EXPERIMENT NO 1

DATA VISUALIZATION TECHNIQUES

1. IMPORTING THE LIBRARIES

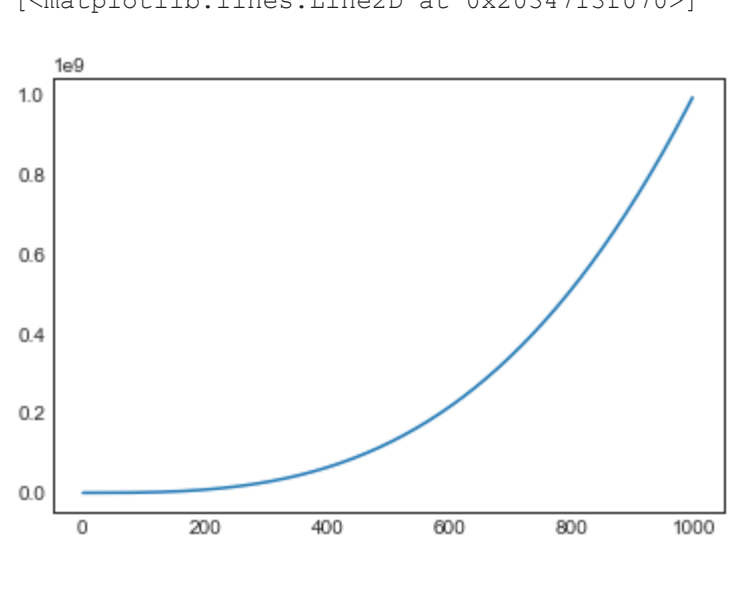
In [1]:
```python
# importing libraries

# !pip install brewer2mpl
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

large = 22
med = 16
small = 12

plt.style.use('seaborn-whitegrid')
sns.set_style("white")
%matplotlib inline
```

2. DIFFRENT PLOTTING TECHNIQUES:

2.1 BASIC PLOTTING WITH MATPLOTLIB

In [2]:
```python
x = np.arange(999)

plt.plot(x, np.power(x, 3))
```

Out[2]:
```
[<matplotlib.lines.Line2D at 0x2034713f070>]
```



2.2 BOX & WHISKER PLOT

In [3]:
```python
# importing dataset

df = pd.read_csv(
    '/DOCUMENTS/COLLEGE/CLASSES/EXPERIMENT_NO_1/StudentsPerformance.csv')

df.head()
```

Out[3]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

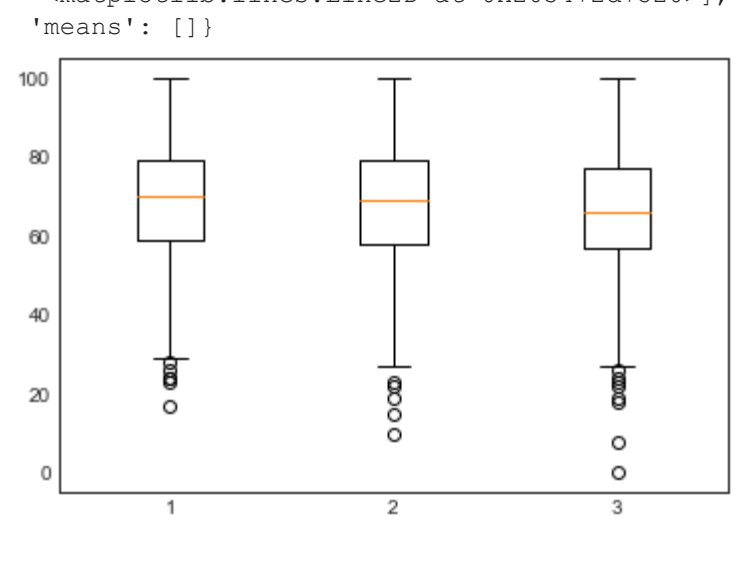
In [4]:
```python
df.describe()
```

Out[4]:

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

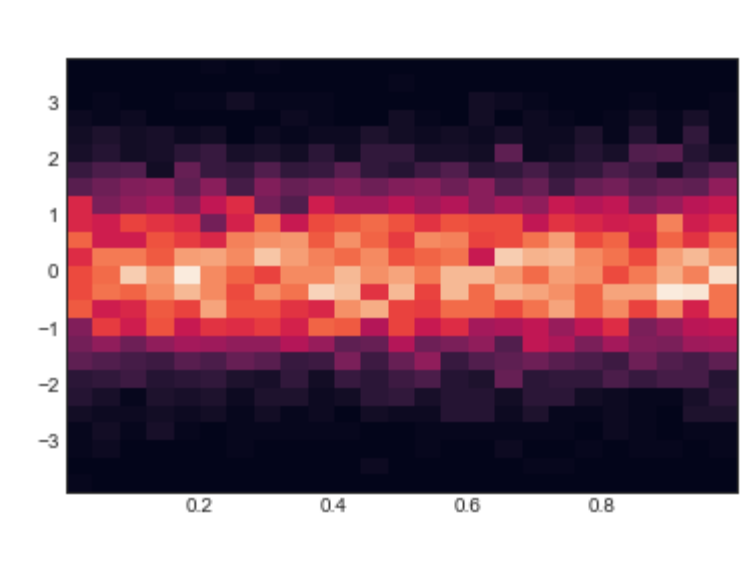
In [5]:
```python
plt.boxplot([df['reading score'], df['writing score'], df['math score']])
```

Out[5]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x203472ab040>,
  <matplotlib.lines.Line2D at 0x203472ab3d0>,
  <matplotlib.lines.Line2D at 0x203472b69a0>,
  <matplotlib.lines.Line2D at 0x203472b6d30>,
  <matplotlib.lines.Line2D at 0x203472ce310>,
  <matplotlib.lines.Line2D at 0x203472ce6a0>],
 'caps': [<matplotlib.lines.Line2D at 0x203472ab760>,
  <matplotlib.lines.Line2D at 0x203472abaf0>,
  <matplotlib.lines.Line2D at 0x203472c3100>,
  <matplotlib.lines.Line2D at 0x203472c3490>,
  <matplotlib.lines.Line2D at 0x203472cea30>,
  <matplotlib.lines.Line2D at 0x203472cedc0>],
 'boxes': [<matplotlib.lines.Line2D at 0x20347299c70>,
  <matplotlib.lines.Line2D at 0x203472b6610>,
  <matplotlib.lines.Line2D at 0x203472c3f40>],
 'medians': [<matplotlib.lines.Line2D at 0x203472abe80>,
  <matplotlib.lines.Line2D at 0x203472c3820>,
  <matplotlib.lines.Line2D at 0x203472d7190>],
 'fliers': [<matplotlib.lines.Line2D at 0x203472b6250>,
  <matplotlib.lines.Line2D at 0x203472c3bb0>,
  <matplotlib.lines.Line2D at 0x203472d7520>],
 'means': []}
```

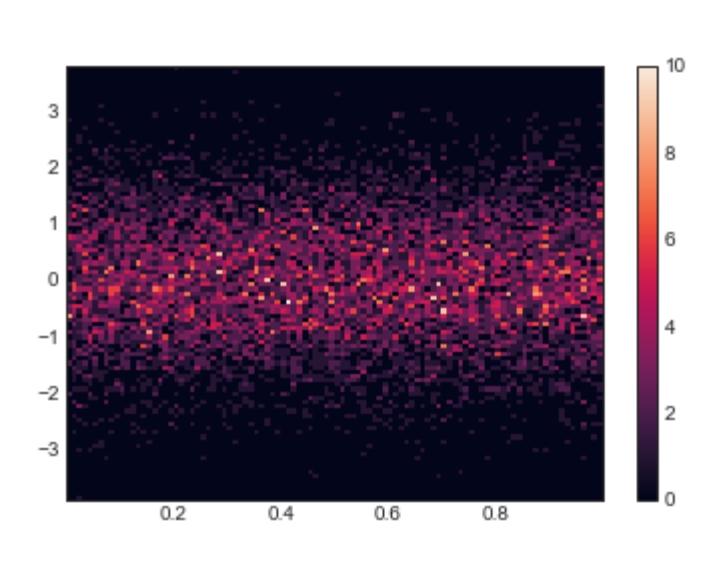


2.3 HEATMAPS PLOT

In [6]:
```python
import numpy as np
import matplotlib.pyplot as plt
%matplotlib notebook

plt.figure()

Y = np.random.normal(loc=0.0, scale=1.0, size=10000)
X = np.random.random(size=10000)

_ = plt.hist2d(X, Y, bins=25)
```



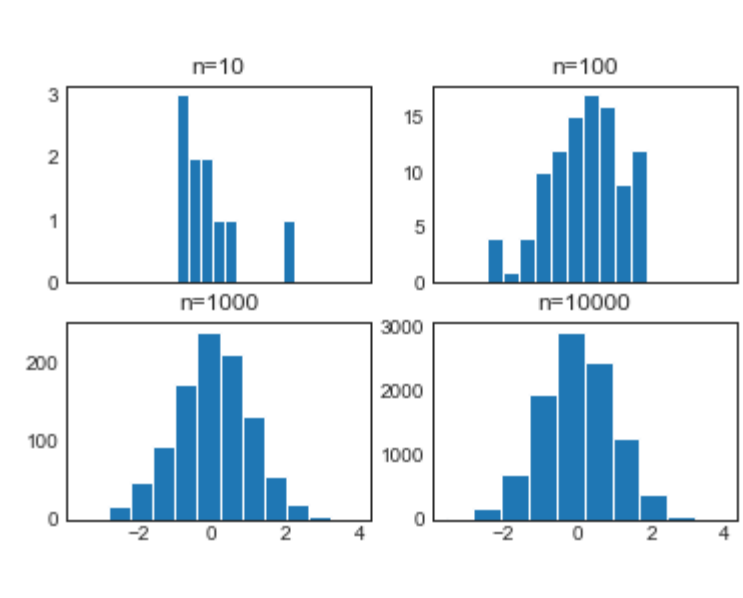
In [7]:
```python
plt.figure()
_ = plt.hist2d(X, Y, bins=100)

plt.colorbar()
```



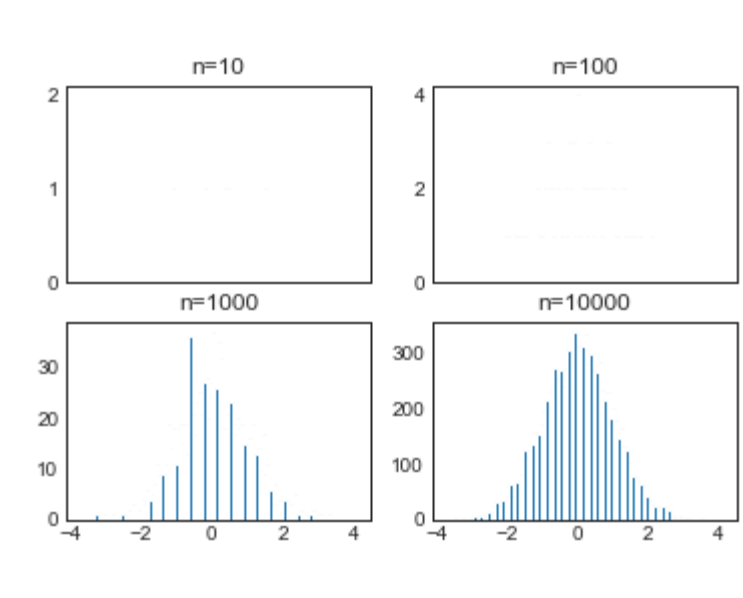
Out[7]:
```
<matplotlib.colorbar.Colorbar at 0x203473e26d0>
```

2.4 HISTOGRAMS PLOT

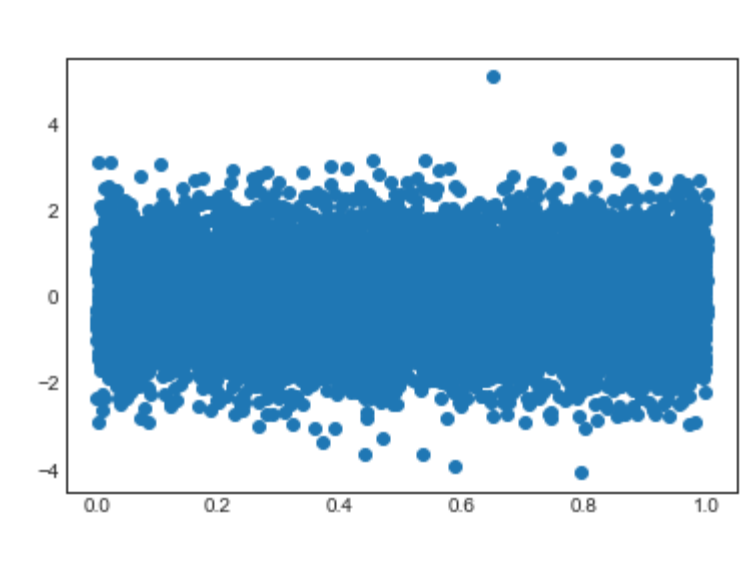
In [8]:
```python
import numpy as np
import matplotlib.pyplot as plt
%matplotlib notebook

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, sharex=True)
axs = [ax1, ax2, ax3, ax4]

for n in range(0, len(axs)):
    sample_size = 10**(n + 1)
    sample = np.random.normal(loc=0.0, scale=1.0, size=sample_size)
    axs[n].hist(sample)
    axs[n].set_title('n={}'.format(sample_size))
```



In [9]:
```python
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, sharex=True)
axs = [ax1, ax2, ax3, ax4]

for n in range(0, len(axs)):
    sample_size = 10**(n + 1)
    sample = np.random.normal(loc=0.0, scale=1.0, size=sample_size)
    axs[n].hist(sample, bins=100)
    axs[n].set_title('n={}'.format(sample_size))
```

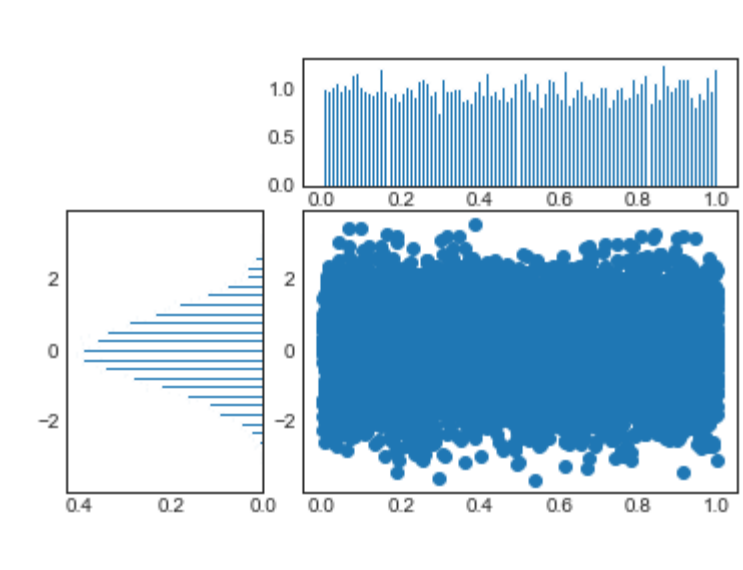


In [10]:
```python
plt.figure()
Y = np.random.normal(loc=0.0, scale=1.0, size=10000)
X = np.random.random(size=10000)
plt.scatter(X, Y)
```



Out[10]:
```
<matplotlib.collections.PathCollection at 0x20347a172b0>
```

In [11]:
```python
import matplotlib.gridspec as gridspec

plt.figure()
gspec = gridspec.GridSpec(3, 3)

top_histogram = plt.subplot(gspec[0, 1:])
side_histogram = plt.subplot(gspec[1:, 0])
lower_right = plt.subplot(gspec[1:, 1:])
```

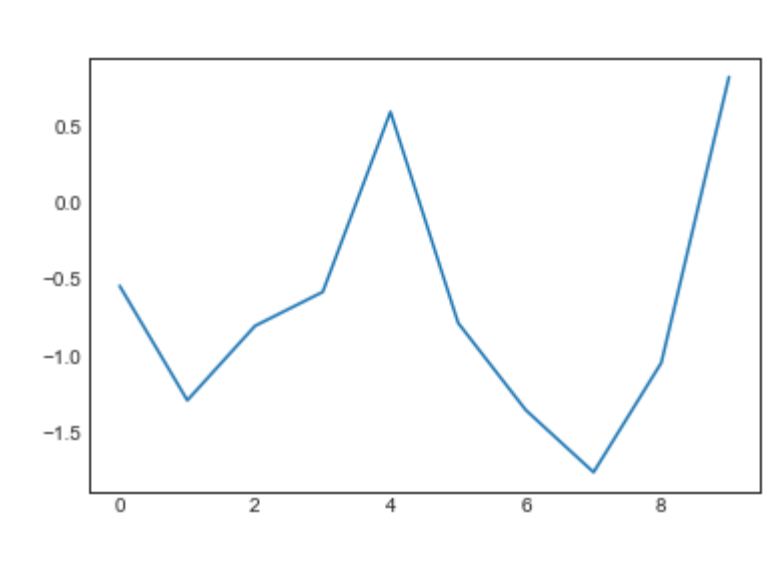


In [12]:
```python
Y = np.random.normal(loc=0.0, scale=1.0, size=10000)
X = np.random.random(size=10000)
lower_right.scatter(X, Y)
top_histogram.hist(X, bins=100)
s = side_histogram.hist(Y, bins=100, orientation='horizontal')
```

In [13]:
```python
top_histogram.clear()
top_histogram.hist(X, bins=100, density=True)
side_histogram.clear()
side_histogram.hist(Y, bins=100, orientation='horizontal', density=True)
side_histogram.invert_xaxis()
```

2.5 INTERACTIVITY PLOT

In [14]:
```python
import numpy as np
import matplotlib.pyplot as plt
%matplotlib notebook

plt.figure()
data = np.random.randn(10)
plt.plot(data)


def onclick(event):
    plt.cla()
    plt.plot(data)
    plt.gca().set_title('Event at pixels {}, {}, and data {}, {}'.format(
        event.x, event.y, '\n', event.xdata, event.ydata))


plt.gcf().canvas.mpl_connect('button_press_event', onclick)
```



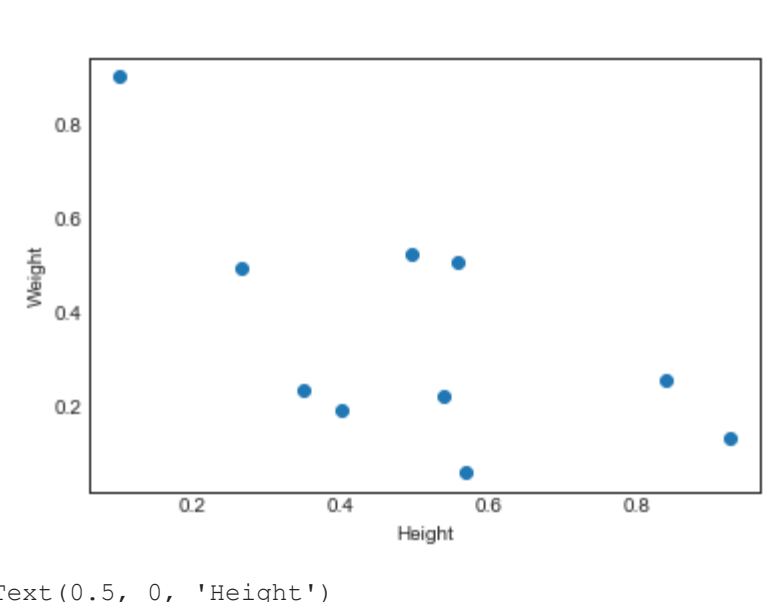
Out[14]:
```
9
```

In [15]:
```python
import pandas as pd
from random import shuffle

origins = [
    'China', 'Brazil', 'India', 'USA', 'Canada', 'UK', 'Germany', 'Iraq',
    'Chile', 'Mexico'
]
shuffle(origins)

df = pd.DataFrame({
    'height': np.random.rand(10),
    'weight': np.random.rand(10),
    'Origin': origins
})
df
```

Out[15]:

	height	weight	Origin
0	0.349221	0.235248	India
1	0.840867	0.258052	USA
2	0.569071	0.059595	Mexico
3	0.100325	0.903816	China
4	0.265135	0.494443	UK
5	0.925780	0.132333	Canada
6	0.400824	0.190511	Brazil
7	0.496310	0.523884	Germany
8	0.556896	0.507058	Iraq
9	0.538894	0.223202	Chile

In [16]:
```python
plt.figure()
plt.scatter(df['height'], df['weight'], picker=5)
plt.gca().set_ylabel('Weight')
plt.gca().set_xlabel('Height')
```



Out[16]:
```
Text(0.5, 0, 'Height')
```

In [17]:
```python
def onpick(event):
    origins = df.iloc[event.ind[0]]['origin']
    plt.gca().set_title('Selected item came from {}'.format(origin))


plt.gcf().canvas.mpl_connect('picl_event', onpick)
```

Out[17]:
```
9
```

3 PLOTTING WITH PANDAS

In [18]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib notebook
```

In [19]:
```python
plt.style.use('seaborn-colorblind')
```

In [20]:
```python
# dataframe plot

np.random.seed(123)
df = pd.DataFrame(
    {
        'A': np.random.randn(365).cumsum(0),
        'B': np.random.randn(365).cumsum(0) + 20,
        'C': np.random.randn(365).cumsum(0) - 20
    },
    index=pd.date_range('1/1/2017', periods=365))
df.head()
```

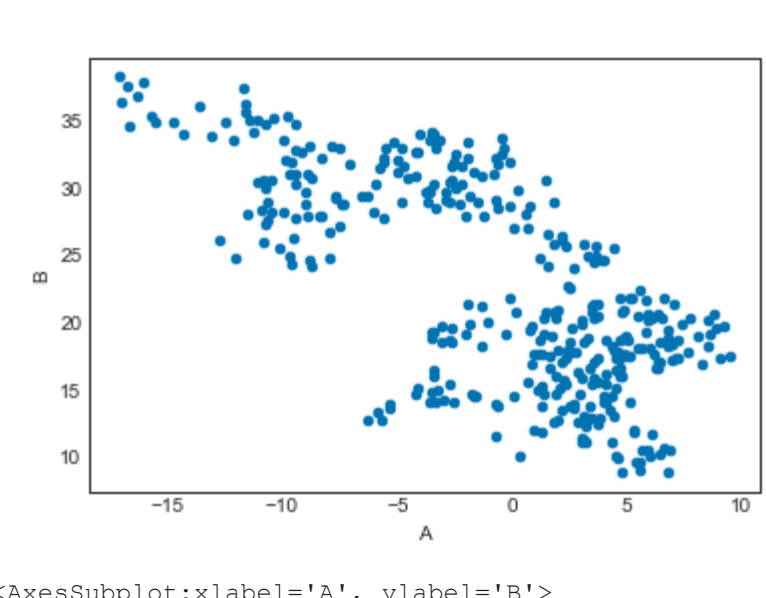
Out[20]:

	A	B	C
2017-01-01	-1.085631	20.059291	-20.230904
2017-01-02	-0.088285	21.803332	-16.659325
2017-01-03	0.194693	20.835588	-17.055481
2017-01-04	-1.311601	21.255156	-17.093802
2017-01-05	-1.890202	21.462083	-19.518638

In [21]:
```python
df.plot()
```

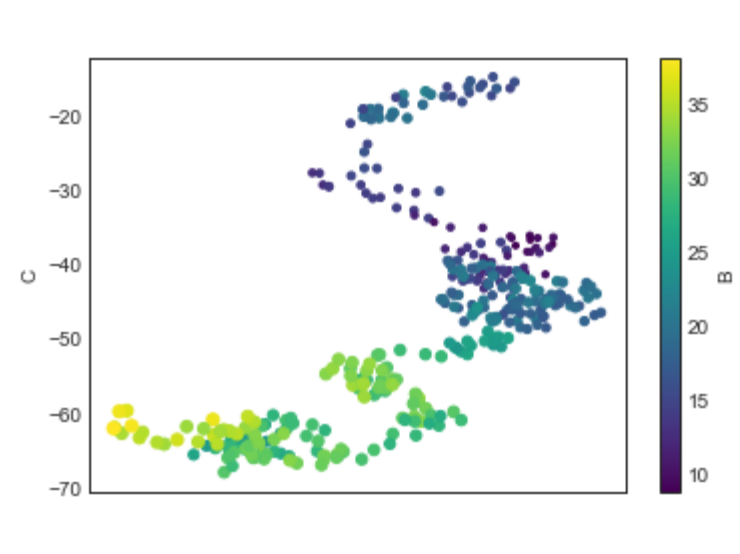
Out[21]:
```
<AxesSubplot:>
```

In [22]:
```python
df.plot('A', 'B', kind='scatter')
```



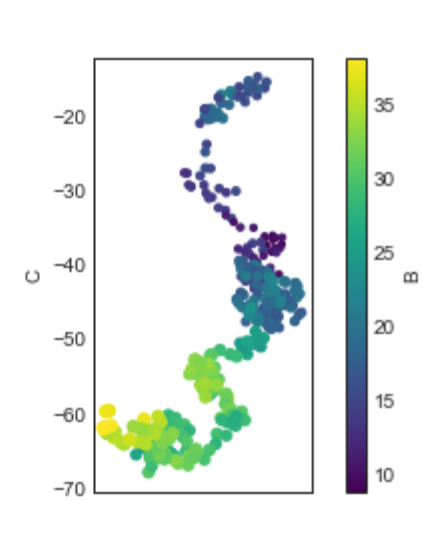
Out[22]:
```
<AxesSubplot:xlabel='A', ylabel='B'>
```

In [23]:
```python
df.plot.scatter('A', 'C', c='B', s=df['B'], colormap='viridis')
```

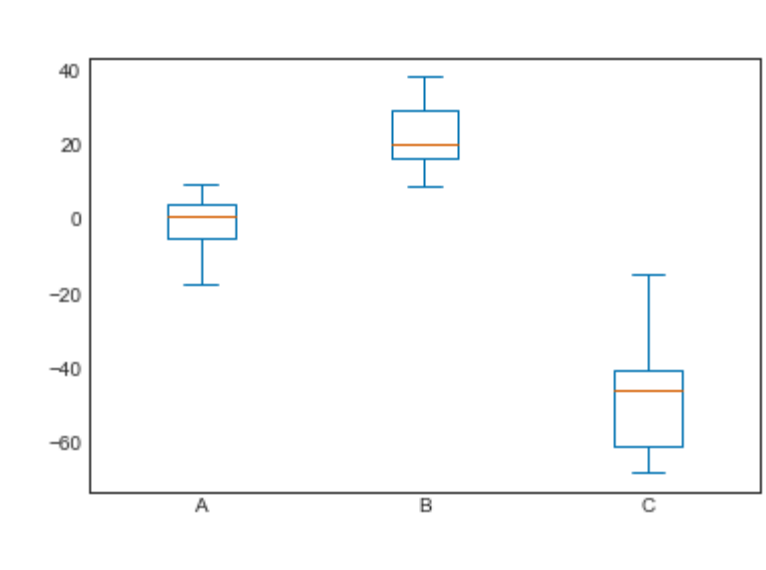


Out[23]:
```
<AxesSubplot:xlabel='A', ylabel='C'>
```

In [24]:
```python
ax = df.plot.scatter('A', 'C', c='B', s=df['B'], colormap='viridis')
ax.set_aspect('equal')
```

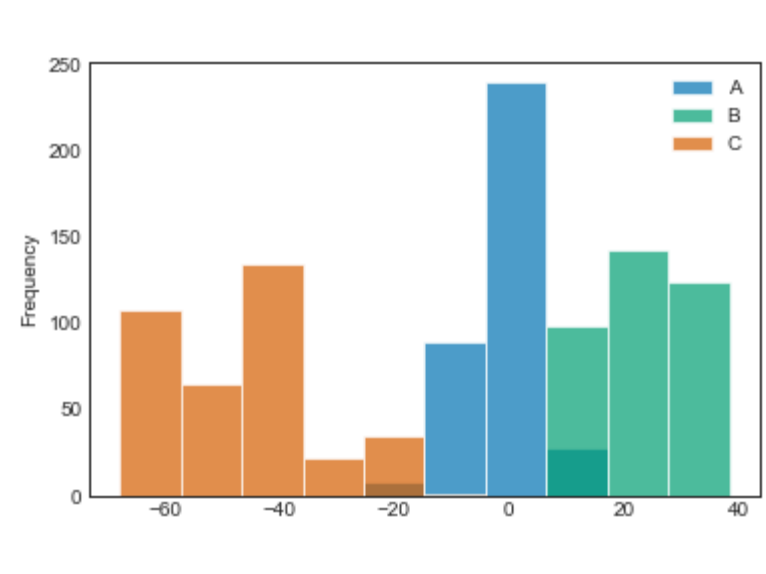


In [25]:
```python
df.plot.box()
```



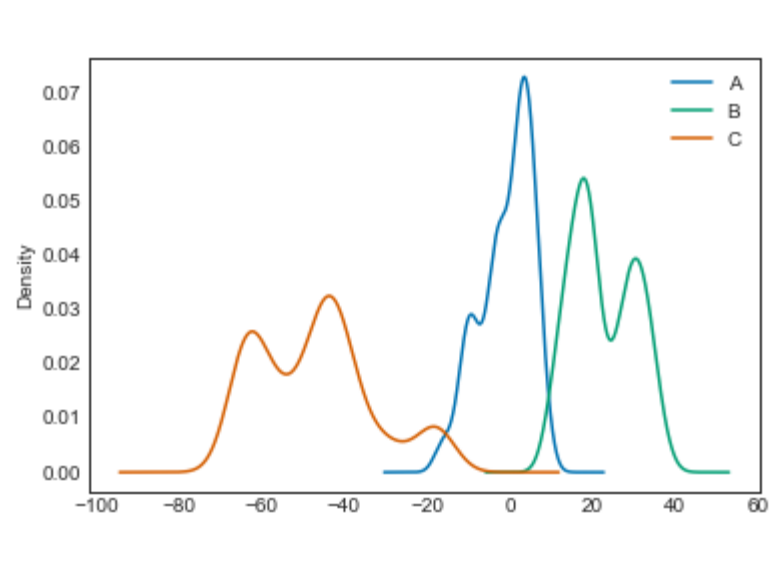
Out[25]:
```
<AxesSubplot:>
```

In [26]:
```python
df.plot.hist(alpha=0.7)
```



Out[26]:
```
<AxesSubplot:ylabel='Frequency'>
```

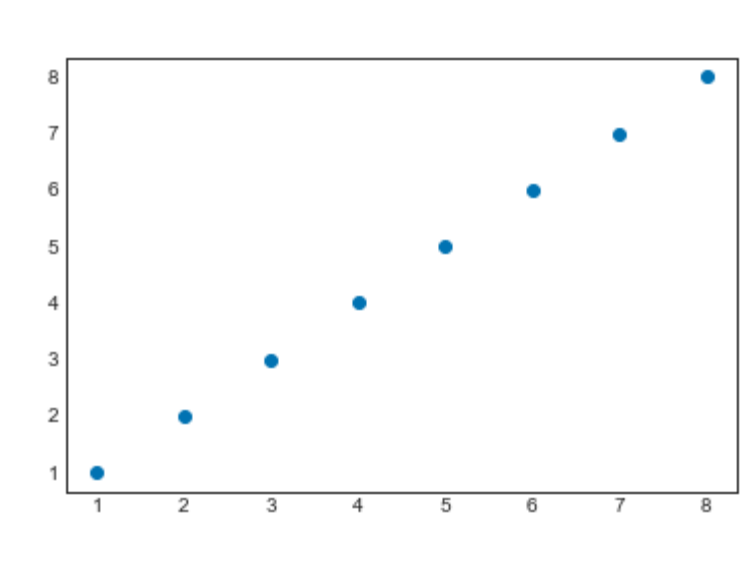
In [28]:
```python
# Kernel density estimate plot
df.plot.kde()
```



Out[28]:
```
<AxesSubplot:ylabel='Density'>
```

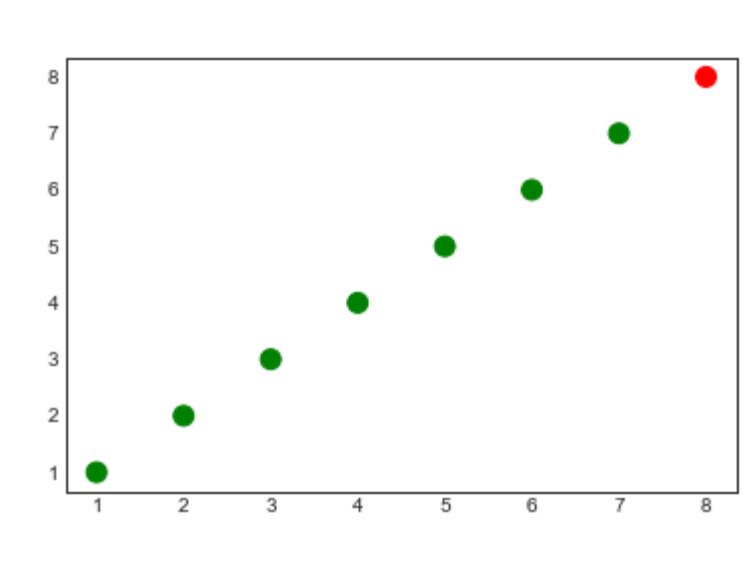
4 SCATTER PLOT

In [29]:
```python
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y = x

plt.figure()
plt.scatter(x, y)
```



Out[29]:
```
<matplotlib.collections.PathCollection at 0x203491ca310>
```

In [30]:
```python
x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y = x
colors = ['green'] * (len(x) - 1)
colors.append('red')

plt.figure()
plt.scatter(x, y, s=100, c=colors)
```

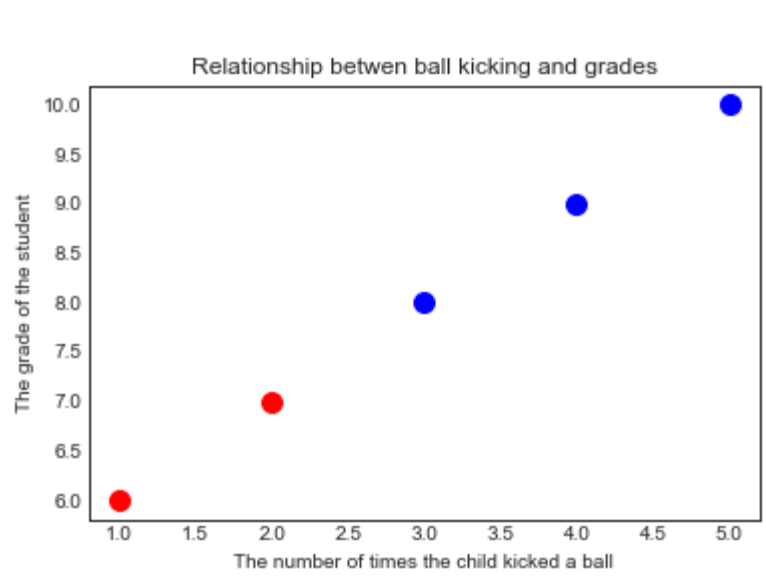


Out[30]:
```
<matplotlib.collections.PathCollection at 0x20349262df0>
```

In [31]:
```python
zip_generator = zip([1, 2, 3, 4, 5], [6, 7, 8, 9, 10])
x, y = zip(*zip_generator)
print(x)
print(y)
```

```
(1, 2, 3, 4, 5)
(6, 7, 8, 9, 10)
```

In [32]:
```python
plt.figure()
plt.scatter(x[:2], y[:2], s=100, c='red', label='Tall Students')
plt.scatter(x[2:], y[2:], s=100, c='blue', label='Short Students')
```

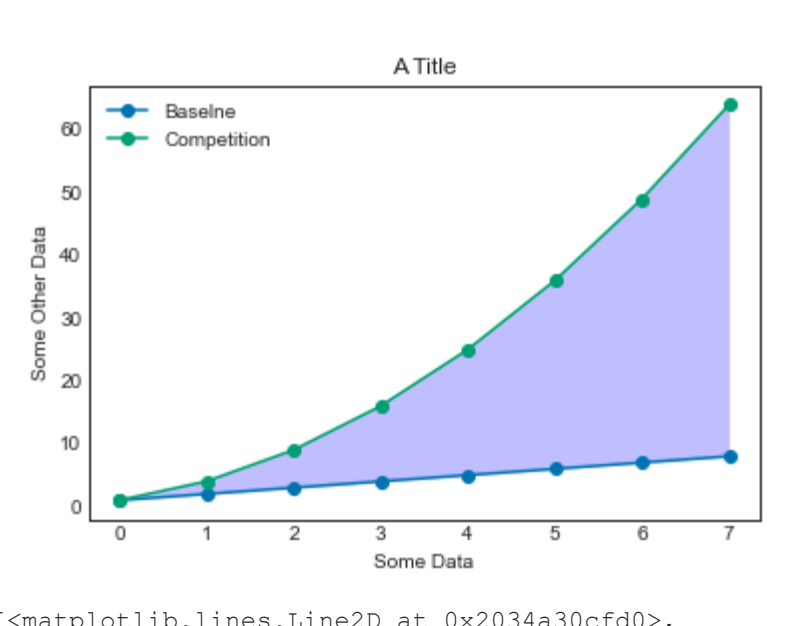


Out[32]:
```
<matplotlib.collections.PathCollection at 0x2034a2419d0>
```

In [33]:
```python
plt.xlabel('The number of times the child kicked a ball')
plt.ylabel('The grade of the student')
plt.title('Relationship betwen ball kicking and grades')
```

Out[33]:
```
Text(0.5, 1.0, 'Relationship betwen ball kicking and grades')
```

In [34]:
```python
# line plots

import matplotlib.pyplot as plt
import numpy as np
%matplotlib notebook

linear_data = np.array([1, 2, 3, 4, 5, 6, 7, 8])
quadratic_data = linear_data**2

plt.figure()
plt.plot(linear_data, '-o', quadratic_data, '-o')
```



Out[34]:
```
[<matplotlib.lines.Line2D at 0x2034a30cfd0>,
 <matplotlib.lines.Line2D at 0x20347465730>]
```

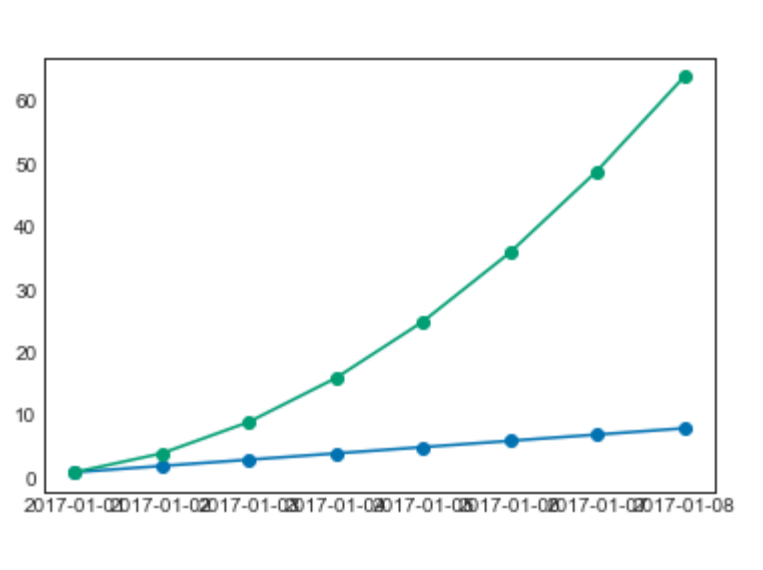
In [35]:
```python
plt.xlabel('Some Data')
plt.ylabel('Some Other Data')
plt.title('A Title')
plt.legend(['Baselne', 'Competition', 'Us'])
```

Out[35]:
```
<matplotlib.legend.Legend at 0x2034a3461f0>
```

In [36]:
```python
plt.gca().fill_between(range(len(linear_data)),
                       linear_data,
                       quadratic_data,
                       facecolor='blue',
                       alpha=0.25)
```

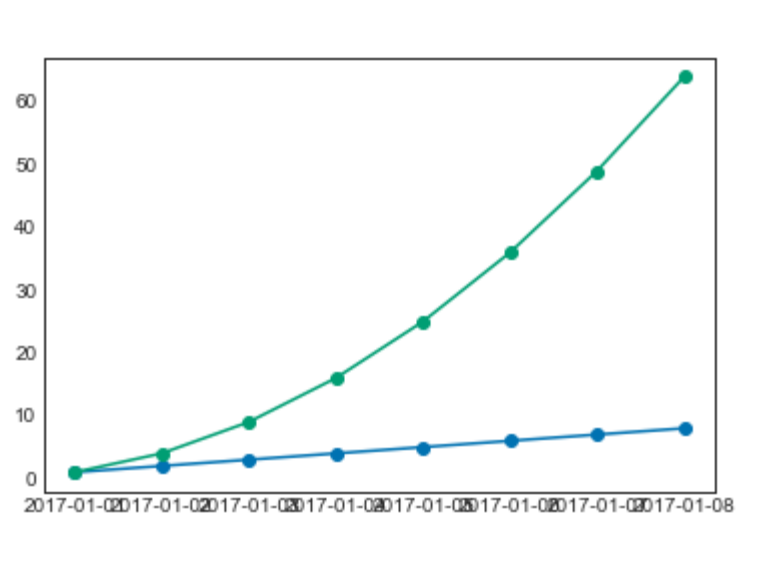
Out[36]:
```
<matplotlib.collections.PolyCollection at 0x2034a34b730>
```

In [37]:
```python
plt.figure()

observation_dates = np.arange('2017-01-01',
                              '2017-01-09',
                              dtype='datetime64[D]')

plt.plot(observation_dates, linear_data, '-o', observation_dates,
         quadratic_data, '-o')
```



Out[37]:
```
[<matplotlib.lines.Line2D at 0x2034a36c430>,
 <matplotlib.lines.Line2D at 0x2034a3631f0>]
```

In [38]:
```python
plt.figure()
observation_dates = np.arange('2017-01-01',
                              '2017-01-09',
                              dtype='datetime64[D]')
observation_dates = list(map(pd.to_datetime, observation_dates))
plt.plot(observation_dates, linear_data, '-o', observation_dates,
         quadratic_data, '-o')
```



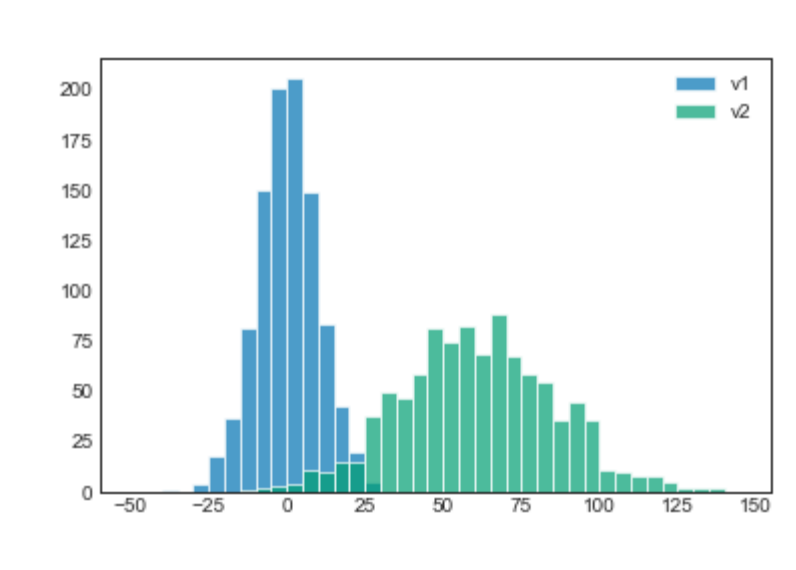
Out[38]:
```
[<matplotlib.lines.Line2D at 0x2034a3cb8e0>,
 <matplotlib.lines.Line2D at 0x2034a3e7dc0>]
```

5 SEABORN LIBRARY

In [39]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib notebook
```

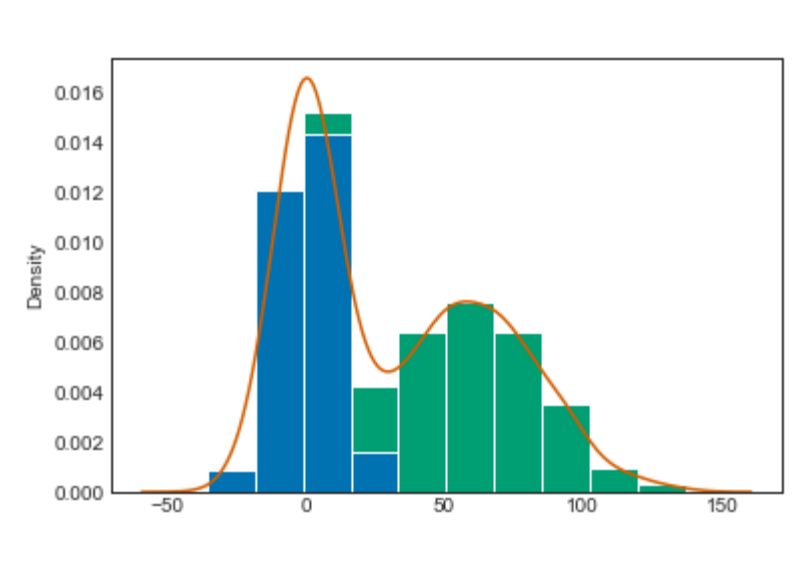
In [40]:
```python
np.random.seed(1234)

v1 = pd.Series(np.random.normal(0, 10, 1000), name='v1')
v2 = pd.Series(2 * v1 + np.random.normal(60, 15, 1000), name='v2')
```

In [41]:
```python
plt.figure()
plt.hist(v1, alpha=0.7, bins=np.arange(-50, 150, 5), label='v1')
plt.hist(v2, alpha=0.7, bins=np.arange(-50, 150, 5), label='v2')
plt.legend()
```



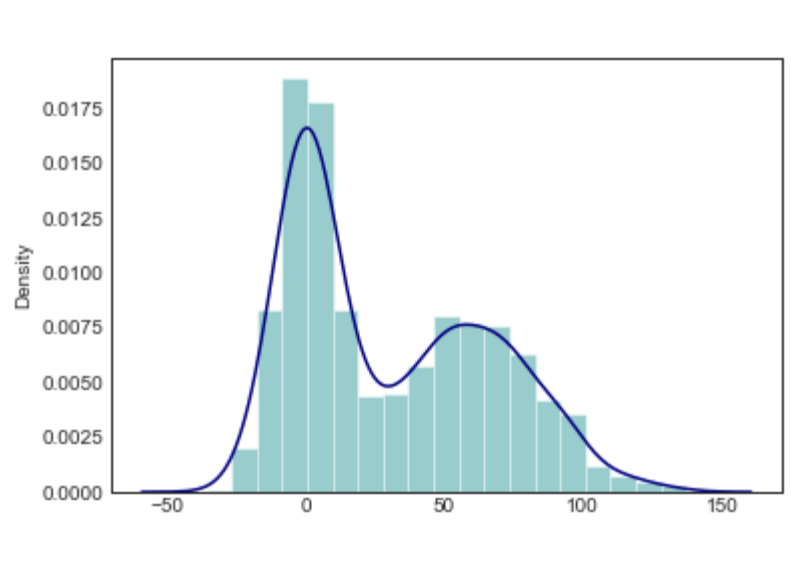
Out[41]: `<matplotlib.legend.Legend at 0x2034a44a460>`

In [42]:
```python
plt.figure()
plt.hist([v1, v2], histtype='barstacked', density=True)
v3 = np.concatenate((v1, v2))
sns.kdeplot(v3)
```



Out[42]: `<AxesSubplot:ylabel='Density'>`

In [43]:
```python
plt.figure()
sns.distplot(v3, hist_kws={'color': 'Teal'}, kde_kws={'color': 'Navy'})
```

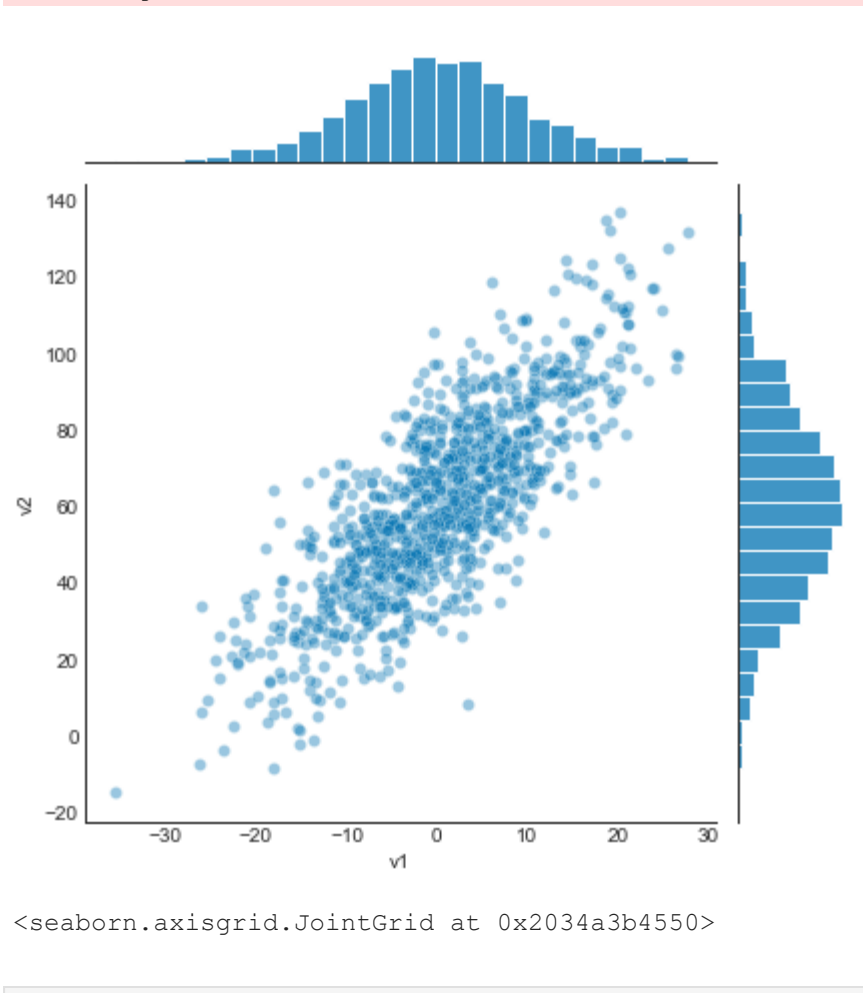



```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[43]: `<AxesSubplot:ylabel='Density'>`

In [45]:
```python
sns.jointplot(v1, v2, alpha=0.4)
```

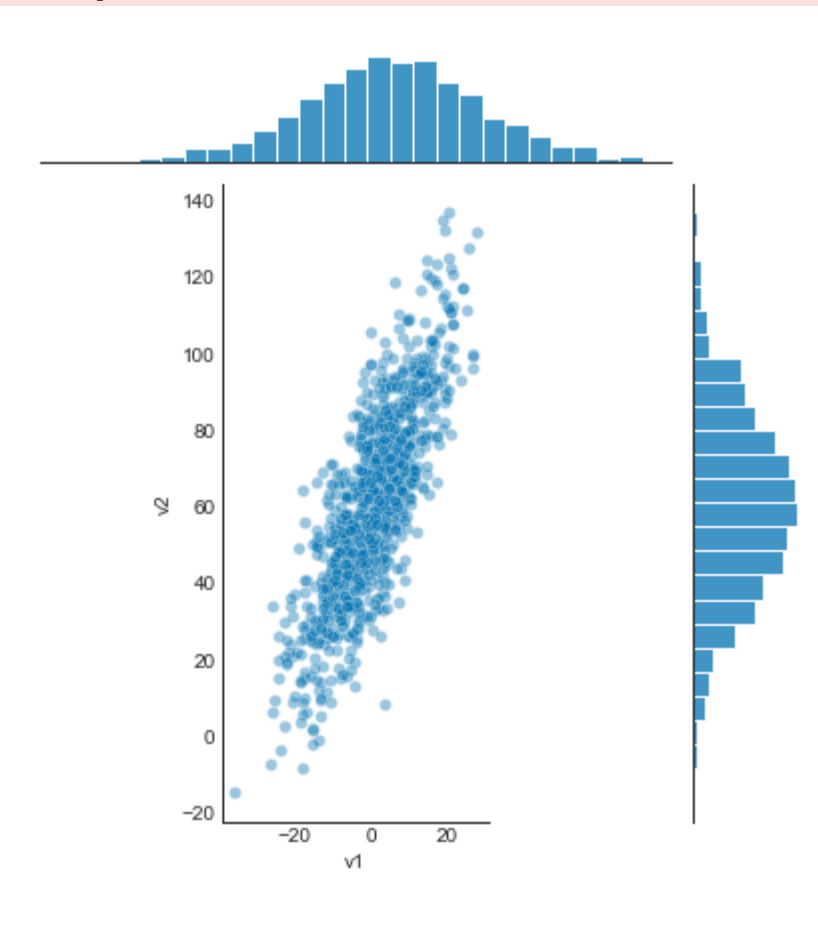
```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Out[45]: `<seaborn.axisgrid.JointGrid at 0x2034a3b4550>`

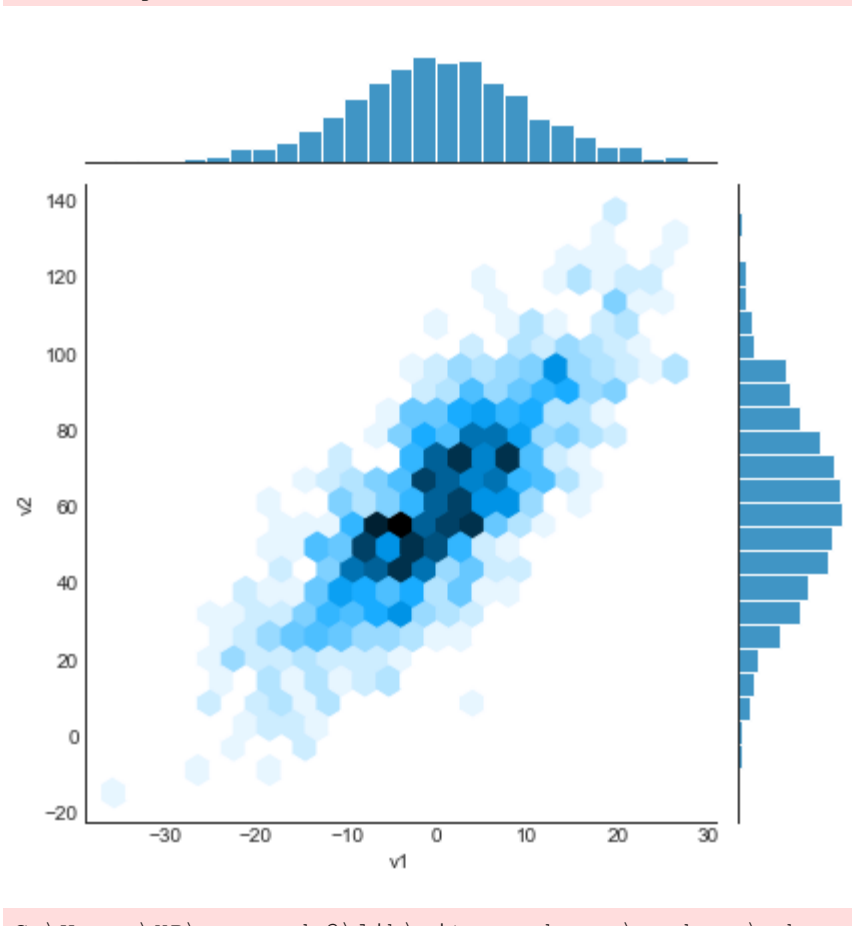
In [46]:
```python
grid = sns.jointplot(v1, v2, alpha=0.4)
grid.ax_joint.set_aspect('equal')
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

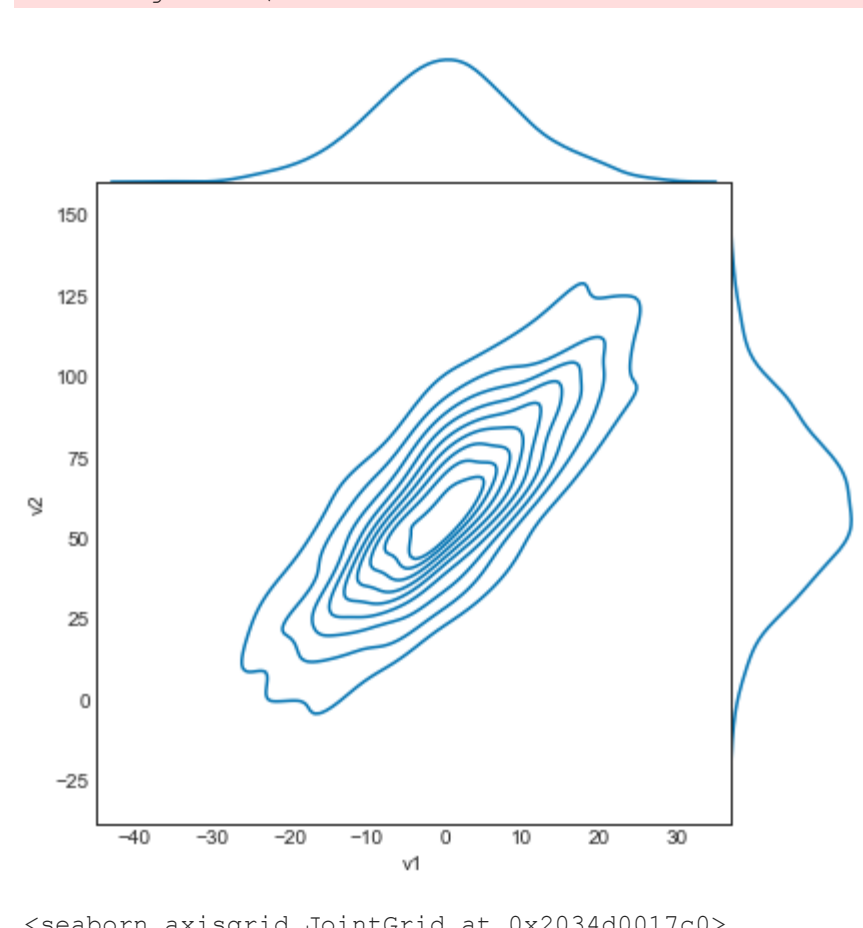


In [50]:
```python
plt.figure()
sns.jointplot(v1, v2, kind='hex')

sns.set_style('white')
sns.jointplot(v1, v2, kind='kde', space=0)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```




```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Out[50]: `<seaborn.axisgrid.JointGrid at 0x2034d0017c0>`

In [51]:
```python
iris = pd.read_csv('/DOCUMENTS/COLLEGE/CLASSES/EXPERIMENT_NO_1/Iris.csv')
```

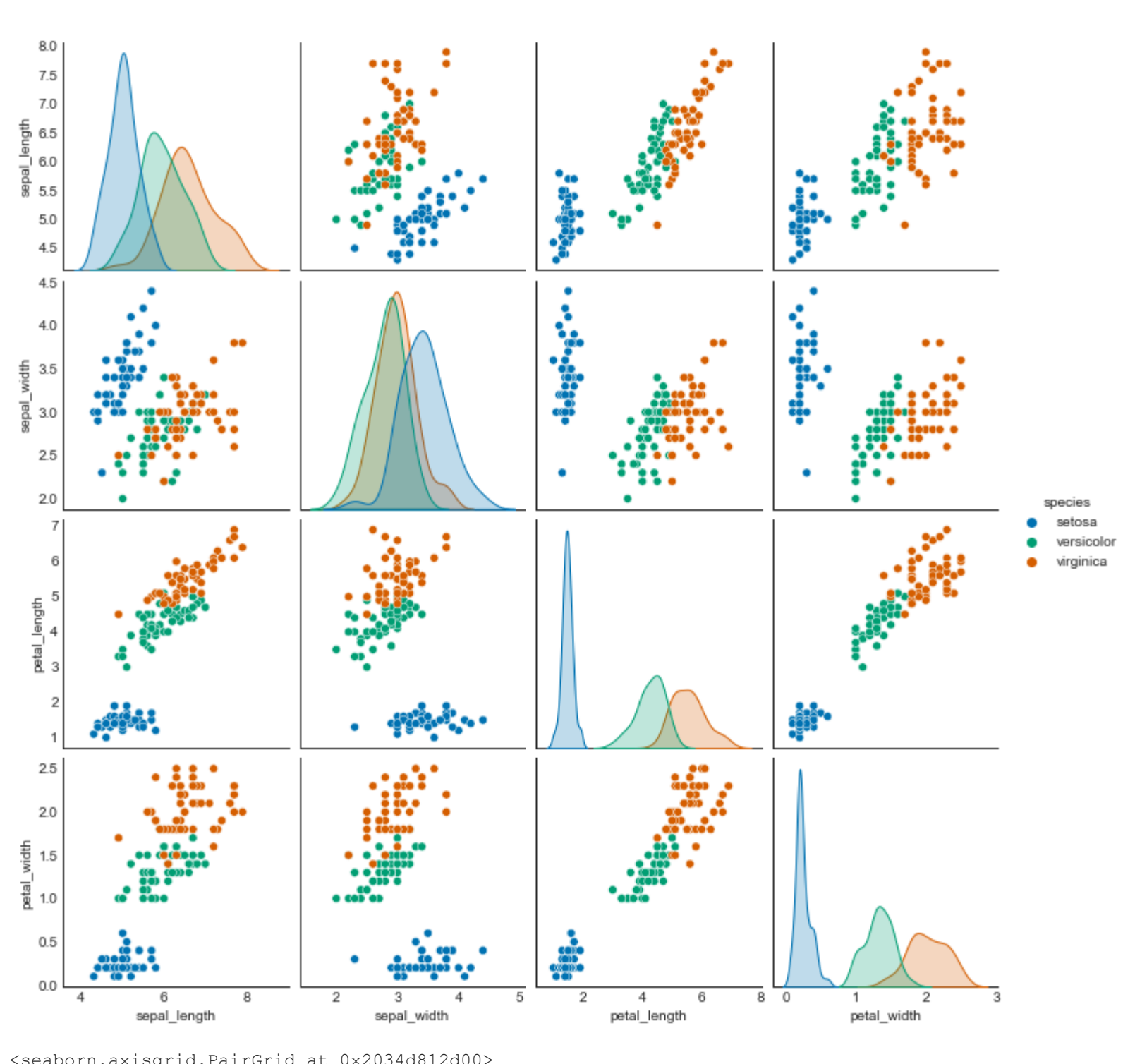
In [52]:
```python
iris.head()
```

Out[52]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [53]:
```python
iris = sns.load_dataset("iris")
```

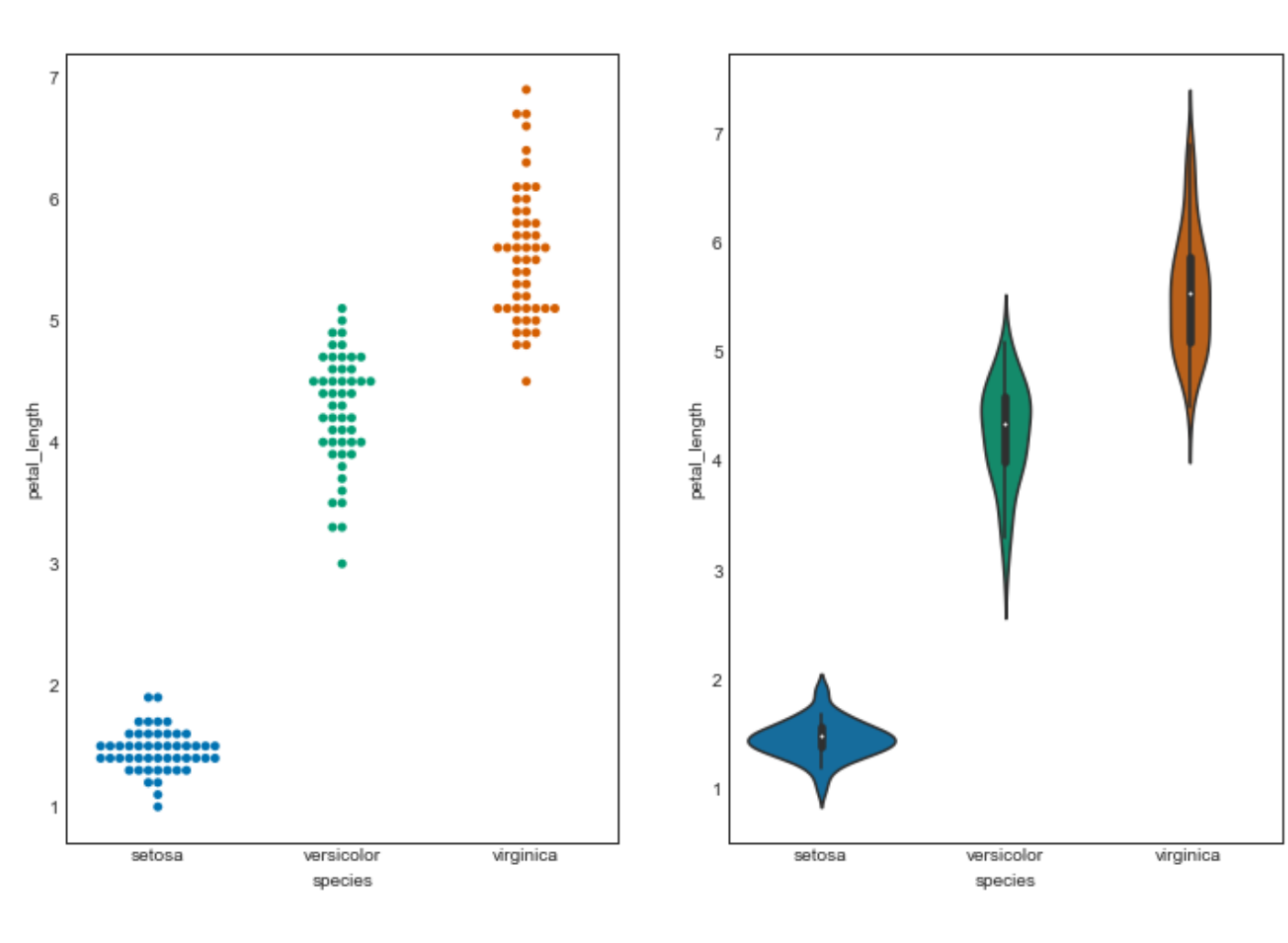
In [54]:
```python
sns.pairplot(iris, hue='species', diag_kind='kde')
```



Out[54]: `<seaborn.axisgrid.PairGrid at 0x2034d812d00>`

In [55]:
```python
# Violin plot and swarm plot
```

In [56]:
```python
plt.figure(figsize=(12, 8))
plt.subplot(121)
sns.swarmplot('species', 'petal_length', data=iris)
plt.subplot(122)
sns.violinplot('species', 'petal_length', data=iris)
```



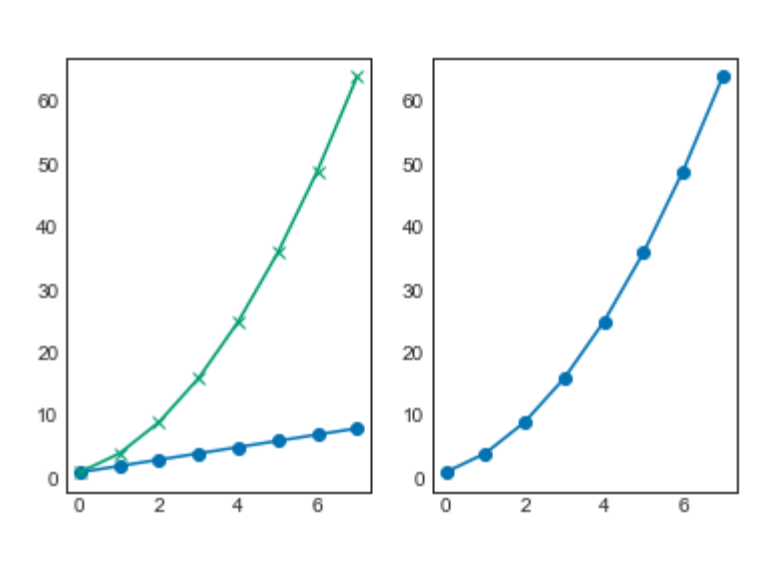

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[56]: `<AxesSubplot:xlabel='species', ylabel='petal_length'>`

6 SUBPLOT

In [57]:
```python
import numpy as np
import matplotlib.pyplot as plt
%matplotlib notebook
```

In [58]:
```python
plt.figure()

plt.subplot(1, 2, 1)
linear_data = np.array([1, 2, 3, 4, 5, 6, 7, 8])

plt.plot(linear_data, '-o')
```



Out[58]: `[<matplotlib.lines.Line2D at 0x2034df71af0>]`

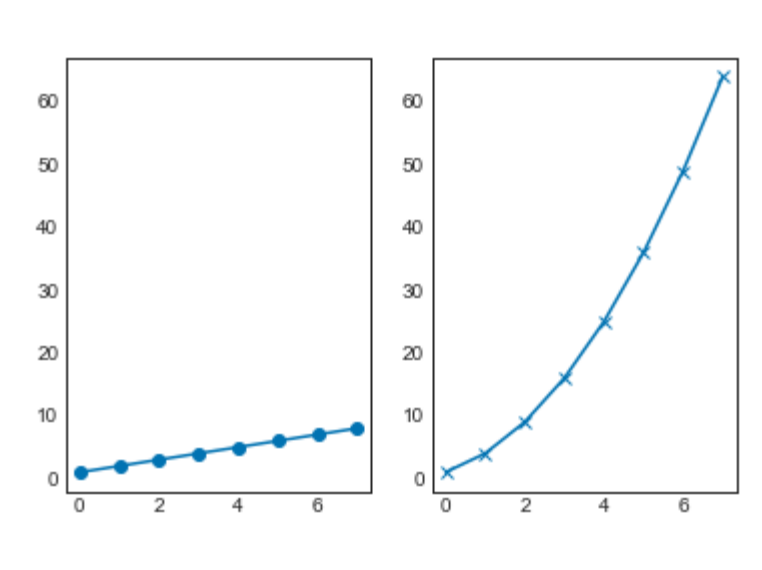
In [59]:
```python
exponential_data = linear_data**2

plt.subplot(1, 2, 2)
plt.plot(exponential_data, '-o')
```

Out[59]: `[<matplotlib.lines.Line2D at 0x2034dfcd580>]`

In [60]:
```python
plt.subplot(1, 2, 1)
plt.plot(exponential_data, '-x')
```

Out[60]: `[<matplotlib.lines.Line2D at 0x2034dff36a0>]`

In [61]:
```python
# To lock the y-axis for two plots side-by-side
plt.figure()
ax1 = plt.subplot(1, 2, 1)
plt.plot(linear_data, '-o')
ax2 = plt.subplot(1, 2, 2, sharey=ax1)
plt.plot(exponential_data, '-x')
```



Out[61]: `[<matplotlib.lines.Line2D at 0x2034e0679a0>]`

----------------X-----------------------X------------------------X-----
-----------------------X-----------