Aula 04 | PosTech | Trabalhando Datetime e Melt

Anotações sobre a quarta aula da PosTech FIAP ✨✨

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

Temas abordados:

- Noção de modificações dentro de um Dataframe;
- Manipulação de dados;
- Utilização da função melt para melhorar análises base.

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

Um dos maiores desafios sobre dados é como tratar em formato de data efetivamente, porque muitas vezes o campo vem no forma

Por isso, o python tem uma biblioteca chamada Datetime que nos ajuda a trabalhar com datas e tempos diversos. Não é a única bib

Na primeira parte da aula, trabalhamos um jeito de modificar nosso índice da datas, utilizando a função nativa map() e a biblioteca D

Se observarmos a videoaula 01, o objeto que retorna naquela função para_dia() é algo do tipo Date (ano, mês, 1), sendo o último pa

Quando plotamos o gráfico em seguida, é interessante verificar se o formato de data padrão é bonito de ser apresentado.

Outro ponto interessante é a utilização da função melt(), que nos permite criar um novo Dataframe baseado em uma **coluna para m**

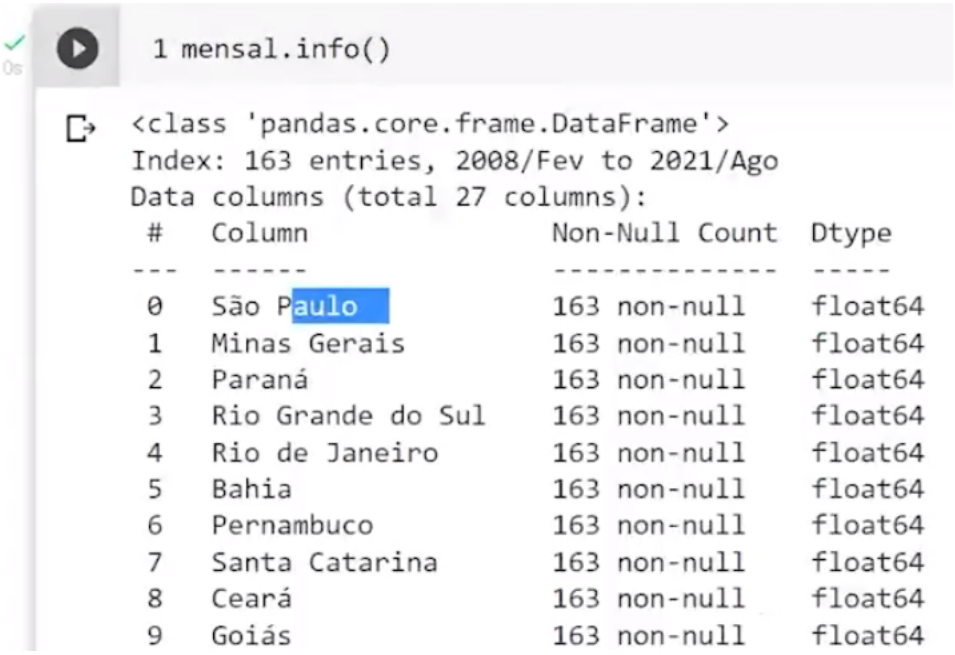
Veja que, no exemplo do vídeo, estávamos trabalhando com uma tabela onde cada coluna era uma unidade da federação diferente,

---

Parte 1 | Trabalhando Datetime e Melt I

É importe saber quais tipos nós estamos usando.

☐ `mensal.info()` → detalha todos os tipos de dados

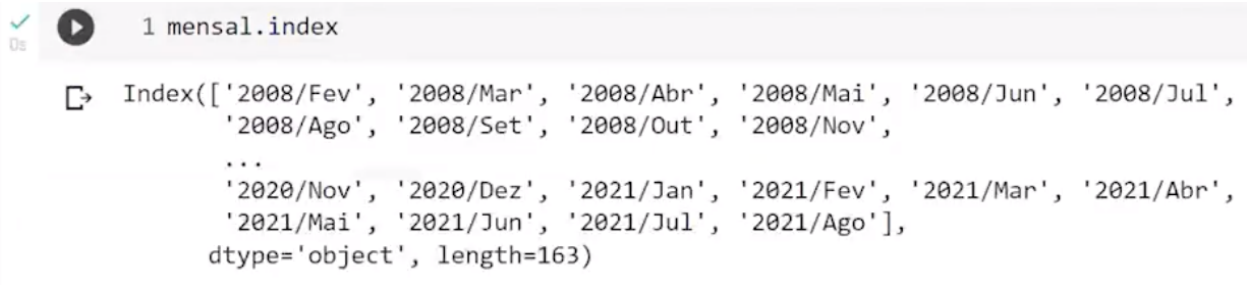


```
1 mensal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 163 entries, 2008/Fev to 2021/Ago
Data columns (total 27 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   São Paulo            163 non-null    float64
 1   Minas Gerais         163 non-null    float64
 2   Paraná               163 non-null    float64
 3   Rio Grande do Sul    163 non-null    float64
 4   Rio de Janeiro       163 non-null    float64
 5   Bahia                163 non-null    float64
 6   Pernambuco           163 non-null    float64
 7   Santa Catarina       163 non-null    float64
 8   Ceará                163 non-null    float64
 9   Goiás                163 non-null    float64
```

O índice (Index) é um texto, é chamado de “object”.

☐ `mensal.index`



```
1 mensal.index
```

```
Index(['2008/Fev', '2008/Mar', '2008/Abr', '2008/Mai', '2008/Jun', '2008/Jul',
       '2008/Ago', '2008/Set', '2008/Out', '2008/Nov',
       ...
       '2020/Nov', '2020/Dez', '2021/Jan', '2021/Fev', '2021/Mar', '2021/Abr',
       '2021/Mai', '2021/Jun', '2021/Jul', '2021/Ago'],
      dtype='object', length=163)
```

É até estranho porque se for ordenar, Agosto vem antes de Fevereiro, então não faz sentido trabalhar com textos. Então, no proces

Alguém teve que fazer essa limpeza. Se não fui eu, alguém ofereceu pra mim. Então temos que aprender a manipular tipos e esse t

Em mensal.index, temos vários valores. Para cada valor, vou mapear esses valores através de uma função “f” que recebe um valor

```
def f(x):
    print(f"valor: {x}")
    mensal.index.map(f)
```

Portanto, para cada um daqueles valores, chamos a função “f”. Map é uma função de mapeamento, é pegar algo e colocar um outro

```
from datetime import date

def para_dia(ano_mes):
    ano = int(ano_mes[:4]) # pegamos os primeiros 4 caracteres que é 2008 e transforma em um int
    return date(ano, mes, 1) # preciso extrair o ano e o mês dentro de ano_mes
```

Então, ano é do tipo int agora. Em python, podemos também fazer:

```
from datetime import date

meses = {
    "Jan" : 1,
    "Fev" : 2,
    "Mar" : 3,
    "Abr" : 4,
    "Mai" : 5
    "Jun" : 6,
    "Jul" : 7,
    "Ago" : 8,
    "Set" : 9,
    "Out" : 10,
    "Nov" : 11,
    "Dez" :  12
}

def para_dia(ano_mes: str):
    ano: int = int(ano_mes[:4]) # pegamos os primeiros 4 caracteres que é 2008 e transforma em um int
    mes_como_string = ano_mes[5:] # a partir do 6º caractere (depois da barra)
    mes: int = meses[mes_como_string]
    return date(ano, mes, 1) # preciso extrair o ano e o mês dentro de ano_mes
```

Outro exemplo clássico é o agrupamento por ano, daí vou precisar tirar os 3 primeiros caracteres e isso é bizarro. O pandas tem um

☐ `mensal.index.map(para_dia)`

```
1 mensal.index.map(para_dia)
2
```

```
Index([2008-02-01, 2008-03-01, 2008-04-01, 2008-05-01, 2008-06-01, 2008-07-01,
       2008-08-01, 2008-09-01, 2008-10-01, 2008-11-01,
       ...
       2020-11-01, 2020-12-01, 2021-01-01, 2021-02-01, 2021-03-01, 2021-04-01,
       2021-05-01, 2021-06-01, 2021-07-01, 2021-08-01],
      dtype='object', length=163)
```

Agora vamos querer sobrescrever:

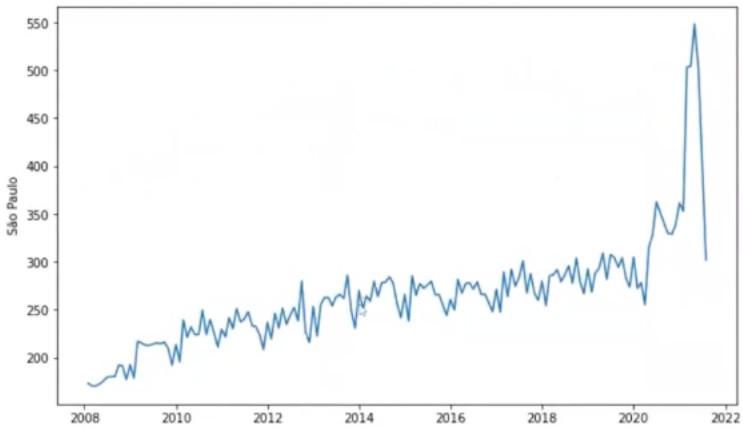
- [ ] `mensal.index = mensal.index.map(para_dia)`
- [ ] `mensal.head()`

```
1 mensal.index = mensal.index.map(para_dia)
2 mensal.head()
```

Unidade da Federação	São Paulo	Minas Gerais	Paraná	Rio Grande do Sul	Rio de Janeiro	Bahia	Pernambuco	Santa Catarina	Ceará	Goiás	Pará	Maranhão	Espírito Santo	Rio Grande do Norte
2008-02-01	173.06	70.30	48.86	45.90	44.13	33.41	26.22	24.41	27.64	18.15	20.13	13.45	10.03	9.11
2008-03-01	170.62	79.12	55.61	52.93	42.91	41.33	30.72	27.48	30.09	20.99	22.15	14.75	12.26	10.13
2008-04-01	170.39	79.05	56.12	51.97	45.32	42.83	30.59	27.81	31.38	21.05	23.44	15.15	12.45	10.96
2008-05-01	172.51	79.27	55.57	51.68	43.95	42.10	31.42	28.35	31.19	19.16	23.86	14.54	11.79	11.03
2008-06-01	175.56	79.63	56.09	53.76	44.12	39.91	28.95	28.99	29.46	20.39	23.50	14.55	11.89	10.45

Se a gente quiser plotar aquele mesmo plot ali em cima:

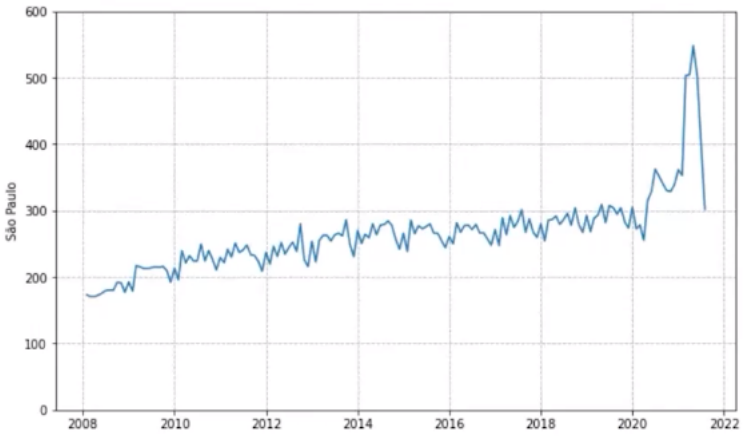
`plt.figure(figsize=(10,6))` # tamanho da figura nós definimos
`axis = sns.lineplot(data=mensal, x=mensal.index, y="São Paulo")`
`#plt.xticks(rotation=30)`
`#plt.ylim(0, 600)`
`#axis.xaxis.set_major_locator(ticker.IndexLocator(base=12, offset=11))`
`#plt.grid(linestyle="—")`
`plt.show()`



Nesse gráfico padrão, começou em 2008 e terminou em 2022. Ele já entendeu que é data, dia, mês e ano. Começou em Janeiro de

Portanto, não precisamos mais rotacionar nada pois já está bem espaçado. O limite de y precisa, é uma questão nossa. O grid preci

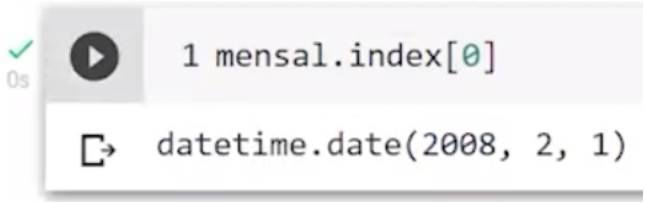
```
plt.figure(figsize=(10,6)) # tamanho da figura nós definimos
axis = sns.lineplot(data=mensal, x=mensal.index, y="São Paulo")
#plt.xticks(rotation=30)
#plt.ylim(0, 600)
#axis.xaxis.set_major_locator(ticker.IndexLocator(base=12, offset=11))
#plt.grid(linestyle="-")
plt.show()
```



O gráfico, portanto, fica assim. 😊

Só pra confirmar…

☐ `mensal.index[0]`



Aqui podemos ver que mensal é do tipo "**datetime**".

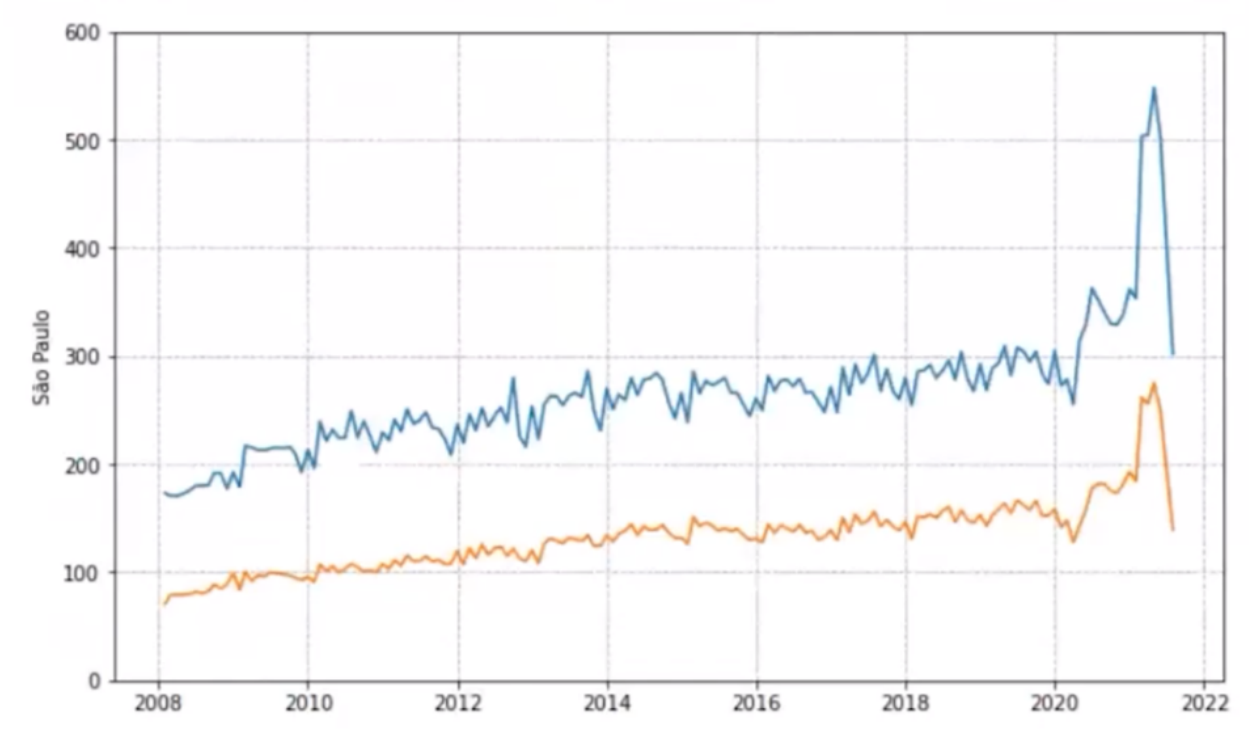
Parte 2 | Trabalhando Datetime e Melt II

Vamos fazer uma mudança naquilo em que estávamos trabalhando.

Quero esse mesmo gráfico que é do estado de São Paulo e quero plotar agora mais dois estados, ou seja, plotar 2x.

```
plt.figure(figsize=(10,6)) # tamanho da figura nós definimos
axis = sns.lineplot(data=mensal, x=mensal.index, y="São Paulo")
axis = sns.lineplot(data=mensal, x=mensal.index, y="Minas Gerais")
```

```
#plt.ylim(0, 600)
#plt.grid(linestyle="—")
plt.show()
```



As duas linhas são razoavelmente parecidas. Porém, não sabemos qual estado é qual. O eixo y também está péssimo, o que é SP?

A tabela foi criada de uma forma que os dados não se repetem. Para quem vem da área de banco de dados, a gente fala bastante d

A gente poderia ter uma outra tabela, uma outra forma de tabela em que se repetem dados. Em vez de ter 27 colunas, eu tenho ape

MES	VALOR	ESTADO
2008-02	173.06	São Paulo
2008-03	170.62	São Paulo
...		
2008-02	70.30	Minas Gerais
2008-03	79.12	Minas Gerais

Nessa tabela, nós temos os mesmos dados que na tabela anterior, está cheio de repetição.

Antes, a coluna / variável São Paulo era contínua, agora não. Agora, a variável “Estado” é categórica, ela assume 26 valores dife

Quando trabalhamos com uma tabela categórica, a gente consegue falar pro Seaborn: “Plota pra mim duas linhas, de acordo coı

Portanto, vou pegar “mensal”, remover o índice (reset_index):

- [ ] `mensal.reset_index()` # vai pegar nosso índice e transformar em uma coluna chamada “índex”.

```
1 mensal.reset_index()
```

	Unidade da Federação	index	São Paulo	Minas Gerais	Paraná	Rio Grande do Sul	Rio de Janeiro	Bahia	Pernambuco	Santa Catarina	Ceará	Goiás	Pará	Maranhão	Espírito Santo
0		2008-02-01	173.06	70.30	48.86	45.90	44.13	33.41	26.22	24.41	27.64	18.15	20.13	13.45	10.03
1		2008-03-01	170.62	79.12	55.61	52.93	42.91	41.33	30.72	27.48	30.09	20.99	22.15	14.75	12.26
2		2008-04-01	170.39	79.05	56.12	51.97	45.32	42.83	30.59	27.81	31.38	21.05	23.44	15.15	12.45
3		2008-05-01	172.51	79.27	55.57	51.68	43.95	42.10	31.42	28.35	31.19	19.16	23.86	14.54	11.79

Vamos “derreter” essa tabela em 3 colunas:

- [ ] `mensal_aberto = mensal.reset_index().melt(id_vars=["index"], value_vars=mensal.columns)` # “quais são as colunas que vão ficar fixas?” →
- [ ]
- [ ] `mensal_aberto.head()`

```
1 mensal_aberto = mensal.reset_index().melt(id_vars=["index"], value_vars=mensal
2 mensal_aberto.head()
```

	index	Unidade da Federação	value
0	2008-02-01	São Paulo	173.06
1	2008-03-01	São Paulo	170.62
2	2008-04-01	São Paulo	170.39
3	2008-05-01	São Paulo	172.51
4	2008-06-01	São Paulo	175.56

- [ ] `mensal_aberto = mensal.reset_index().melt(id_vars=["index"], value_vars=mensal.columns)`
- [ ] `mensal_aberto.columns = [“mes”, “uf”, “gasto”]`
- [ ] `mensal_aberto.head()`

```
[138] 1 mensal_aberto = mensal.reset_index().melt(id_vars=["index"],
      2 mensal_aberto.columns = ["mes", "uf", "gasto"]
      3 mensal_aberto.head()
```

	mes	uf	gasto
0	2008-02-01	São Paulo	173.06
1	2008-03-01	São Paulo	170.62
2	2008-04-01	São Paulo	170.39
3	2008-05-01	São Paulo	172.51
4	2008-06-01	São Paulo	175.56

“Por que quando a gente derrete a tabela, é melhor?”

Desafio 01: como criar uma tabela equivalente ao ‘mensal’ baseada no ‘mensal_abe

Agora, vamos plotar. Como fazer?

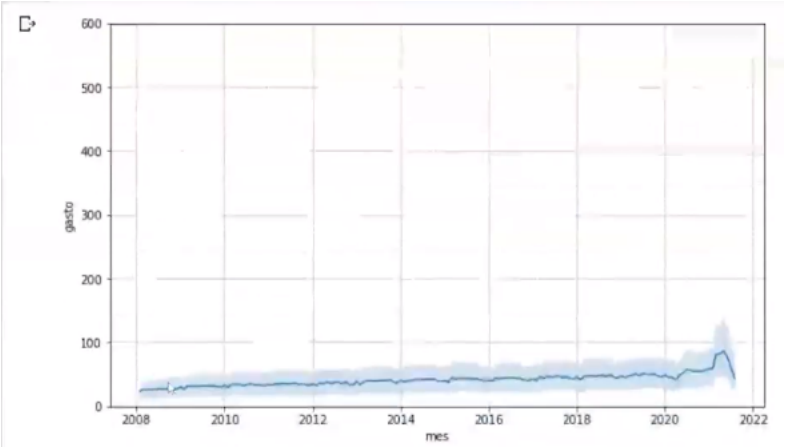
`plt.figure(figsize=(10,6))` # tamanho da figura nós definimos

`axis = sns.lineplot(data=mensal_aberto, x="mes", y=”gasto”)` # x é a coluna mês

`#plt.ylim(0, 600)`

`#plt.grid(linestyle=”–”)`

`plt.show()`



O que aconteceu? Para cada um dos meses, a gente tem 26 valores diferentes.

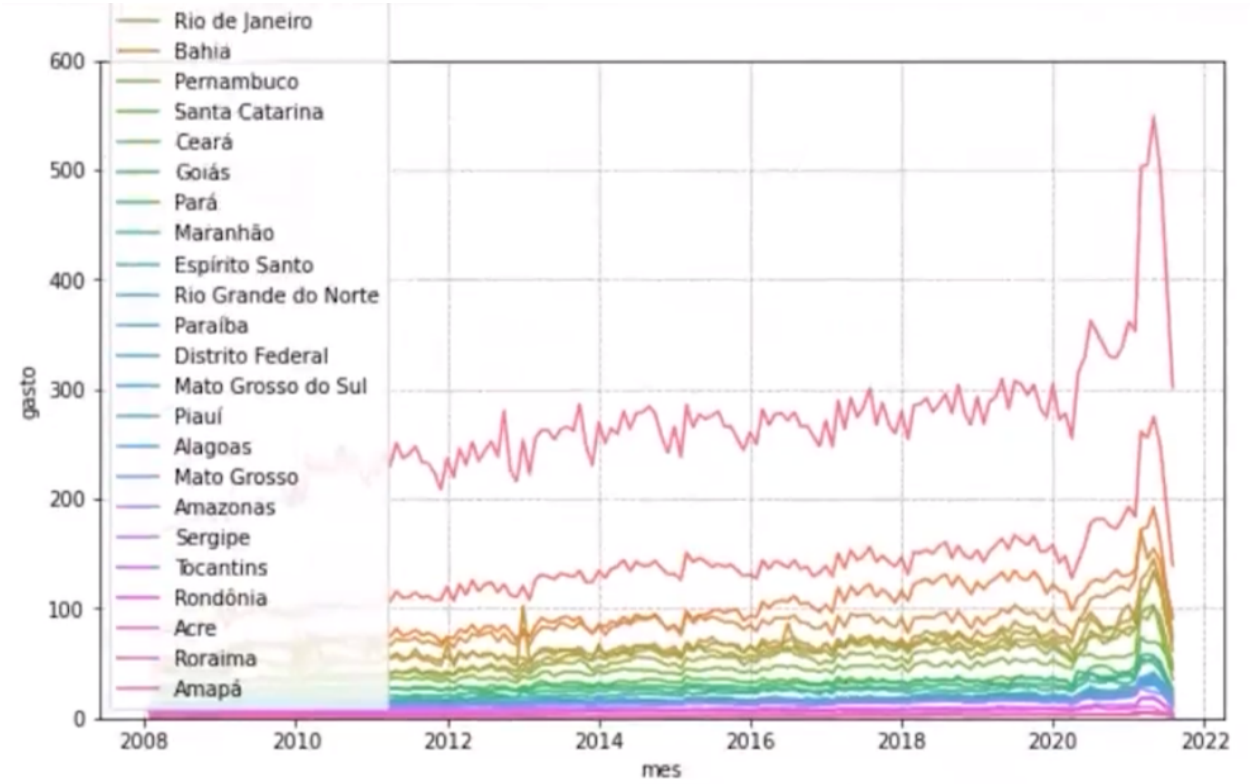
Ali tem uma linha e uma barrinha (que é intervalo de confiança estatístico em cima dos 26 valores que estão sendo lidos). Tem nada

Agora, eu quero é uma linha com uma tonalidade, de cada uma das categorias da coluna “uf”.

```
plt.figure(figsize=(10,6))
```
tamanho da figura nós definimos

```
axis = sns.lineplot(data=mensal_aberto, x="mes", y=”gasto”, hue="uf")
```
hue para colocar tonalidade

```
#plt.ylim(0, 600)
#plt.grid(linestyle=”–”)
plt.show()
```



Errata: Não são 26 valores, são 27 contando com o Distrito Federal.

Desafio 02: plotar o mesmo gráfico, porém com somente 2 estados que escolherei.

Desafio 03: escolher uma paleta de cores adequada, de tamanho adequado (pelo me

Sugestão:

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

“uf” é categórico, uma paleta cujos tons são bem distintos talvez seja o ideal. Escolher uma paleta de cores com pelo menos 27 core

Desafio 04: formatar legenda e eixos.

Desafio 05: como configurar o grid com o Seaborn?