
(http://www.pieriandata.com)

Pandas Built-in Data Visualization

In this lecture we will learn about pandas built-in capabilities for data visualization! It's built-off of matplotlib, but it baked into pandas for easier usage!

Let's take a look!

Imports

In [1]:

```
import numpy as np
import pandas as pd
%matplotlib inline
```

The Data

There are some fake data csv files you can read in as dataframes:

In [2]:

```
df1 = pd.read_csv('df1',index_col=0)
df2 = pd.read_csv('df2')
```

Style Sheets

Matplotlib has style sheets (http://matplotlib.org/gallery.html#style_sheets) you can use to make your plots look a little nicer. These style sheets include plot_bmh,plot_fivethirtyeight,plot_ggplot and more. They basically create a set of style rules that your plots follow. I recommend using them, they make all your plots have the same look and feel more professional. You can even create your own if you want your company's plots to all have the same look (it is a bit tedious to create on though).

Here is how to use them.

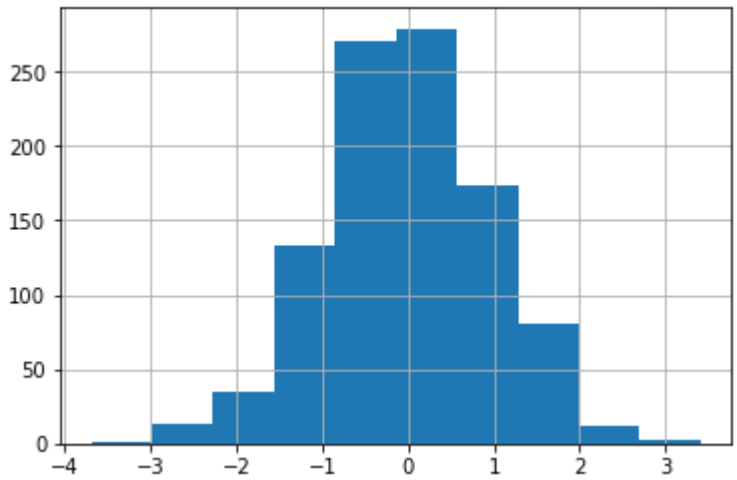
Before plt.style.use() your plots look like this:

In [3]:

```
df1['A'].hist()
```

Out[3]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d195898d0>
```



Call the style:

In [4]:

```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

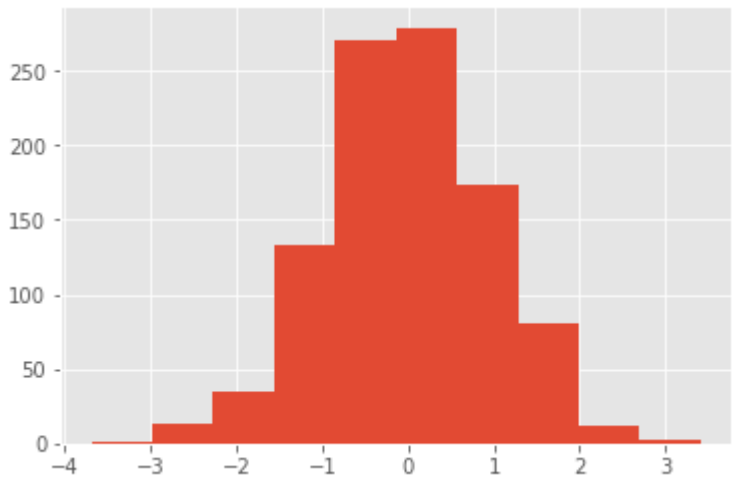
Now your plots look like this:

In [5]:

```
df1['A'].hist()
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d19a51a58>
```

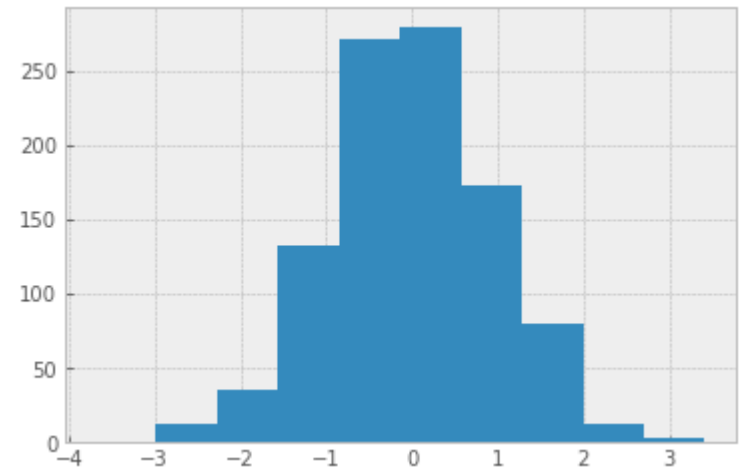


In [6]:

```
plt.style.use('bmh')
df1['A'].hist()
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d19a45d68>
```

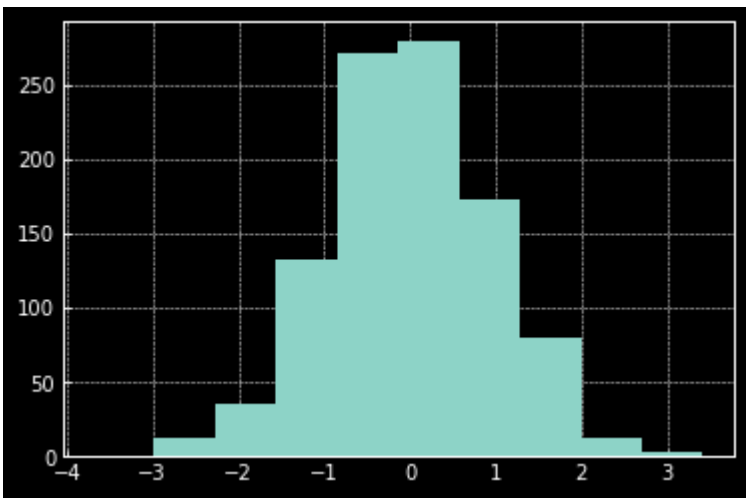


In [7]:

```
plt.style.use('dark_background')
df1['A'].hist()
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d19bd5b38>
```

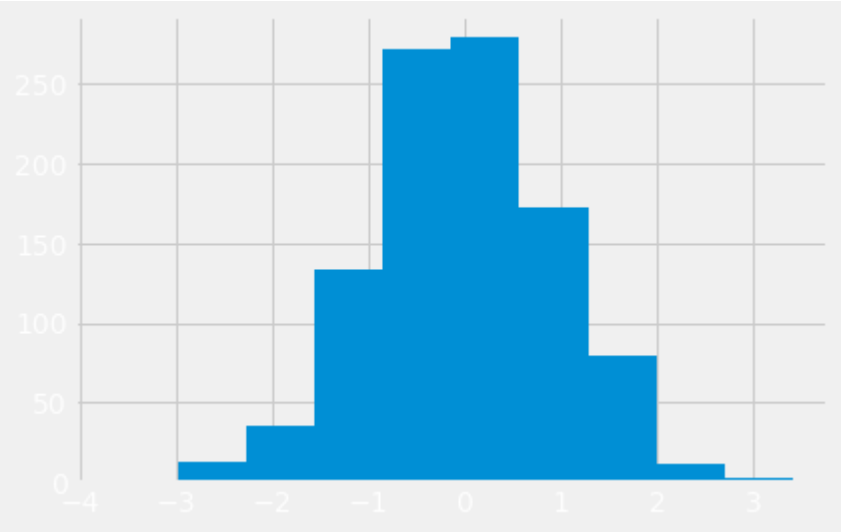


In [8]:

```
plt.style.use('fivethirtyeight')
df1['A'].hist()
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d19c51160>
```



In [9]:

```
plt.style.use('ggplot')
```

Let's stick with the ggplot style and actually show you how to utilize pandas built-in plotting capabilities!

Plot Types

There are several plot types built-in to pandas, most of them statistical plots by nature:

- df.plot.area
- df.plot.barh
- df.plot.density
- df.plot.hist
- df.plot.line
- df.plot.scatter
- df.plot.bar
- df.plot.box
- df.plot.hexbin
- df.plot.kde
- df.plot.pie

You can also just call df.plot(kind='hist') or replace that kind argument with any of the key terms shown in the list above (e.g. 'box','barh', etc..)

Let's start going through them!

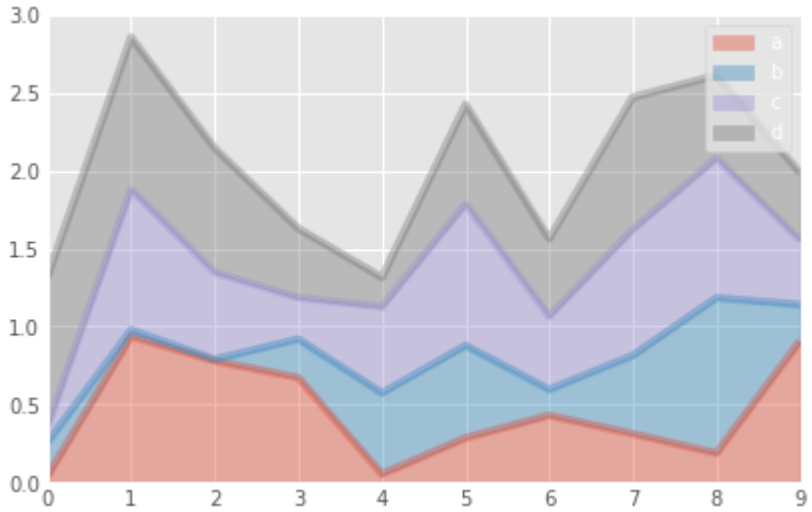
Area

In [10]:

```
df2.plot.area(alpha=0.4)
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d19d22828>
```



Barplots

In [11]:

```
df2.head()
```

Out[11]:

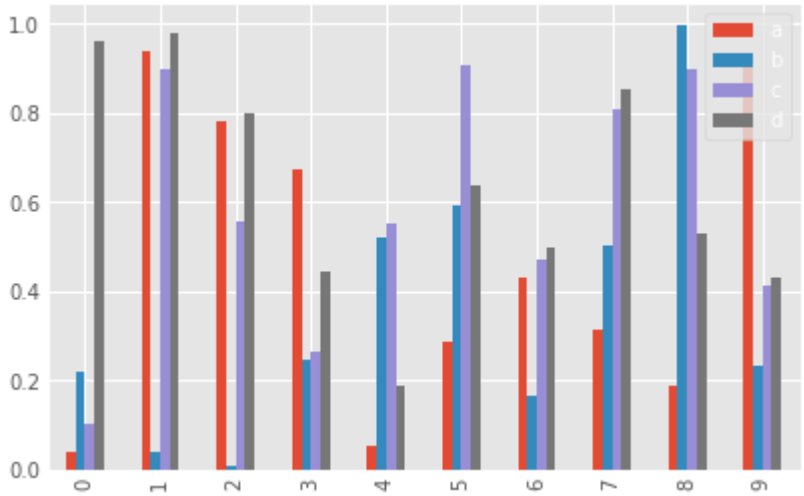
	a	b	c	d
0	0.039762	0.218517	0.103423	0.957904
1	0.937288	0.041567	0.899125	0.977680
2	0.780504	0.008948	0.557808	0.797510
3	0.672717	0.247870	0.264071	0.444358
4	0.053829	0.520124	0.552264	0.190008

In [12]:

```
df2.plot.bar()
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d19c1b470>
```

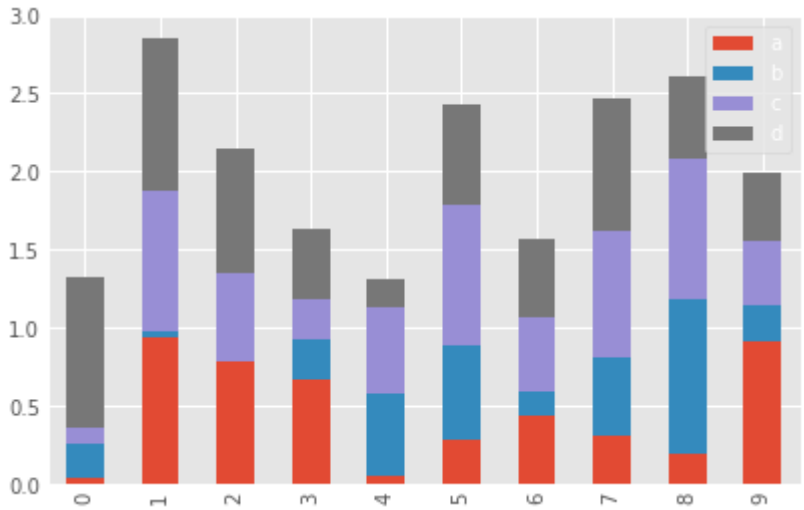


In [13]:

```
df2.plot.bar(stacked=True)
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d19fa7978>
```



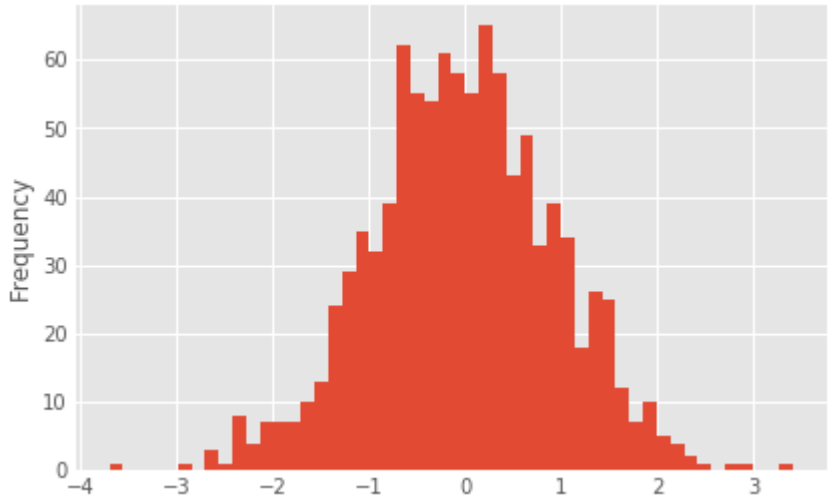
Histograms

In [14]:

```
df1['A'].plot.hist(bins=50)
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d1a0ece10>
```



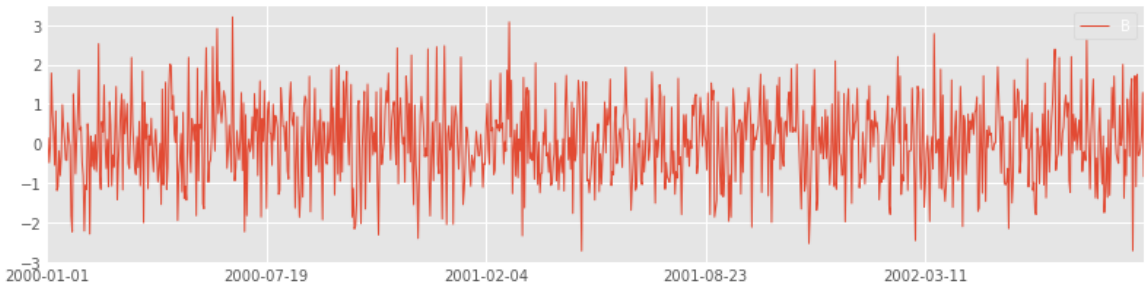
Line Plots

In [15]:

```
df1.plot.line(x=df1.index,y='B',figsize=(12,3),lw=1)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d1b305940>
```



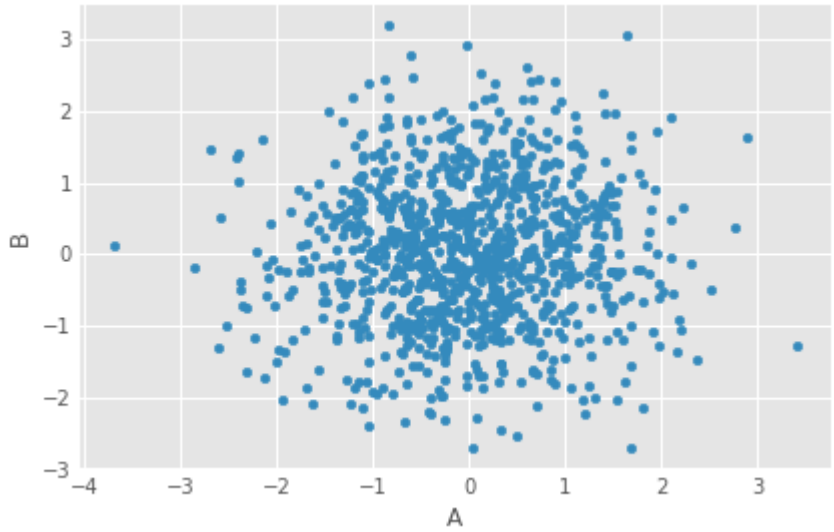
Scatter Plots

In [16]:

```
df1.plot.scatter(x='A',y='B')
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d1b336be0>
```



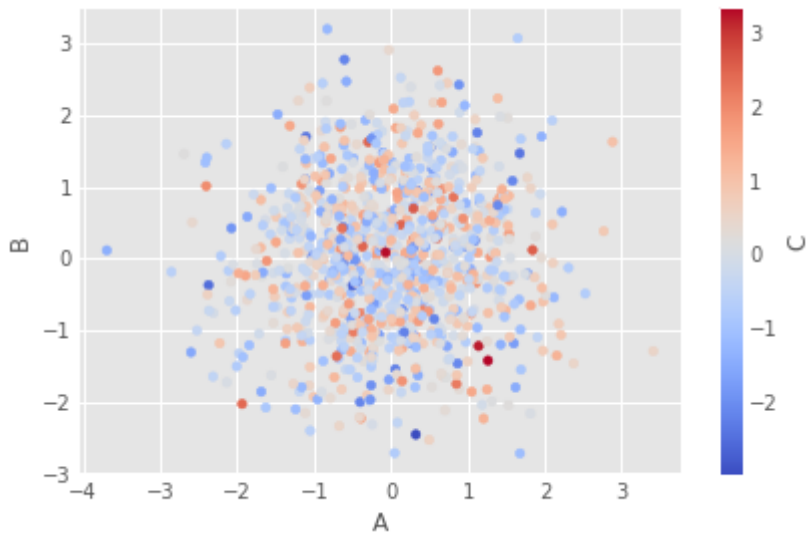
You can use c to color based off another column value Use cmap to indicate colormap to use. For all the colormaps, check out: http://matplotlib.org/users/colormaps.html (http://matplotlib.org/users/colormaps.html)

In [17]:

```
df1.plot.scatter(x='A',y='B',c='C',cmap='coolwarm')
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d1b47ccc0>
```



Or use s to indicate size based off another column. s parameter needs to be an array, not just the name of a column:

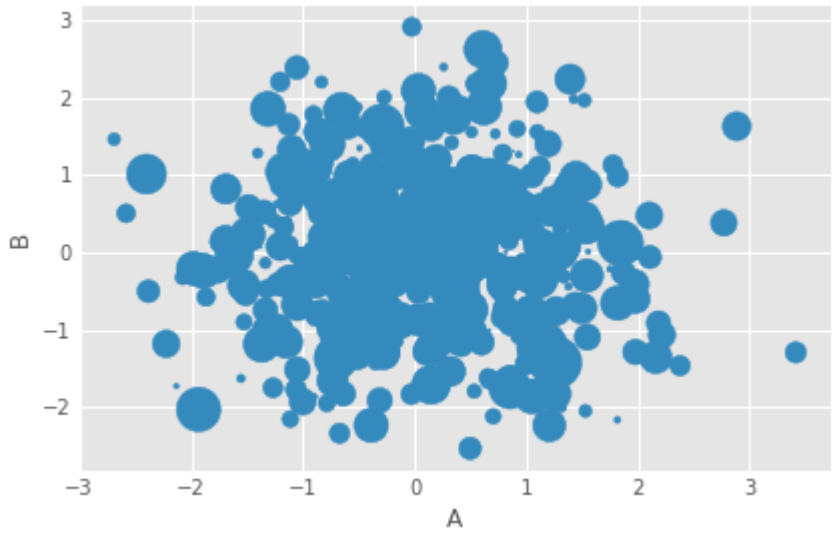
In [23]:

```
df1.plot.scatter(x='A',y='B',s=df1['C']*200)
```

```
C:\Users\Marcial\Anaconda3\lib\site-packages\matplotlib\collections.
py:877: RuntimeWarning: invalid value encountered in sqrt
  scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d1c355908>
```



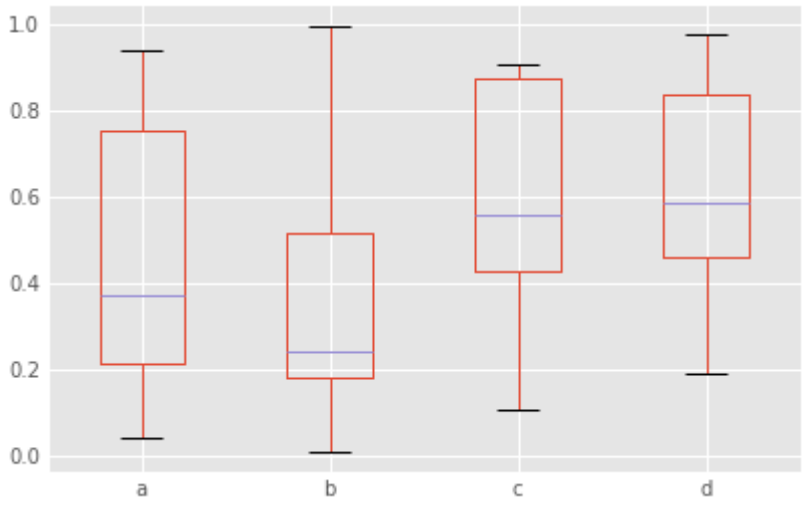
BoxPlots

In [19]:

```
df2.plot.box() # Can also pass a by= argument for groupby
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d1b5e3e48>
```



Hexagonal Bin Plot

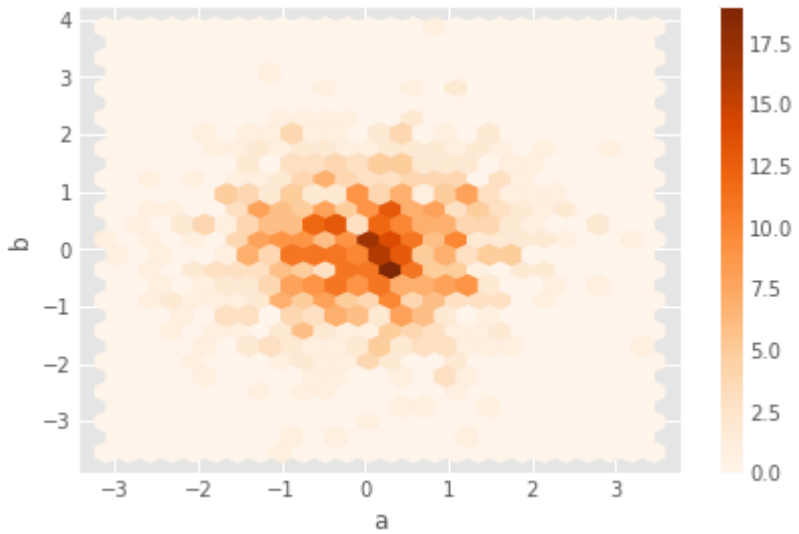
Useful for Bivariate Data, alternative to scatterplot:

In [20]:

```
df = pd.DataFrame(np.random.randn(1000, 2), columns=['a', 'b'])
df.plot.hexbin(x='a',y='b',gridsize=25,cmap='Oranges')
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d1b761978>
```



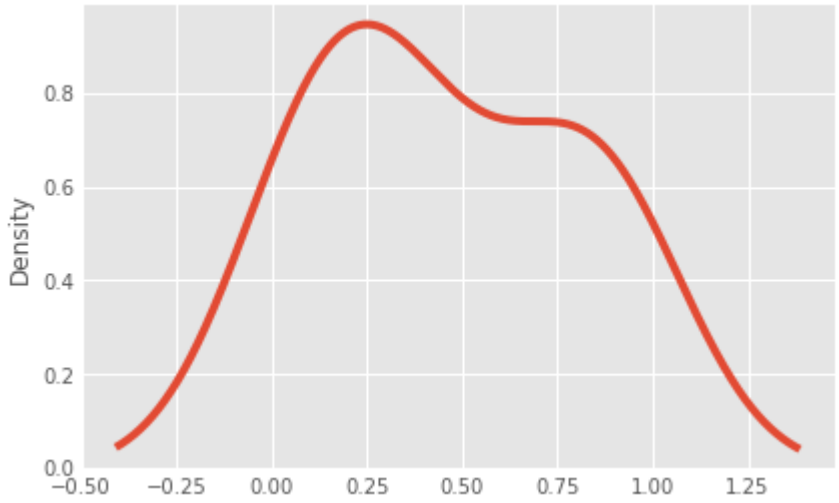
Kernel Density Estimation plot (KDE)

In [21]:

```
df2['a'].plot.kde()
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d1b88d630>
```

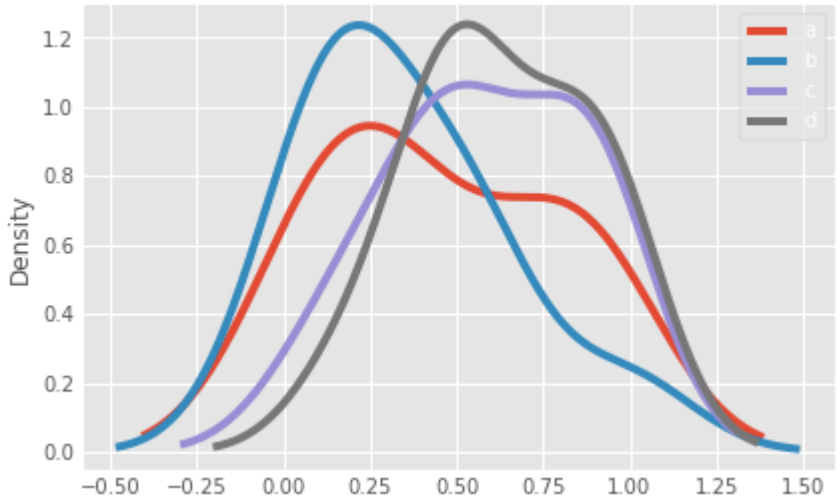


In [22]:

```
df2.plot.density()
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24d1b67cf60>
```



That's it! Hopefully you can see why this method of plotting will be a lot easier to use than full-on matplotlib, it balances ease of use with control over the figure. A lot of the plot calls also accept additional arguments of their parent matplotlib plt. call.

Great Job!