex”
* import seaborn as sns  
  import matplotlib.pyplot as plt  
  tips = sns.load\_dataset("tips")  
  sns.jointplot(x="total\_bill",y="tip",data=tips)  
  plt.show()

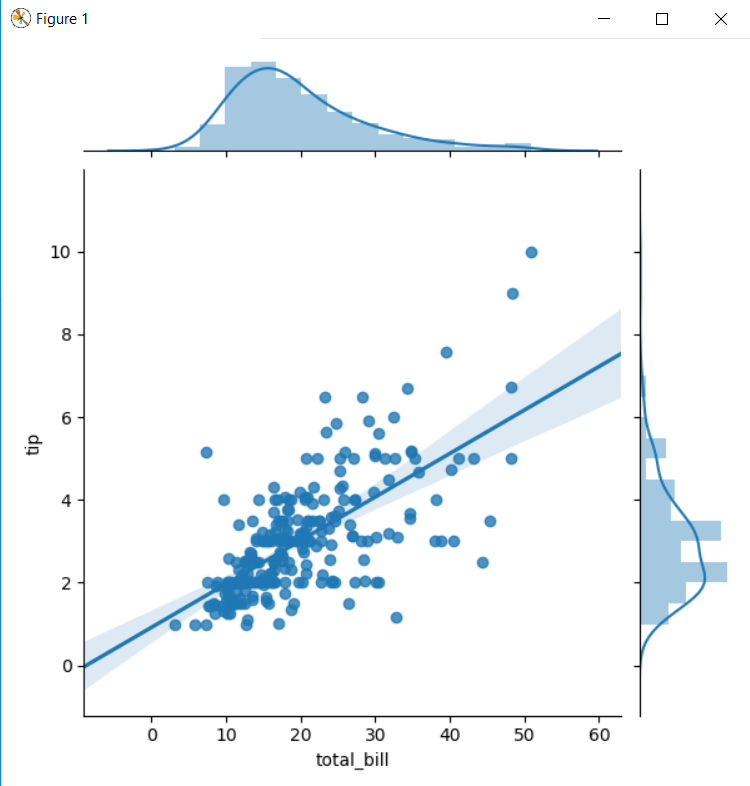


We have one parameter “kind” through which we can specify what kind of joint plot we want. The default one is scatter which is shown above.

import seaborn as sns  
import matplotlib.pyplot as plt  
tips = sns.load\_dataset("tips")  
sns.jointplot(x="total\_bill",y="tip",data=tips,kind="hex")  
plt.show()

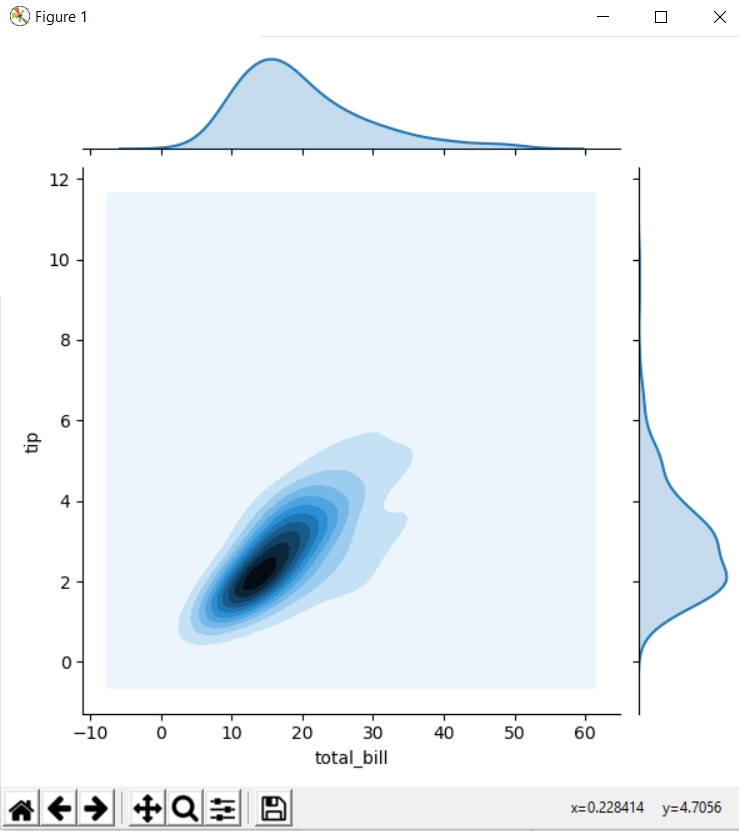


import seaborn as sns  
import matplotlib.pyplot as plt  
tips = sns.load\_dataset("tips")  
sns.jointplot(x="total\_bill",y="tip",data=tips,kind="reg")  
plt.show()



“reg” stands for regression. Its similar to scatter plot but here seaborn also draws a regression line. (same as for linear regression)

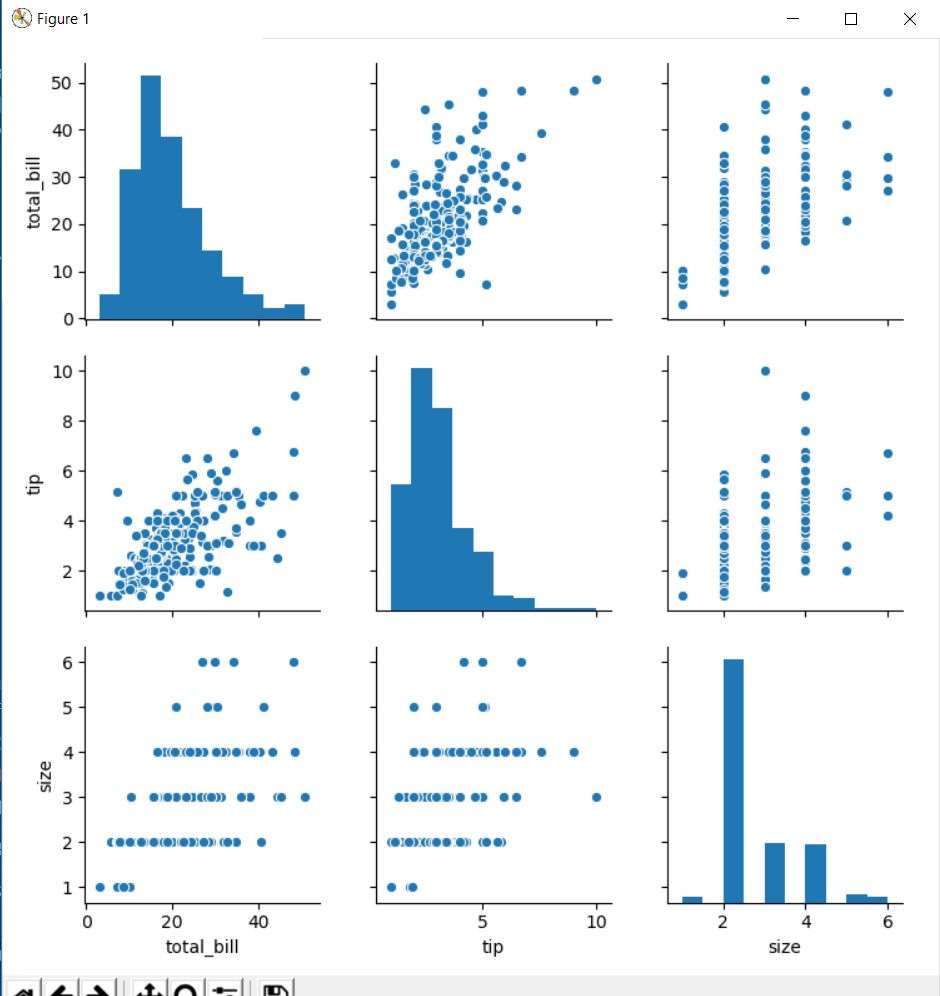
import seaborn as sns  
import matplotlib.pyplot as plt  
tips = sns.load\_dataset("tips")  
sns.jointplot(x="total\_bill",y="tip",data=tips,kind="kde")  
plt.show()



## pairplot

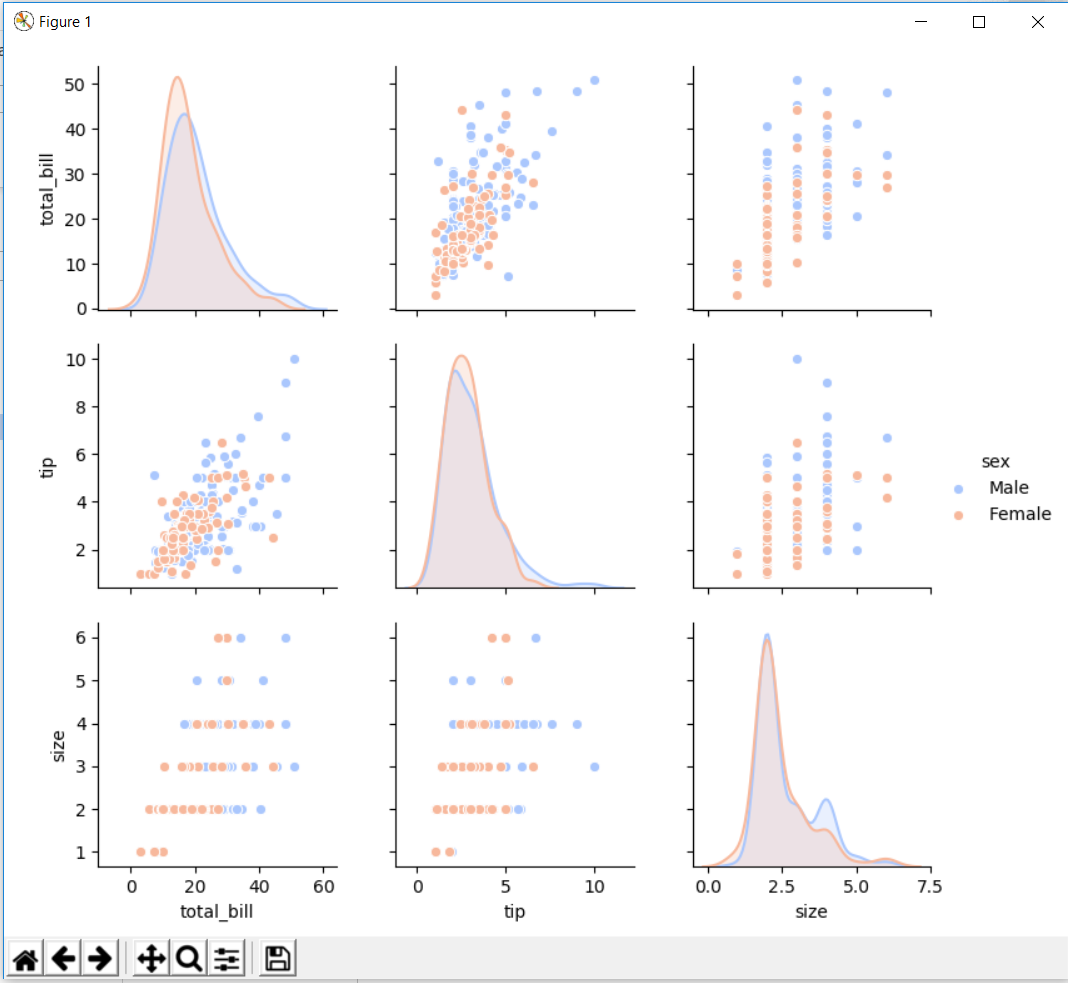
pairplot will plot pairwise relationships across an entire dataframe (for the numerical columns) and supports a color hue argument (for categorical columns).

Its basically joint plot between columns present in the data frame.



* You can notice here that plot between same columns is not scattered plot but a histogram
* We can also add hue argument and in hue argument, we mention the name of categorical column (non numeric).
* We can also mention palette argument for better colour combination

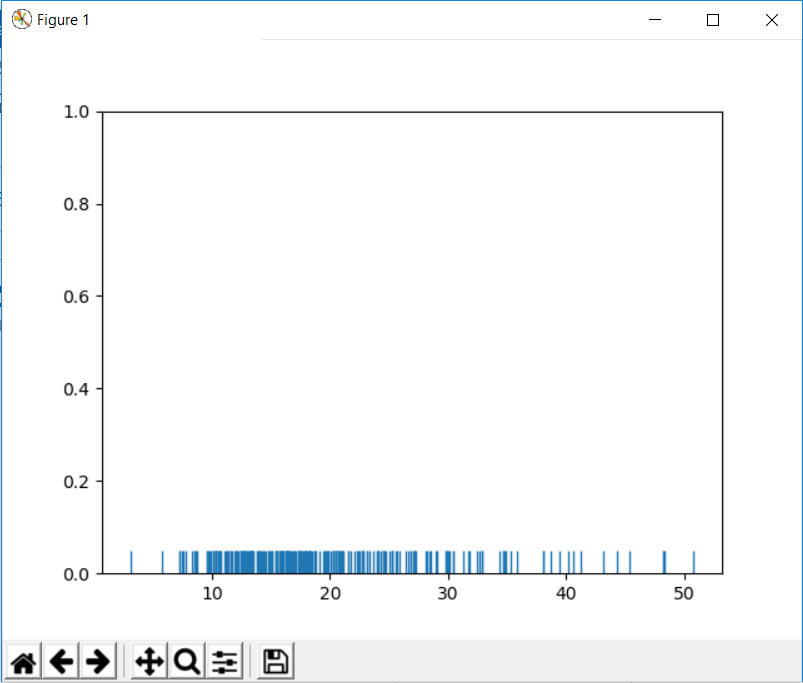
import seaborn as sns  
import matplotlib.pyplot as plt  
tips = sns.load\_dataset("tips")  
sns.pairplot(tips,hue="sex",palette="coolwarm")  
plt.show()



## rugplot

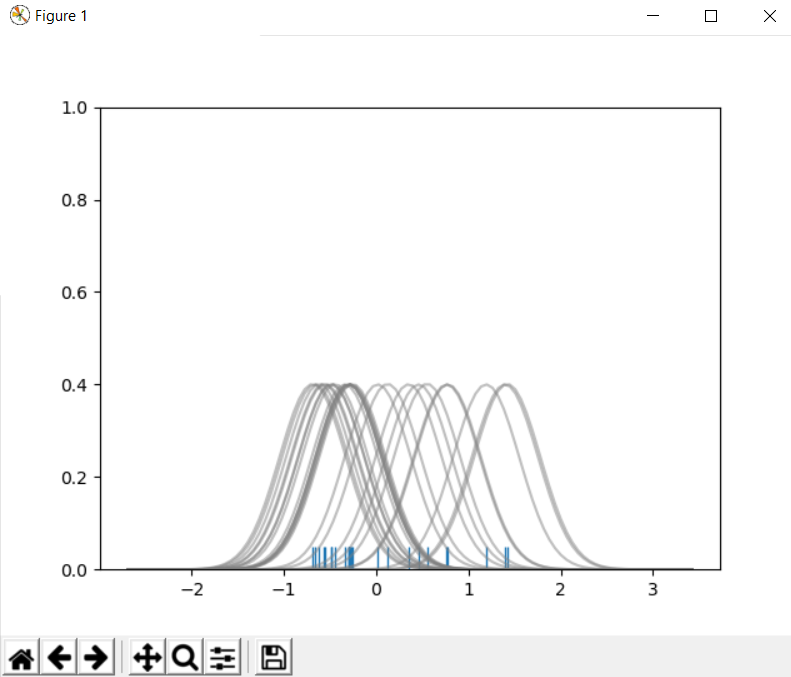
rugplots are actually a very simple concept, they just draw a dash mark for every point on a univariate distribution. They are the building block of a KDE plot:

import seaborn as sns  
import matplotlib.pyplot as plt  
tips = sns.load\_dataset("tips")  
sns.rugplot(tips["total\_bill"])  
plt.show()



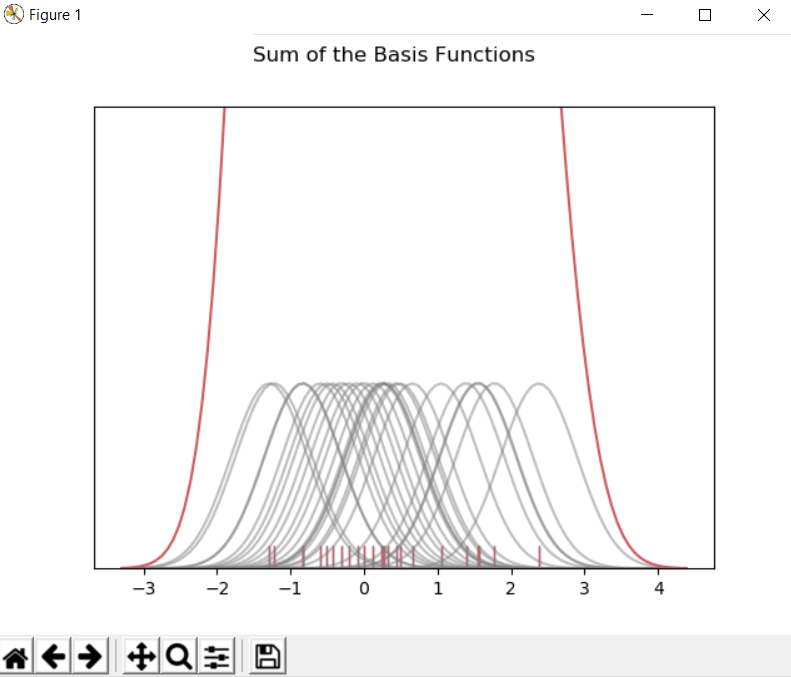
We will generate kde line using rugplot as below:

# Don't worry about understanding this code!  
# It's just for the diagram below  
import seaborn as sns  
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\_estimation\_of\_the\_bandwidth'  
  
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)  
  
plt.show()



Adding below code to above one will give us the kdeplot:

# 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")



## 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:

import seaborn as sns  
import matplotlib.pyplot as plt  
tips = sns.load\_dataset("tips")  
sns.kdeplot(tips["total\_bill"])  
plt.show()

