Interactive Widgets

- https://ipywidgets.readthedocs.io/en/stable/examples/Using%20Interact.html (https://ipywidgets.readthedocs.io/en/stable/examples/Using%20Interact.html)
    - pip install ipywidgets
    - jupyter nbextension enable --py widgetsnbextension

In [1]:
```python
import ipywidgets as widgets

from ipywidgets import interact, fixed
```

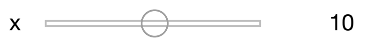
In [2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```python
%matplotlib inline
plt.style.use('seaborn-whitegrid')
```

Basic *interact*

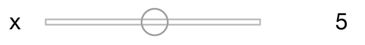
In [4]:
```python
def f(x):
    return x
```

In [5]:
```python
interact(f, x=10);
```



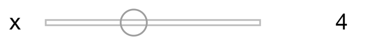
```
10
```

In [6]:
```python
# (min,max)

interact(f, x=(0,10));
```



```
5
```

In [7]:
```python
# (min, max, step)

interact(f, x=(0,10, 2));
```



```
4
```

In [ ]:
```python

```

In [8]:
```python
interact(f, x=True);
```

```
True
```

In [9]:
```python
interact(f, x="Hello!");
```

x `Hello!`

```
'Hello!'
```

In [10]:
```python
interact(f, x=['Python', 'R']);
```

x `Python`

```
'Python'
```

In [11]:
```python
# ('label', value) pairs

interact(f, x=[('Python', 'cs521'), ('R', 'cs544')]);
```

x `Python`

```
'cs521'
```

In [ ]:
```python

```

In [ ]:
```python

```

In [12]:
```python
@interact(x=True, y=1.5)
def g(x, y):
    return (x, y)
```

✓ x

y 1.50

```
(True, 1.5)
```

In [13]:
```python
@interact(x=5, y=fixed(10))
def h(x, y):
    return (x, y)
```

x 5

```
(5, 10)
```

Case Study

In [14]:
```python
np.random.seed(59367)

N = 100
df = pd.DataFrame(np.random.randint(60, 100, (N, 3)),
                  columns=['Q1', 'Q2', 'Q3'])

r = [[4,12,-16], [12,37,-43], [-16,-43,98]]

df = pd.DataFrame(np.random.multivariate_normal((60,70,80), r, N),
                  columns=['Q1', 'Q2', 'Q3'])

df.index = (["S" + str(i) for i in range(df.shape[0])])
df.head()
```

Out[14]:

	Q1	Q2	Q3
S0	61.234876	73.634151	74.208367
S1	58.800626	65.168959	82.813534
S2	62.202802	75.438902	61.816529
S3	59.045809	67.080244	77.770865
S4	59.220672	66.791932	77.349053

In [15]:
```python
@interact
def show_scores_more_than(quiz=['Q1','Q2','Q3'], x=(10,100)):
    return df.loc[df[quiz] > x]
```

quiz Q1

x 55

	Q1	Q2	Q3
S0	61.234876	73.634151	74.208367
S1	58.800626	65.168959	82.813534
S2	62.202802	75.438902	61.816529
S3	59.045809	67.080244	77.770865
S4	59.220672	66.791932	77.349053
...	...	...	...
S95	58.418013	64.654888	80.354792
S96	64.085862	82.741171	61.632443
S97	59.408635	68.963582	83.041888
S98	61.056778	74.158991	79.198236
S99	61.710324	74.600197	67.723088

100 rows × 3 columns

In [16]:
```python
cols = list(df.columns)
cols
```

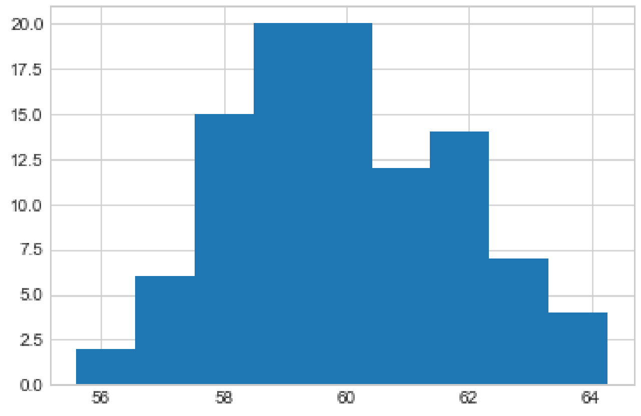
Out[16]:
```
['Q1', 'Q2', 'Q3']
```

In [ ]:

In [17]:
```
@interact
def histogram_plots(quiz = cols, bins=(3,15)):
    plt.hist(df[quiz], bins=bins);
```

quiz Q1

bins 9



In [18]:
```
@interact
def correlations(first = cols,
                 second = cols[1:]):
    print("Correlation {:.2f}".format(df[first].corr(df[second])))
```

first Q1

second Q2

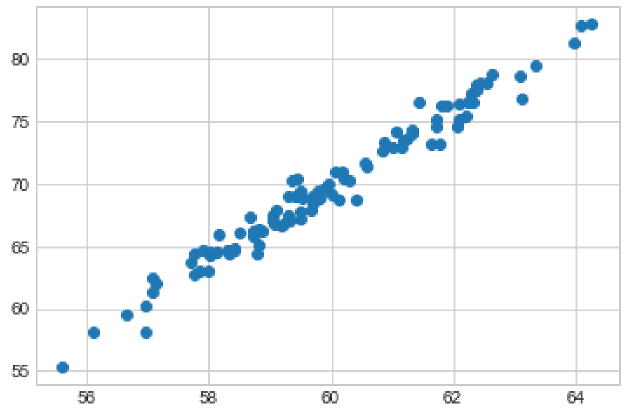
```
Correlation 0.99
```

In [19]:
```
@interact
def scatter_plots(first = cols,
                  second = cols[1:]):
    print("Correlation {:.2f}".format(df[first].corr(df[second])))
    plt.scatter(df[first], df[second]);
```

first Q1

second Q2

```
Correlation 0.99
```



In [ ]:

```

```

In [ ]:

```

```

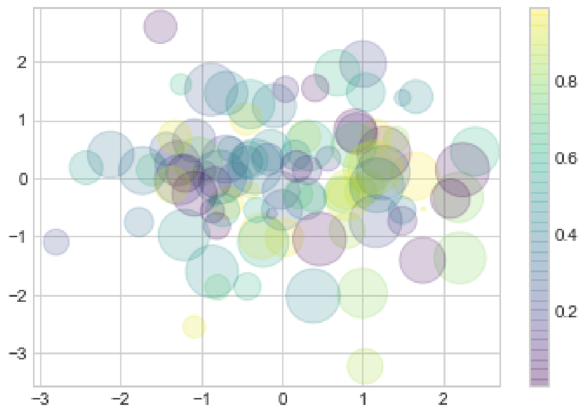
In [20]:

```
cscales = ['viridis', 'plasma', 'inferno', 'magma', 'cividis',
           'PiYG', 'PRGn', 'BrBG', 'PuOr', 'RdGy', 'RdBu',
            'RdYlBu', 'RdYlGn', 'Spectral', 'coolwarm', 'seismic']
```

In [21]:

```
np.random.seed(123)
x = np.random.randn(100)
y = np.random.randn(100)
colors = np.random.rand(100)
sizes = 1000 * np.random.rand(100)

@interact
def plot_scatter(colorscale=cscales):
    plt.scatter(x, y, c=colors, s=sizes, alpha=0.2,
            cmap=colorscale);
    plt.colorbar();  # show color scale
```

colorscale viridis



Dependent Widgets

In [22]:

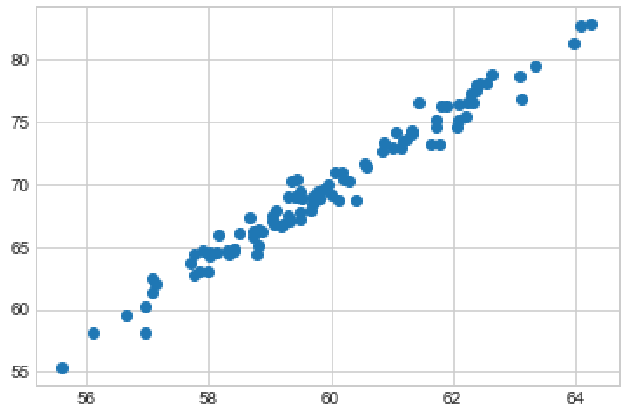
```
quiz1 = widgets.Dropdown(options=cols)
quiz2 = widgets.Dropdown(options=cols[1:] + cols[:1])

def update_options(*args):
    quiz2.options = cols[(quiz1.index+1):] + cols[:(quiz1.index+1)]

quiz1.observe(update_options, 'value')
```

```
In [23]: @interact
         def correlations(first = quiz1,
                          second = quiz2):
             print("Correlation {:.2f}".format(df[first].corr(df[second])))
             plt.scatter(df[first], df[second]);
```

first Q1

second Q2

```
Correlation 0.99
```



```
In [ ]:
```