IESB

Pós Graduação em Inteligência Artificial

Disciplina: Aprendizado não Supervisionado

Docente: Mateus Mendelson

Discente: Henrique Brandão

Atividades com dados `base-covid-19-us.csv`

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib notebook
```

In [2]:
```
df = pd.read_csv('base-covid-19-us.csv')
df.shape
```

```
(1570, 3)
```

In [3]:
```
df.head()
```

	county	cases	deaths
0	Abbeville	84	0
1	Acadia	741	21
2	Accomack	116	0
3	Ada	4264	41
4	Adair	325	8

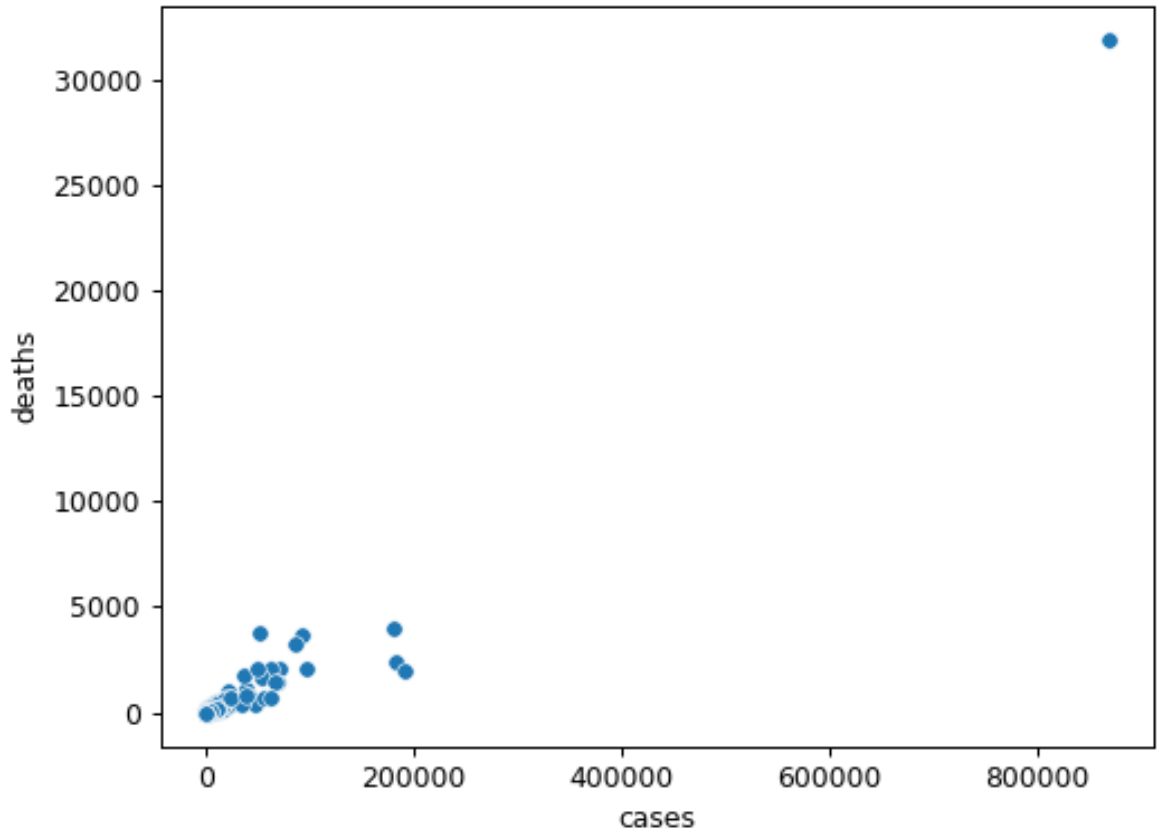
In [4]:
```
df.describe()
```

	cases	deaths
count	1570.000000	1570.000000
mean	2633.903822	69.952229
std	24352.089822	842.722792
min	1.000000	0.000000
25%	34.000000	0.000000
50%	133.500000	0.000000

	cases	deaths
75%	632.750000	17.000000

In [5]:
```
pontos = df[['cases', 'deaths']].values
```

In [6]:
```
sns.scatterplot(x=df['cases'], y=df['deaths'])
plt.show()
```



In [7]:
```
from math import sqrt
from random import choice
```

In [8]:
```
def sortear_centroide_inicial(pontos):
    """Sorteia um dos pontos da amostra para ser um centroide inicial
    return choice(pontos)
```

In [9]:
```
def distancia(u, v):
    """
    Calcula a distância euclidiana entre vetores.
    Análogo a np.linalg.norm().

    :param list(float) u: vetor n-dimensional
```

```
        :param list(float) v: vetor n-dimensional
        :return distância: float
        """
        delta = []
        for i in range(len(u)):
            delta.append((u[i] - v[i])**2)
        return sqrt(sum(delta))
```

In [10]:

```
def otimo(antigos, novos, parada: float):
    delta = []
    for i in range(len(novos)):
        delta.append(distancia(novos[i], antigos[i]))
    if sum(delta) <= parada:
        return True
    else:
        return False
```

In [11]:

```
def fit_k_means(pontos, k: int, parada: float, max_iter: int):
    print(f'Treinando com {pontos.shape[0]} registros')
    clusters = {}
    centroides = []
    for i in range(k):
        # 2 - sortear centroides iniciais
        centroides.append(sortear_centroide_inicial(pontos))
    for i in range(max_iter):
        print(f'\n# Iteração no. {i} --------------------------------')
        print(f'Centroides:\n{centroides}')
        # 1 - definir a quantidade de clusters/zerar os clusters
        for c in range(k):
            clusters[f'c_{c}'] = []
        # 3 - calcular a distância de cada ponto para cada centroide
        for p in pontos:
            _dists = [distancia(p, c) for c in centroides]
            # 4 - associar cada ponto ao centroide mais próximo
            _clust = _dists.index(min(_dists))
            clusters[f'c_{_clust}'].append(p)
        for c in clusters:
            print(f"Cluster {c} tem {len(clusters[c])} elementos")
        # 5 - atualizar as coords de cada centroide
        novos_centroides = []
        for c in range(k):
            _pontos = clusters[f'c_{c}']
```

```
                novos_centroides.append(sum(_pontos)/len(_pontos))
            # 6 - verificar parada
            if otimo(centroides, novos_centroides, parada):
                break
            else:
                centroides = novos_centroides
        # 7 - Retornar os centroides
        return centroides, clusters
```

In [12]:
```
centroides, clusters = fit_k_means(pontos=pontos, k=3, parada=0.001,
```

```
Treinando com 1570 registros

# Iteração no. 0 --------------------------------------- #
Centroides:
[array([836,  11]), array([45,  0]), array([8680,  417])]
Cluster c_0 tem 359 elementos
Cluster c_1 tem 1095 elementos
Cluster c_2 tem 116 elementos

# Iteração no. 1 --------------------------------------- #
Centroides:
[array([1526.60445682,   36.63509749]), array([103.8347032 ,   2.91324201]), array([2994
Cluster c_0 tem 313 elementos
Cluster c_1 tem 1221 elementos
Cluster c_2 tem 36 elementos

# Iteração no. 2 --------------------------------------- #
Centroides:
[array([3652.4057508 ,   85.11501597]), array([154.56019656,   3.96150696]), array([7786
Cluster c_0 tem 204 elementos
Cluster c_1 tem 1348 elementos
Cluster c_2 tem 18 elementos

# Iteração no. 3                                        #
```

In [13]:
```
def _to_df(clusters: dict):
    saida = []
    for k in clusters:
        for ponto in clusters[k]:
            _ = (ponto[0], ponto[1], k)
            saida.append(_)
    return saida
```

In [14]:
```
df_kmeans = pd.DataFrame(_to_df(clusters), columns=['cases', 'deaths'
df_kmeans.shape
```

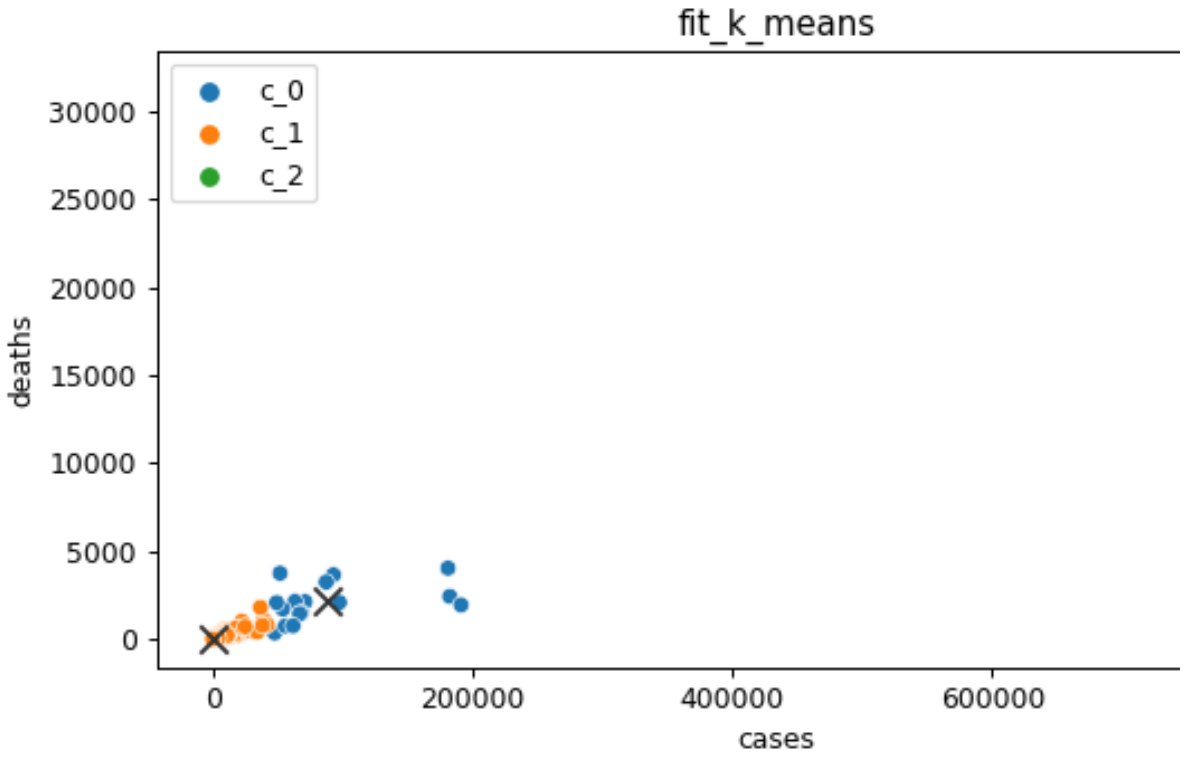
```
(1570, 3)
```

In [15]:
```
df_kmeans.head()
```

	cases	deaths	cluster
0	71121	2117	c_0
1	97593	2079	c_0
2	63344	2129	c_0
3	54023	1704	c_0
4	51873	3741	c_0

In [16]:
```
cent_xs = [c[0] for c in centroides]
cent_ys = [c[1] for c in centroides]
```

In [17]:
```
plt.figure(figsize=(8, 4), dpi=100)
sns.scatterplot(data=df_kmeans, x='cases', y='deaths', hue=df_kmeans[
sns.scatterplot(x=cent_xs, y=cent_ys, marker='x', s=100, color='.2',
plt.title('fit_k_means')
plt.show()
```




```
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/seaborn/relat
a edgecolor/edgecolors ('w') for an unfilled marker ('x').  Matplotlib is ignoring the e
his behavior may change in the future.
  points = ax.scatter(*args, **kws)
```

In [18]:
```
from sklearn.cluster import KMeans
```

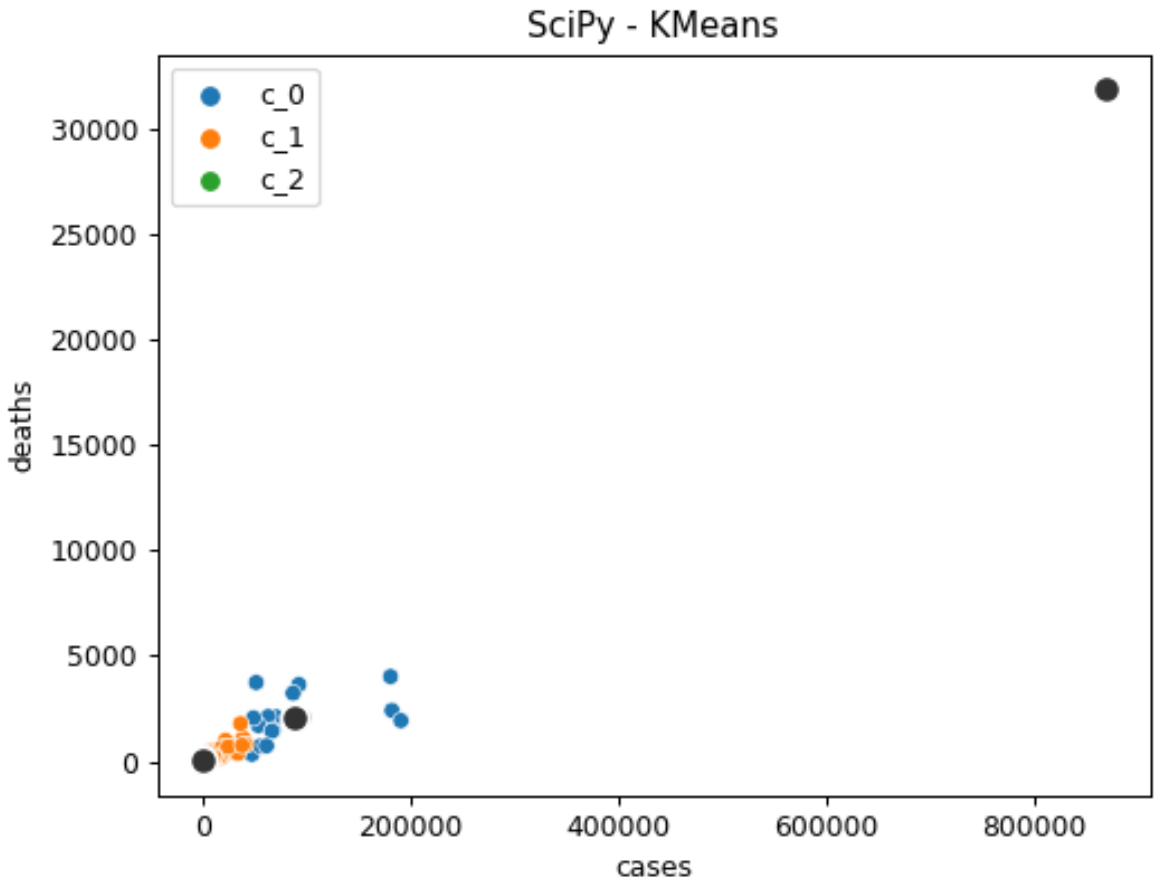
In [19]:
```
kmeans = KMeans(init='random', n_clusters=3, max_iter=300)
```

```
kmeans.fit(df[['cases','deaths']])
centroides_km = kmeans.cluster_centers_
n_iter_km = kmeans.n_iter_
```

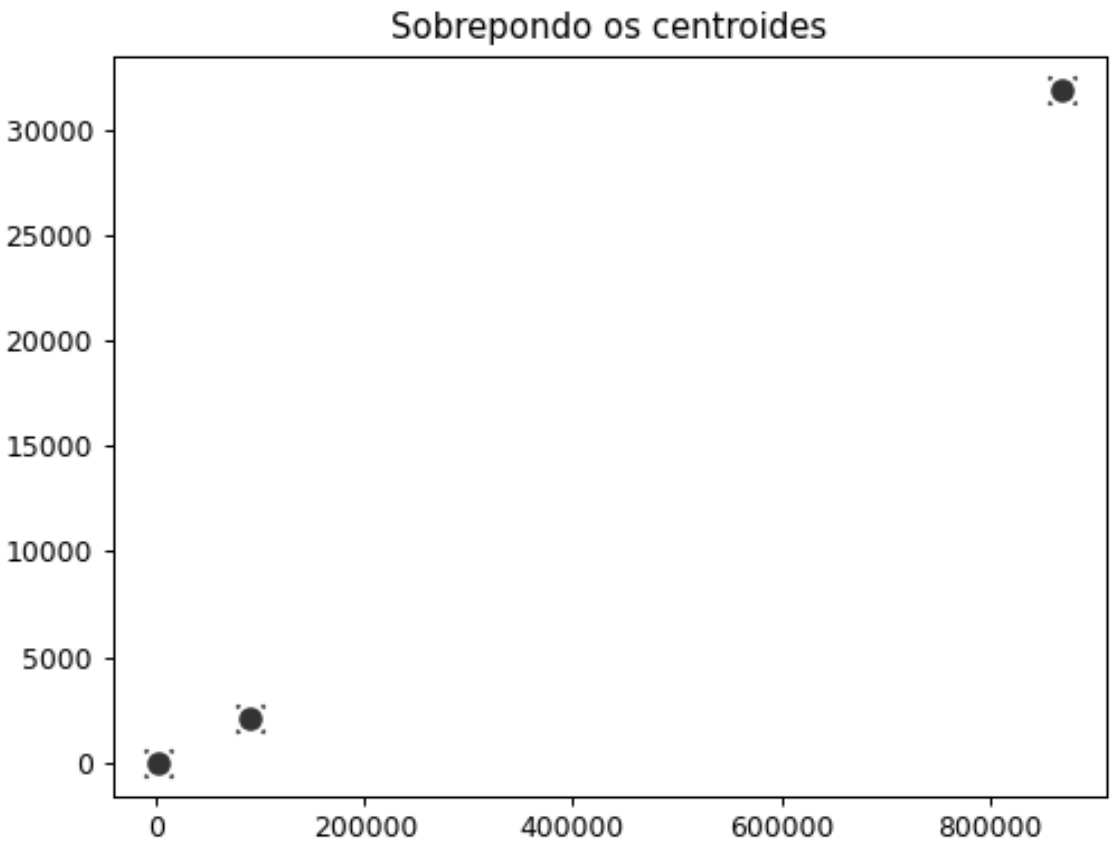
In [20]:
```
centroides_km, n_iter_km
```

```
(array([[1.18691500e+03, 2.83966516e+01],
        [8.68824000e+05, 3.18870000e+04],
        [8.89453750e+04, 2.11487500e+03]]),
 13)
```

In [21]:
```
sns.scatterplot(data=df_kmeans, x='cases', y='deaths', hue=df_kmeans[
sns.scatterplot(x=centroides_km[:, 0], y=centroides_km[:, 1], marker=
plt.title('SciPy - KMeans')
plt.show()
```



In [22]:
```
sns.scatterplot(x=cent_xs, y=cent_ys, marker='x', s=100, color='.2',
sns.scatterplot(x=centroides_km[:, 0], y=centroides_km[:, 1], marker=
plt.title('Sobrepondo os centroides')
plt.show()
```

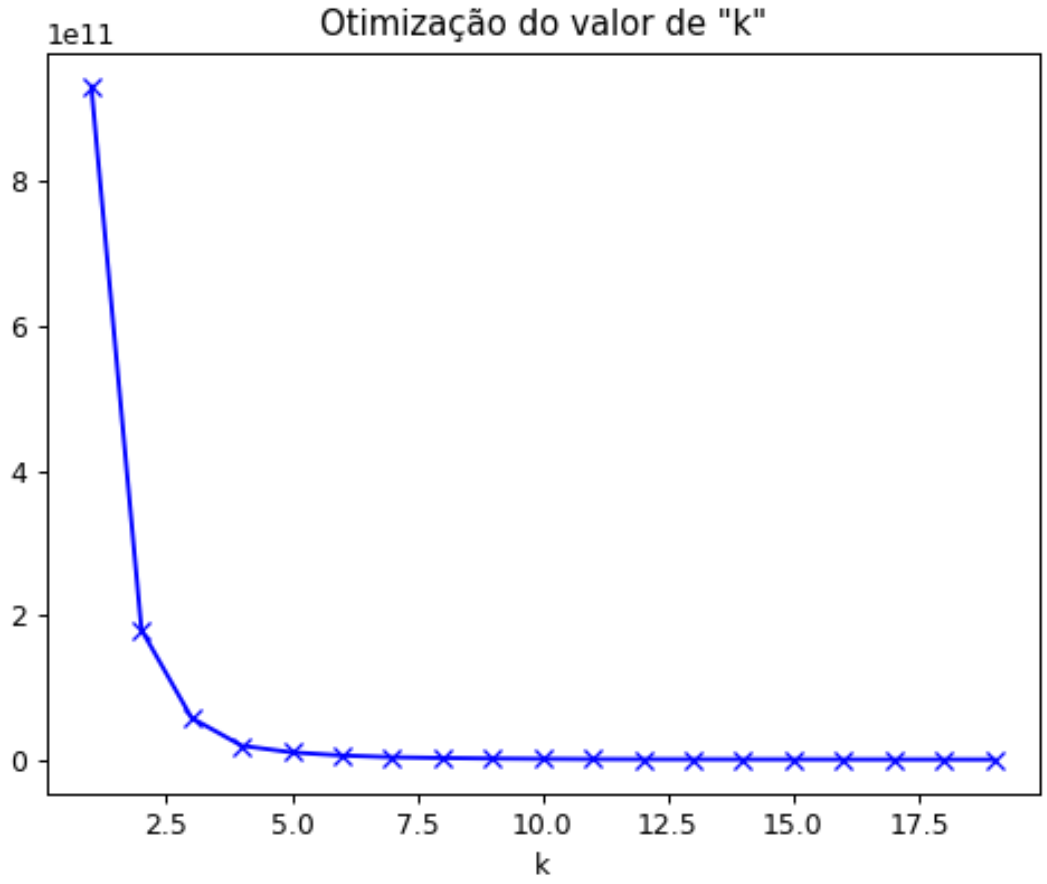



```
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/seaborn/relat
a edgecolor/edgecolors ('w') for an unfilled marker ('x').  Matplotlib is ignoring the e
his behavior may change in the future.
  points = ax.scatter(*args, **kws)
```

In [23]:
```
custo = []
qnt_k = range(1,20)
for k in qnt_k:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(df[['cases','deaths']])
    custo.append(kmeanModel.inertia_)
```

In [24]:
```
plt.plot(qnt_k, custo, 'bx-')
plt.xlabel('k')
```

```
plt.title('Otimização do valor de "k"')
plt.show()
```



PAM

In [25]:
```
def custo(u, v):
    dx = u[0] - v[0]
    dy = u[1] - v[1]
    return dx**2 + dy**2
```

In [26]:
```
def mse(medoide, pontos):
    n = len(pontos)
    if n == 0:
        return 0
    else:
        acc = []
        for p in pontos:
            dx = (medoide[0] - p[0])**2
            dy = (medoide[1] - p[1])**2
            acc.append(dx + dy)
```

```
            return sum(acc)/n
```

In [27]:

```
def diff(li1, li2):
    li_dif = [i for i in li1 + li2 if i not in li1 or i not in li2]
    return li_dif

def alternar_medoide(pontos, todos_medoides):
#     tup_pontos = [tuple(p) for p in pontos]
#     tup_meds = [tuple(med) for med in todos_medoides]
#     diff = list(set(tup_pontos) - set(tup_meds))

#     print(f'Qnt medoides usados: {len(todos_medoides)}')
#     _diff = diff(pontos, todos_medoides)
#     print(f'Pontos restantes: {len(_diff)}\n')
    return sortear_centroide_inicial(_diff)
```

In [28]:

```
def fit_pam(pontos, k, alfa):
    num_pontos = len(pontos)

    custo_total_medio = 0
    clusters = {}
    medoides = {1: {}}
    custos_totais = {1: {}}
    pontos_usados = []

    for i in range(k):
        medoides[1][i] = sortear_centroide_inicial(pontos)
        pontos_usados.append(medoides[1][i])

    itr = 1
    parada = 0
    while True:
        print(f'\n# {itr} ---------------------------------------- #
        print(f'Medoides: {list(medoides[itr].values())}')

        for c in range(k):
            clusters[c] = []

        for p in pontos:
            _custos = [custo(m, p) for m in medoides[itr].values()]
            clusters[_custos.index(min(_custos))].append(p)
        for c in clusters:
```

```
                print(f"- Cluster {c} tem {len(clusters[c])} elementos")

            custos_totais[itr] = {}
            for m in medoides[itr]:
                custos_totais[itr][m] = mse(medoides[itr][m], clusters[m]
            novo_custo_total_medio = sum(custos_totais[itr].values())/k
#             print(f'$1: {custo_total_medio:.2f}\n$2: {novo_custo_total_

            if novo_custo_total_medio < custo_total_medio:
                parada = 0
            else:
                parada += 1
                print(f'\t\t\t\tParada: {parada}')
                if itr > 1:
                    medoides[itr] = medoides[itr-1]
                if parada == alfa:
                    break
                custo_total_medio = novo_custo_total_medio

            if len(pontos_usados) == num_pontos:
                break

            itr += 1
            medoides[itr] = medoides[itr - 1]
            # sortear sem reposição
            p = sortear_centroide_inicial(pontos)
            dd = [distancia(m, p) for m in medoides[itr].values()]
            ind = dd.index(min(dd))
            medoides[itr][ind] = p
            pontos_usados.append(p)

        return medoides[itr], clusters

    medoides, clusters = fit_pam(pontos.tolist(), k=3, alfa=300)
```

```
 # 1 ---------------------------------------- #
 Medoides: [[250, 24], [23, 0], [1475, 11]]
 - Cluster 0 tem 439 elementos
 - Cluster 1 tem 795 elementos
 - Cluster 2 tem 336 elementos
                                Parada: 1

 # 2 ---------------------------------------- #
```

```
Medoides: [[250, 24], [106, 0], [1475, 11]]
- Cluster 0 tem 373 elementos
- Cluster 1 tem 861 elementos
- Cluster 2 tem 336 elementos
                                Parada: 2

# 3 ---------------------------------------- #
Medoides: [[624, 11], [106, 0], [1475, 11]]
- Cluster 0 tem 222 elementos
- Cluster 1 tem 1037 elementos
- Cluster 2 tem 311 elementos
                                Parada: 3

# 4 ---------------------------------------- #
```

In [29]:
```
df_pam = pd.DataFrame(_to_df(clusters), columns=['cases', 'deaths', '
df_pam.shape
```

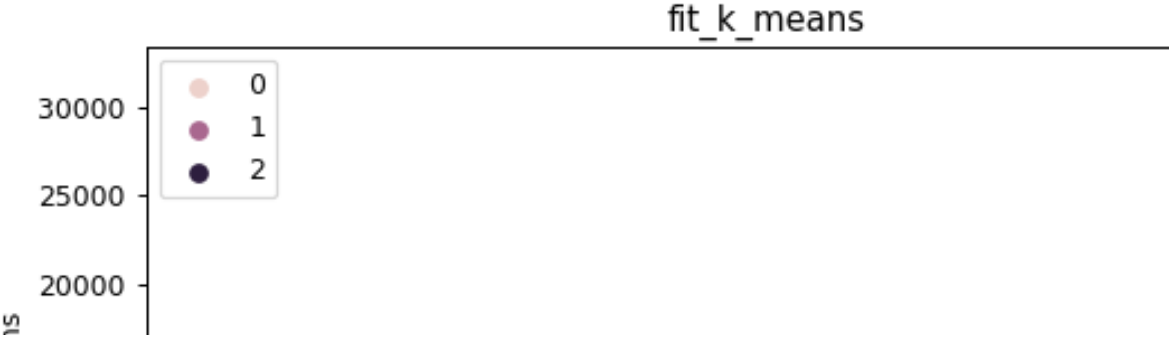
```
(1570, 3)
```

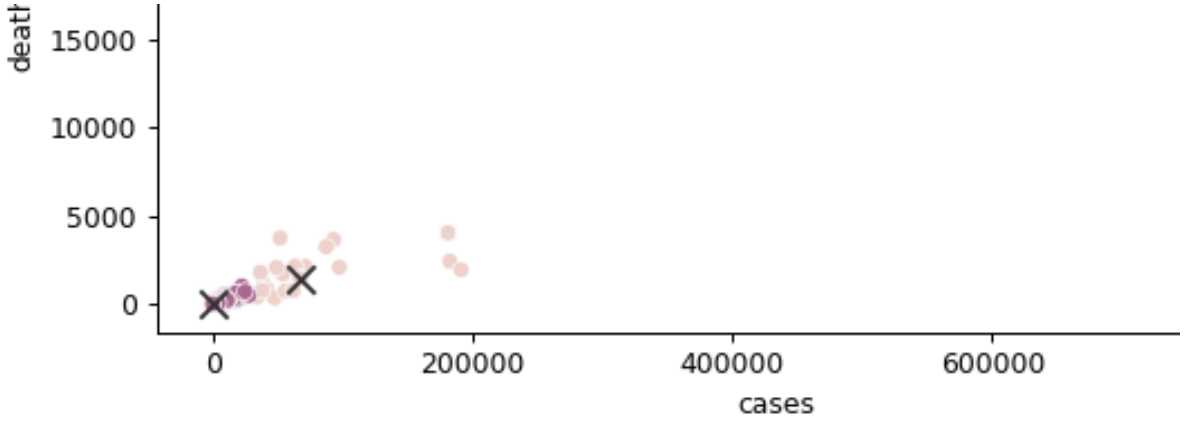
In [30]:
```
df_pam.head()
```

	cases	deaths	cluster
0	71121	2117	0
1	97593	2079	0
2	63344	2129	0
3	39302	1106	0
4	41917	800	0

In [31]:
```
medoides_xs = [c[0] for c in medoides.values()]
medoides_ys = [c[1] for c in medoides.values()]
```

In [32]:
```
plt.figure(figsize=(8, 4), dpi=100)
sns.scatterplot(data=df_pam, x='cases', y='deaths', hue=df_pam['clust
sns.scatterplot(x=medoides_xs, y=medoides_ys, marker='x', s=100, colo
plt.title('fit_k_means')
plt.show()
```






```
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/seaborn/relat
a edgecolor/edgecolors ('w') for an unfilled marker ('x').  Matplotlib is ignoring the e
his behavior may change in the future.
  points = ax.scatter(*args, **kws)
```

In [34]:
```
from sklearn_extra.cluster import KMedoids
```

In [35]:
```
kmedoid = KMedoids(n_clusters=3)
```

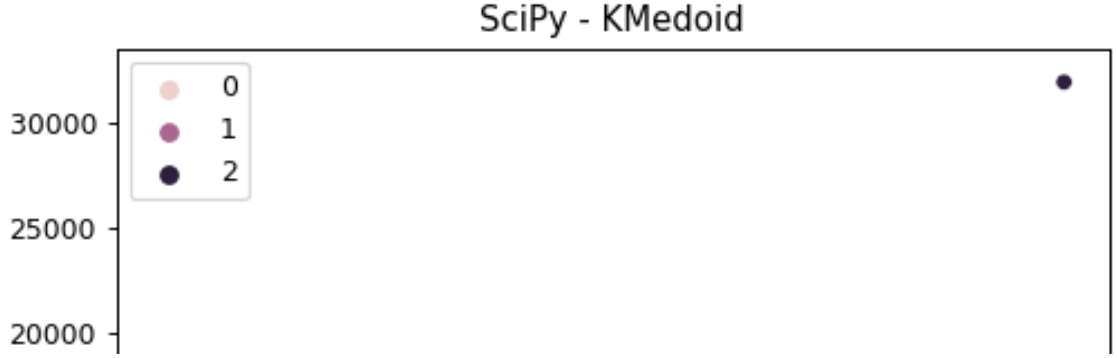
In [36]:
```
kmedoid.fit(df[['cases','deaths']])
```

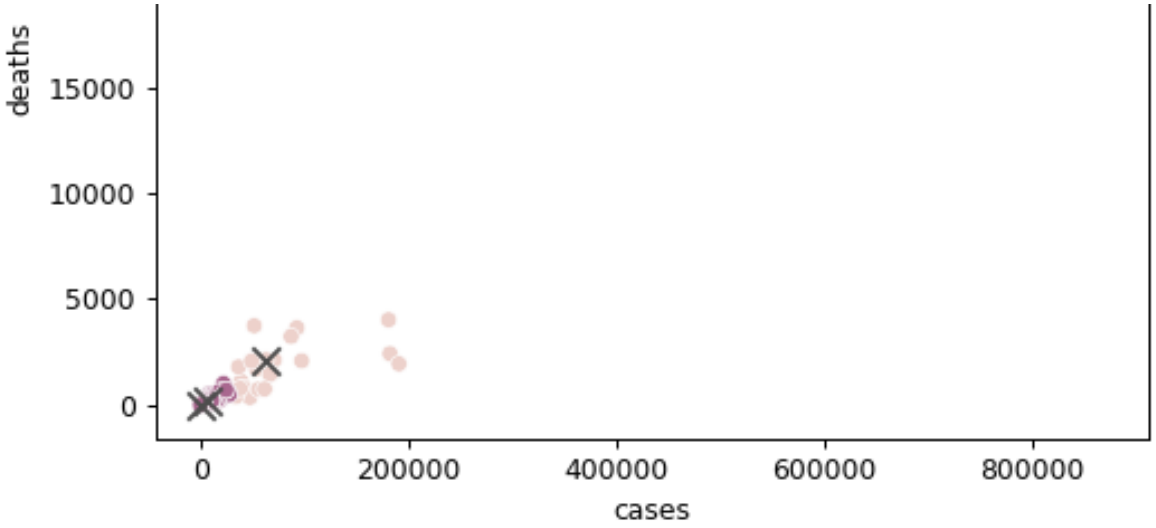
```
KMedoids(n_clusters=3)
```

In [37]:
```
centroides_kmedoid = kmedoid.cluster_centers_
centroides_kmedoid
```

```
array([[ 7294,   181],
       [  103,     3],
       [63344,  2129]])
```

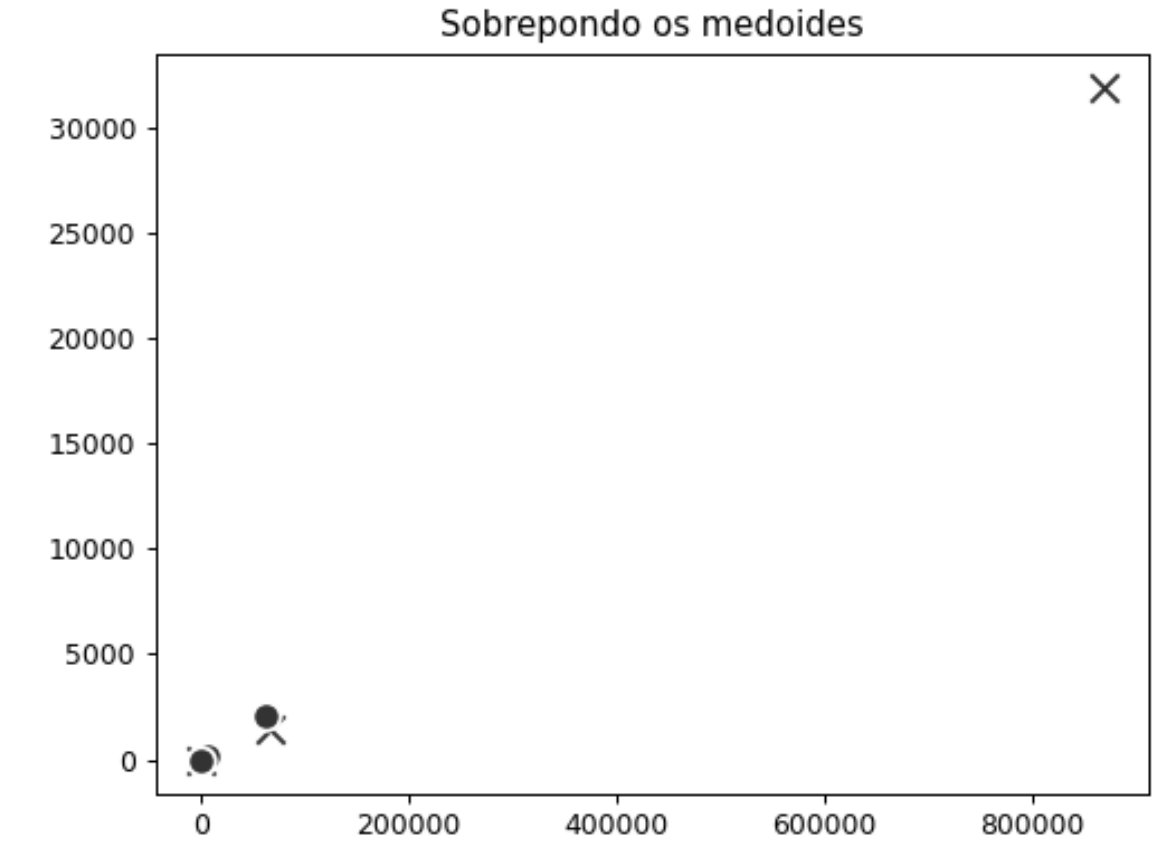
In [42]:
```
sns.scatterplot(data=df_pam, x='cases', y='deaths', hue=df_pam['clust
sns.scatterplot(x=centroides_kmedoid[:, 0], y=centroides_kmedoid[:, 1
plt.title('SciPy - KMedoid')
plt.show()
```






```
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/seaborn/relat
a edgecolor/edgecolors ('w') for an unfilled marker ('x').  Matplotlib is ignoring the e
his behavior may change in the future.
  points = ax.scatter(*args, **kws)
```

In [44]:
```
sns.scatterplot(x=medoides_xs, y=medoides_ys, marker='x', s=100, colo
sns.scatterplot(x=centroides_kmedoid[:, 0], y=centroides_kmedoid[:, 1
plt.title('Sobrepondo os medoides')
plt.show()
```



```
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/seaborn/relat
a edgecolor/edgecolors ('w') for an unfilled marker ('x').  Matplotlib is ignoring the e
his behavior may change in the future.
  points = ax.scatter(*args, **kws)
```

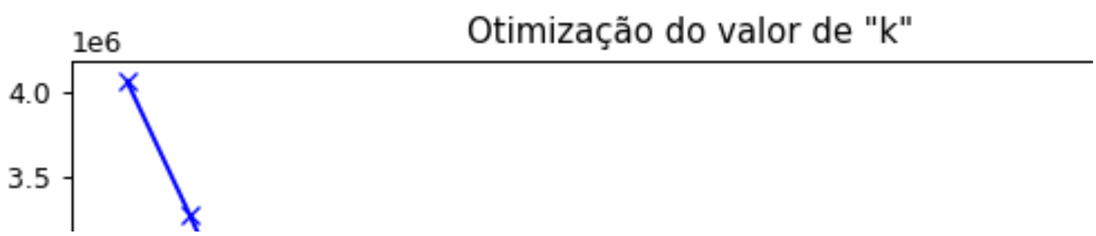
In [40]:

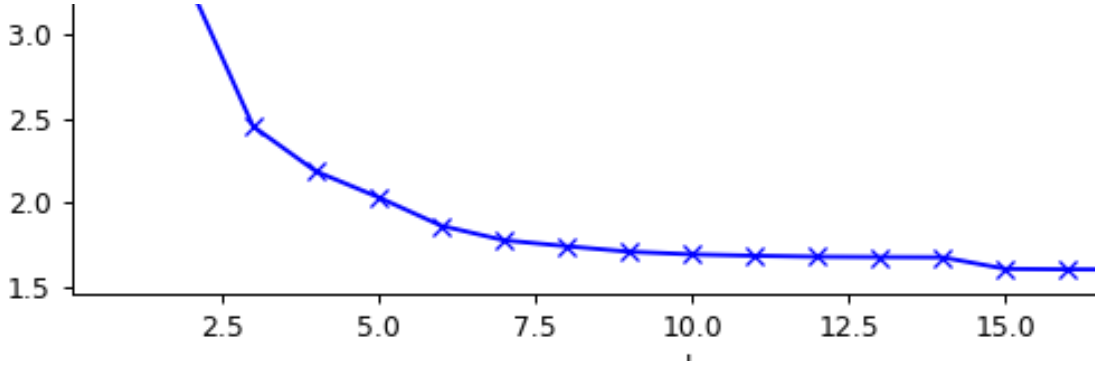
```python
custo = []
qnt_k = range(1,20)
for k in qnt_k:
    kmedoidmodel = KMedoids(n_clusters=k)
    kmedoidmodel.fit(df[['cases','deaths']])
    custo.append(kmedoidmodel.inertia_)
```

```
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/sklearn_extra
g: Cluster 4 is empty! self.labels_[self.medoid_indices_[4]] may not be labeled with its
  warnings.warn(
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/sklearn_extra
g: Cluster 4 is empty! self.labels_[self.medoid_indices_[4]] may not be labeled with its
  warnings.warn(
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/sklearn_extra
g: Cluster 4 is empty! self.labels_[self.medoid_indices_[4]] may not be labeled with its
  warnings.warn(
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/sklearn_extra
g: Cluster 6 is empty! self.labels_[self.medoid_indices_[6]] may not be labeled with its
  warnings.warn(
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/sklearn_extra
g: Cluster 4 is empty! self.labels_[self.medoid_indices_[4]] may not be labeled with its
  warnings.warn(
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/sklearn_extra
g: Cluster 6 is empty! self.labels_[self.medoid_indices_[6]] may not be labeled with its
  warnings.warn(
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/sklearn_extra
g: Cluster 4 is empty! self.labels_[self.medoid_indices_[4]] may not be labeled with its
  warnings.warn(
/home/hbrandao/Projects/post-grad-iesb-ai/venv/lib/python3.8/site-packages/sklearn_extra
g: Cluster 6 is empty! self.labels_[self.medoid_indices_[6]] may not be labeled with its
  warnings.warn(
```

In [41]:

```python
plt.figure(figsize=(8,3))
plt.plot(qnt_k, custo, 'bx-')
plt.xlabel('k')
plt.title('Otimização do valor de "k"')
plt.show()
```





DBSCAN

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: