Aula 02 | PosTech | Proporcionalidade e Seaborn

Anotações sobre a segunda aula da PosTech FIAP ✨✨

https://on.fiap.com.br/mod/conteudoshtml/view.php?id=307796&c=8729&sesskey=ZuKoJQwSR0

Temas abordados:

- Enxergar a proporcionalidade dos dados;
- Utilizar biblioteca gráfica Seaborn;
- Como visualizar minhas análises com a Seaborn.

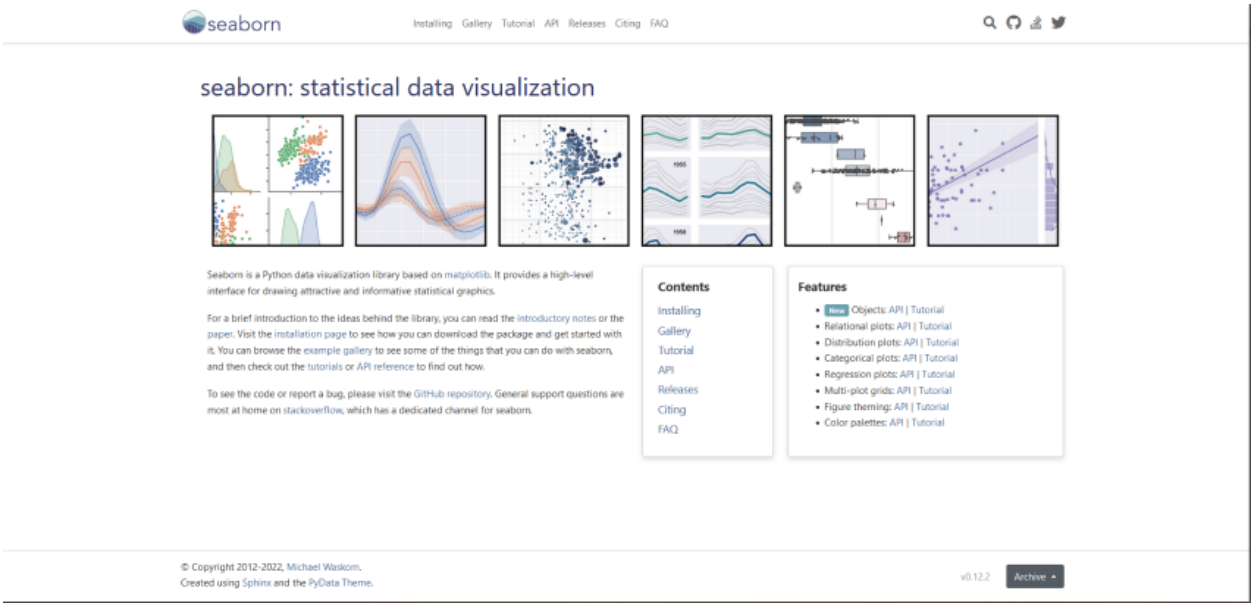
Pré-requisitos:

- Base de dados:

```
https://github.com/alura-tech/pos-datascience-introducao-a-visualizacao/archive/refs/heads/dados.zip
```

- Importar essa base de dados no Colaboratory

A biblioteca Seaborn é uma opção além da Matplotlib. É famosa por ter lindos gráficos e um leque de variedades interessantes. <3 já gostei! 😮



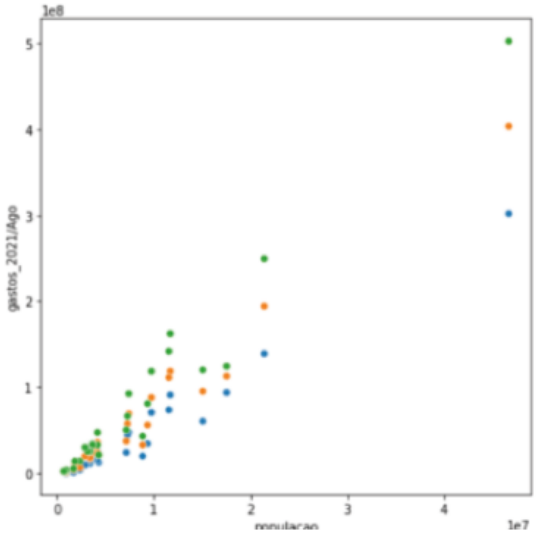
https://seaborn.pydata.org/

Seaborn é uma bilbioteca de visualização de dados baseada em Matplotlib para o Python. É projetada para fornecer uma interface de alto nível para criar gráficos estatísticos atrativos e informativos. Suporta vários tipos de gráficos, incluindo gráficos de distribuição, regressão, de violin, de barra, de área, entre outros. Além disso, a biblioteca inclui uma série de temas e paletas de cores para personalizar a aparência dos gráficos.

Torna-se mais fácil trabalhar com dados categóricos, além de fornecer recursos avançados de visualização de dados, como plotagem de séries temporais, mapas de calor e gráficos de matriz de correlação. Também permite ajustar facilmente modelos estatísticos aos seus dados, como regressão linear e regressão logística.

Uma maneira de gerar três informações com o Scatterplot do Seaborn, utilizando 3 linhas:

```
plt.figure(figsize=(7,7))
sns.scatterplot(data = gastos_e_populacao,

                    x = "populacao",

                    y ="gastos_2021/Ago")

sns.scatterplot(data = gastos_e_populacao,

                    x = "populacao",

                    y ="gastos_2021/Jul")

sns.scatterplot(data = gastos_e_populacao,

                    x = "populacao",

                    y ="gastos_2021/Jun")
```



Scatter Múltiplo.

Os dados foram diferenciados por cor sem declararmos, ou seja, a biblioteca gerou automaticamente, mas podemos mudar os parâmetros e colocar outras cores.

Em comparação com o Matplotlib, o Seaborn pode ser uma opção melhor em algumas situações:

1 | Estilo visual - O Seaborn vem com um estilo padrão que é mais atrativo e profissional do que o Matplotlib. Isso pode ser útil quando você precisa criar visualizações para apresentações ou relatórios;

2 | Plotagem avançada - O Seaborn fornece suporte a vários tipos de gráficos comuns, como gráficos de densidade, histogramas, gráficos de dispersão e muito mais. Além disso, é possível criar visualizações mais avançadas, como gráficos de correlação, gráficos de matriz e mapas de calor;

3 | Análise de dados - O Seaborn inclui recursos para análise exploratória de dados, como gráficos de distribuição e testes estatísticos. Pode ser útil para entender rapidamente a estrutura de seus dados e identificar tendências ou padrões.

Pode ser uma melhor opção do que o Matplotlib quando você precisa de visualizações mais atrativas e com recursos avançados para análise exploratória de dados. No entanto, é importante notar que o Matplotlib ainda é uma biblioteca poderosa e versátil, que pode ser usada em combinação com o Seaborn.

Ilustrações de código:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Carregar o dataset de exemplo iris

iris = sns.load_dataset("iris")
```

```
# Criar um gráfico de distribuição usando o atributo "sepal_width"

sns.distplot(iris["sepal_width"])

# Mostrar o gráfico

plt.show()
```

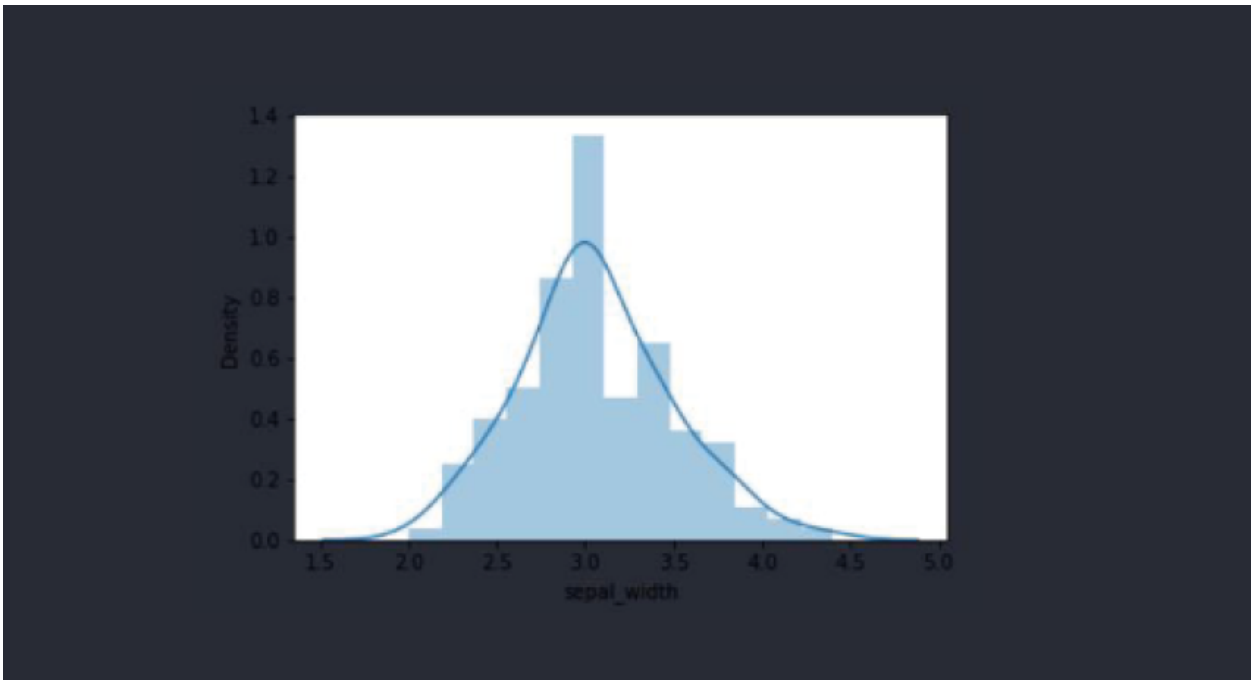
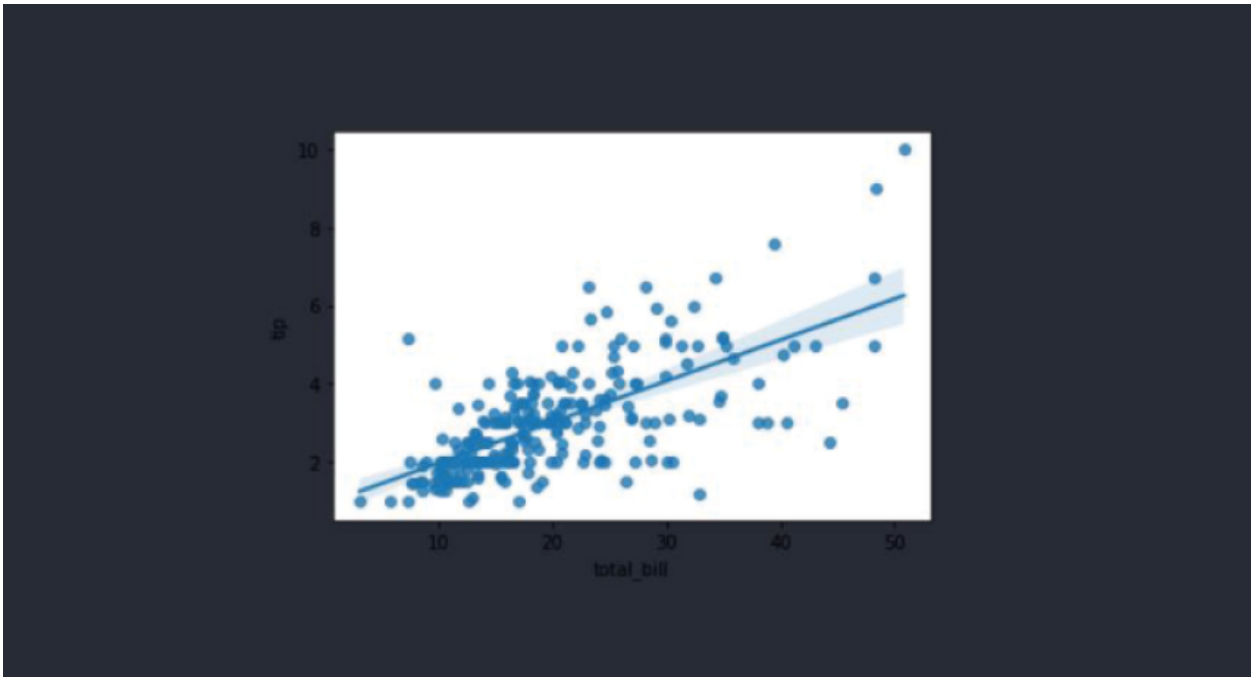


Gráfico de Dispersão.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Carregar o dataset de exemplo tips
tips = sns.load_dataset("tips")

# Criar um gráfico de regressão linear usando "total_bill" como
# variável independente e "tip" como variável dependente
sns.regplot(x="total_bill", y="tip", data=tips)

# Mostrar o gráfico
plt.show()
```



Regressão Linear.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Carregar o dataset de exemplo tips
tips = sns.load_dataset("tips")

# Criar um gráfico de barra agrupado por "smoker" e "day"
sns.barplot(x="day", y="total_bill", hue="smoker", data=tips)

# Mostrar o gráfico
plt.show()
```

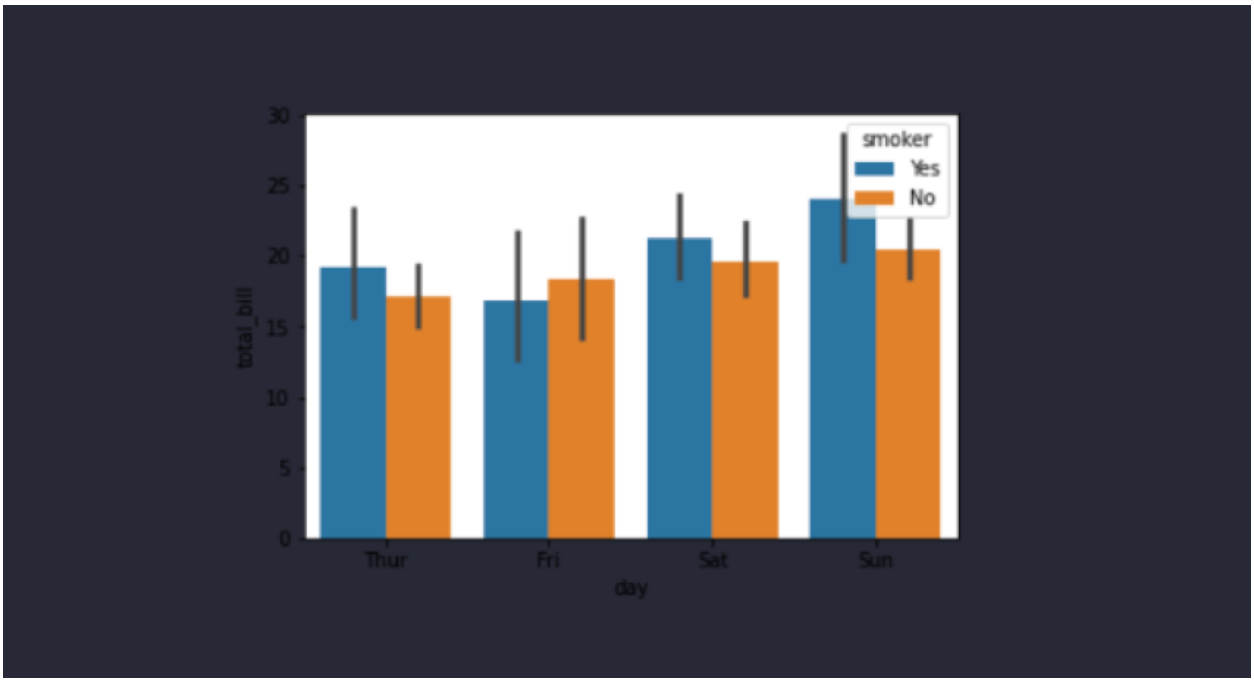


Gráfico de barra.

Parte 1 | Proporcionalidade e Seaborn I

https://seaborn.pydata.org/

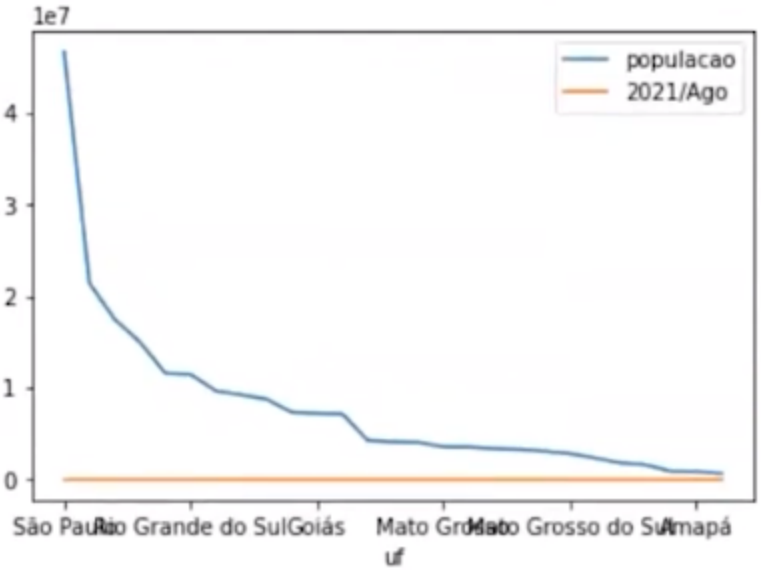
```
1 gastos_e_populacao = populacao.join(gastos_do_mais_recente)
2 gastos_e_populacao.head()
```

uf	populacao	2021/Ago
São Paulo	46649132	301.99
Minas Gerais	21411923	139.16
Rio de Janeiro	17463349	94.14
Bahia	14985284	61.65
Paraná	11597484	91.19

Agora, o que queremos fazer é finalmente calcular o gasto por população (ou algo do gênero). Primeiro, vamos **pegar o gastos_e_populacao e plotar**.

```
gastos_e_populacao.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf79bd50d0>
```



Esse é um gráfico de linhas e, como já sabemos, não serve para o que queremos.

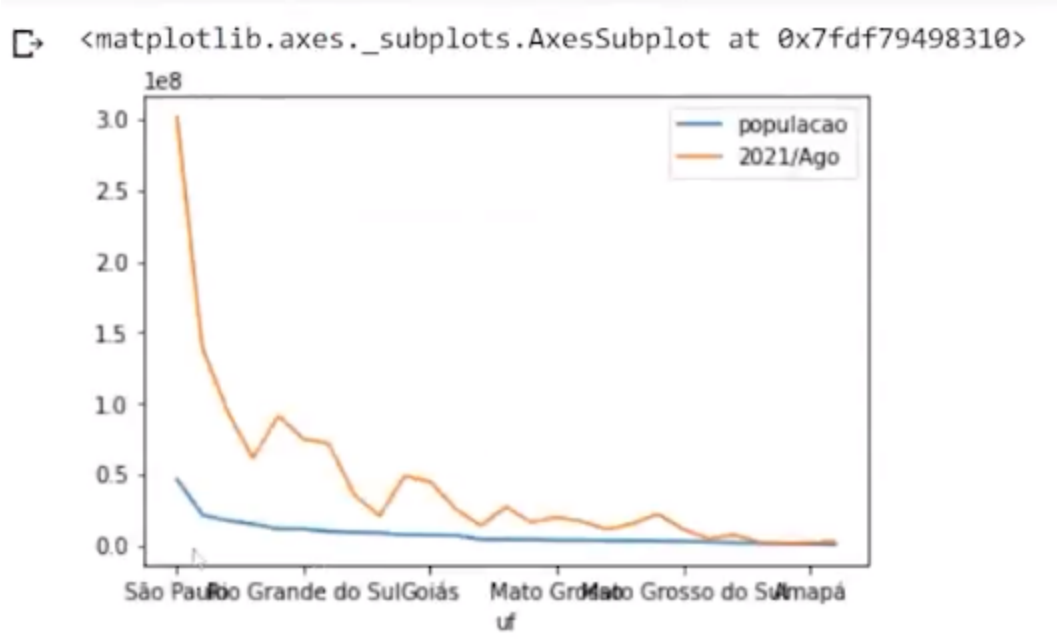
O eixo y está em milhões.

- [ ] `gastos_e_populacao[-1] = gastos_e_populacao[-1] * 1_000_000`
- [ ] `gastos_e_populacao.plot()`

Isso dá erro, porque não conseguimos acessar a coluna do eixo y por [-1].

O correto é:

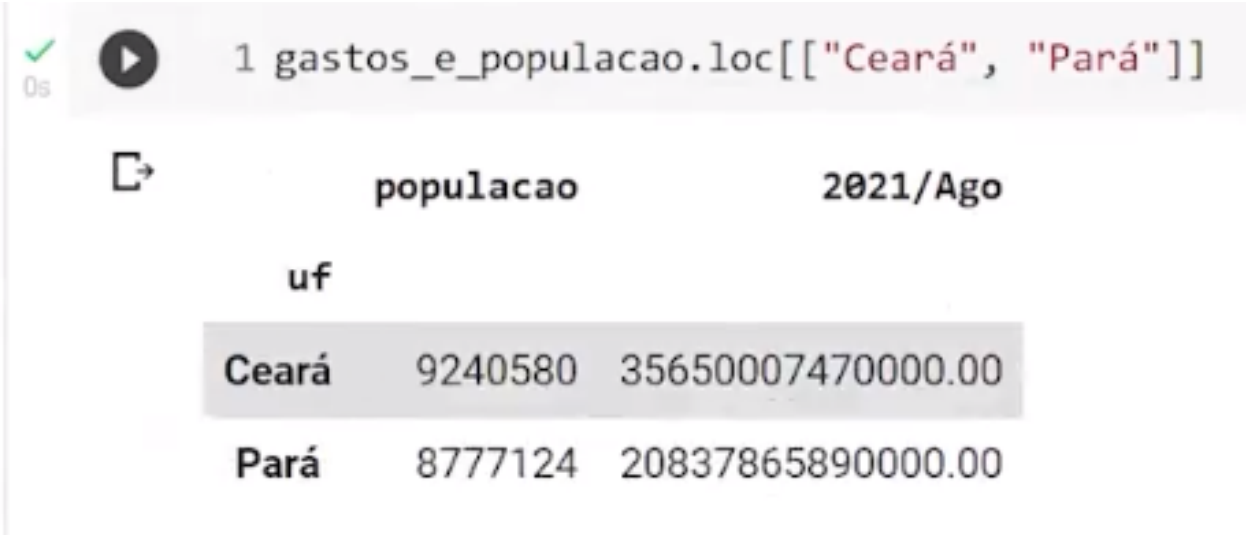
- [ ] `ultima_coluna = gastos_e_populacao.columns[-1]`
- [ ] `gastos_e_populacao[ultima_coluna] = gastos_e_populacao[ultima_coluna] * 1_000_000`
- [ ] `gastos_e_populacao.plot()`



Agora pronto, o valor foi remultiplicado. Porém, o gráfico ainda não faz o menor sentido. 😃 kkkkkkk gráfico de linha está um horror.

Vamos agora pegar um estado específico. Exemplo: Ceará.

- [ ] `gastos_e_populacao.loc[[”Ceará”, "Pará"]]`



Lembrando que o .loc **localiza o índice**! Como multiplicamos o valor ali em cima, na terceira coluna temos esse número gigante. Para arrumar isso:

- [ ] `ultima_coluna = gastos_e_populacao.columns[-1]`
- [ ] `gastos_e_populacao["gastos"] = gastos_e_populacao[ultima_coluna] * 1_000_000` # aqui
- [ ] `gastos_e_populacao.plot()`

```
1 gastos_e_populacao.loc[["Ceará", "Pará"]]
```

uf	populacao	2021/Ago	gastos
Ceará	9240580	35.65	35650007.47
Pará	8777124	20.84	20837865.89

E pronto, fica assim.

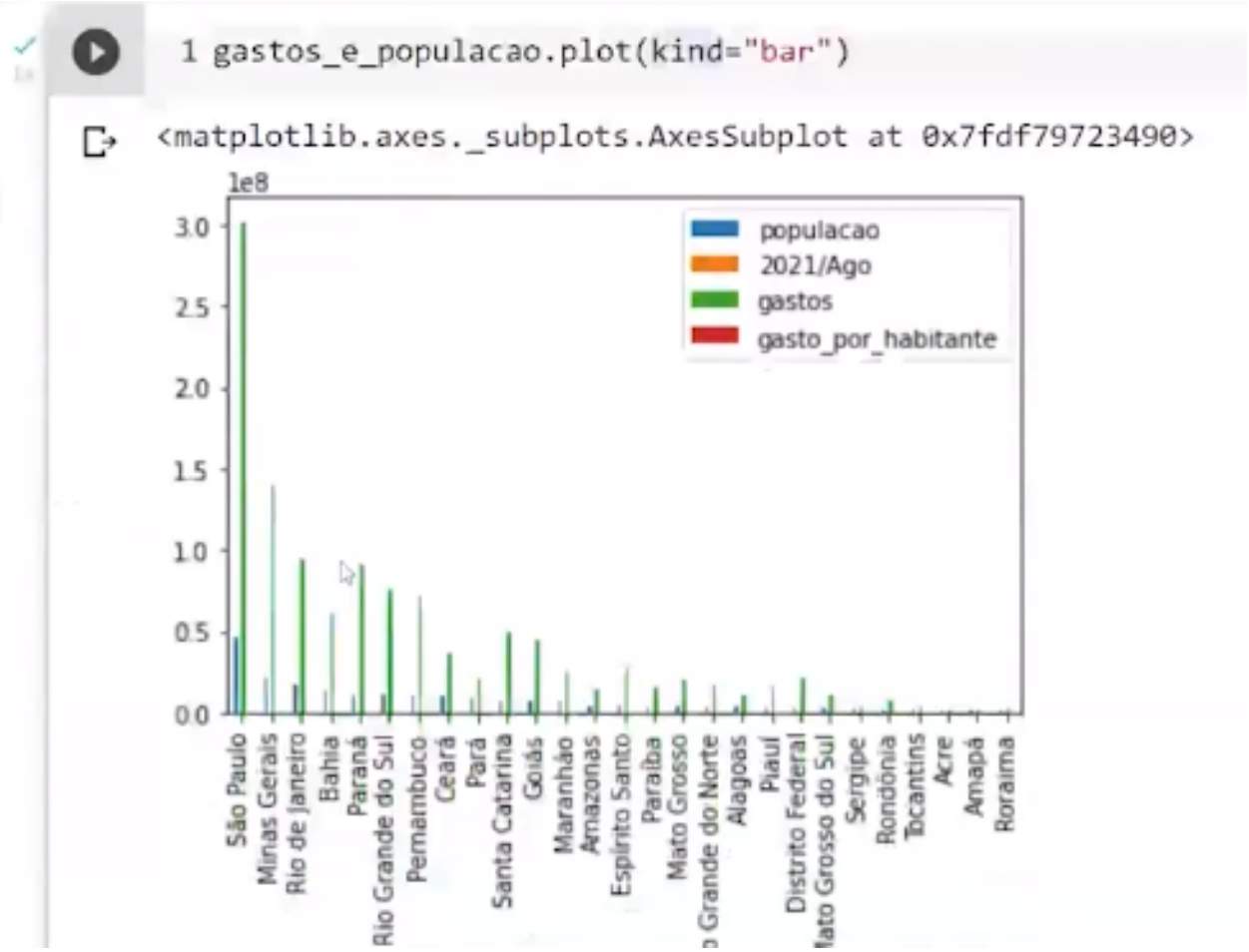
Agora, vamos criar uma coluna nova para gastos por habitantes.

- [ ] `gastos_e_populacao["gasto_por_habitante"] = gastos_e_populacao["gastos"] / gastos_e_populacao["populacao"]`
- [ ] `gastos_e_populacao.head()`

uf	populacao	2021/Ago	gastos	gasto_por_habitante
São Paulo	46649132	301.99	301986341.98	6.47
Minas Gerais	21411923	139.16	139157823.59	6.50
Rio de Janeiro	17463349	94.14	94137361.39	5.39
Bahia	14985284	61.65	61645689.06	4.11
Paraná	11597484	91.19	91187722.64	7.86

Agora, vamos plotar o gasto_por_habitante:

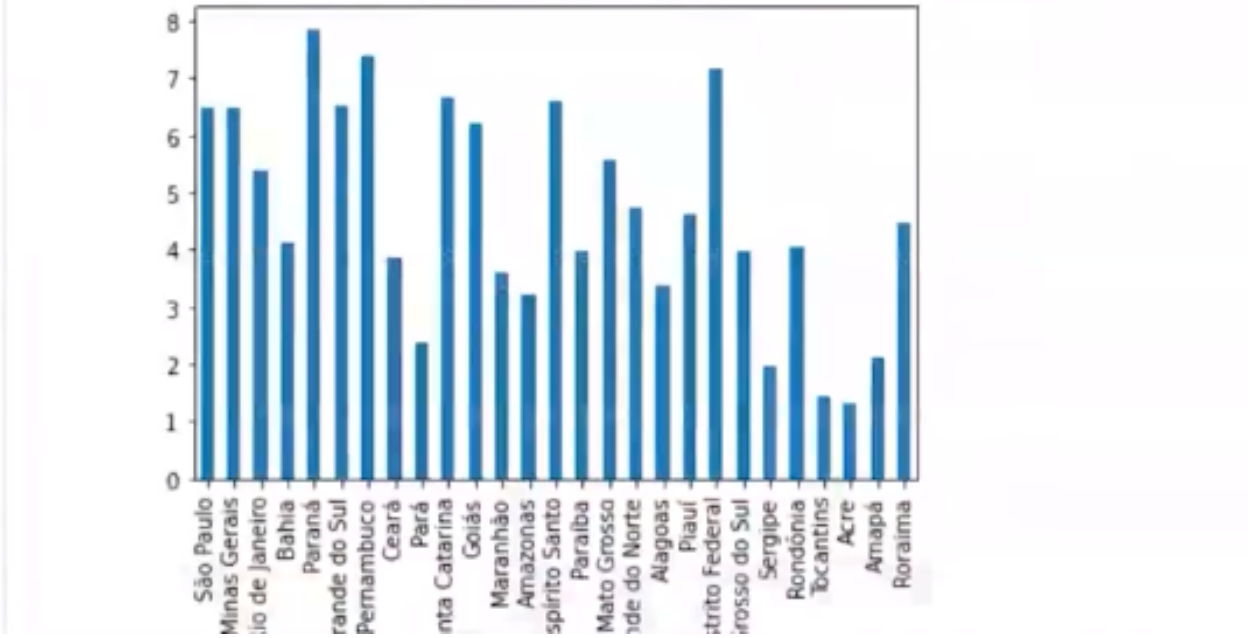
☐ `gastos_e_populacao.plot(kind="bar")` # aqui temos todas as colunas, certo?



☐ `gastos_e_populacao["gasto_por_habitante"].plot(kind="bar")` # aqui temos apenas a gasto_por_habitante

```
[142]  1 gastos_e_populacao["gasto_por_habitante"].plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf74b6ea50>
```



DESAFIO 01: deixar o gráfico mais arrumado com legenda adequada, títulos, eixos e ordenar do maior gasto por habitante para o menor.

DESAFIO 02: colocar a média dos gastos como uma linha horizontal.

O gráfico acima é uma média de médias. Isso tem seus problemas: estados com diferença de população muito grandes vão ter diferenças no custo muito grande. Um valor maior altera muito a média, isso é um problema comum. Poderíamos usar mediana, por exemplo.

DESAFIO 03: ordenar por número de habitantes. Será que a média de gastos é maior ou não?

Na media em que escala o número de habitantes, cresce ou cai o gráfico?

Até aqui, estamos trabalhando com a função .plot() do Pandas. Agora, vamos de fato usar a Seaborn. 😃 uhul! Ela serve para visualizar dados estatísticos.

- [ ] `import seaborn as sns`
- [ ] `sns.scatterplot(data = gastos_e_populacao,  x="populacao", y="gastos")` # é um gráfico que faz pontos!

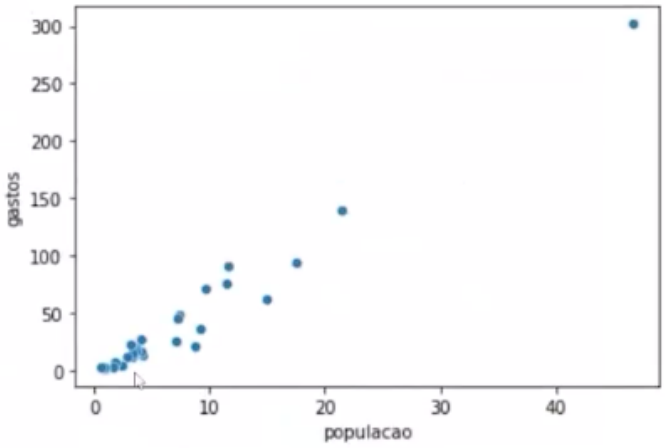
https://seaborn.pydata.org/generated/seaborn.scatterplot.html?highlight=scatterplot#seaborn.scatterplot

Aqui podemos ver que, na medida em que a população aumenta, os gastos também aumentam. Contudo, “gastos_e_populacao” está em milhões. Portanto, vou dividir em 1_000_000:

☐ `sns.scatterplot(data = gastos_e_populacao / 1_000_000,  x=”populacao”, y=”gastos”)` # é um gráfico que faz pontos!

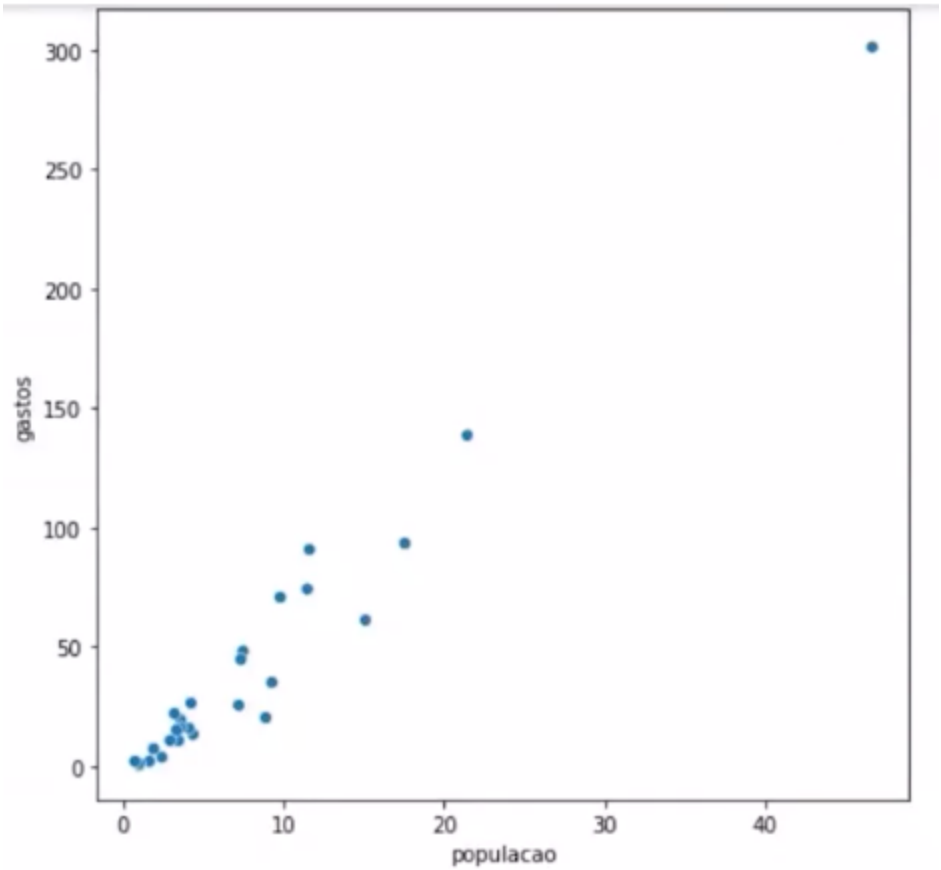
```
1 import seaborn as sns
2
3 sns.scatterplot(data = gastos_e_populacao / 1_000_000, x="populacao", y="gastos")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf6608bd10>
```

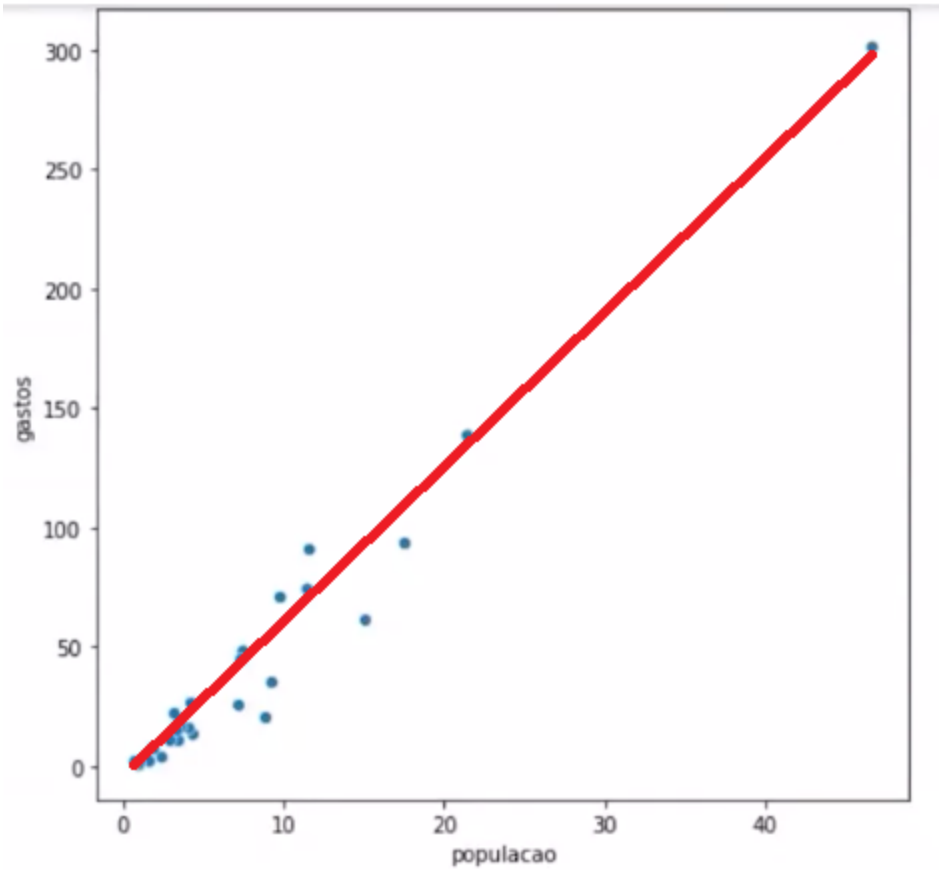


Assim, o gráfico fica mais fácil de ler, a unidade de medida mudou.

- [ ] `plt.figure(figsize=(7,7))`
- [ ] `sns.scatterplot(data = gastos_e_populacao / 1_000_000,  x="populacao", y="gastos")`



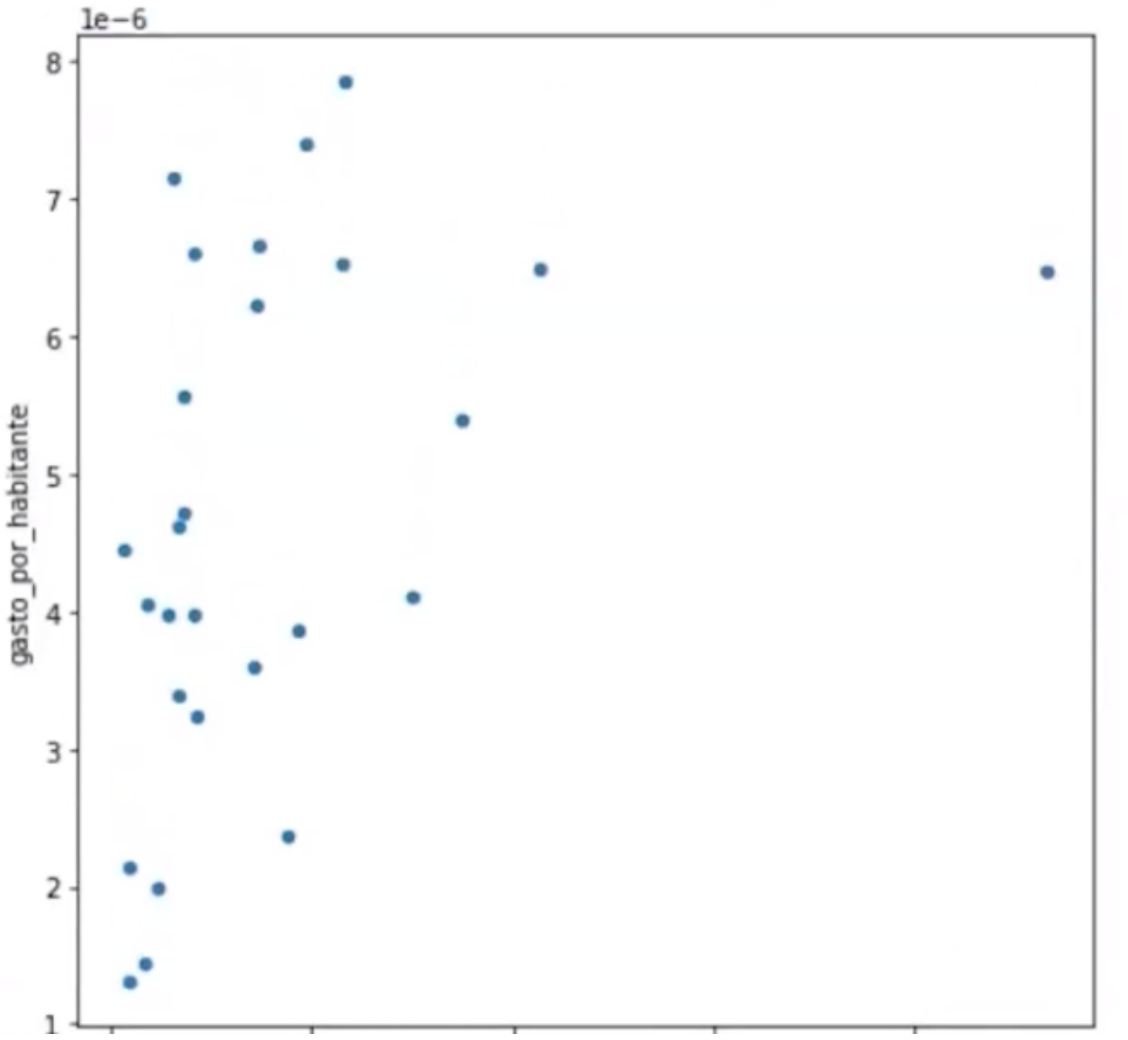
Os gastos crescem em função da população, é fato. A gente consegue ver uma tendência de crescimento aqui. Mas a gente espera que se, à medida que os gastos crescem junto com a população “”””uniforme””” / “”””linear””””, esperamos que essa linha seja reta. Lá em cima, na direita, temos o estado de maior população e maior gasto.



Todo mundo que estiver abaixo dessa diagonal, gasta proporcionalmente à São Paulo - por habitante. Acima da diagonal, gasta proporcionalmente à São Paulo + por habitante. De graça, ganhei essa informação.

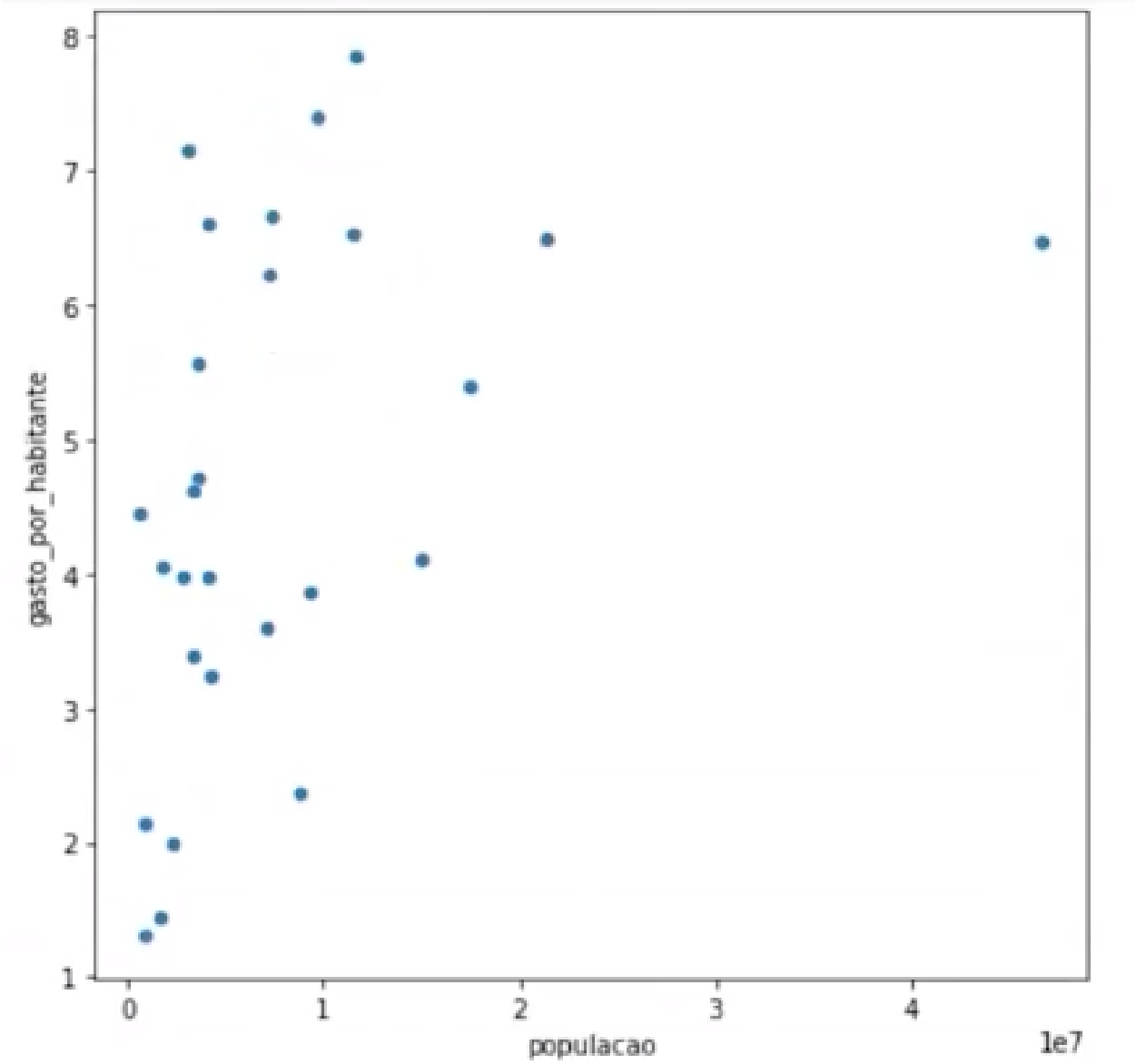
- [ ] `plt.figure(figsize=(7,7))`
- [ ] `sns.scatterplot(data = gastos_e_populacao / 1_000_000,  x="populacao", y="gasto_por_habitante")` # eixo y mudou

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf65fbfb50>
```



Agora, no formato quadradinho, já não fez mais tanto sentido. Além do mais, gastos_por_habitantes já não se divide mais por milhão, vamos tirar.

- [ ] `plt.figure(figsize=(7,7))`
- [ ] `sns.scatterplot(data = gastos_e_populacao, x="populacao", y="gasto_por_habitante")` # tirou a divisão por milhão



Agora, vemos que SP não é o que mais gasta por habitante (estava no topo do gráfico, agora não mais). Mas, na medida em que a população aumenta, até existe uma tendência do gasto_por_habitante crescer.

Na população até 10 milhões, a maior parte dos estados gasta qualquer valor entre 1 e 8 reais. Mas o eixo y está horrível, está começando com 1 em vez de 0. No gráfico

anterior, era interessante aquela linha diagonal para poder concluir algo sobre o gráfico. Nesse acima, não mais.

DESAFIO 04: melhorar todos os gráficos. Já tem nomes nos eixos x e y, mas tem muito o que melhorar!

DESAFIO 05: executar o mesmo gráfico para o penúltimo mês.

DESAFIO 06: plote o gráfico de gasto por população para os dois meses simultaneamente no mesmo gráfico, diferenciando pontos com uma tonalidade de cor diferente para cada mês.

Vamos precisar do `ordenados_por_total`! Lembra? Lá atrás nas aulas, tinha tudo ordenado por colunas.

Parte 2 | Proporcionalidade e Seaborn II

Então, de alguma forma, vou precisar `insere_gastos_e_gasto_por_habitante(ordenados_por_total, gastos_e_populacao, ordenados_por_total.columns[-1])`. Essa parte ficou um pouco confusa. 😟

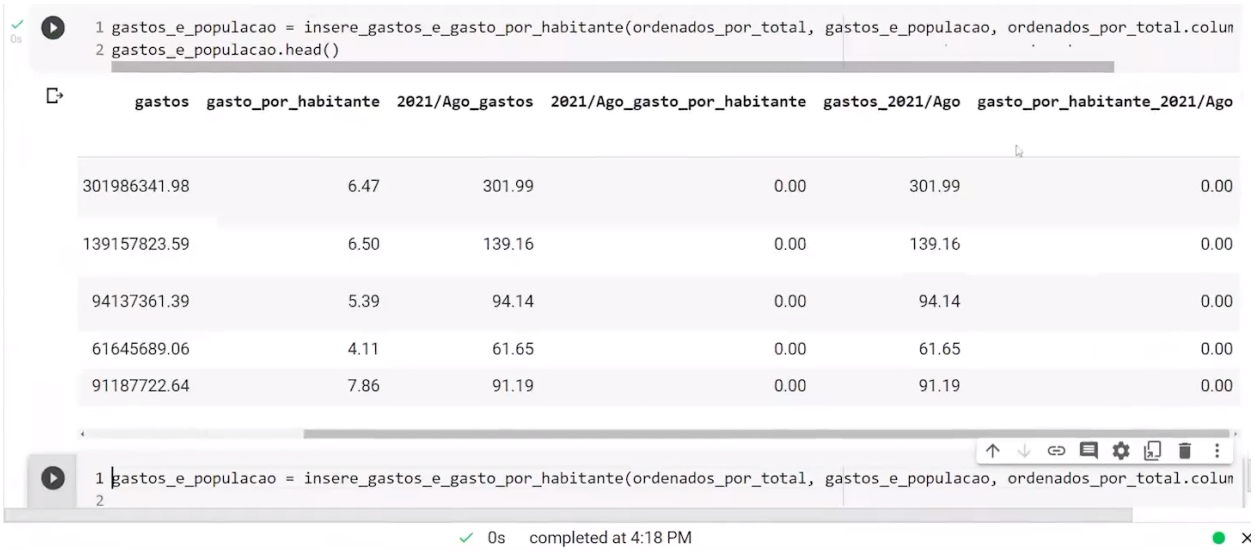
- [ ] `def insere_gastos_e_gasto_por_habitante(ordenados_por_total, gastos_e_populacao, mes):`
- [ ] `gastos = ordenados_por_total[mes]`
- [ ] `gastos_e_populacao[f”gastos_{mes}”] = gastos`
- [ ] `#gastos_e_populacao[f"{mes}_gasto_por_habitante"] = gastos_e_populacao[f"gastos_{mes}"] / gastos_e_populacao[”populacao”]`

- [ ] `return gastos_e_populacao`

- [ ] `#print(gastos.head())`
- [ ] `#print(gastos_e_populacao.head())`

- [ ] `print(ordenados_por_total.index)`
- [ ] `print(gastos_e_populacao.index)`

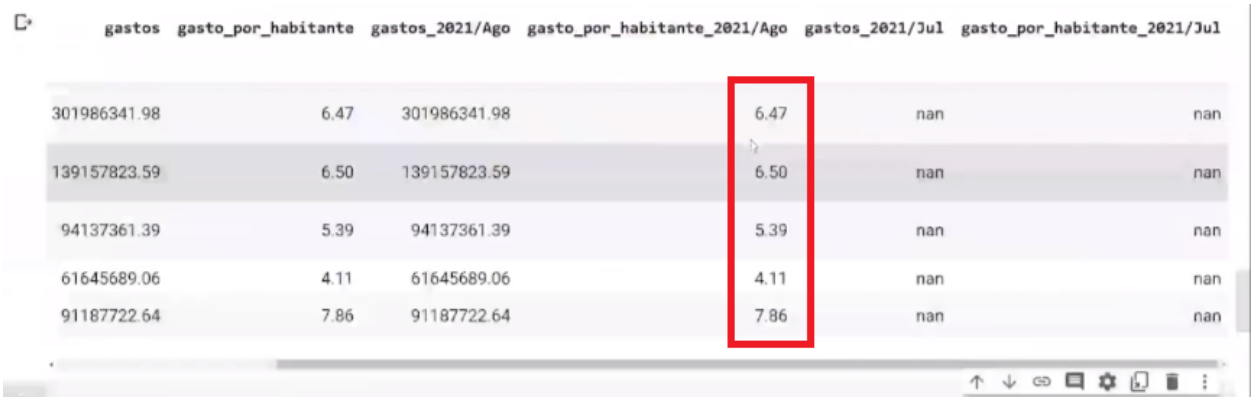
- [ ] `insere_gastos_e_gasto_por_habitante(ordenados_por_total, gastos_e_populacao, ordenados_por_total.columns[-1])`
- [ ] `ordenados_por_total.head()`



021/Ago_gasto_por_habitante	gastos_2021/Ago	gasto_por_habitante_2021/Ago	gastos_2021/Jul	gasto_por_habitante_2021/Jul
0.00	301.99	0.00	nan	nan
0.00	139.16	0.00	nan	nan
0.00	94.14	0.00	nan	nan
0.00	61.65	0.00	nan	nan
0.00	91.19	0.00	nan	nan

Porém, aqui temos um problema! Tem alguns valores que não foram encontrados! Isso se deve pelo fato de não ter **multiplicado por milhões** lá em cima:

```
gastos_e_populacao[f"gastos_{mes}"] = gastos * 1_000_000
```



gastos	gasto_por_habitante	gastos_2021/Ago	gasto_por_habitante_2021/Ago	gastos_2021/Jul	gasto_por_habitante_2021/Jul
301986341.98	6.47	301986341.98	6.47	nan	nan
139157823.59	6.50	139157823.59	6.50	nan	nan
94137361.39	5.39	94137361.39	5.39	nan	nan
61645689.06	4.11	61645689.06	4.11	nan	nan
91187722.64	7.86	91187722.64	7.86	nan	nan

Mas, de alguma forma, o índice dos **gastos** e o índice dos **gastos_e_populacao** não está batendo. O índice ainda está na frente das unidades da federação.

Porque aqui, não foi feita a limpeza:

```
def insere_gastos_e_gasto_por_habitante(ordenados_por_total, gastos_e_populacao, mes):
    gastos = ordenados_por_total[mes]
    gastos_e_populacao[f"gastos_{mes}"] = gastos * 1_000_000
    gastos_e_populacao[f"gastos_por_habitante_{mes}"] = gastos_e_populacao[f"gastos_{mes}"] / gastos_e_populacao["populacao"]
    return gastos_e_populacao
```

Como forma de limpar, podemos fazer:

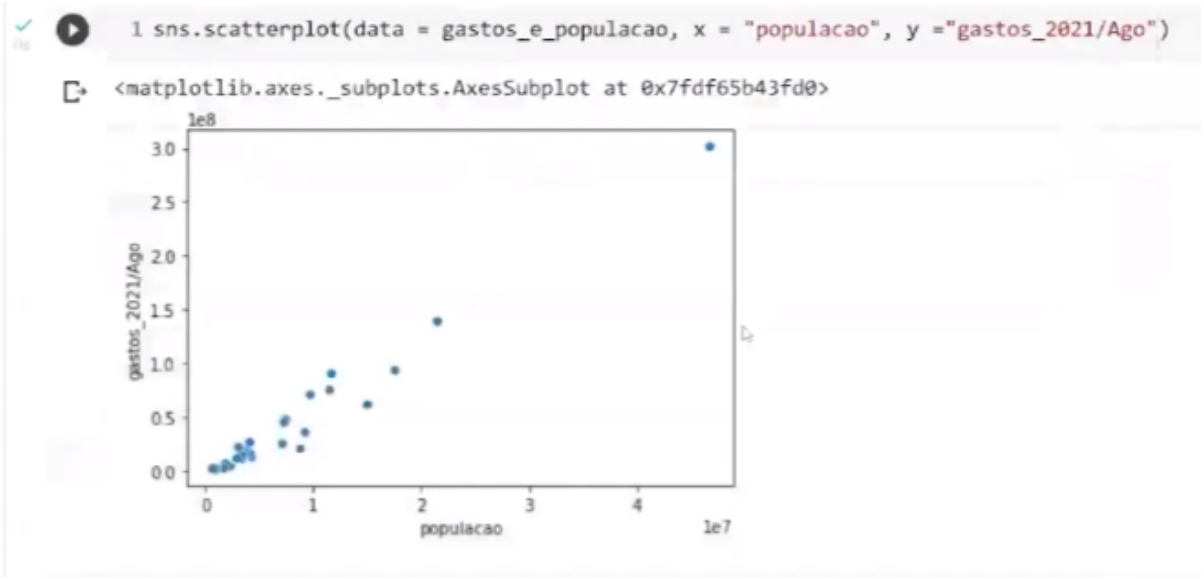
```
def insere_gastos_e_gasto_por_habitante(ordenados_por_total, gastos_e_populacao, mes):
    gastos = ordenados_por_total[mes]
    gastos.index = gastos.index.str[3:].str.strip() # aqui!
    print(gastos) # aqui!
    gastos_e_populacao[f"gastos_{mes}"] = gastos * 1_000_000
```

```
gastos_e_populacao[f"gastos_por_habitante_{mes}"] = gastos_e_populacao[f"gastos_{mes}"] / gastos_e_populacao["populacao"]
```

```
return gastos_e_populacao
```

Agora, queremos plotar isso:

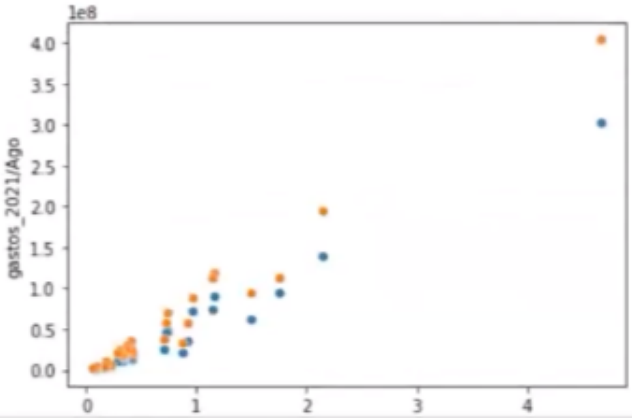
- [ ] `sns.scatterplot(data = gastos_e_populacao, x = "populacao", y = "gastos_2021/Ago")`



- [ ] `sns.scatterplot(data = gastos_e_populacao, x = "populacao", y = "gastos_2021/Jul")`

```
1 sns.scatterplot(data = gastos_e_populacao, x = "populacao", y ="gastos_2021/Ago")
2 sns.scatterplot(data = gastos_e_populacao, x = "populacao", y ="gastos_2021/Jul")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf655bec40>
```



Agora, podemos ver que tem duas cores de bolinhas, ou seja, um é Agosto e outro é Julho.

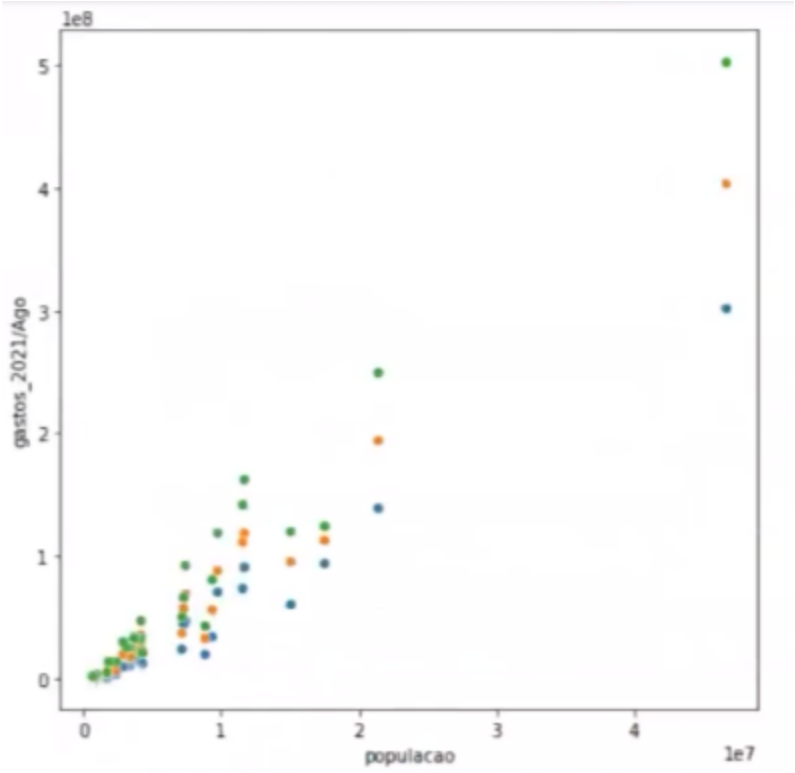
E se quero 3? Fácil. Copio aquela linha que fiz por mês:

☐ `gastos_e_populacao = insere_gastos_e_gasto_por_habitante(ordenados_por_total, gastos_e_populacao, ordenados_por_total.columns[-3])` # aqui faço -3 para três meses

uf	populacao	2021/Ago	gastos	gasto_por_habitante	gastos_2021/Ago	gasto_por_habitante_2021/Ago	gastos_2021/Jul
São Paulo	46649132	301.99	301986341.98	6.47	301986341.98	6.47	404373175.61
Minas Gerais	21411923	139.16	139157823.59	6.50	139157823.59	6.50	194734136.81
Rio de Janeiro	17463349	94.14	94137361.39	5.39	94137361.39	5.39	113660882.20
Bahia	14985284	61.65	61645689.06	4.11	61645689.06	4.11	95691895.48
Paraná	11597484	91.19	91187722.64	7.86	91187722.64	7.86	119408026.56

Aí fazemos o mesmo para scatterplot, mas antes vamos mudar o tamanho da figura:

- [ ] `plt.figure(figsize=(7,7))` # aqui
- [ ] `sns.scatterplot(data = gastos_e_populacao, x = "populacao", y = "gastos_2021/Ago")`
- [ ] `sns.scatterplot(data = gastos_e_populacao, x = "populacao", y = "gastos_2021/Jul")`
- [ ] `sns.scatterplot(data = gastos_e_populacao, x = "populacao", y = "gastos_2021/Jun")`



Agora, temos 3 meses distintos plotados aqui! 😊

DESAFIO 07: arrumar legenda e gráfico, diminuir marcações;

DESAFIO 08: explorem esse gráfico e levantem alguma hipótese ou questão;

DESAFIO 09: comprar os últimos 12 meses com os 12 meses anteriores. Somar os últimos 12 meses em uma única coluna.