Visualizing the distribution of a dataset

- When dealing with a set of data, often the first thing youll want to do is get a sense for how the variables are distributed.
- This chapter of the tutorial will give a brief introduction to some of the tools in seaborn for examining univariate and bivariate distributions.
- You may also want to look at the categorical plots chapter for examples of functions that make it easy to compare the distribution of a variable across levels of other variables.

```
%matplotlib inline
import numpy as np
import pandas as pd
from scipy import stats, integrate
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color_codes=True)
np.random.seed(sum(map(ord,
      "distributions")))
```

The most convenient way to take a quick look at a univariate distribution in seaborn is the distplot() function. By default, this will draw a histogram and fit a kernel density estimate (KDE).

```
x = np.random.normal(size=100)
sns.distplot(x);
```

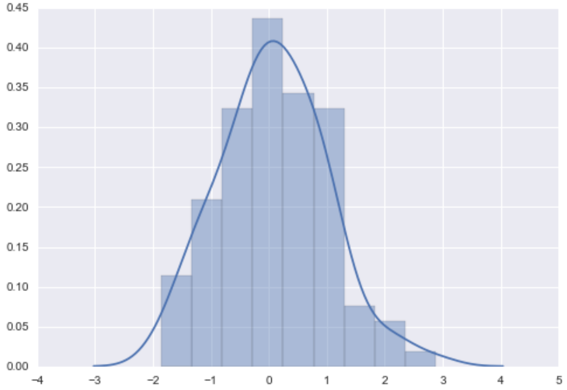


Figure:

Seaborn Workshop

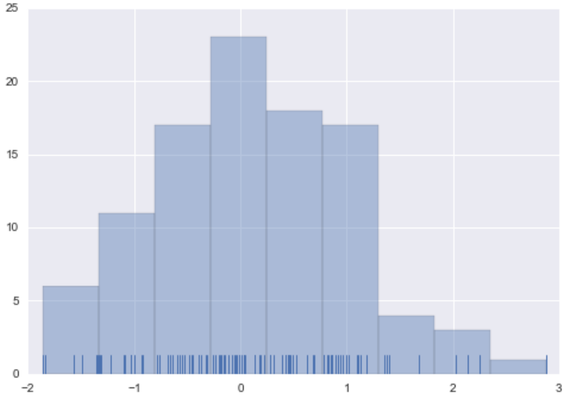
Histograms

- Histograms are likely familiar, and a hist function already exists in matplotlib.
- 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.

Seaborn Workshop

- To illustrate this, lets remove the density curve and add a rug plot, which draws a small vertical tick at each observation.
- You can make the rug plot itself with the rugplot() function, but it is also available in `distplot()`:

sns.distplot(x, kde=False, rug=True);

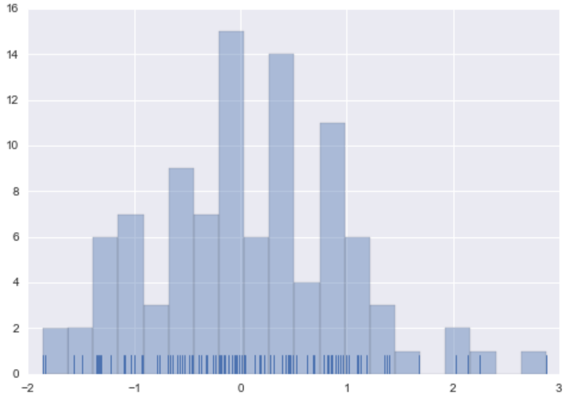


Seaborn Workshop

- When drawing histograms, the main choice you have is the number of bins to use and where to place them.
- `distplot()` uses a simple rule to make a good guess for what the right number is by default, but trying more or fewer bins might reveal other features in the data:

Seaborn Workshop

```
sns.distplot(x, bins=20, kde=False, rug=True);
```



16
14
12
10
8
6
4
2
0
−2
−1
0
1
2
3