- Many datasets contain multiple quantitative variables, and the goal of an analysis is often to relate those variables to each other.
- We previously discussed functions that can accomplish this by showing the joint distribution of two variables. It can be very helpful, though, to use statistical models to estimate a simple relationship between two noisy sets of observations.
- The functions discussed in this chapter will do so through the common framework of linear regression.

- In the spirit of Tukey, the regression plots in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses.
- That is to say that seaborn is not itself a package for statistical analysis. To obtain quantitative measures related to the fit of regression models, you should use statsmodels.
- The goal of seaborn, however, is to make exploring a dataset through visualization quick and easy, as doing so is just as (if not more) important than exploring a dataset through tables of statistics.

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color_codes=True)
np.random.seed(sum(map(ord, "regression")))
tips = sns.load_dataset("tips")
```

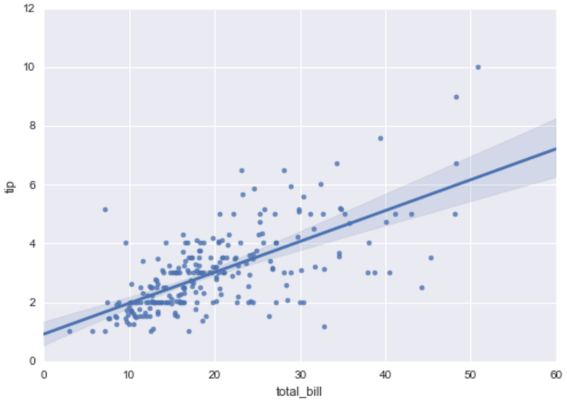
Two main functions in seaborn are used to visualize a linear relationship as determined through regression. These functions, regplot() and lmplot() are closely related, and share much of their core functionality. It is important to understand the ways they differ, however, so that you can quickly choose the correct tool for particular job.

Seaborn Workshop

In the simplest invocation, both functions draw a scatterplot of two variables, x and y, and then fit the regression model y x and plot the resulting regression line and a 95% confidence interval for that regression:

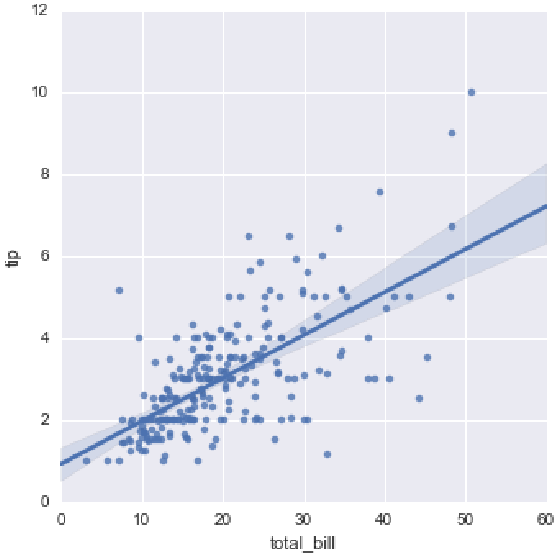
Seaborn Workshop

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



Seaborn Workshop

```
sns.lmplot(x="total_bill", y="tip", data=tips);
```



Seaborn Workshop

You should note that the resulting plots are identical, except that the figure shapes are different. We will explain why this is shortly. For now, the other main difference to know about is that regplot() accepts the x and y variables in a variety of formats including simple numpy arrays, pandas Series objects, or as references to variables in a pandas DataFrame object passed to data. In contrast, `lmplot()` has data as a required parameter and the x and y variables must be specified as strings.

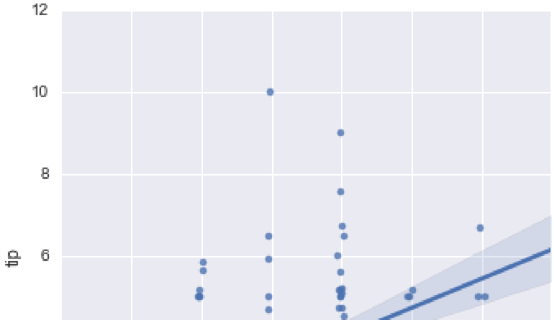
- This data format is called long-form or tidy data. Other than this input flexibility, `regplot()` possesses a subset of lmplot()s features, so we will demonstrate them using the latter.
- Its possible to fit a linear regression when one of the variables takes discrete values, however, the simple scatterplot produced by this kind of dataset is often not optimal:

```
sns.lmplot(x="size", y="tip", data=tips);
```

One option is to add some random noise (jitter) to the discrete values to make the distribution of those values more clear. Note that jitter is applied only to the scatterplot data and does not influence the regression line fit itself:

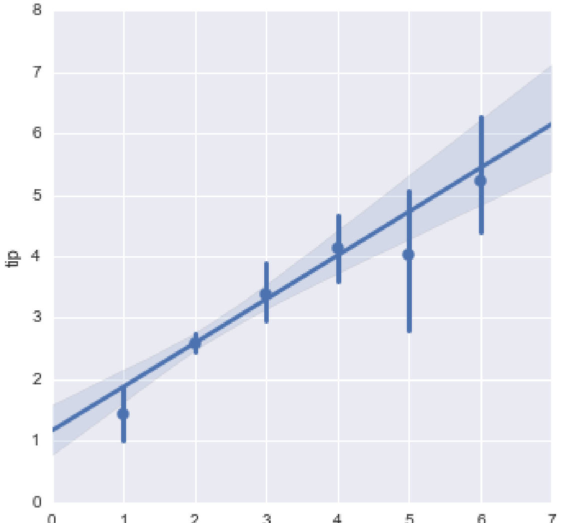
```
sns.lmplot(x="size", y="tip", data=tips, x_jitter=.05
```

/



A second option is to collapse over the observations in each discrete bin to plot an estimate of central tendency along with a confidence interval:

```
sns.lmplot(x="size", y="tip", data=tips, x_estimator=np
```


tip