

Aula 06 | PosTech | Agrupando dados e analisando por categorias

Anotações sobre a sexta aula da PosTech FIAP ✨✨

<https://on.fiap.com.br/mod/conteudoshtml/view.php?id=307799&c=8729&sesskey=mp0PcE8JII>

Temas abordados:

- Como usar o Groupby;
- Como alterar a paleta de cores do seu gráfico Seaborn;
- Como usar a paleta de cores para ajudar na interpretação de gráficos;
- O que é e como usar o catplot para comparar categorias;

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


Parte 1 | Agrupando dados e analisando por categoria I

Escolhendo o estado de SP...

☐ `mensal_do_meu_estado = mensal_aberto.query("uf=='São Paulo'")`

☐ `mensal_do_meu_estado.head()`

☒  `1 mensal_do_meu_estado = mensal_aberto.query("uf=='São Paulo'")`
`2 mensal_do_meu_estado.head()`



	dia_mes_ano	uf	gasto	ano	mes	gasto_diario
0	2008-02-01	São Paulo	173.06	2008	2	6.18
1	2008-03-01	São Paulo	170.62	2008	3	5.50
2	2008-04-01	São Paulo	170.39	2008	4	5.68
3	2008-05-01	São Paulo	172.51	2008	5	5.56
4	2008-06-01	São Paulo	175.56	2008	6	5.85

Agora, quero pegar os gastos por ano. Quando digo por ano, é porque quero fazer um cálculo de agrupamento:

☐ `gastos_por_ano = mensal_do_meu_estado.groupby("ano").sum()` # leia sempre a documentação! importante

☐ `gastos_por_ano.head()`

```
1 gastos_por_ano = mensal_do_meu_estado.groupby("ano").sum()
2 gastos_por_ano.head()
```

	gasto	mes	gasto_diario
ano			
2008	1962.42	77	64.67
2009	2490.55	78	81.87
2010	2700.26	78	88.74
2011	2796.46	78	91.97
2012	2877.14	78	94.57

Na tabela original, tínhamos 6 variáveis. Agora, ela possui apenas 3 variáveis (colunas) e 1 índice, que é o ano. Índice é o que a gente agrupou e faz sentido, porque só vai ter uma linha por ano.

```
1 mensal_do_meu_estado = mensal_aberto.query("uf=='São Paulo'")
2 mensal_do_meu_estado.head()
```

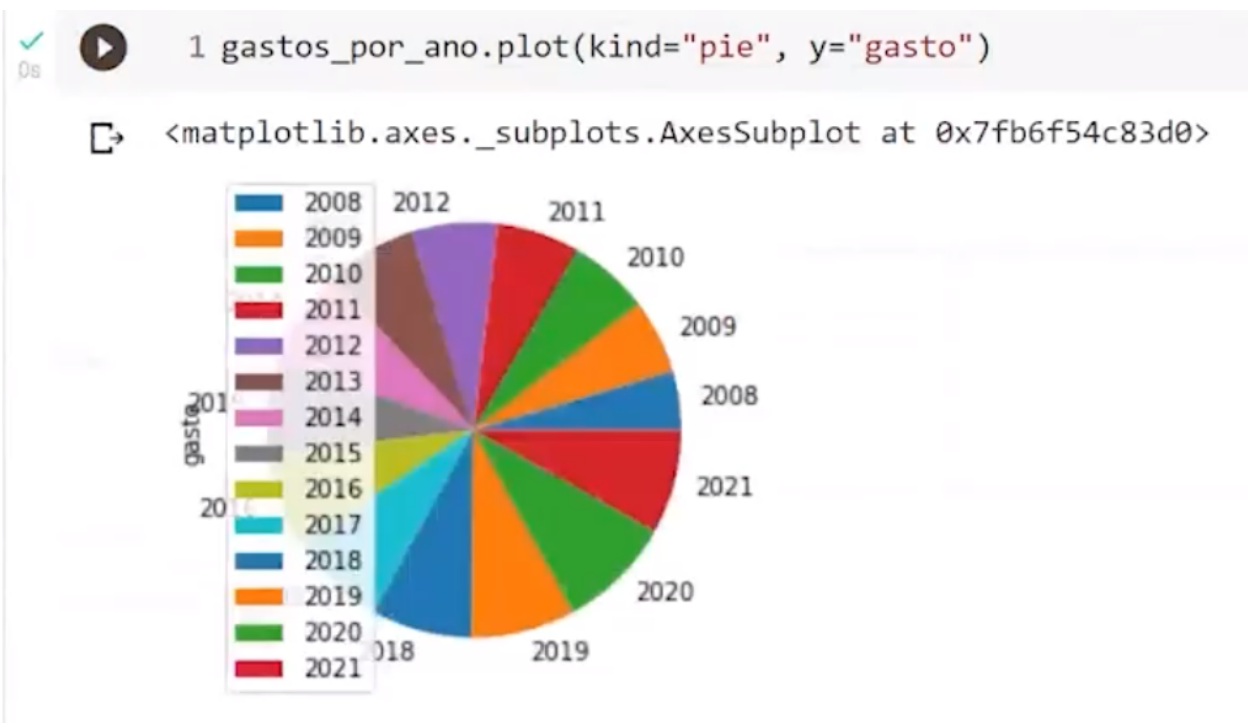
	dia_mes_ano	uf	gasto	ano	mes	gasto_diario
0	2008-02-01	São Paulo	173.06	2008	2	6.18
1	2008-03-01	São Paulo	170.62	2008	3	5.50
2	2008-04-01	São Paulo	170.39	2008	4	5.68
3	2008-05-01	São Paulo	172.51	2008	5	5.56
4	2008-06-01	São Paulo	175.56	2008	6	5.85

Aqui nessa tabela anterior, tínhamos 6 colunas, mas tirando a “ano” sobram 5 ainda.

☐ `gastos_por_ano = mensal_do_meu_estado.groupby("ano").sum()`

Nessa parte aqui do código, quando fazemos `.sum()` podemos notar que não dá pra somar "uf" porque é string, não é valor numérico. Portanto, se eu for plotar aqui o gasto por ano, talvez a coluna "mês" seja um fator de divisão, de normalização para os nossos dados, se a gente achar que faz sentido.

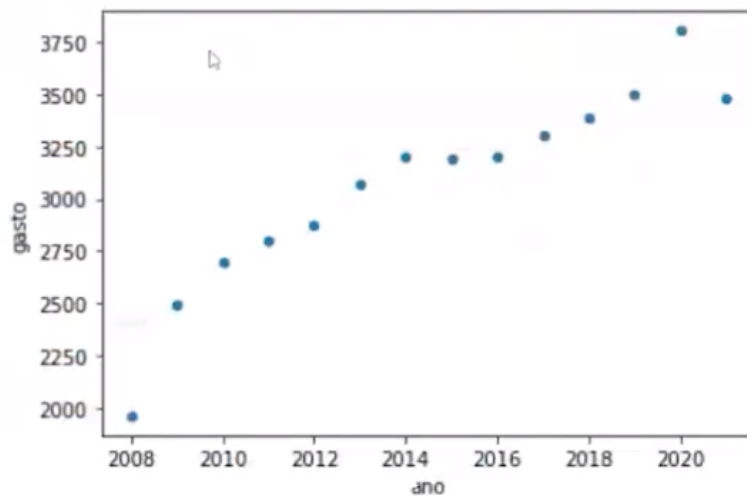
☐ `gastos_por_ano.plot(kind="pie", y="gasto")`



☐ `sns.scatterplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto")`

```
1 sns.scatterplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto")
```

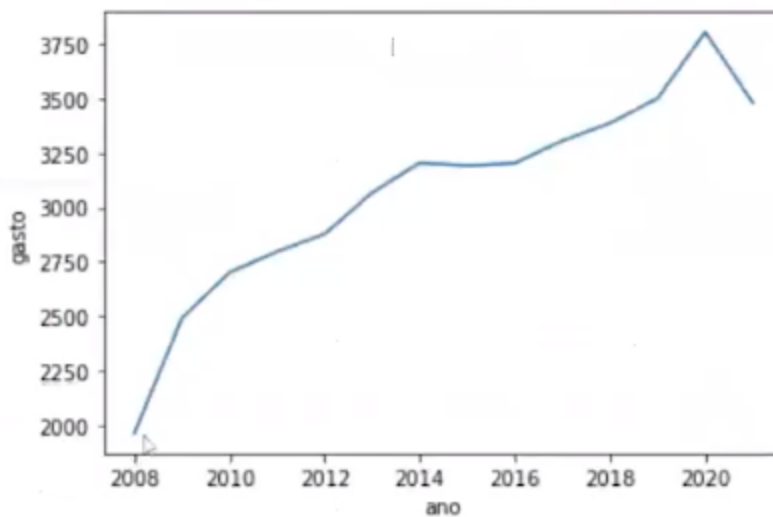
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb6f58445d0>
```



Talvez fosse mais interessante um **lineplot** que um **scatterplot**:

```
sns.lineplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb6f5278990>
```



Contudo, com esse gráfico, pode-se criar uma ilusão de que em Março de 2008 gasto x, em Dezembro de 2009 gastou y, sendo que não é verdade. A verdade é que em 2008 gastou-se 2 bilhões.

Nesse caso, portanto, pontos são mais valiosos do que linhas! 😊 A linha cria uma ilusão de que existem valores intermediários, sendo que não existem, são realmente pontos.

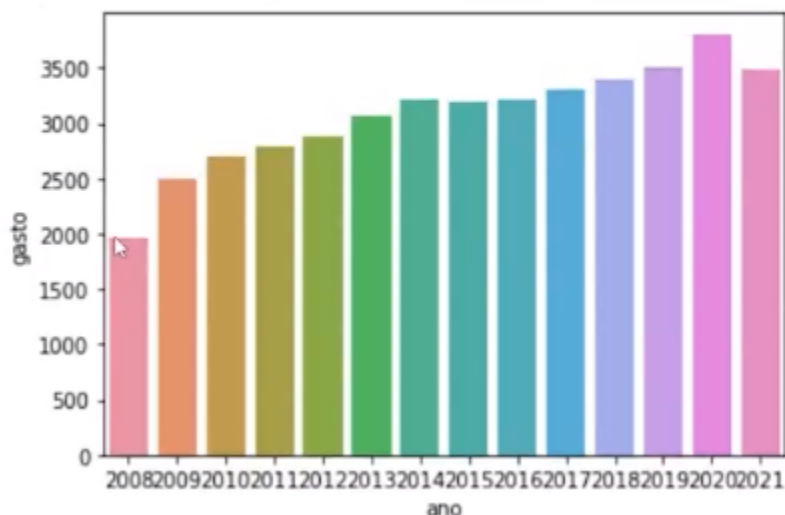
O lineplot é pior pois gera a sensação da existência de valores intermediários! Esse gráfico perde também a sazonalidade dentro de um ano.

Por outro lado, também podemos fazer um gráfico de barra!

```
□ sns.barplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto")
```

```
1 sns.barplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto")
```

```
↳ <matplotlib.axes._subplots.AxesSubplot at 0x7fb6f58a4490>
```



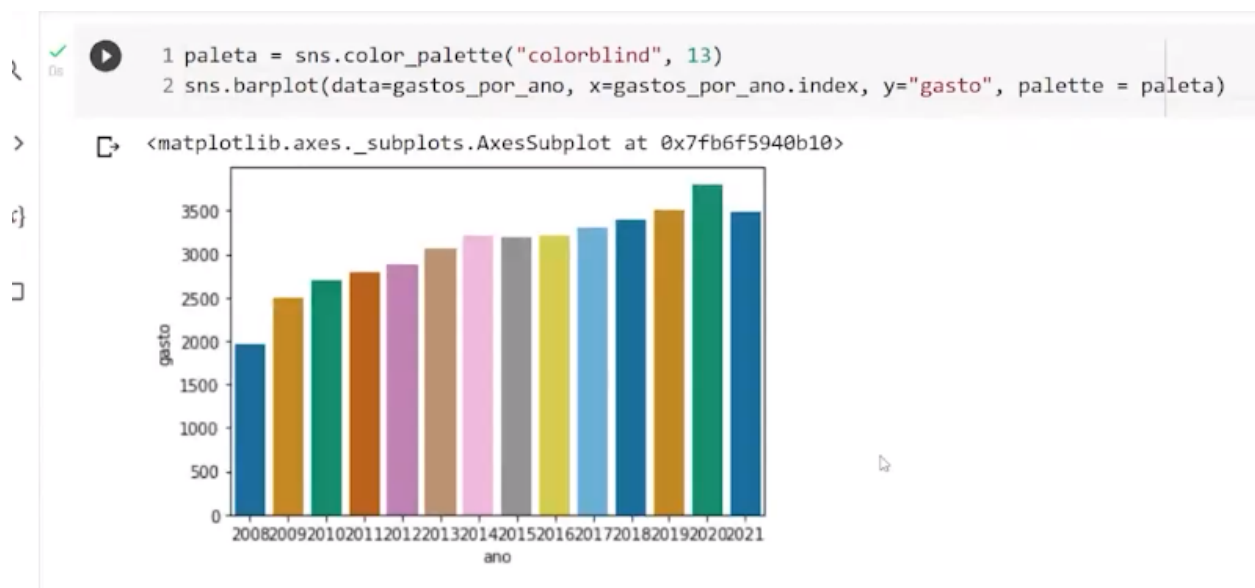
Aqui nós temos as barras. Ele diz para nós que em 2015 / 2016 não houve crescimento, 2017 muito pouco, 2018 muito pouco, 2019 pouco de novo (...) e aí em 2020 teve um salto.

Se cada barra é uma categoria, cada barra tem que ter uma cor diferente:

https://seaborn.pydata.org/tutorial/color_palettes.html

```
❑ paleta = sns.color_palette("colorblind", 13)
```

```
❑ sns.barplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto", palette = paleta)
```



2021 - 2008 = 13, mas a diferença entre o número de barras entre 2021 e 2008 é 14.

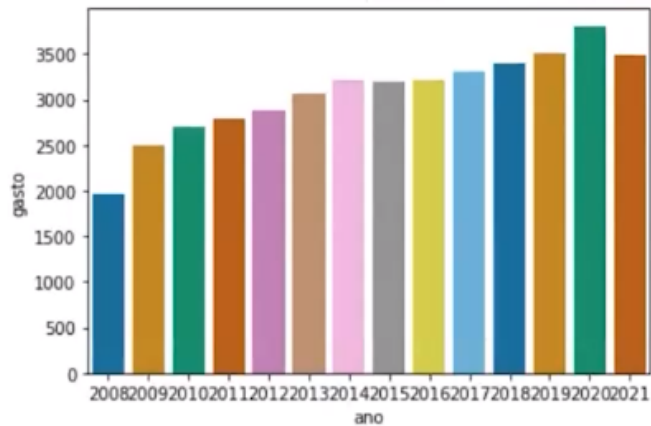
Então:

```

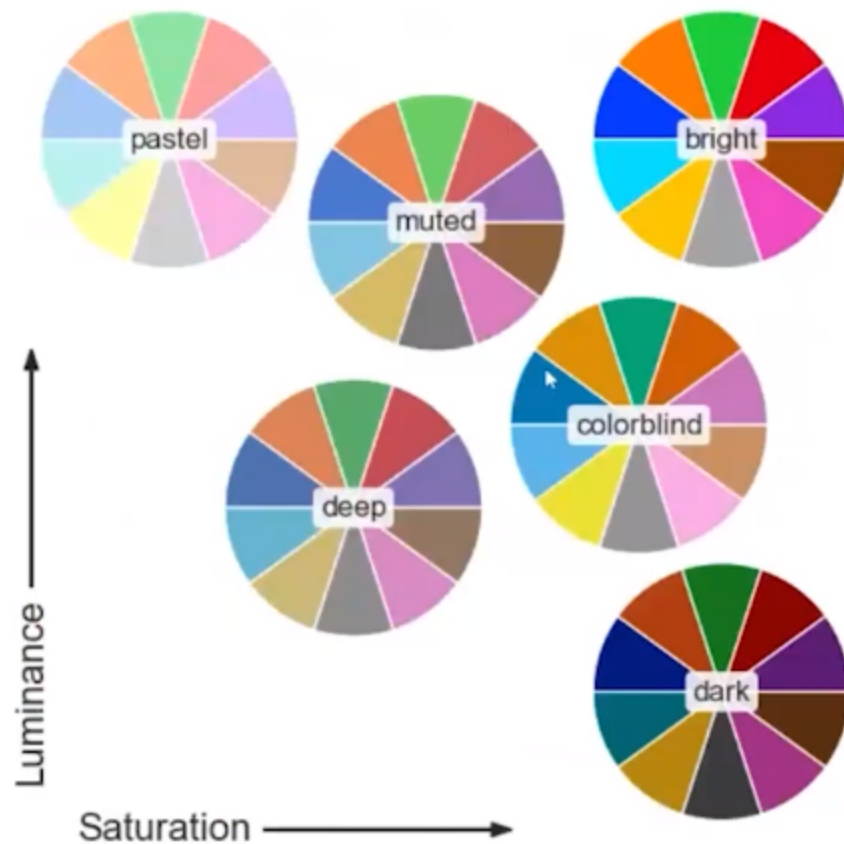
1 paleta = sns.color_palette("colorblind", 14)
2 sns.barplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto", palette = paleta)

```

<matplotlib.axes._subplots.AxesSubplot at 0x7fb6f5ea8190>



Entretanto, a paleta ali em cima se repetiu, porque a paleta de cores do Seaborn acabou se repetindo, só consegue fazer 10 cores.



Se quiser mais de 10 cores, tem que escolher uma paleta que tenha mais de 10.

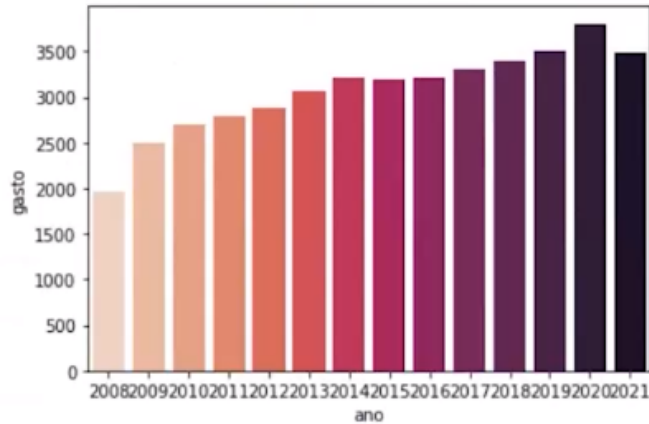
A paleta “rocket”, por exemplo...

☐ `paleta = sns.color_palette("rocket_r", 14)`

☐ `sns.barplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto", palette = paleta)`

```
1 paleta = sns.color_palette("rocket_r", 14)
2 sns.barplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto", palette = paleta)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fb6f087d7d0>

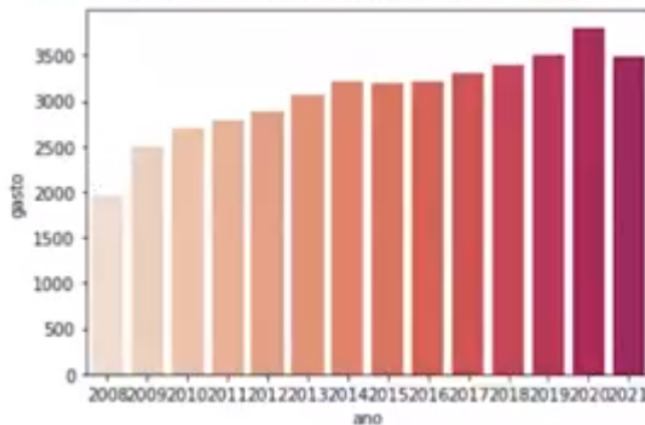


Aqui a gente consegue ver que tem uma tendência de crescimento. Como tem 14 tons (começou com o mais claro na esquerda e o mais escuro na direita). Se eu mudar pra 24 tons, olha o que acontece:

```
1 paleta = sns.color_palette("rocket_r", 24)
```

```
2 sns.barplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto", palette = paleta)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fb6f08de190>



Contudo, isso não quer dizer que quanto mais alto, mais escuro.

Desafio 01: colocar a cor de acordo com um valor mais baixo ou mais alto, não de acordo com esquerda e direita.

Obs: se quiser exemplos, clicar em “Gallery” na documentação do Seaborn.

Parte 2 | Agrupando dados e analisando por categoria II

```
estados = ["Amazonas", "Mato Grosso", "Ceará"]  
  
por_ano_dos_estados = mensal_aberto.query("uf in @estados").groupby(["uf", "ano"]).sum()  
# todas as variáveis que vamos colocar nas queries vamos colocar como "@estados"  
  
por_ano_dos_estados.head()
```

```
1 estados = ["Amazonas", "Mato Grosso", "Ceará"]  
2 por_ano_dos_estados = mensal_aberto.query("uf in @estados").groupby(["uf", "ano"]).sum()  
3 por_ano_dos_estados.head()
```



uf	ano	gasto_mes	gasto_diario
Amazonas	2008	91.28	77
	2009	106.49	78
	2010	114.18	78
	2011	120.26	78
	2012	122.92	78

Mas aí eu quero tirar o index...

☐ `estados = ["Amazonas", "Mato Grosso", "Ceará"]`

☐ `por_ano_dos_estados = mensal_aberto.query("uf in @estados").groupby(["uf",`

`"ano"]).sum().reset_index()` # todas as variáveis que vamos colocar nas queries vamos colocar como "@estados"

☐ `por_ano_dos_estados.head()`

```
1 estados = ["Amazonas", "Mato Grosso", "Ceará"]
2 por_ano_dos_estados = mensal_aberto.query("uf in @estados").groupby(["uf", "ano"]).sum().reset_index()
3 por_ano_dos_estados.head()
```

uf ano gasto mes gasto_diario

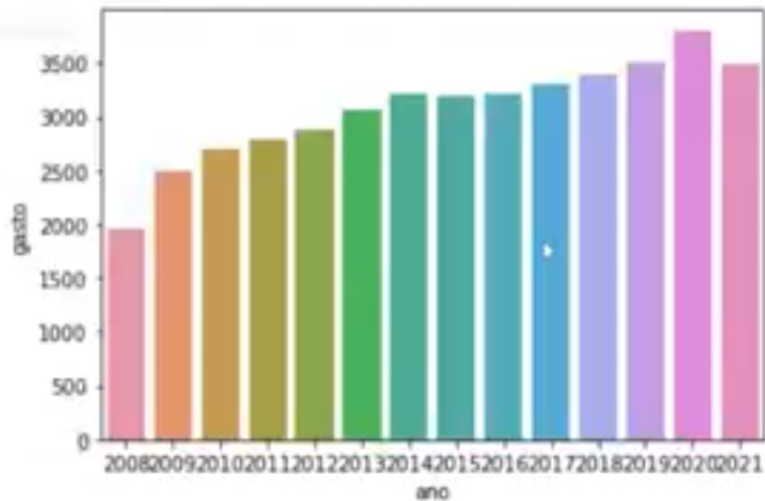
0	Amazonas	2008	91.28	77	3.01
1	Amazonas	2009	106.49	78	3.50
2	Amazonas	2010	114.18	78	3.75
3	Amazonas	2011	120.26	78	3.96
4	Amazonas	2012	122.92	78	4.04

Agora sim, as colunas ficaram certinhas.

☐ `sns.barplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto")`

```
1 sns.barplot(data=gastos_por_ano, x=gastos_por_ano.index, y="gasto")
```

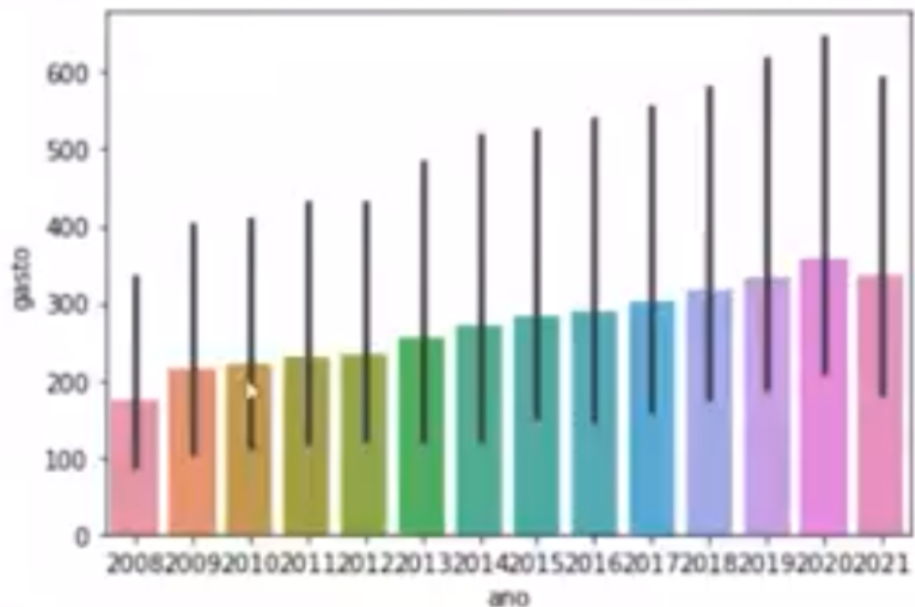
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb6f05fd790>
```



```
sns.barplot(data=gastos_por_ano, x="ano", y="gasto")
```

```
1 sns.barplot(data=por_ano_dos_estados, x="ano", y="gasto")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb6f08e4d90>
```

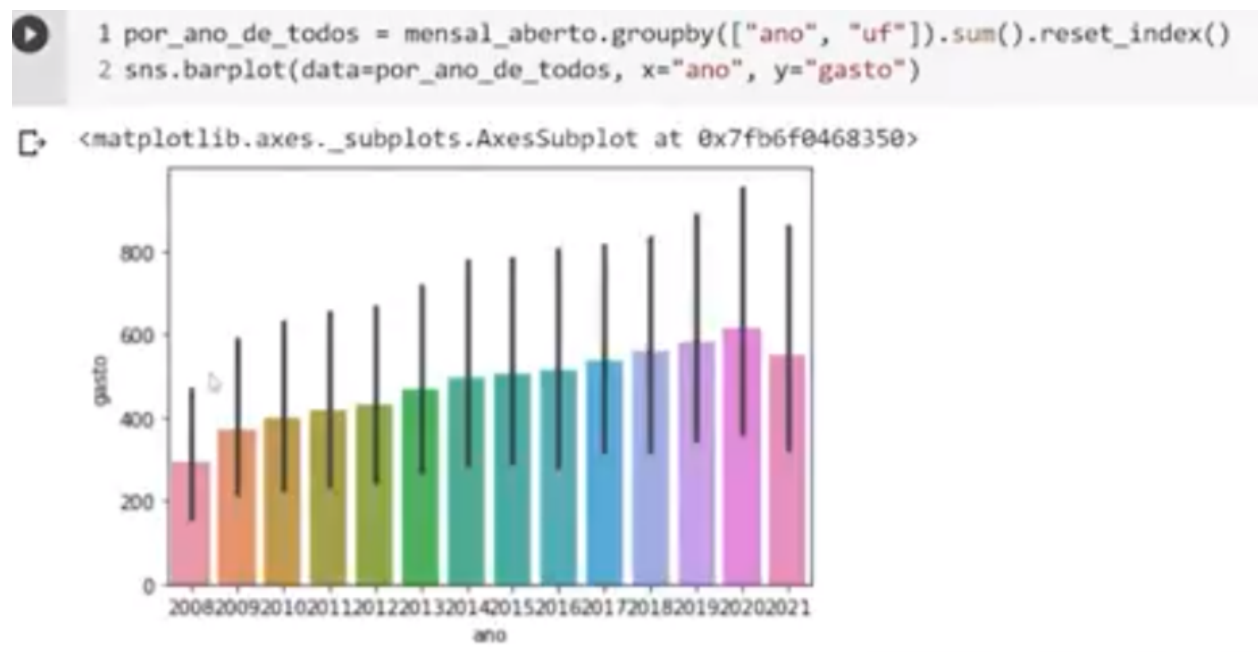


Essas barrinhas pretas é como se fosse um “intervalo de confiança”.

Para ver quanto um estado gasta, seria interessante fazer isso:

```
☐ por_ano_de_todos = mensal_aberto.groupby(["ano", "uf"]).sum().reset_index()
```

```
☐ sns.barplot(data=por_ano_de_todos, x="ano", y="gasto")
```



Mas esse intervalo de confiança, na verdade, serve pra nada pois pegamos todos os estados.

Intervalo de confiança serve pra gente poder supor informações.

Não é o que a gente quer...

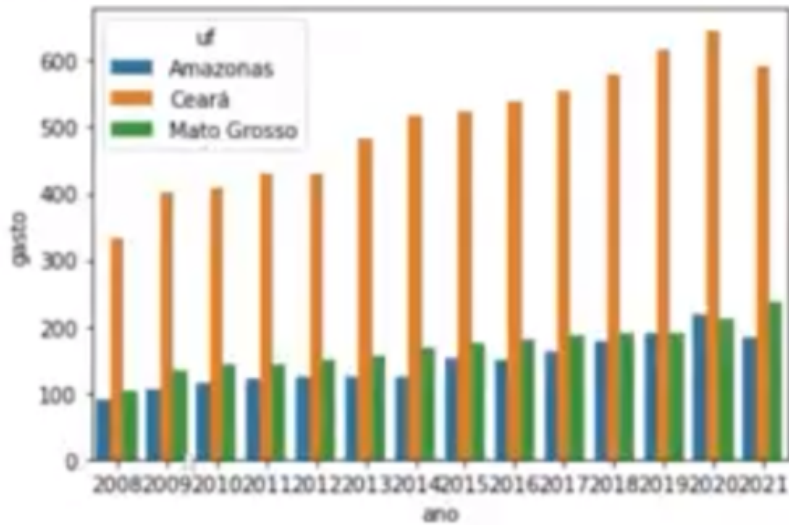
```
☐ sns.barplot(data=por_ano_de_todos, x="ano", y="gasto", hue="uf")
```



```
1 sns.barplot(data=por_ano_dos_estados, x="ano", y="gasto", hue="uf")
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb6f5af3b10>
```



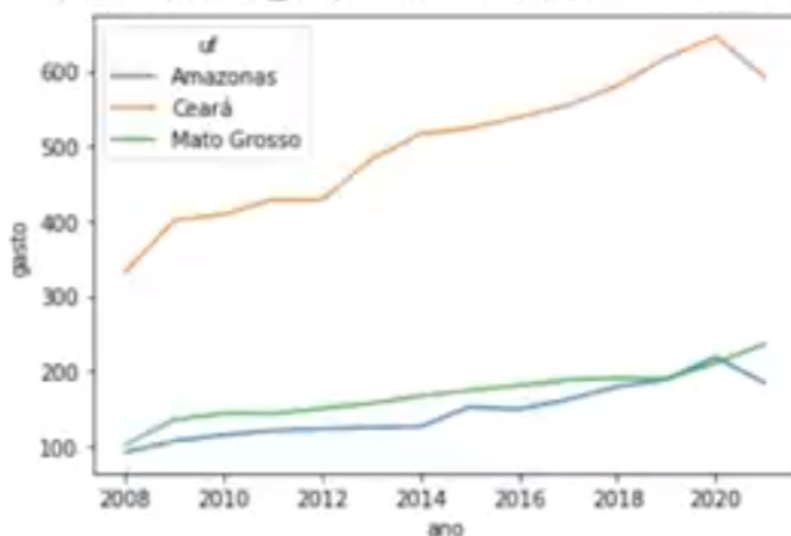
Esse gráfico traz várias informações pra gente, mas nada muito claro, está confuso! barplot pra tantas barras assim não serve pra gente.



```
sns.lineplot(data=por_ano_de_todos, x="ano", y="gasto", hue="uf")
```

```
1 sns.lineplot(data=por_ano_dos_estados, x="ano", y="gasto", hue="uf")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb6f00a38d0>
```

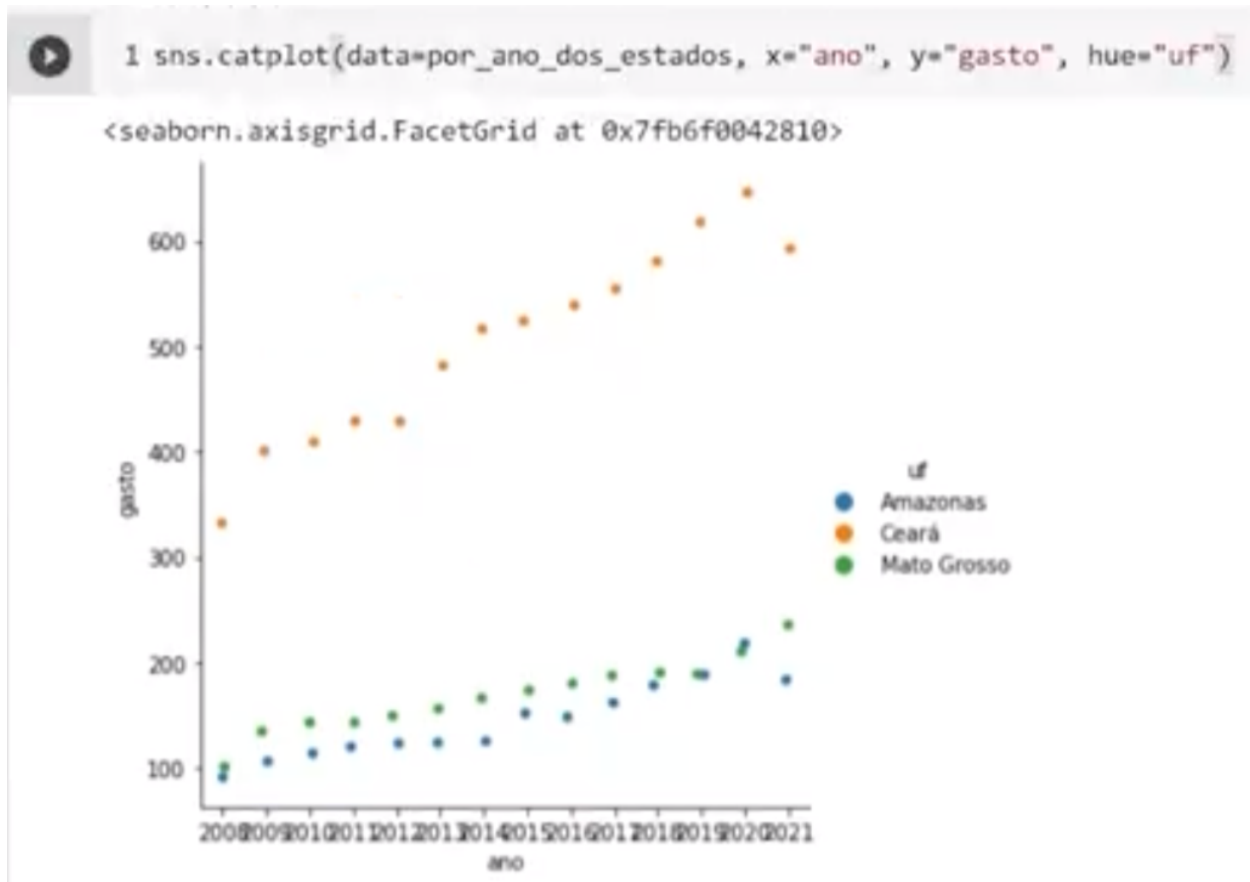


Se a gente pegar o gráfico em formato de linha, é muito claro que houve uma decaída no Amazonas em 2020, algo que o gráfico anterior não mostrava!

Até agora, estávamos trabalhando com uma função de baixo nível do Seaborn, mas temos funções de alto nível também!

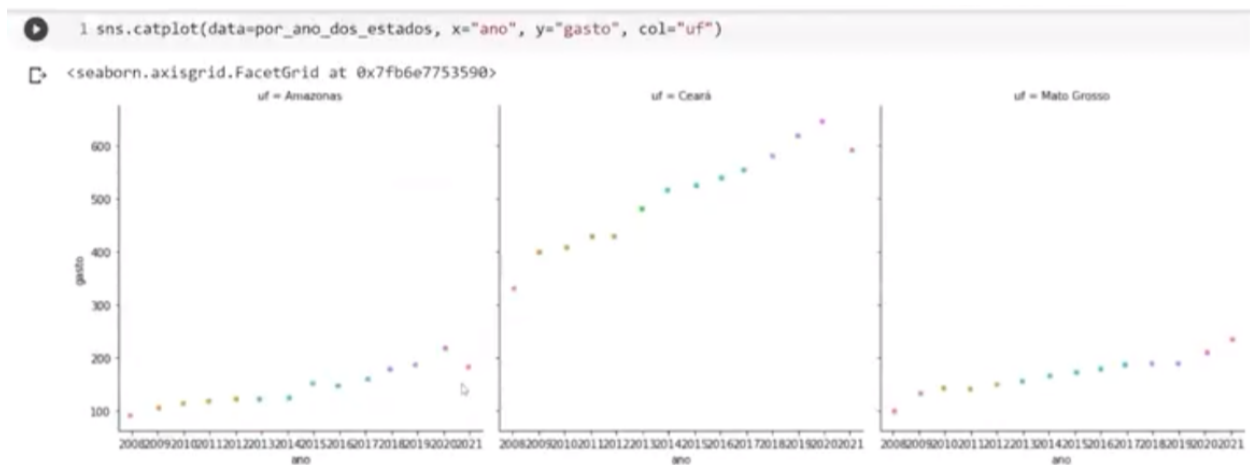
O “catplot” é nível de figura, diferente do barplot:

```
1 sns.catplot(data=por_ano_de_todos, x="ano", y="gasto", hue="uf")
```

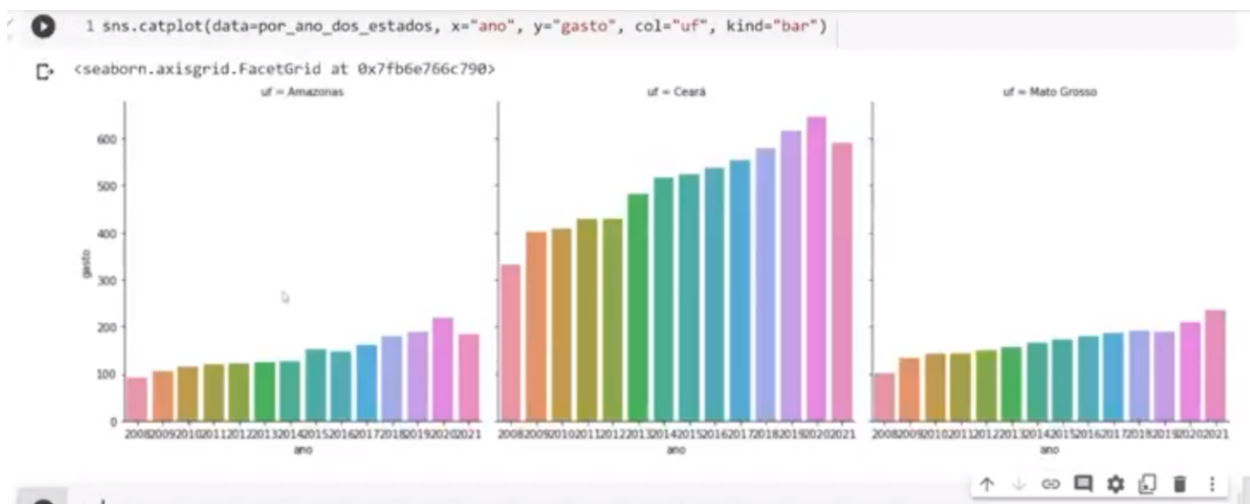
Aqui ele fez um gráfico de scatter pra gente, plotou alguns pontos. Em vez de hue, eu vou querer colunas, então:

```
☐ sns.catplot(data=por_ano_de_todos, x="ano", y="gasto", col="uf")
```



Aqui temos 3 gráficos: Amazonas, Ceará e Mato Grosso.

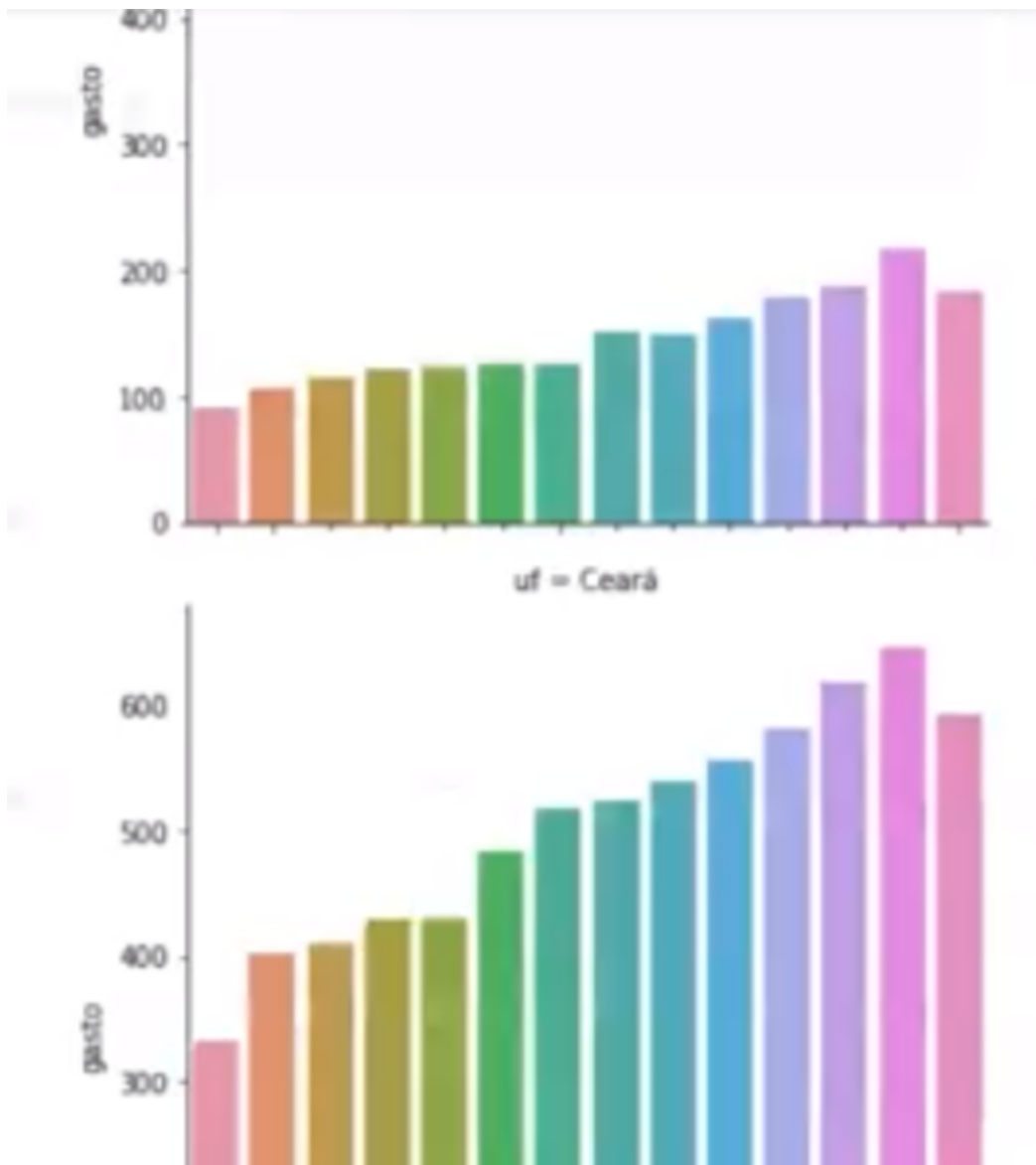
☐ `sns.catplot(data=por_ano_de_todos, x="ano", y="gasto", col="uf", kind="bar")`



Quando olhamos para um gráfico ao lado do outro, conseguimos compará-los. Contudo, conseguimos ver que Ceará gasta mais do que Amazonas e Mato Grosso.

Se mudarmos col para row:

☐ `sns.catplot(data=por_ano_de_todos, x="ano", y="gasto", row="uf", kind="bar")`



Vai plotar na vertical.

Desafio 02: explorar a documentação da Seaborn e Matplotlib. Escolher um gráfico novo e plotar para deixar redondinho.

Desafio 03: refinar o penúltimo gráfico, onde usamos o “col”. Alterar o grid.

Desafio 04: plotar 4 estados, 2 na linha de cima e 2 na linha de baixo.