

## Plotting Bivariate Bistributions

- ▶ It can also be useful to visualize a bivariate distribution of two variables.
- ▶ The easiest way to do this in seaborn is to just the `jointplot()` function, which creates a multi-panel figure that shows both the bivariate (or joint) relationship between two variables along with the univariate (or marginal) distribution of each on separate axes.

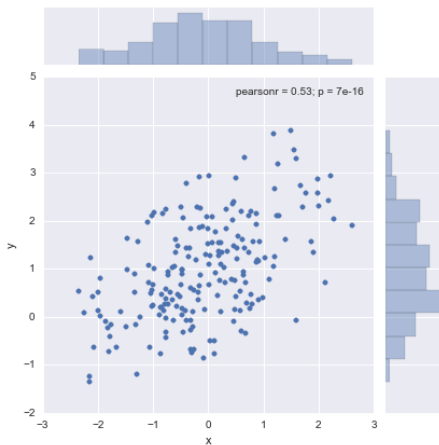
# Seaborn Workshop

```
mean, cov = [0, 1], [(1, .5), (.5, 1)]  
  
data = np.random.multivariate_normal(mean,  
                                     cov, 200)  
  
df = pd.DataFrame(data, columns=["x", "y"])
```

- ▶ The most familiar way to visualize a bivariate distribution is a scatterplot, where each observation is shown with point at the  $x$  and  $y$  values.
- ▶ This is analogous to a rug plot on two dimensions.
- ▶ You can draw a scatterplot with the matplotlib `plt.scatter` function, and it is also the default kind of plot shown by the `jointplot()` function:

# Seaborn Workshop

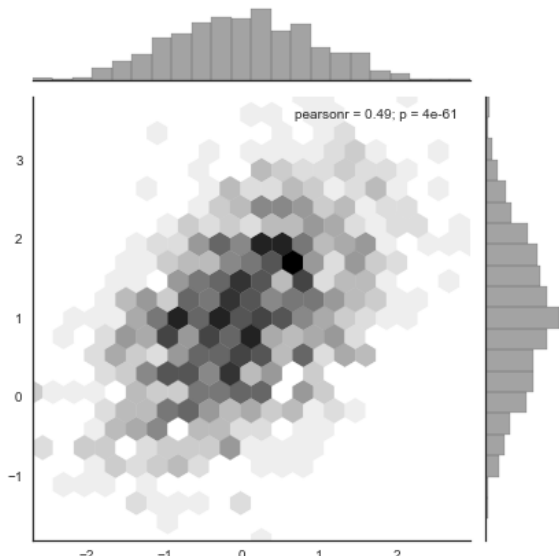
```
sns.jointplot(x="x", y="y", data=df);
```



## Hexbin plots

- ▶ The bivariate analogue of a histogram is known as a hexbin plot, because it shows the counts of observations that fall within hexagonal bins.
- ▶ This plot works best with relatively large datasets.
- ▶ Its available through the matplotlib `plt.hexbin` function and as a style in `jointplot()`.
- ▶ It looks best with a white background:

```
x, y = np.random.multivariate_normal(mean, cov, 1000).T  
with sns.axes_style("white"):  
sns.jointplot(x=x, y=y, kind="hex", color="k");
```

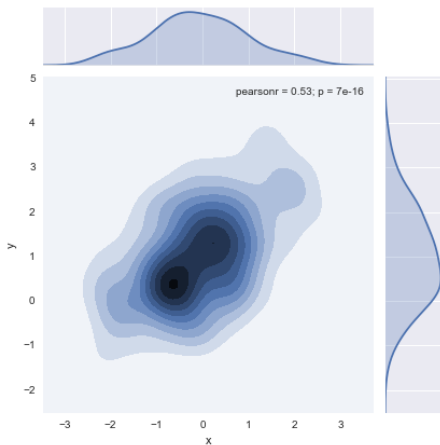


# Kernel density estimation

- ▶ It is also possible to use the kernel density estimation procedure described above to visualize a bivariate distribution.
- ▶ In seaborn, this kind of plot is shown with a contour plot and is available as a style in `jointplot()`:

# Seaborn Workshop

```
sns.jointplot(x="x", y="y", data=df, kind="kde");
```

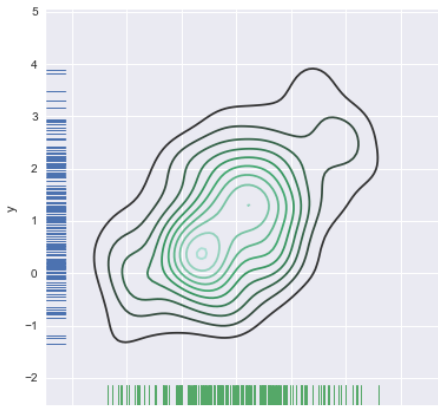




# Seaborn Workshop

- ▶ You can also draw a two-dimensional kernel density plot with the `kdeplot()` function.
- ▶ This allows you to draw this kind of plot onto a specific (and possibly already existing) matplotlib axes, whereas the `jointplot()` function manages its own figure:

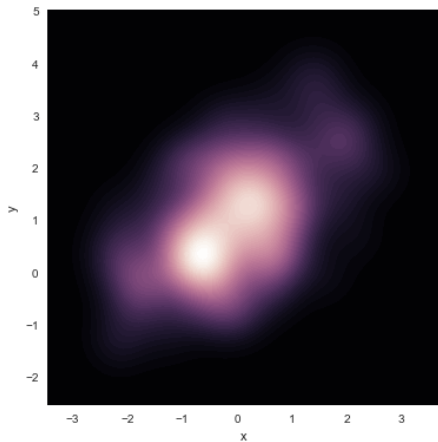
```
f, ax = plt.subplots(figsize=(6, 6))  
sns.kdeplot(df.x, df.y, ax=ax)  
sns.rugplot(df.x, color="g", ax=ax)  
sns.rugplot(df.y, vertical=True, ax=ax);
```



If you wish to show the bivariate density more continuously, you can simply increase the number of contour levels:

```
f, ax = plt.subplots(figsize=(6, 6))  
cmap = sns.cubehelix_palette(as_cmap=True, dark=0, light=1)  
sns.kdeplot(df.x, df.y, cmap=cmap, n_levels=60, shade=True)
```

# Hexbin Plots



# Seaborn Workshop

- ▶ The `jointplot()` function uses a `JointGrid` to manage the figure.
- ▶ For more flexibility, you may want to draw your figure by using `JointGrid` directly.
- ▶ `jointplot()` returns the `JointGrid` object after plotting, which you can use to add more layers or to tweak other aspects of the visualization:

# Seaborn Workshop

```
g = sns.jointplot(x="x", y="y",  
                  data=df, kind="kde", color="m")  
  
g.plot_joint(plt.scatter, c="w",  
              s=30, linewidth=1, marker="+")  
  
g.ax_joint.collections[0].set_alpha(0)  
  
g.set_axis_labels("$X$", "$Y$");
```

