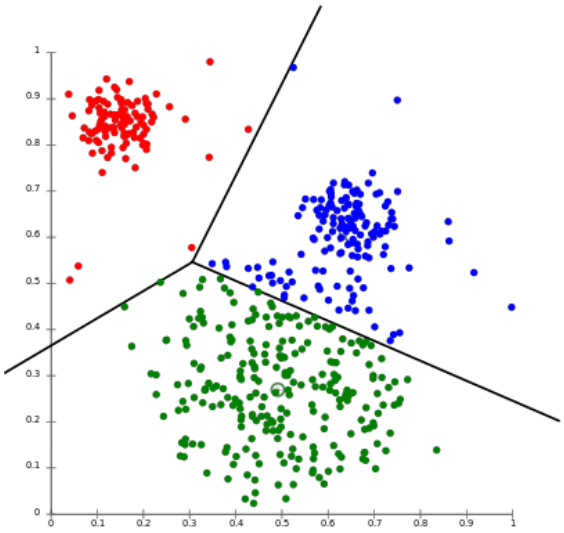
Projeto 3: Algoritmos de Machine Learning

Aluno: Romulo Barros de Freitas
Matrícula: 521353

Agrupamento (Clustering)

K- Means

É um método de aprendizado não supervisionado que particiona objetos de dados em k-grupos onde cada observação pertence ao grupo mais próximo da média (centróides).



Objetiva-se, com a aplicação do algoritimo, agrupar os municipios do estado do Ceará, levando em consideração as seguintes variáveis: Densidade Demográfica, Taxa de Escolarização, Índice de Desenvolvimento Humano Municipal (IDHM), Taxa de Mortalidade Infantil, Receitas Realizadas, Despesas Empenhadas e Produto Interno Bruto (PIB).

Sumário

In [14]:
```
# Bibliotecas que serão utilizadas no trabalho

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [15]:
```
from sklearn.metrics import davies_bouldin_score
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

In [16]:
```
# Dataframe utilizado - todos os municípios do estado do ceará
municipios = pd.read_csv('DF_Cidades.csv')
```

In [17]:
```
municipios.head()
```

Out[17]:

	codigo	cidade	prefeito	gentilico	area_territorial	populacao_estimada	densidade_d
0	2300101	Abaiara	AFONSO TAVARES LEITE	abaiarense	180.833	11965	
1	2300150	Acarape	FRANCISCO EDILBERTO BESERRA BARROSO	acarapense	130.002	15140	
2	2300200	Acaraú	ANA FLÁVIA RIBEIRO MONTEIRO	acarauense	842.471	63556	
3	2300309	Acopiara	ANTÔNIO ALMEIDA NETO	acopiarense	2254.279	54687	
4	2300408	Aiuaba	RAMILSON ARAUJO MORAES	aiuabense	2438.563	17584	

In [18]:
```
# Selecionando as colunas que serão utilizadas no agrupamento
features = ['densidade_demografica', 'escolarizacao', 'idhm', 'mortalidade_inf
```

In [19]:
```
# Excluindo os valores nulos das variáveis selecionadas
municipios = municipios.dropna(subset=features)
```

In [20]:
```
# criando o dataframe com as colunas que serão utilizadas para fazer o agrupam
df = municipios[features].copy()
```

In [21]:
```
df
```

Out[21]:

	densidade_demografica	escolarizacao	idhm	mortalidade_infantil	receitas_realizadas	despe
0	58.69	96.7	0.628	19.87	29043.96	
1	98.52	96.8	0.606	5.35	38762.54	
2	68.31	96.8	0.601	5.92	132656.39	
3	22.58	97.2	0.595	20.18	153148.71	
4	6.66	97.5	0.569	15.08	37794.63	
...	...	...	...	...	...	
177	203.61	95.9	0.639	14.16	46252.58	
178	18.49	97.7	0.566	18.52	36175.66	
179	98.07	96.6	0.611	7.41	51275.61	
180	45.99	97.0	0.629	10.68	79368.16	
181	41.90	96.9	0.571	8.14	126024.79	

169 rows × 7 columns

Etapas

1. Dimensionar os dados
2. Iniciar centróides aleatórios
3. Rotular cada ponto com base na distância para o centróide
4. Atualizar os centróides
5. Repetir etapas 3 e 4 até que os centróides se estabilizem

1. Dimensionar os Dados

In [22]:
```
# Dimensionando os valores do novo dataframe
df = (df - df.min()) / (df.max() - df.min()) * 9 + 1
```

In [23]:
```
df.describe()
```

Out[23]:

	densidade_demografica	escolarizacao	idhm	mortalidade_infantil	receitas_realizadas
count	169.000000	169.000000	169.000000	169.000000	169.000000
mean	1.131079	7.135672	4.184566	4.073438	1.169573
std	0.716866	1.445865	1.356066	1.997359	0.697428
min	1.000000	1.000000	1.000000	1.000000	1.000000
25%	1.018128	6.400000	3.397196	2.659000	1.053145
50%	1.035284	7.300000	3.985981	3.550000	1.072529
75%	1.071111	8.071429	4.869159	5.200000	1.132268
max	10.000000	10.000000	10.000000	10.000000	10.000000

In [24]:
```
df.head()
```

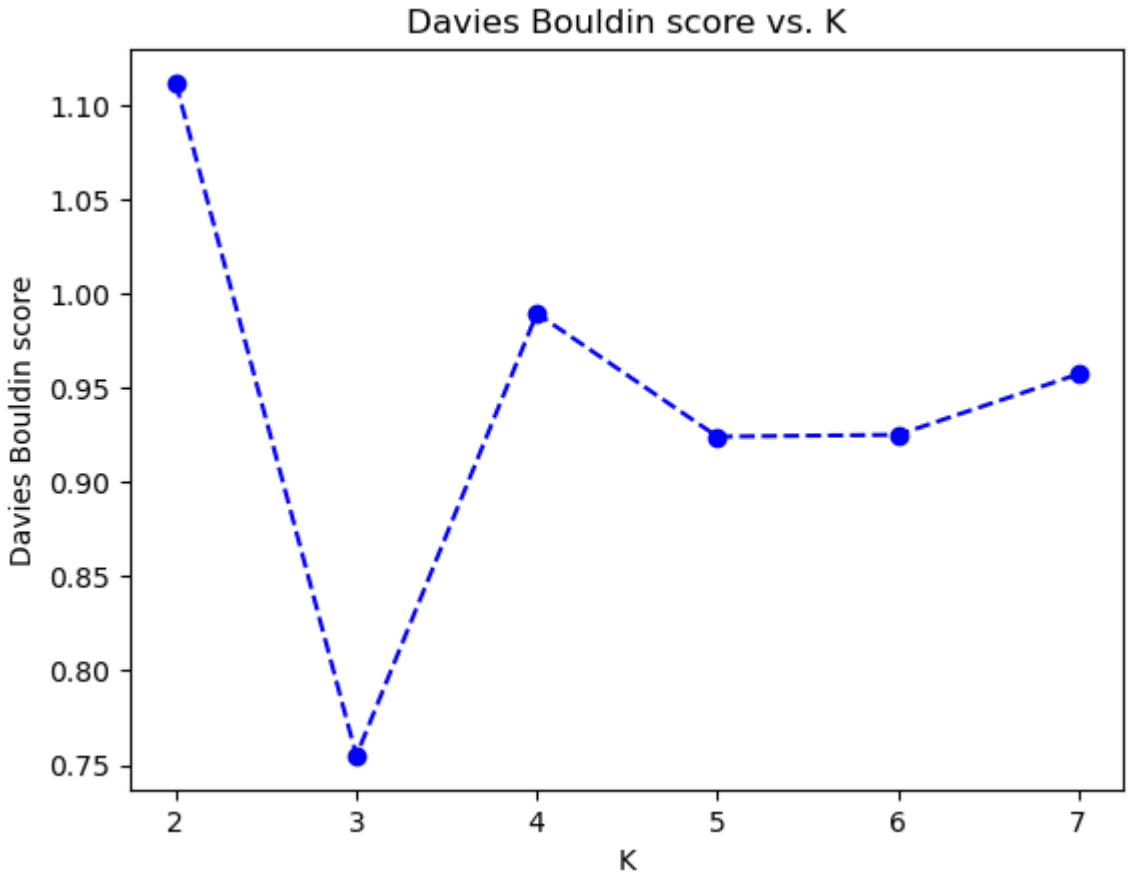
Out[24]:

	densidade_demografica	escolarizacao	idhm	mortalidade_infantil	receitas_realizadas	desp
0	1.060191	6.400000	4.700935	6.178	1.038366	
1	1.106268	6.528571	3.775701	1.822	1.051209	
2	1.071319	6.528571	3.565421	1.993	1.175288	
3	1.018417	7.042857	3.313084	6.271	1.202369	
4	1.000000	7.428571	2.219626	4.741	1.049930	

Plotando Davies-Bouldin score

In [25]:
```
# Definindo o número total de clusters para fazer o agrupamento

def get_kmeans_score(data, center):

    kmeans = KMeans(n_clusters=center, init='k-means++')
    model = kmeans.fit_predict(df)

    # Calculate Davies Bouldin score
    score = davies_bouldin_score(df, model)

    return score
```

In [26]:
```
scores = []
centers = list(range(2,8))
for center in centers:
    scores.append(get_kmeans_score(df, center))

plt.plot(centers, scores, linestyle='--', marker='o', color='b');
plt.xlabel('K');
plt.ylabel('Davies Bouldin score');
plt.title('Davies Bouldin score vs. K');
```



De acordo com o Índice Davies-Bouldin score, devemos utilizar 3 clusters para a realização do agrupamento

2. Iniciar centróides aleatórios

In [27]:
```
def random_centroids(data, k_clusters):
    centroids = []
    for i in range(k_clusters):
        centroid = df.apply(lambda x: float(x.sample()))
        centroids.append(centroid)
    return pd.concat(centroids, axis = 1)
```

In [28]:
```
centroids = random_centroids(df, 3)
```

In [29]:
```
centroids
```

Out[29]:

	0	1	2
densidade_demografica	1.034671	1.091032	1.035284
escolarizacao	5.757143	8.971429	9.100000
idhm	2.387850	3.901869	1.925234
mortalidade_infantil	7.321000	3.037000	1.831000
receitas_realizadas	1.039834	1.053776	1.076127
despesas_empenhadas	1.020505	1.016623	1.070277
pib	2.507142	1.000000	1.383213

3. Rotular cada ponto com base na distância para o centróide

In [30]:
```
def get_labels(data, centroids):
    distances = centroids.apply(lambda x: np.sqrt(((df - x) ** 2).sum(axis = 1
    return distances.idxmin(axis = 1)
```

In [31]:
```
labels = get_labels(df, centroids)
```

In [32]:
```
labels
```

Out[32]:
```
0      0
1      1
2      1
3      0
4      1
      ..
177    0
178    0
179    1
180    1
181    2
Length: 169, dtype: int64
```

In [33]:
```
labels.value_counts()
```

Out[33]:
```
1    113
0     41
2     15
dtype: int64
```

De acordo com a nossa clusterização, o cluster 0 englobou 36 municípios, o cluster 1 englobou 111 municípios e o cluster 2 englobou 22 municípios.

4. Atualizar os centróides

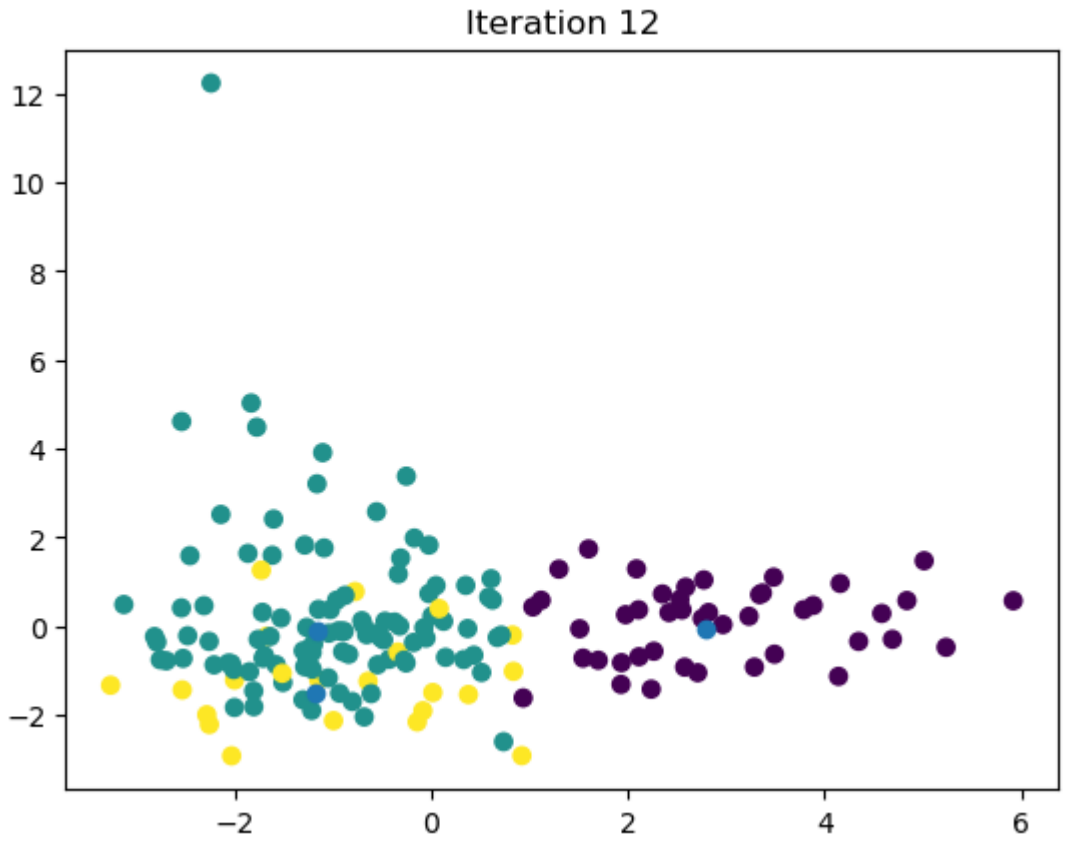
In [34]:
```
def new_centroids(data, labels, k_clusters):
    return df.groupby(labels).apply(lambda x: np.exp(np.log(x).mean())).T
```

5. Repetir etapas 3 e 4 até que os centróides se estabilizem

In [35]:
```
# Análise de componenentes principais
from sklearn.decomposition import PCA

# Plotagem de gráficos
import matplotlib.pyplot as plt

# Limpar a saída do jupyter notebook cada vez que criarmos um novo gráfico
from IPython.display import clear_output
```

In [36]:
```
def plot_clusters(data, labels, centroids, iteration):
    pca = PCA(n_components = 2) # dados 2D
    df_2d = pca.fit_transform(df)
    centroids_2d = pca.transform(centroids.T)
    clear_output(wait=True)
    plt.title(f'Iteration {iteration}')
    plt.scatter(x=df_2d[:,0], y=df_2d[:,1], c=labels)
    plt.scatter(x=centroids_2d[:,0], y=centroids_2d[:,1])
    plt.show()
```

In [37]:
```
max_iterations = 100
k_clusters = 3

centrois = random_centroids(df, k_clusters)
old_centroids = pd.DataFrame()
iteration = 1

while iteration < max_iterations and not centroids.equals(old_centroids):
    old_centroids = centroids

    labels = get_labels(df, centroids)
    centroids = new_centroids(df, labels, k_clusters)
    plot_clusters(df, labels, centroids, iteration)
    iteration += 1
```



In [38]:
```
centroids
```

Out[38]:

	0	1	2
densidade_demografica	1.039033	1.100527	1.081118
escolarizacao	7.408323	7.460768	4.255031
idhm	3.860702	4.166191	3.381181
mortalidade_infantil	6.816817	2.844471	3.116128
receitas_realizadas	1.069674	1.156746	1.090823
despesas_empenhadas	1.045286	1.126637	1.062935
pib	1.414555	1.699432	1.315480

In [39]:
```
# Municípios pertencentes ao cluster 0
municipios[labels == 0]
```

Out[39]:

	codigo	cidade	prefeito	gentilico	area_territorial	populacao_estimad
0	2300101	Abaiara	AFONSO TAVARES LEITE	abaiarense	180.833	119
3	2300309	Acopiara	ANTÔNIO ALMEIDA NETO	acopiarense	2254.279	546
4	2300408	Aiuaba	RAMILSON ARAUJO MORAES	aiuabense	2438.563	175
17	2301505	Arneiroz	ANTÔNIO MONTEIRO PEDROSA FILHO	arneirozense	1068.437	784
23	2301950	Barreira	ANTONIO ALAILSON OLIVEIRA SALDANHA MICHELE	barreirense	260.003	2271

In [40]:
```
# Municípios pertencentes ao cluster 1
municipios[labels == 1]
```

Out[40]:

	codigo	cidade	prefeito	gentilico	area_territorial	populacao_estimada	densid
1	2300150	Acarape	FRANCISCO EDILBERTO BESERRA BARROSO	acarapense	130.002	15140	
2	2300200	Acaraú	ANA FLÁVIA RIBEIRO MONTEIRO	acarauense	842.471	63556	
7	2300705	Alto Santo	JOSE JOENI HOLANDA DE ARAUJO	alto-santense	1147.208	16077	
8	2300754	Amontada	FLAVIO CESAR BRUNO TEIXEIRA FILHO	amontadense	1175.044	44195	
11	2301000	Aquiraz	BRUNO BARROS	aquirazense	480.236	81581	

In [41]:
```
# Municípios pertencentes ao cluster 2
municipios[labels == 2]
```

Out[41]:

	codigo	cidade	prefeito	gentilico	area_territorial	populacao_estimada	d
6	2300606	Altaneira	FRANCISCO DARIOMAR RODRIGUES SOARES	altaneirense	72.675	7712	
9	2300804	Antonina do Norte	ANTONIO ROSENO FILHO	antonino ou antoninense	259.706	7402	
15	2301307	Araripe	CICERO FERREIRA DA SILVA	araripense	1097.339	21707	
26	2302107	Baturité	HERBERLH FREITAS REIS CAVALCANTE MOTA	baturiteense	314.075	36127	
30	2302503	Brejo Santo	MARIA GISLAINE SANTANA SAMPAIO LANDIM	brejo-santense	654.658	50195	
34	2302909	Capistrano	ANTONIO SOARES SARAIVA JUNIOR	capistranense	226.549	17830	
36	2303204	Caririaçu	JOSÉ EDMILSON LEITE BARBOSA	caririaçuense	634.179	27008	
41	2303600	Catarina	THIAGO PAES DE ANDRADE RODRIGUES	catarinense	488.153	21041	
61	2304608	General Sampaio	FRANCISCO CORDEIRO MOREIRA	sampaiense	230.371	7767	
62	2304707	Granja	JULIANA FROTA LOPES DE ALDIGUERI ARRUDA	granjense	2663.174	55170	
66	2304954	Guaiúba	IZABELLA MARIA FERNANDES DA SILVA	guaiubano	256.053	26508	
71	2305266	Ibaretama	ELIRIA MARIA FREITAS DE QUEIROZ	ibaretamense	879.255	13385	

	codigo	cidade	prefeito	gentilico	area_territorial	populacao_estimada	d
82	2306108	Irauçuba	PATRÍCIA MARIA SANTOS BARRETO	irauçubense	1466.412	24450	
86	2306504	Itapiúna	FRANCISCO DÁRIO DE OLIVEIRA COELHO	itapiunense	593.231	20653	
102	2307700	Maranguape	ATILA CORDEIRO CAMARA	maranguapense	583.505	131677	
112	2308500	Mombaça	ORLANDO BENEVIDES CAVALCANTE FILHO	mombacense	2115.748	43917	
125	2309706	Pacatuba	CARLOMANO GOMES MARQUES	pacatubano	133.236	85647	
132	2310308	Parambu	ROMULO MATEUS NORONHA	parambuense	2313.868	31391	
143	2311207	Potengi	FRANCISCO EDSON VERIATO DA SILVA	potengiense	343.264	11165	
173	2313559	Tururu	FRANCISCA HILZETE MALVEIRA BATISTA	tururuense	201.270	16588	
176	2313757	Umirim	FELIPE CARLOS UCHÔA SALES RIBEIRO	umiriense	315.648	19976	
177	2313807	Uruburetama	FRANCISCO ALDIR CHAVES DA SILVA	uruburetamense	99.400	22223	

fazendo diretamente por meio do sklearn

In [42]:
```
from sklearn.cluster import KMeans
```

In [43]:
```
kmeans = KMeans(3)
kmeans.fit(df)
```

Out[43]:
```
KMeans
KMeans(n_clusters=3)
```

In [45]:
```
centroids = kmeans.cluster_centers_
```

In [46]:
```
pd.DataFrame(kmeans.cluster_centers_, columns=features).T
```

Out[46]:

	0	1	2
densidade_demografica	1.040659	1.092903	10.000000
escolarizacao	7.275380	7.093861	5.628571
idhm	3.940346	4.231366	10.000000
mortalidade_infantil	6.812851	3.011438	3.823000
receitas_realizadas	1.070676	1.135009	10.000000
despesas_empenhadas	1.045923	1.104510	10.000000
pib	1.438642	1.779976	3.058255

Tópico Extra

In [47]:
```
new_df = np.array(df)
```

In [48]:
```
new_df
```

Out[48]:
```
array([[1.06019065, 6.4       , 4.70093458, ..., 1.03836597, 1.01385969,
        1.28869597],
       [1.10626779, 6.52857143, 3.77570093, ..., 1.05120894, 1.02764908,
        1.42460578],
       [1.0713195 , 6.52857143, 3.56542056, ..., 1.17528833, 1.12872518,
        1.76975432],
       ...,
       [1.10574721, 6.27142857, 3.98598131, ..., 1.06774478, 1.04287885,
        2.48816808],
       [1.04549872, 6.78571429, 4.74299065, ..., 1.10486869, 1.0753531 ,
        1.45885079],
       [1.04076722, 6.65714286, 2.30373832, ..., 1.16652477, 1.12082808,
        1.18376655]])
```

In [49]:
```
kmeans = KMeans(n_clusters = 3, random_state = 0)
```

In [50]:
```
kmeans.fit(new_df)
```

Out[50]:
```
KMeans(n_clusters=3, random_state=0)
```

In [51]:
```
kmeans.labels_
```

Out[51]:
```
array([2, 1, 1, 2, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 2, 1, 1, 1, 0, 2, 1,
       1, 1, 2, 2, 1, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 2, 1, 2, 0, 2, 0, 2, 2, 1, 1, 2, 2, 2, 1, 1, 1, 0, 1,
       1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 0,
       0, 1, 1, 0, 2, 0, 1, 2, 1, 2, 2, 1, 2, 2, 1, 1, 1, 2, 1, 1, 2, 1,
       1, 2, 1, 1, 1, 2, 0, 0, 2, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1,
       1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 0, 1, 1, 1, 2, 1, 1, 1, 0, 1, 1, 0,
       1, 2, 1, 1, 2, 0, 1, 1, 2, 1, 1, 2, 1, 1, 1])
```

In [52]:
```
df['k_classes'] = kmeans.labels_
```

In [53]:
```
df
```

Out[53]:

	densidade_demografica	escolarizacao	idhm	mortalidade_infantil	receitas_realizadas	de
0	1.060191	6.400000	4.700935	6.178	1.038366	
1	1.106268	6.528571	3.775701	1.822	1.051209	
2	1.071319	6.528571	3.565421	1.993	1.175288	
3	1.018417	7.042857	3.313084	6.271	1.202369	
4	1.000000	7.428571	2.219626	4.741	1.049930	
...	...	...	...	...	...	
177	1.227841	5.371429	5.163551	4.465	1.061107	
178	1.013685	7.685714	2.093458	5.773	1.047790	
179	1.105747	6.271429	3.985981	2.440	1.067745	
180	1.045499	6.785714	4.742991	3.421	1.104869	
181	1.040767	6.657143	2.303738	2.659	1.166525	

169 rows × 8 columns

In [54]:
```
sb.pairplot(df, hue = 'k_classes', palette = 'muted')
```

Out[54]:
```
<seaborn.axisgrid.PairGrid at 0x1ffe40798d0>
```