Plotando no Pandas

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy

%matplotlib notebook
```

1 - Estilos predefinidos do matplotlib

In [2]:

```
plt.style.available
```

Out[2]:

```
['Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',
 'seaborn-paper',
 'seaborn-pastel',
 'seaborn-poster',
 'seaborn-talk',
 'seaborn-ticks',
 'seaborn-white',
 'seaborn-whitegrid',
 'tableau-colorblind10']
```

In [3]:

```
plt.style.use('seaborn-colorblind')
```

2 - Criando um dataframe

In [4]:

```
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[4]:

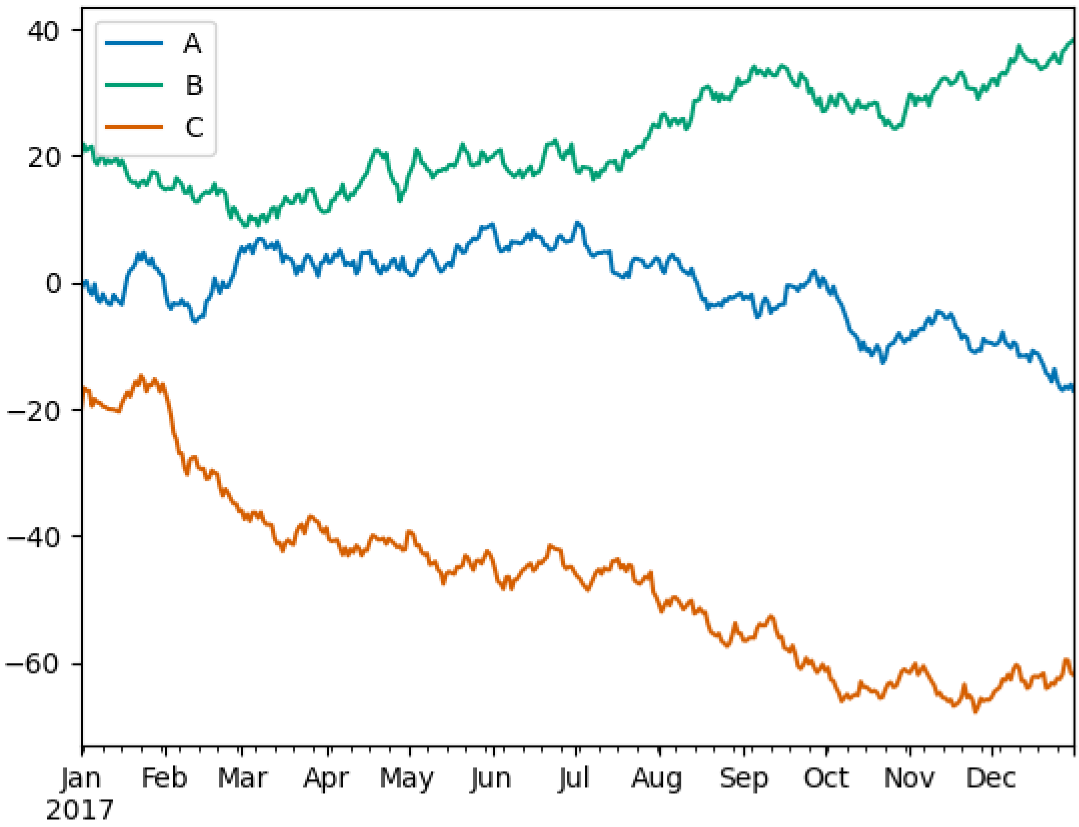
	A	B	C
2017-01-01	-1.085631	20.059291	-20.230904
2017-01-02	-0.088285	21.803332	-16.659325
2017-01-03	0.194693	20.835588	-17.055481

	A	B	C
2017-01-04	-1.311601	21.255156	-17.093802
2017-01-05	-1.890202	21.462083	-19.518638

3 - Plot direto no dataframe

In [5]:

```
df.plot()
```



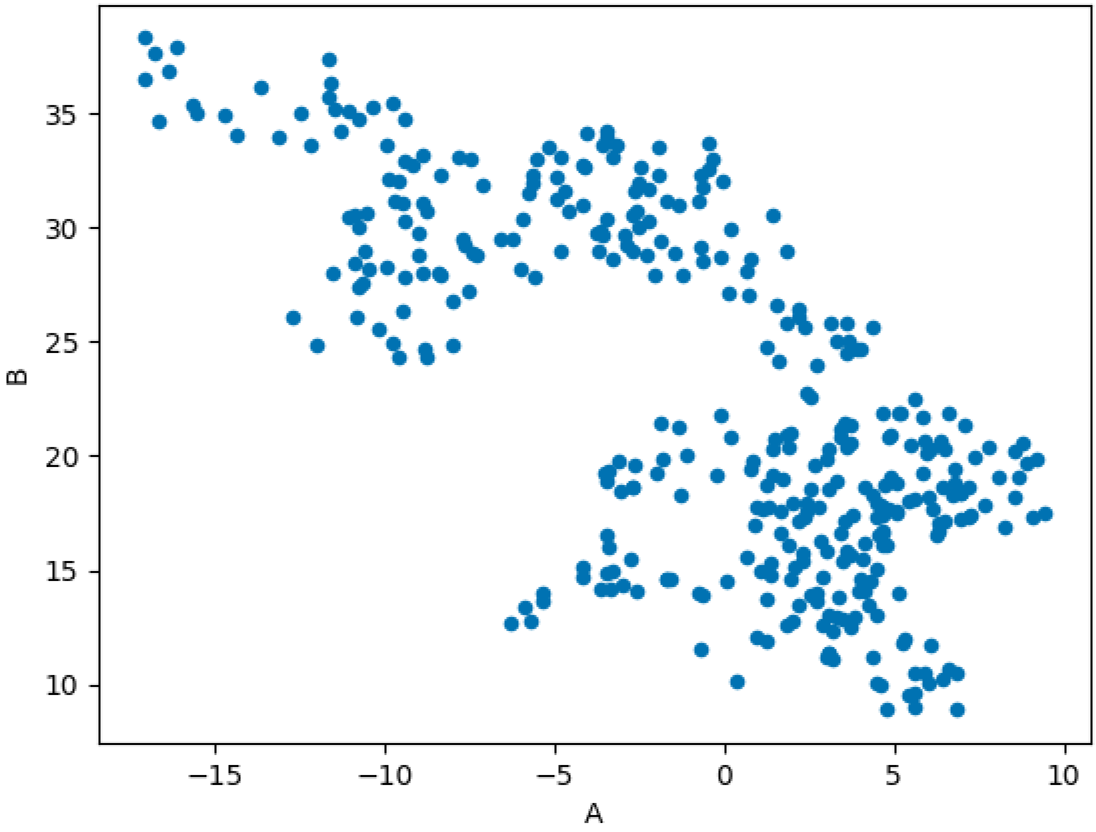
Out[5]:

```
<AxesSubplot:>
```

4 - Tipo de plotagem

In [6]:

```
df.plot('A', 'B', kind='scatter')
```



```
Out[6]: <AxesSubplot:xlabel='A', ylabel='B'>
```

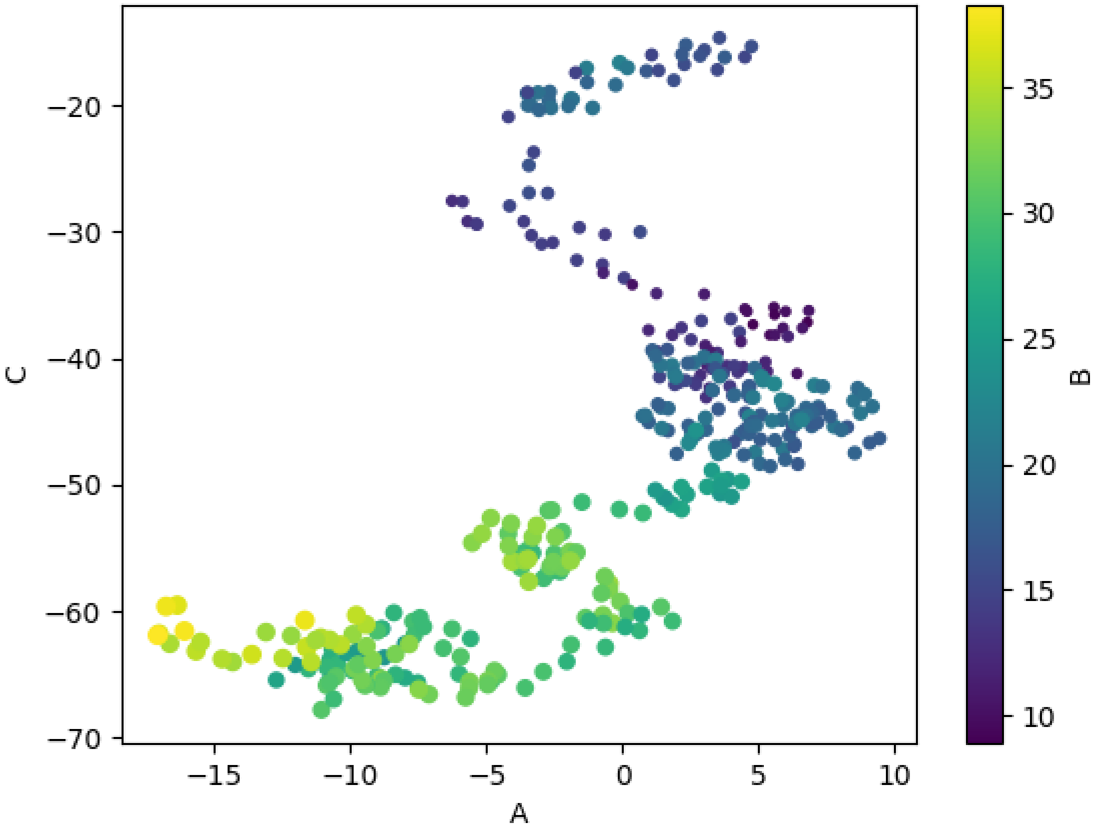
Você também pode escolher o tipo de plotagem usando os métodos `DataFrame.plot.kind` em vez de fornecer o argumento da palavra-chave `kind` .

`kind` :

- `'line'` : line plot (default)
- `'bar'` : vertical bar plot
- `'barh'` : horizontal bar plot
- `'hist'` : histogram
- `'box'` : boxplot
- `'kde'` : Kernel Density Estimation plot
- `'density'` : same as 'kde'
- `'area'` : area plot
- `'pie'` : pie plot
- `'scatter'` : scatter plot
- `'hexbin'` : hexbin plot

5 - A contra C, com tamanho e cor conforme os valores de B

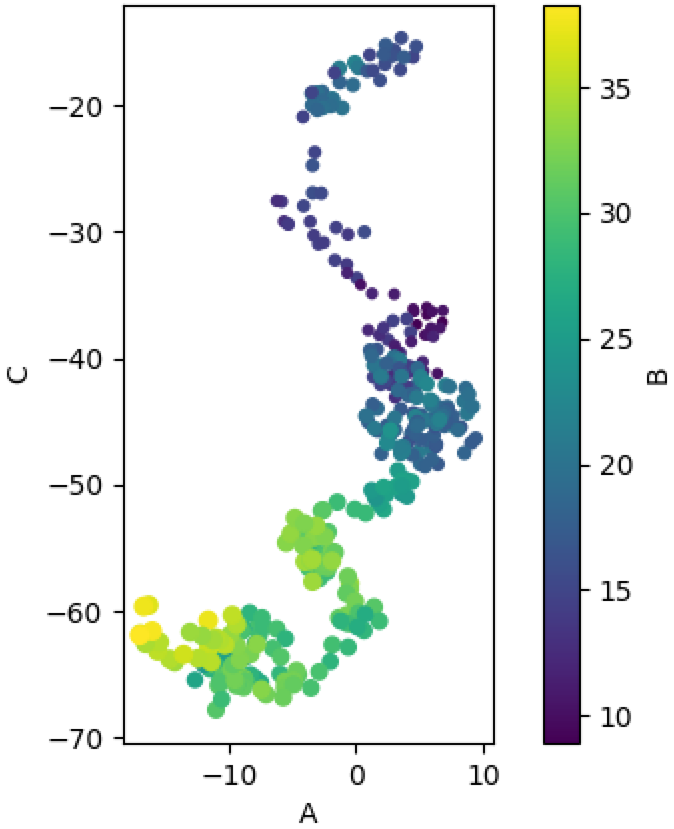
```
In [7]: df.plot.scatter?
        df.plot.scatter('A', 'C', c='B', s=df['B'], colormap='viridis')
```



```
Out[7]: <AxesSubplot:xlabel='A', ylabel='C'>
```

6 - Mudar aspecto do objeto "eixos"

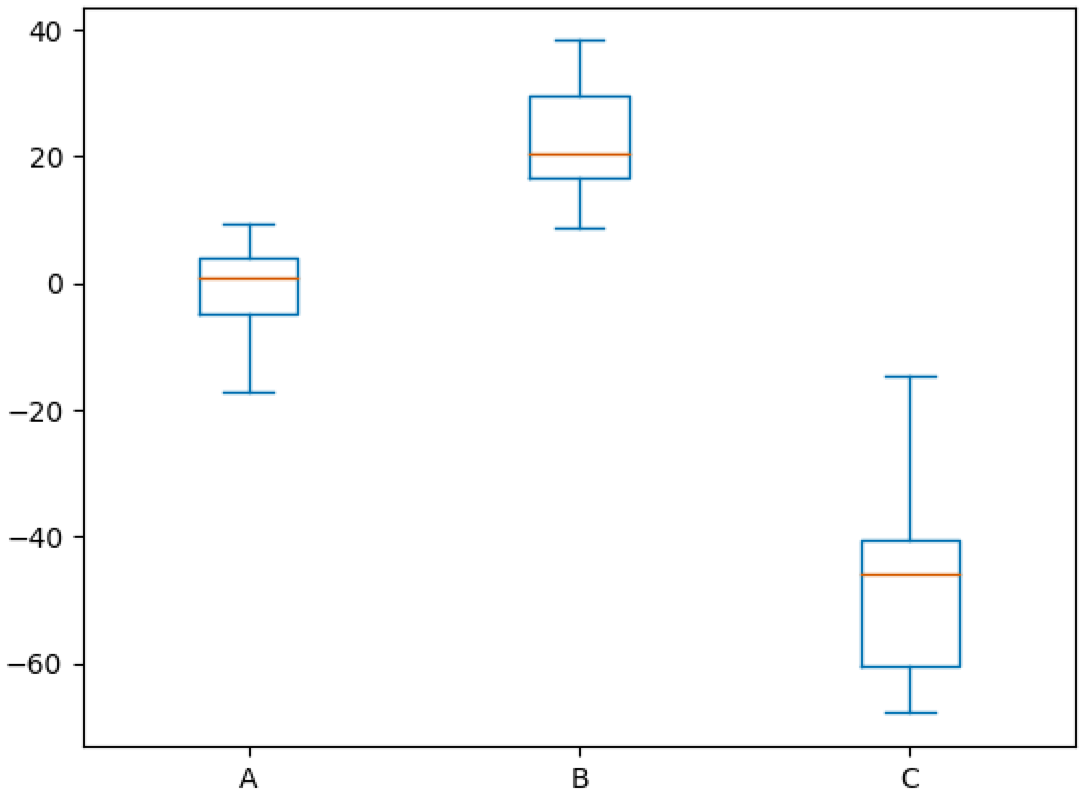
```
In [8]:
ax = df.plot.scatter('A', 'C', c='B', s=df['B'], colormap='viridis')

# Define o aspecto da escala do eixo, ou seja...
# a proporção da unidade y para a unidade x.
# 'equal' significa mesma escala de dados em ambos os eixos

ax.set_aspect('equal')
```



7 - Plot em caixa do dataframe

In [9]:

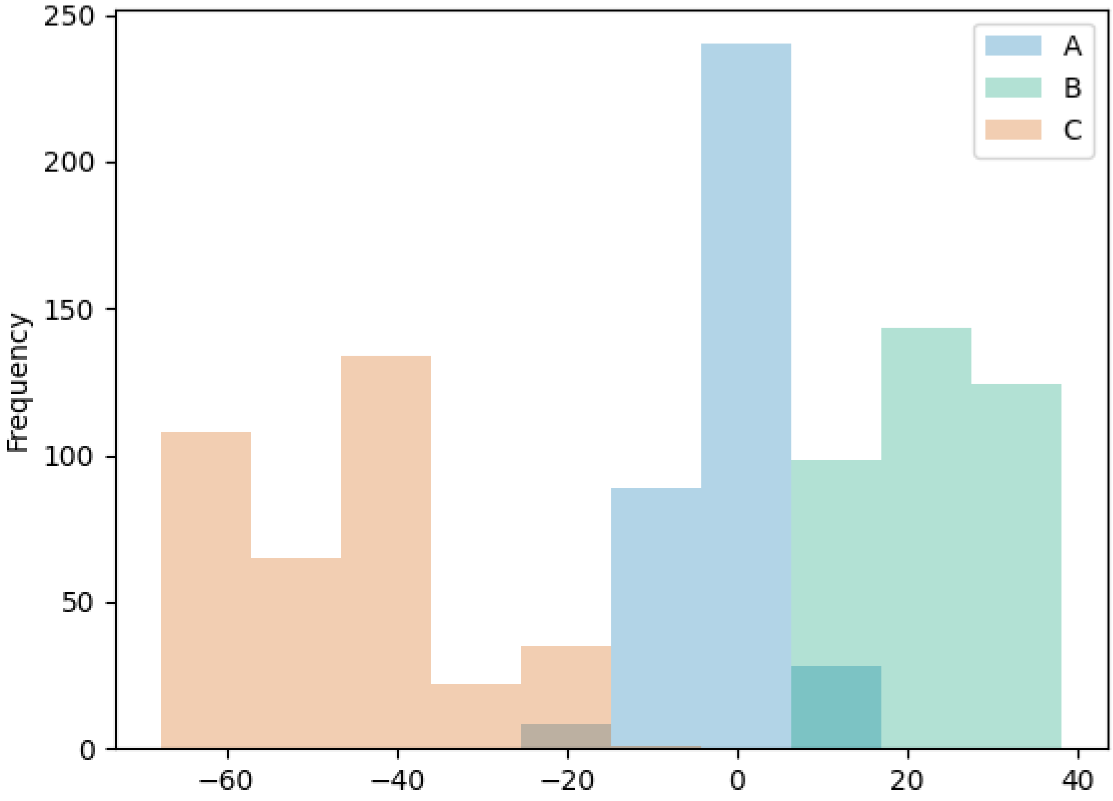
```
df.plot.box()
```



```
Out[9]: <AxesSubplot:>
```

8 - Histograma do dataframe

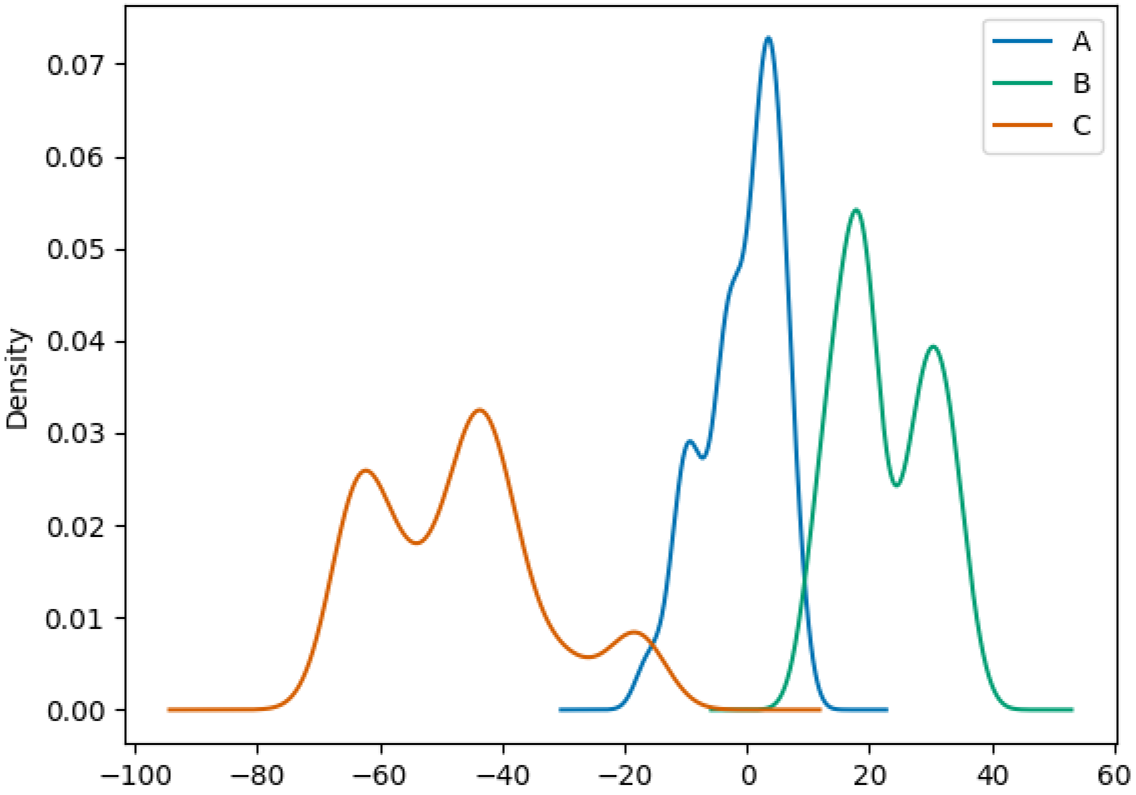
```
In [10]: df.plot.hist(alpha=0.3)
```



```
Out[10]: <AxesSubplot:ylabel='Frequency'>
```

9 - Gráficos de estimativa de densidade do kernel (KDE)

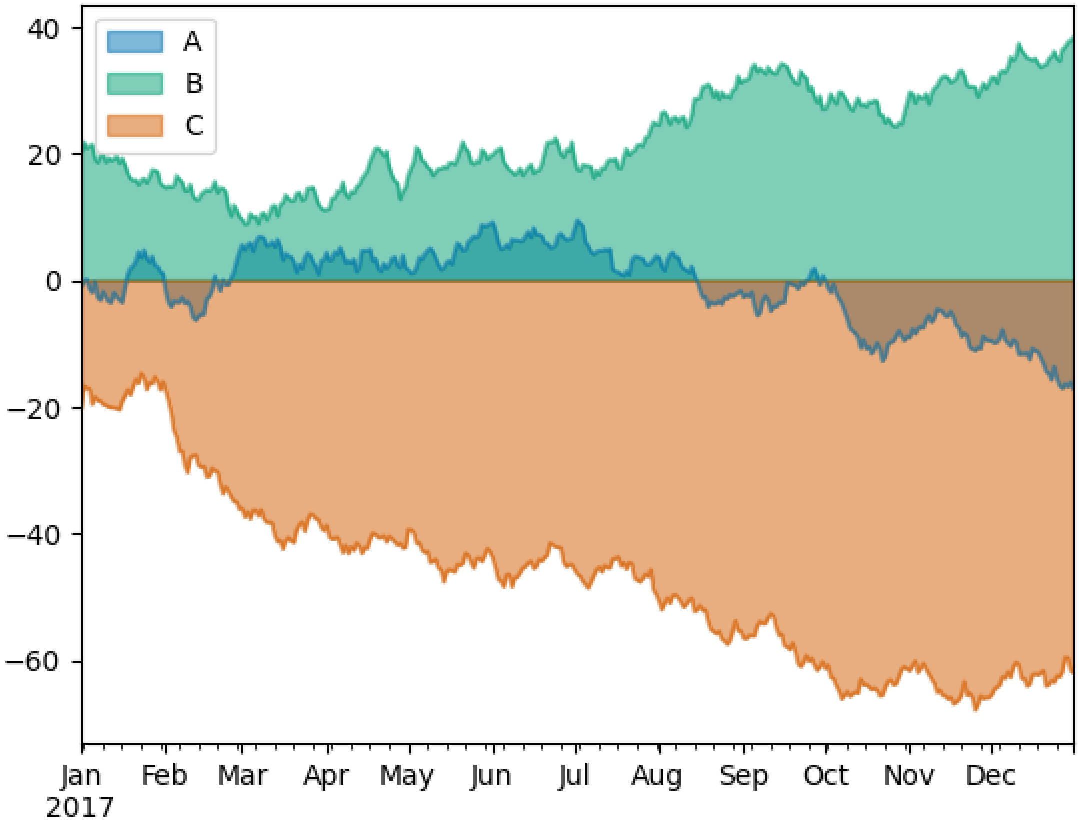
```
In [11]: df.plot.kde();
```



10 - Para produzir um gráfico de área não empilhado

In [12]:

```
df.plot.area(stacked=False);
```



11 - Carregando a base de dados Íris

In [14]:

```
iris = pd.read_csv('./Data/iris.csv')
iris.head()
```

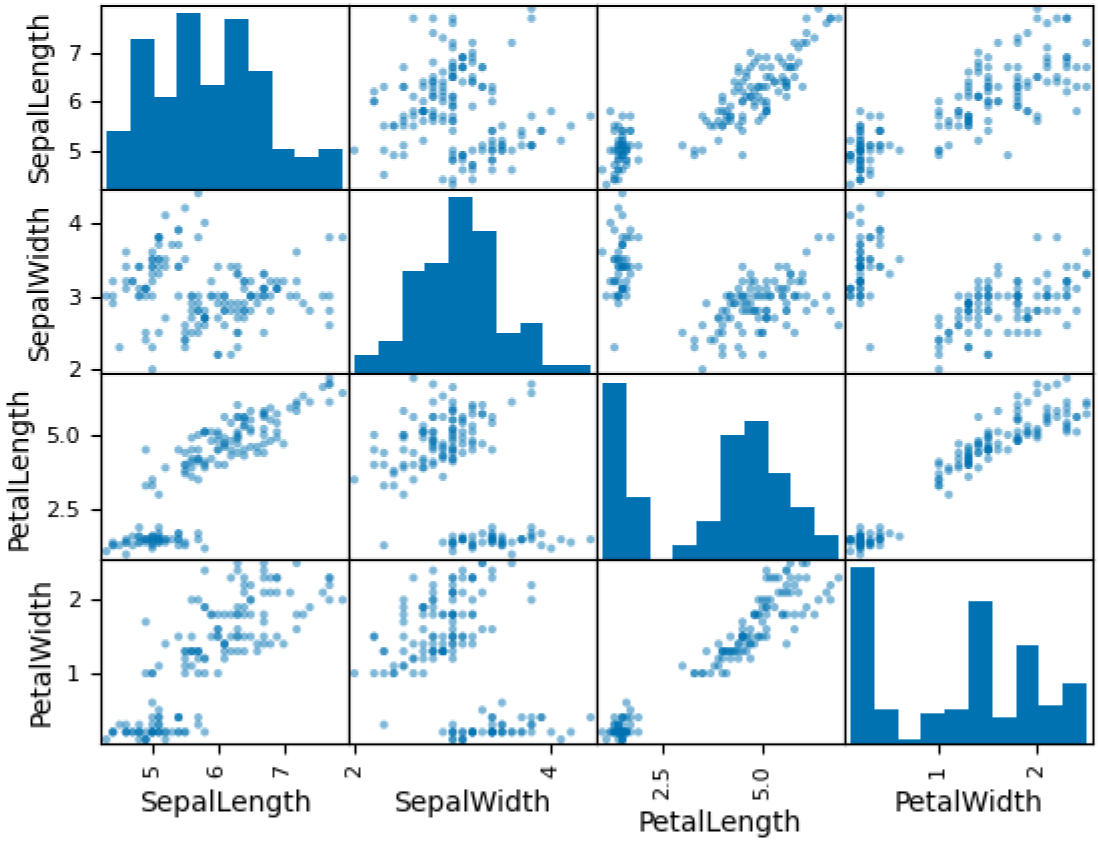
Out[14]:

	SepalLength	SepalWidth	PetalLength	PetalWidth	Name
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

12 - Matriz de gráficos de dispersão automática

In [15]:

```
pd.plotting.scatter_matrix(iris);
```



13 - Plotagem de coordenadas paralelas

In [16]:

```
pd.plotting?
```

Seaborn

14 - Importar Seaborn e criar séries pandas

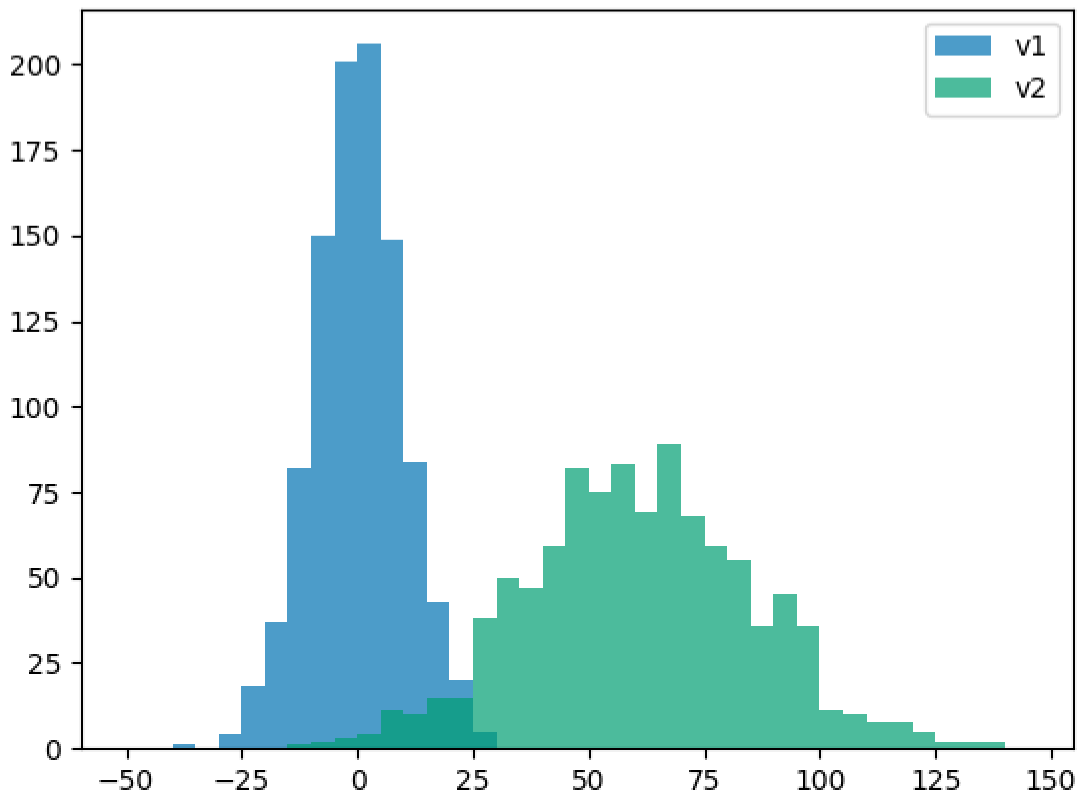
In [17]:

```
import seaborn as sns

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')
```

15 - Plotando histogramas

In [18]:

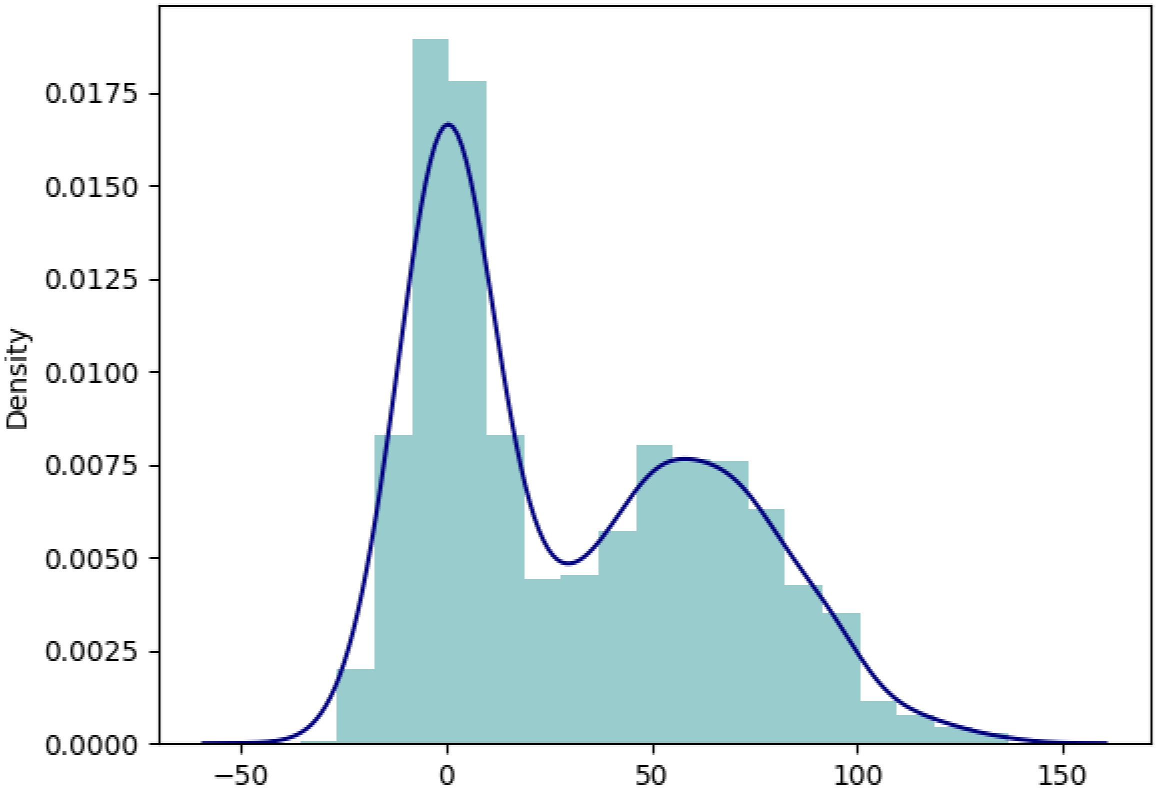
```
plt.figure()
plt.hist(v1, alpha=0.7, bins=np.arange(-50,150,5), label='v1');
plt.hist(v2, alpha=0.7, bins=np.arange(-50,150,5), label='v2');
plt.legend();
v1.describe()
```



Out[18]:
```
count    1000.000000
mean        0.157406
std         9.735531
min       -35.635167
25%        -6.243196
50%         0.177609
75%         6.688061
max        27.638441
Name: v1, dtype: float64
```

16 - Função distplot()

In [19]:
```
v3 = np.concatenate((v1,v2))
plt.figure()
sns.distplot(v3, hist_kws={'color':'Teal'}, kde_kws={'color':'Navy'});
```



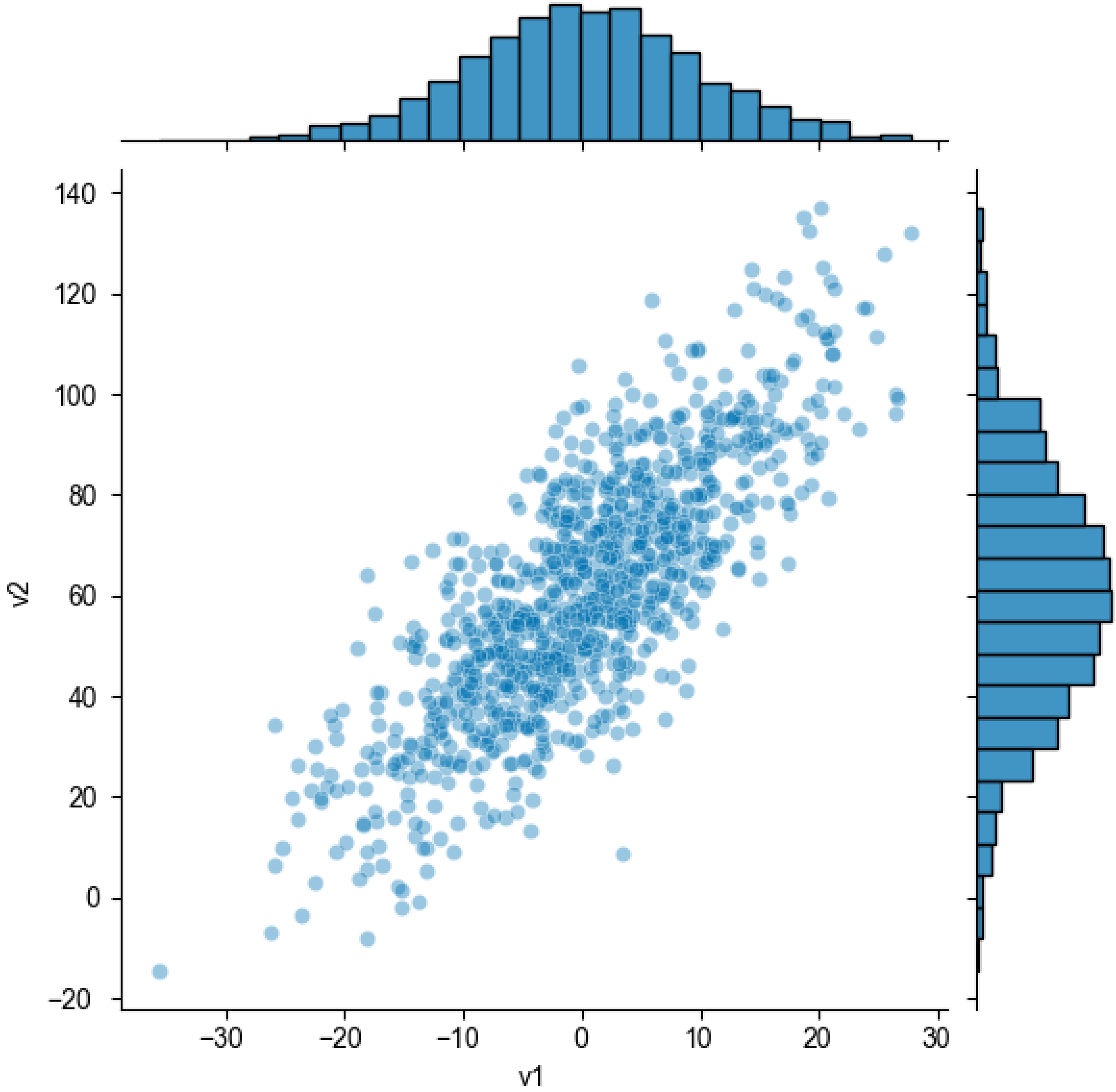
```
/Users/marinaramalhetedesouza/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/seabo
rn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be r
emoved in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

17 - Plotagem conjunta

In [21]:

```
sns.jointplot(v1, v2, alpha=0.4);
```

```
/Users/marinaramalhetedesouza/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/seabo
rn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. F
rom version 0.12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

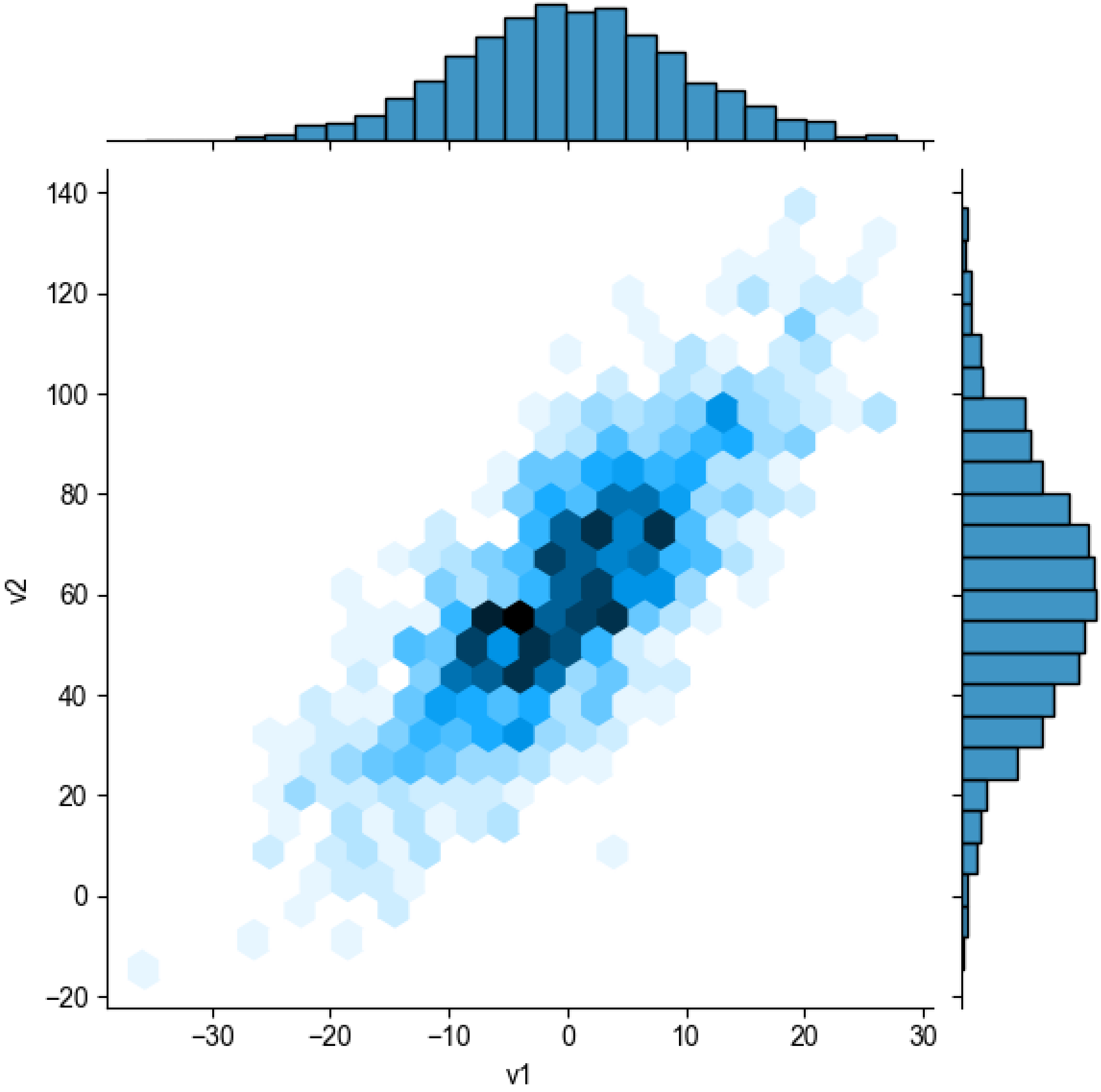


18 - Estilo de bins hexagonal

In [22]:

```
sns.jointplot(v1,v2,kind='hex')
```

```
/Users/marinaramalhetedesouza/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/seabo
rn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. F
rom version 0.12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



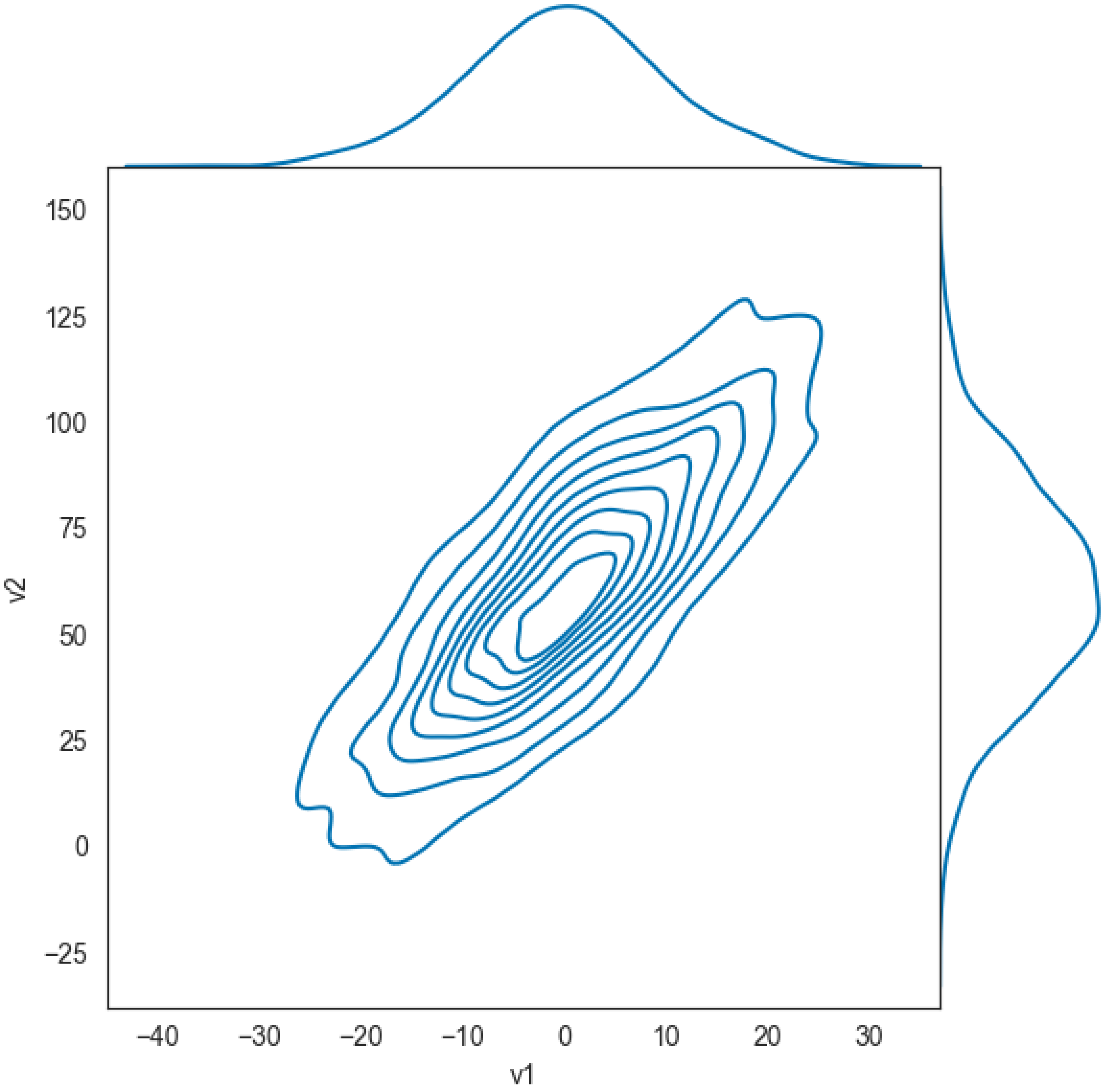
```
Out[22]: <seaborn.axisgrid.JointGrid at 0x7fd2bef0d970>
```

19 - Estimativas de densidade do kernel com mudança de estilo seaborn

In [23]:

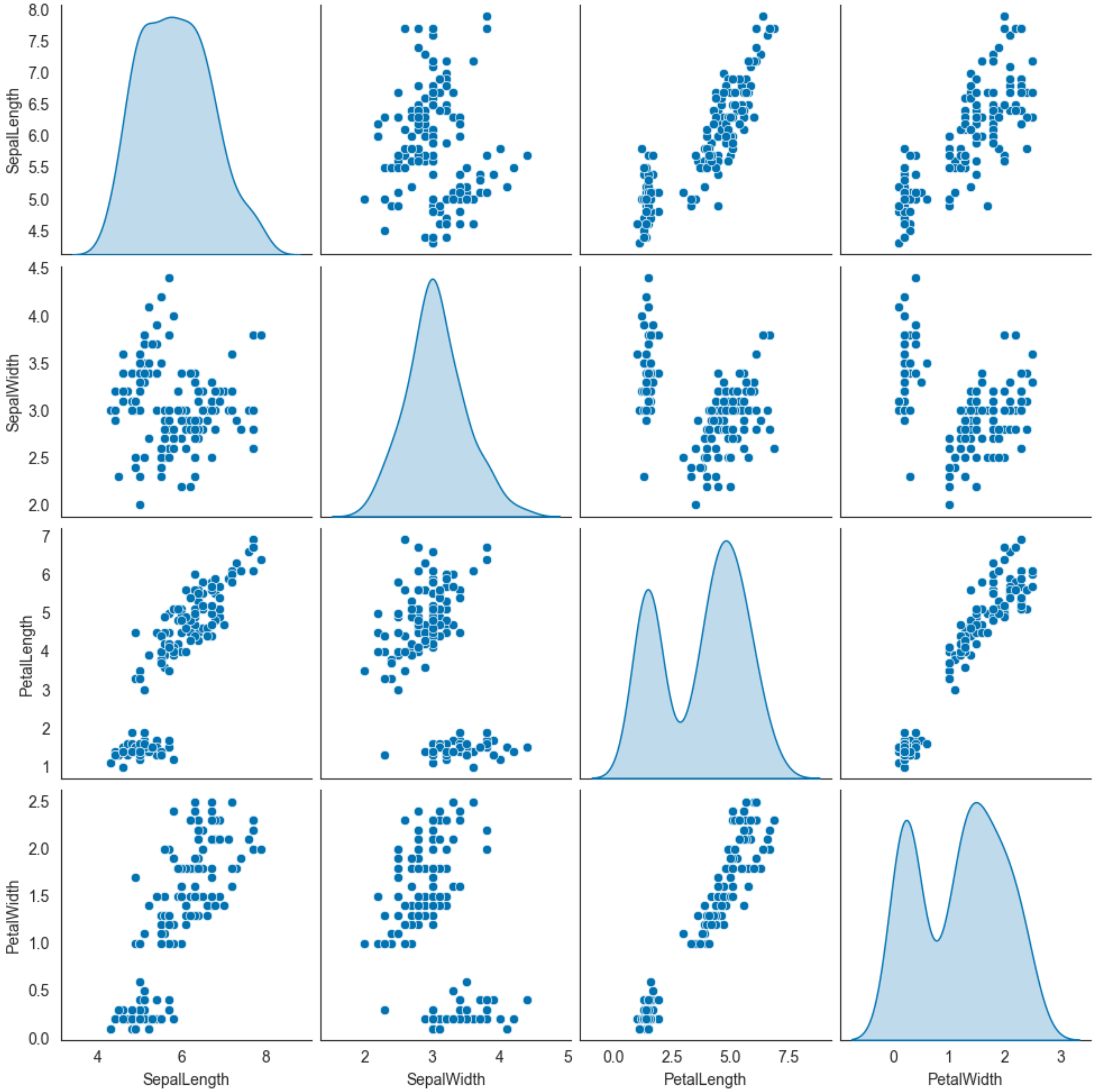
```
sns.set_style('white')

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

```
/Users/marinaramalhetedesouza/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/seabo
rn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. F
rom version 0.12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



20 - Matriz de dispersão do seaborn

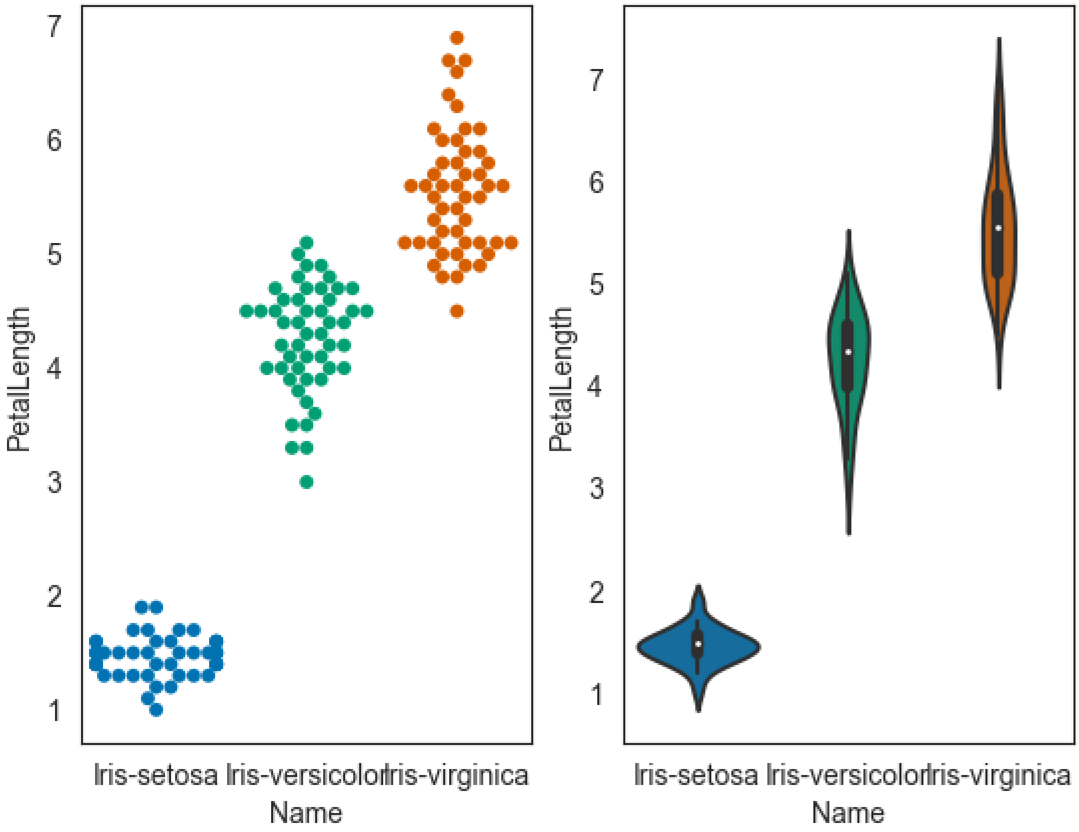
```
In [25]: sns.pairplot(iris, diag_kind='kde')
```



Out[25]:
```
<seaborn.axisgrid.PairGrid at 0x7fd2c0960a60>
```

21 - Plotagem swarm e de violino

In [27]:
```
plt.figure()
plt.subplot(121)
sns.swarmplot('Name', 'PetalLength', data=iris);
plt.subplot(122)
sns.violinplot('Name', 'PetalLength', data=iris);
```



```
/Users/marinaramalhetedesouza/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/seabo
rn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. F
rom version 0.12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/Users/marinaramalhetedesouza/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/seabo
rn/categorical.py:1296: UserWarning: 44.0% of the points cannot be placed; you may want to
decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
/Users/marinaramalhetedesouza/opt/anaconda3/envs/ml-impa/lib/python3.8/site-packages/seabo
rn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. F
rom version 0.12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Exercícios de consultas no Pandas

22 - Carregando dataframe de medalhas nas olimpíadas

In [29]:

```python
import pandas as pd

df = pd.read_csv('./Data/olympics.csv', index_col=0, skiprows=1)

for col in df.columns:
    if col[:2]=='01':
        df.rename(columns={col:'Gold'+col[4:]}, inplace=True)
    if col[:2]=='02':
        df.rename(columns={col:'Silver'+col[4:]}, inplace=True)
    if col[:2]=='03':
        df.rename(columns={col:'Bronze'+col[4:]}, inplace=True)
    if col[:1]=='№':
        df.rename(columns={col:'#'+col[1:]}, inplace=True)
```

```
names_ids = df.index.str.split('\s\(') # split the index by '('

df.index = names_ids.str[0] # the [0] element is the country name (new index)
df['ID'] = names_ids.str[1].str[:3] # the [1] element is the abbreviation or ID (take firs

df = df.drop('Totals')
df.head()
```

Out[29]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gold.2
Afghanistan	13	0	0	2	2	0	0	0	0	0	13	0
Algeria	12	5	2	8	15	3	0	0	0	0	15	5
Argentina	23	18	24	28	70	18	0	0	0	0	41	18
Armenia	5	1	2	9	12	6	0	0	0	0	11	1
Australasia	2	3	4	5	12	0	0	0	0	0	2	3

23 - Escreva uma função que crie uma série chamada "Points", ela deve conter o valor ponderado onde cada medalha de ouro conta 3 pontos, medalhas de prata conta 2 pontos e medalhas de bronze conta 1 ponto. A função deve retornar apenas um objeto Série, com os nomes dos países como índices. Esta função deve retornar uma série de comprimento 146.

In [30]:

```
def points():

    return (3*df['Gold.2'] + 2*df['Silver.2'] + df['Bronze.2']).sort_values(ascending=Fals

points().loc['Brazil']
```

Out[30]:

```
184
```

24 - Carregando dataframe do censo dos EUA

In [32]:

```
census_df = pd.read_csv('./Data/census.csv')
census_df.head()
```

Out[32]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESBASE2(
0	40	3	6	1	0	Alabama	Alabama	4779736	4780
1	50	3	6	1	1	Alabama	Autauga County	54571	545
2	50	3	6	1	3	Alabama	Baldwin County	182265	1822
3	50	3	6	1	5	Alabama	Barbour County	27457	274
4	50	3	6	1	7	Alabama	Bibb County	22915	229

5 rows × 100 columns

25 - Podemos verificar que os Estados Unidos são divididos em quatro regiões. Criar uma consulta que encontre os municípios pertencentes às regiões 1 ou 2, cujo nome começa com 'Washington' e cujo população estimada foi maior em 2015 que em 2014. Essa função deve retornar um DataFrame 5x2 com as colunas = ['STNAME', 'CTYNAME'] e o mesmo índice que o dataframe original (classificado em ordem crescente por índice).

In [33]:

```
def function():
    return census_df.loc[(census_df['SUMLEV'] == 50) & (census_df['REGION'] <= 2) & (censu

function()
```

Out[33]:

	STNAME	CTYNAME
896	Iowa	Washington County
1419	Minnesota	Washington County
2345	Pennsylvania	Washington County
2355	Rhode Island	Washington County
3163	Wisconsin	Washington County

26 - Qual estado tem mais municípios (condados)?

In [34]:

```
def max_counties():
    return census_df['STNAME'].value_counts().idxmax()
    #return census_df[census_df['SUMLEV'] == 50].groupby(['STNAME']).size().idxmax()
max_counties()
```

Out[34]:

```
'Texas'
```

In [ ]: