Aula 04 | PosTech | Análise Exploratória de Dados

Anotações sobre a quarta aula da PosTech FIAP ✨✨

https://on.fiap.com.br/mod/conteudoshtml/view.php?id=307787&c=8729&sesskey=AlUOg2UtXh

Temas abordados:

- Importância de uma boa representação gráfica, para não passar uma imagem errada sobre os dados apresentados ou uma interpretação dicotômica;
- Gráficos com menos poluição visual, nosso cérebro tende a aceitar melhor informações otimizadas;
- Olhar e intepretar um gráfico de linhas realizando comparações.

Dependências:

- Baixar o arquivo zip:

https://github.com/alura-tech/pos-datascience-analise-e-exploracao-de-dados/tree/aula1

- Documentação Pandas:

https://pandas.pydata.org/

- Documentação Matplotlib:

https://matplotlib.org/

- Google Colab:

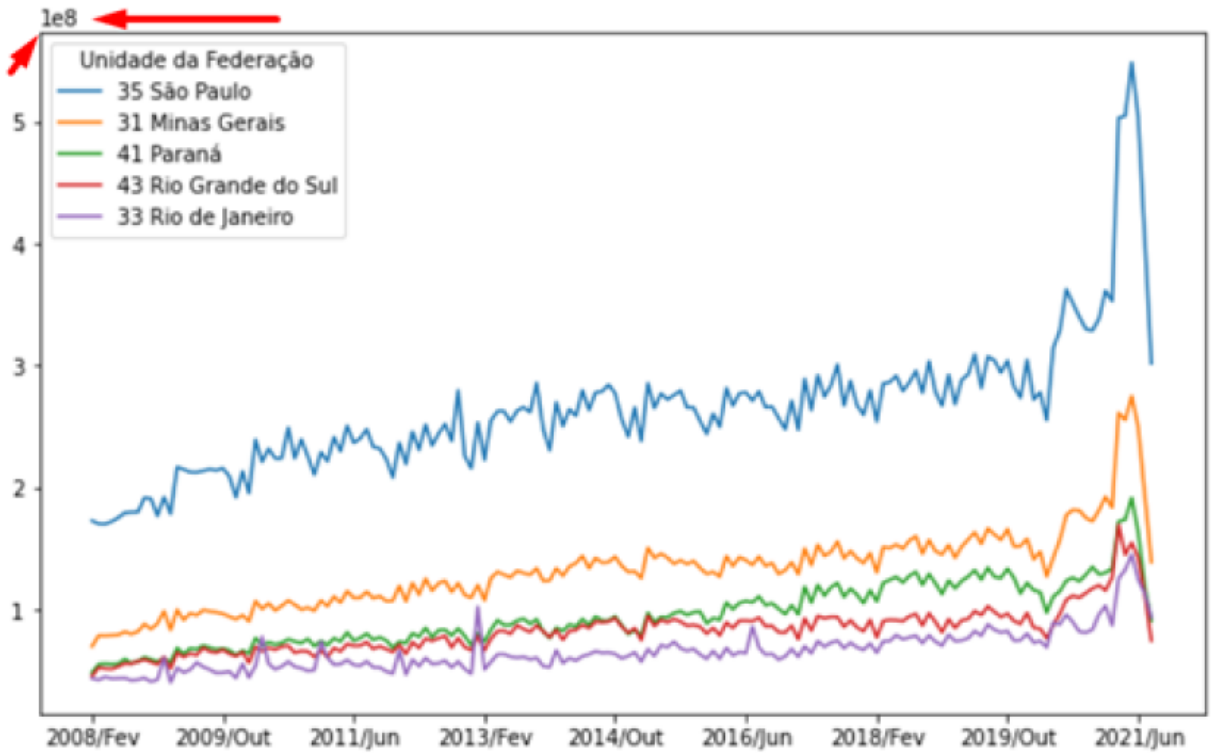
https://colab.research.google.com/

- TabNet:

https://datasus.saude.gov.br/informacoes-de-saude-tabnet/

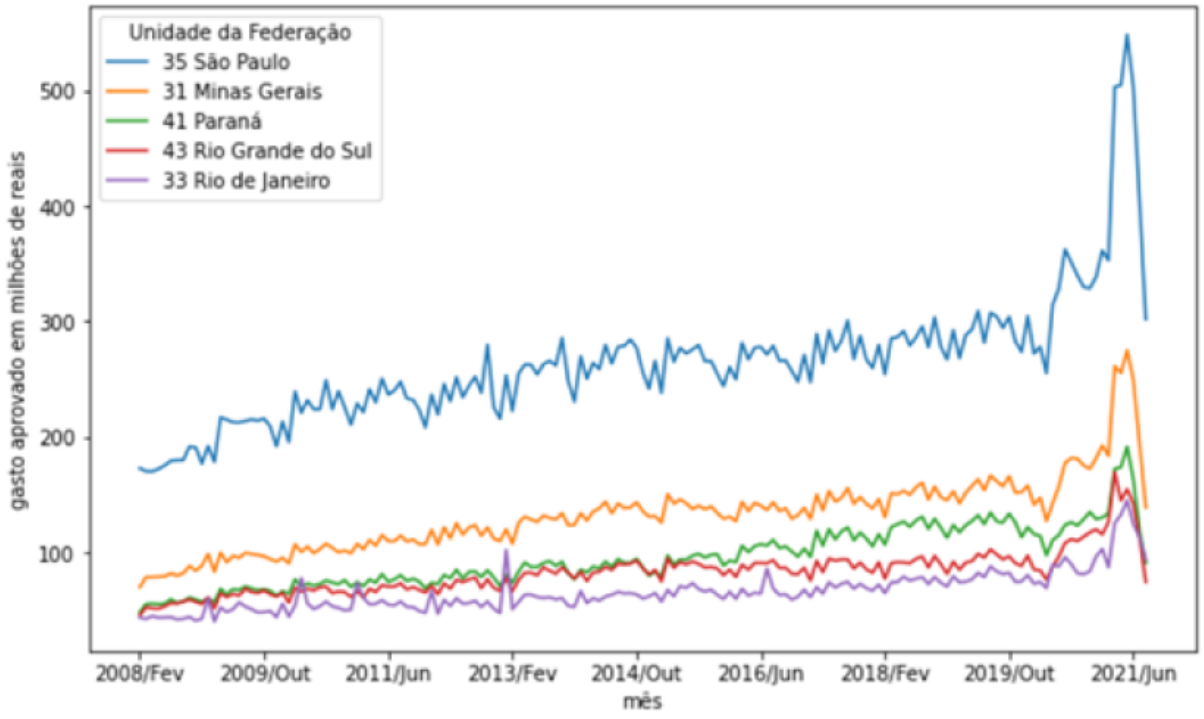
Aula 4 - Análise de gráficos e criação de hipóteses

Imagine este primeiro gráfico (**não ajustado em y**) que apareceu anteriormente na aula 03:



As setas vermelhas mostram a escala em que o eixo y está, então, se você não domina matemática elementar e suas notações, ficará difícil de interpretar a situação. Essa escala “1e8” significa “10 elevado a 8” ou 100 milhões.

De maneira visual, isso não é nada bom. Em contrapartida, existe uma solução (que é apresentada na nossa aula), onde temos uma adição importante no gráfico:



Aqui, o gráfico está **ajustado em y**.

Adicionando uma simples legenda e ajustando a escala, a visualização se tornou mais fácil de compreender, deixando a mensagem de que o eixo vertical quer passar simplesmente ajustando os labels e “normalizando” os valores, e para isso, apenas **duas células** foram necessárias.

- [ ] `ordenados_por_total = ordenados_por_total / 1_000_000`

- [ ] `axis = ordenados_por_total.head(5).T.plot(figsize=(10,6))`
- [ ] `axis.set_ylabel("gasto aprovado em milhões de reais")` # uso de set_ylabel
- [ ] `axis.set_xlabel("mês")` # uso de set_xlabel

Nota-se a importância de uma simples mudança, principalmente se esse tipo de gráfico for apresentado por alguém que não seja técnico, mas que precisa entender a mensagem descrita ali.

Outro ponto é o eixo “x”, que ao ser alongado ou encurtado pode passar mensagens visuais diferentes, mas que pode variar o tipo de problema em que você vai atuar, principalmente quando o assunto é a tal “série temporal”.

Uma maneira interessante de ver o mesmo gráfico em tamanhos diferentes é fazer o mesmo que é apresentado nas figuras 3 - “Gráfico ajustado em x“ (1º), figura 4 - “Gráfico ajustado em y” (2º), figura 5 - “Gráfico ajustado em x” (3º).

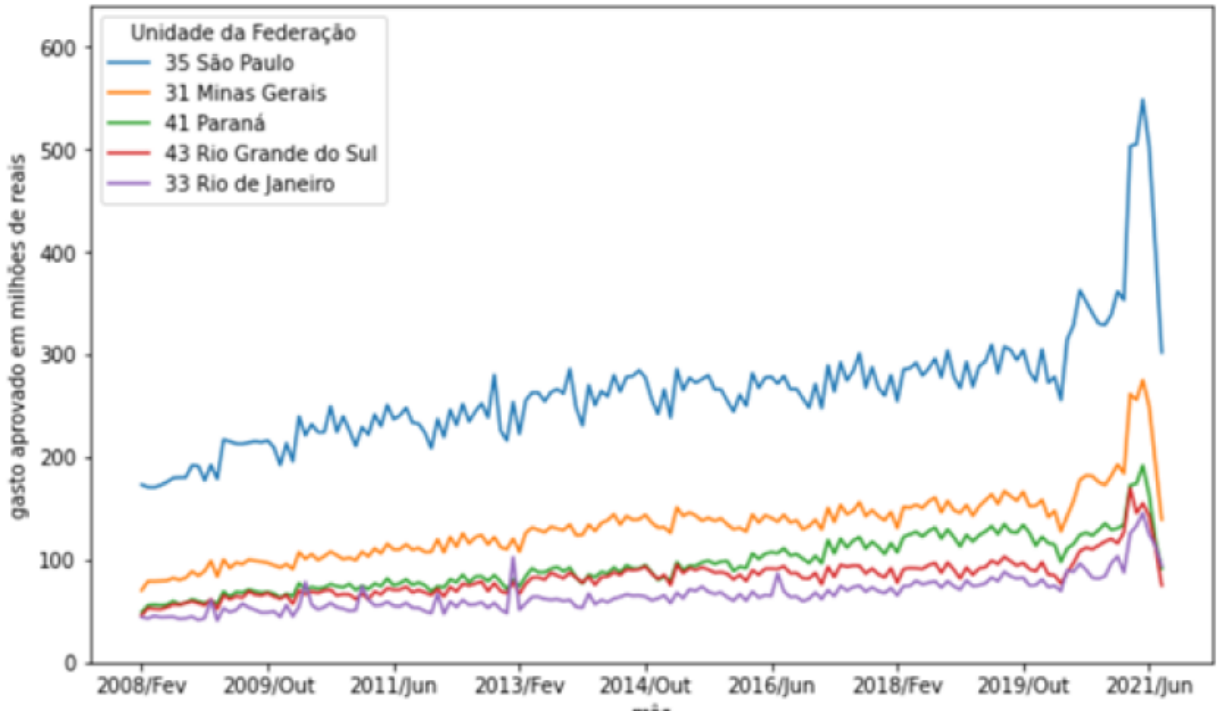


figura 3 - “Gráfico ajustado em x“ (1º)

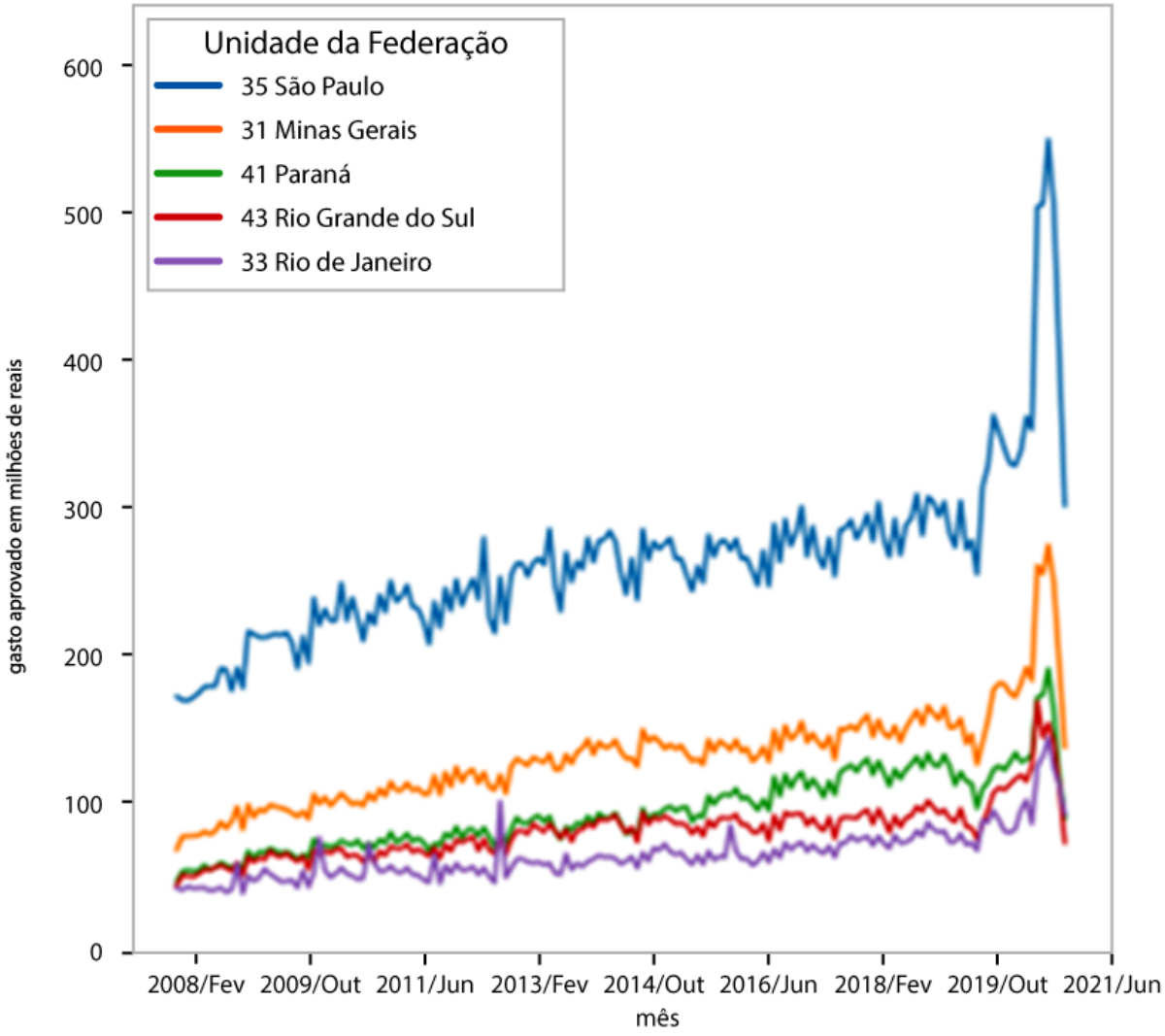


figura 4 - “Gráfico ajustado em y” (2º)

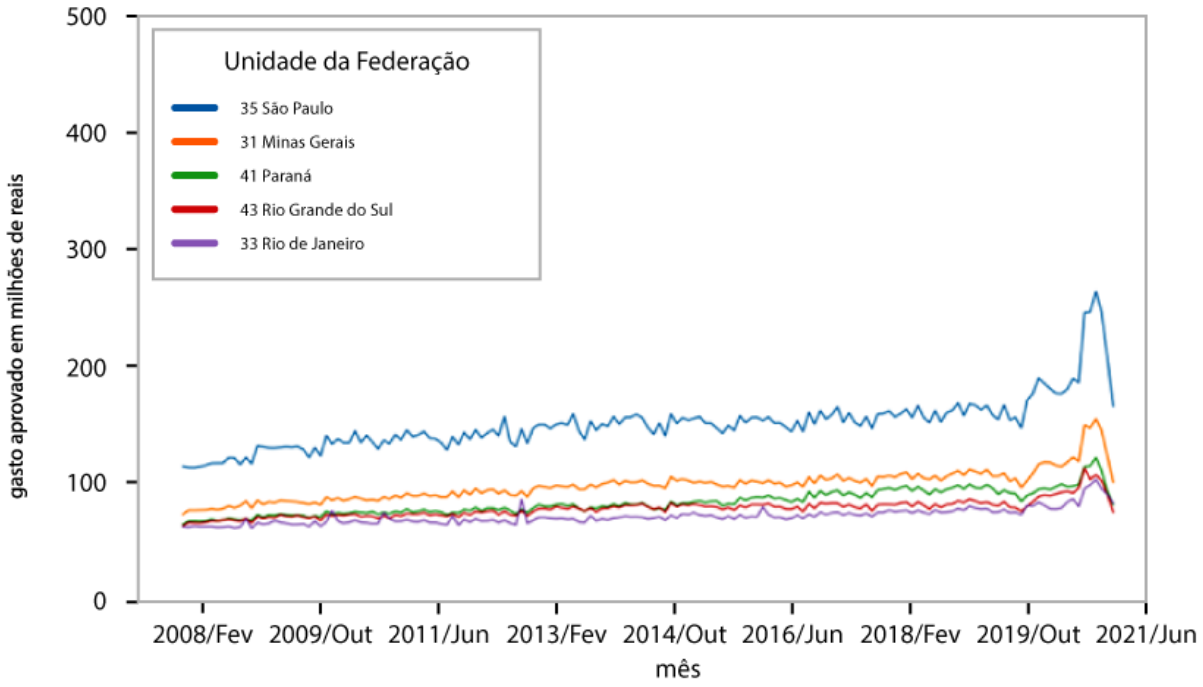


figura 5 - "Gráfico ajustado em x" (3º)

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html

Ajustar as visualziações no Matplotlib é importante, porque permite que você controle a aparência e a estética dos gráficos produzidos.

Isso inclui aspectos como títulos, legendas, rótulos de eixos, escala, cores e muito mais. Ajustar as visualizações ajuda a tornar os gráficos mais claros e fáceis de entender para o público-alvo, o que pode ser importante em contextos científicos ou de negócios.

Além disso, o ajuste adequado das visualizações pode destacar informações relevantes e tornar os dados mais atraentes e impactantes.

Recursos mais comumente usados no Matplotlib para ajustar visualizações:

- Títulos e legendas: fornecer contexto e explicação para o gráfico;
- Rótulos de eixos: identificar as escalas de valores no gráfico;
- Escala: controlar a escala dos valores exibidos nos eixos;
- Cores: destacar informações específicas ou tornar o gráfico mais atraente;
- Marcadores: identificar pontos específicos no gráfico;
- Grade: tornar o gráfico mais claro e fácil de ler;
- Estilos de linha: controlar o tipo de linha usado no gráfico, por exemplo, contínua, tracejada, pontilhada, etc.

O Matplotlib é uma biblioteca poderosa e versátil para a visualização de dados, mas pode ter limitações em algumas situações. Exemplo:

Complexidade: O Matplotlib pode ser complexo de ser usado para criar gráficos mais avançados ou interativos.

Visualizações limitadas: O matplotlib tem suporte para uma ampla gama de tipos de gráficos, mas pode não ter suporte para algum tipo de visualização específica que você precise.

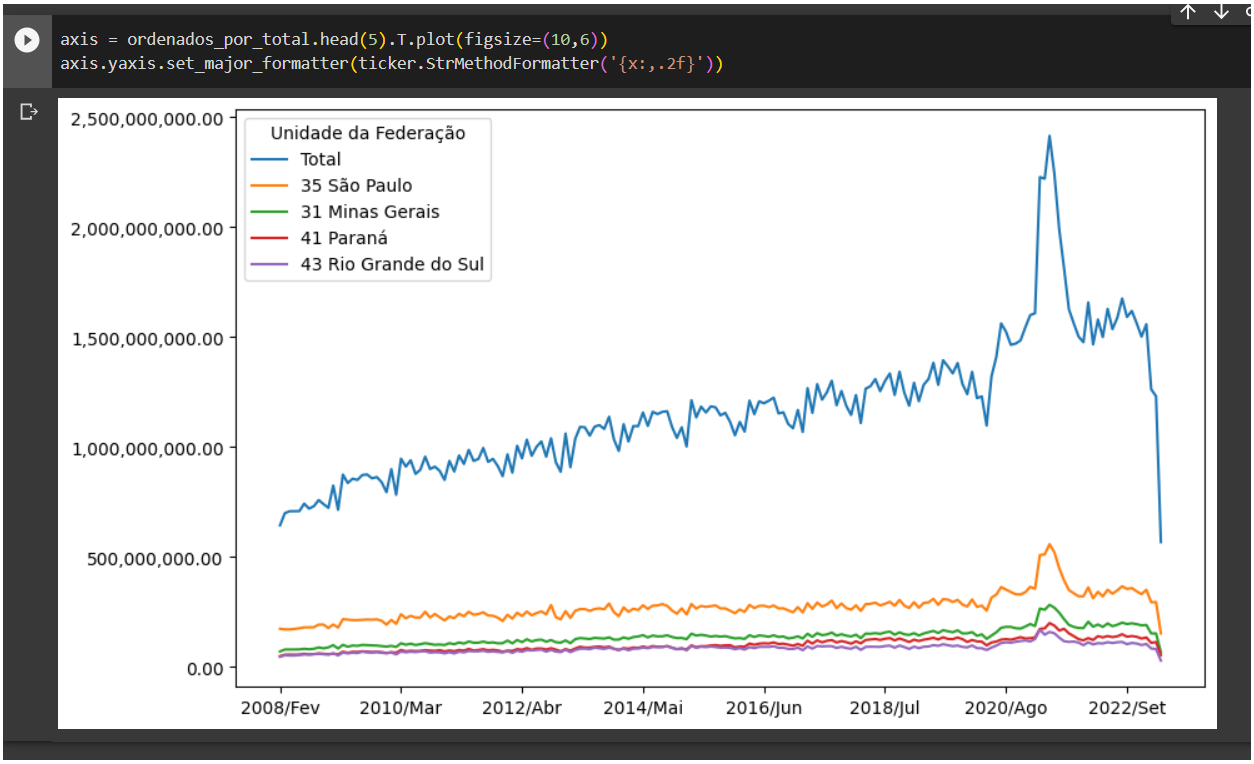
Gráficos interativos: O matplotlib tem recursos limitados para criar gráficos interativos, que permitem ao usuário manipular o gráfico com o mouse ou outros dispositivos de entrada.

Nessas 3 situações, pode ser necessário usar outras bibliotecas adicionais, como o Bokeh ou o Plotly, que oferecem **recursos adicionais para criar gráficos interativos e mais avançados**. No entanto, o Matplotlib é uma escolha sólida para a maioria das

necessidades de visualização de dados e é frequentemente usado como uma base para outras bibliotecas gráficas.

1 | ANÁLISE DE GRÁFICOS E CRIAÇÃO DE HIPÓTESE

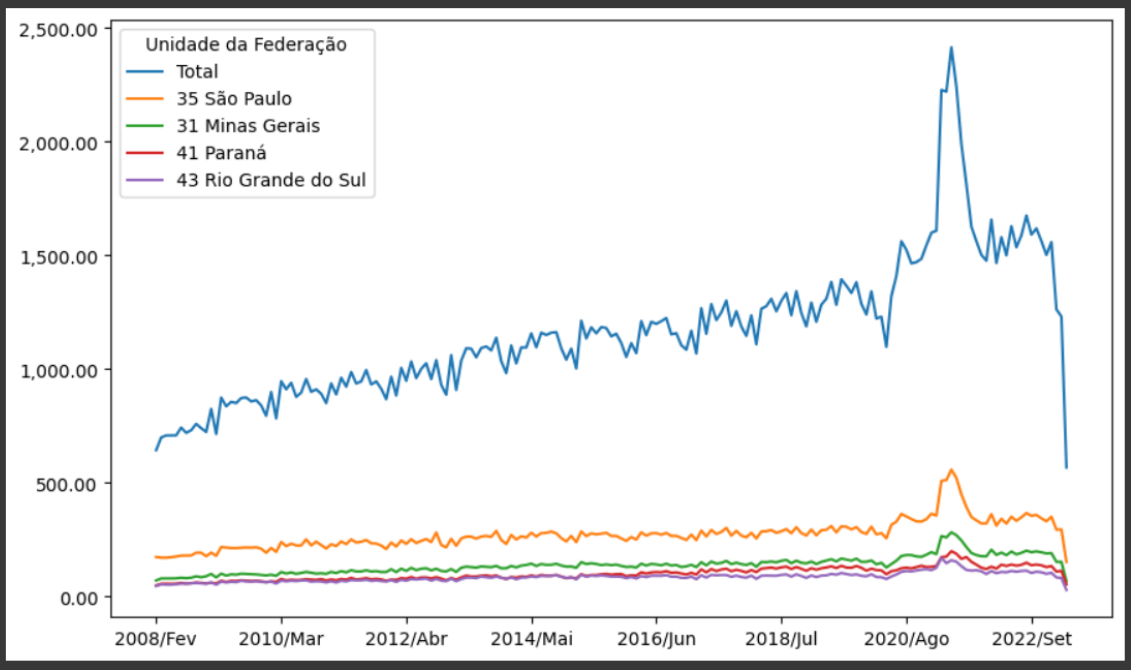
- [ ] `axis = ordenados_por_total.head(5).T.plot(figsize=(10,6))`
- [ ] `axis.yaxis.set_major_formatter(ticker.StrMethodFormatter("{x:,.2f}"))`



Porém, o número ainda está muito grande. Tem uma outra forma de trabalhar isso:

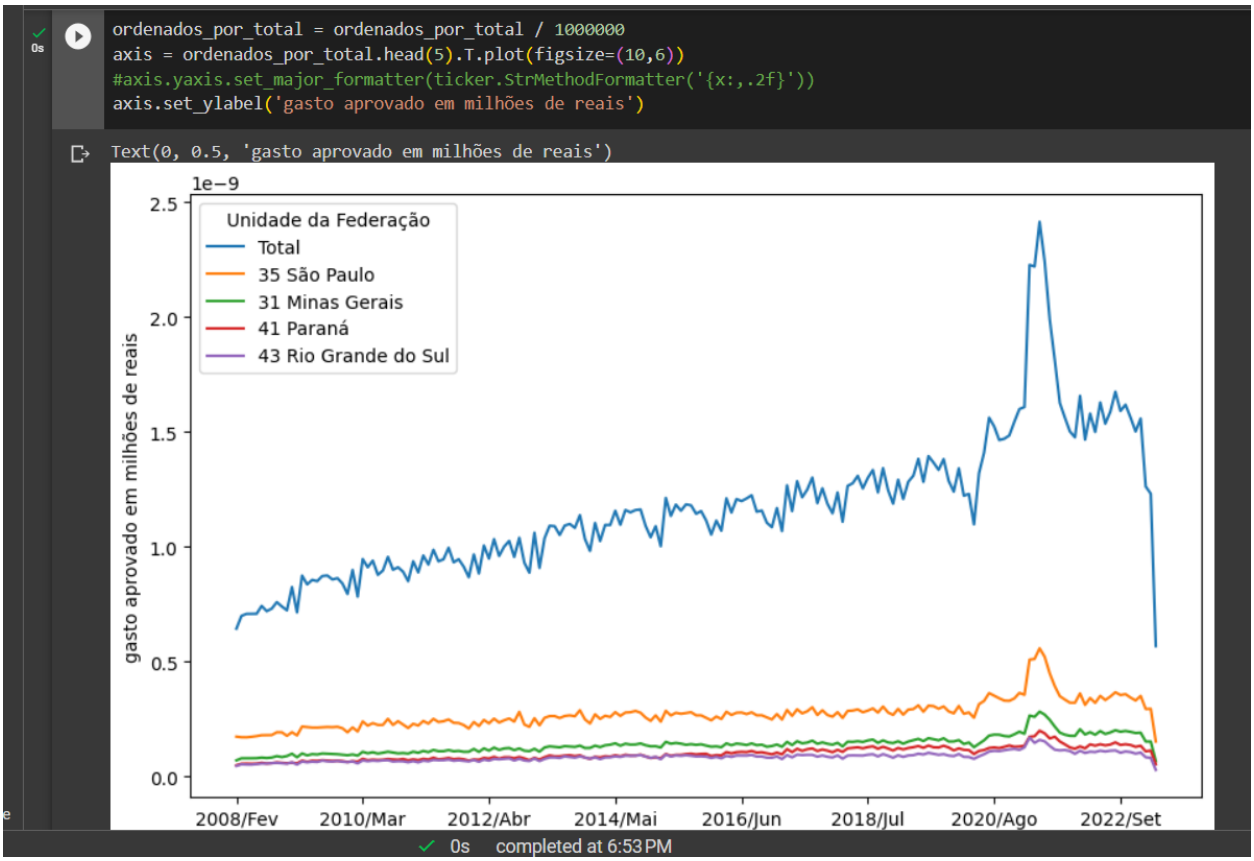
- [ ] `ordenados_por_total = ordenados_por_total / 1000000`
- [ ] `axis = ordenados_por_total.head(5).T.plot(figsize=(10,6))`
- [ ] `axis.yaxis.set_major_formatter(ticker.StrMethodFormatter("{x:,.2f}"))`

```
ordenados_por_total = ordenados_por_total / 1000000
axis = ordenados_por_total.head(5).T.plot(figsize=(10,6))
axis.yaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.2f}'))
```



O problema é que agora não sabemos o que são esses 500.00, pode ser qualquer unidade de medida.

- [ ] `axis.set_ylabel(”gasto aprovado em milhões de reais”)`




```
axis = ordenados_por_total.head(5).T.plot(figsize=(10,6))
axis.set_ylabel('gasto aprovado em milhões de reais')
axis.set_xlabel('mês')
```

Agora, vamos explorar um pouco esse conjunto de dados visualmente.

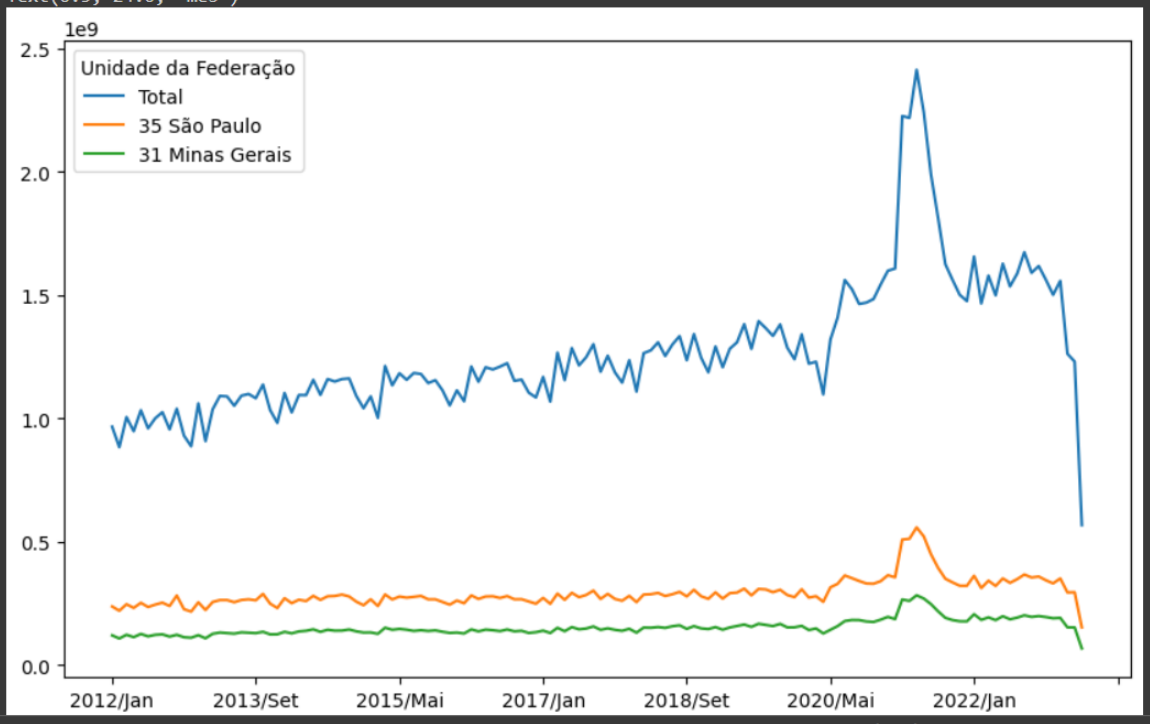
Queremos, agora, escolher apenas alguns meses. Vou jogar fora alguns deles e vou pegar apenas algumas colunas.

- [ ] `ordenados_por_total.columns[47:]` # da coluna 47 em diante

- [ ] `meses_filtrados = ordenados_por_total.columns[47:]`
- [ ] `ordenados_por_total[meses_filtrados].head(3).T.plot(figsize=(10,6))` # vou pegar apenas os 3 primeiros estados
- [ ] `axis.set_ylabel(”gasto aprovado em milhões de reais”)`
- [ ] `axis.set_xlabel(”mês”)`

```
meses_filtrados = ordenados_por_total.columns[47:]
ordenados_por_total[meses_filtrados].head(3).T.plot(figsize=(10,6))
axis.set_ylabel('gasto aprovado em milhões de reais')
axis.set_xlabel('mês')
```

```
Text(0.5, 24.0, 'mês')
```

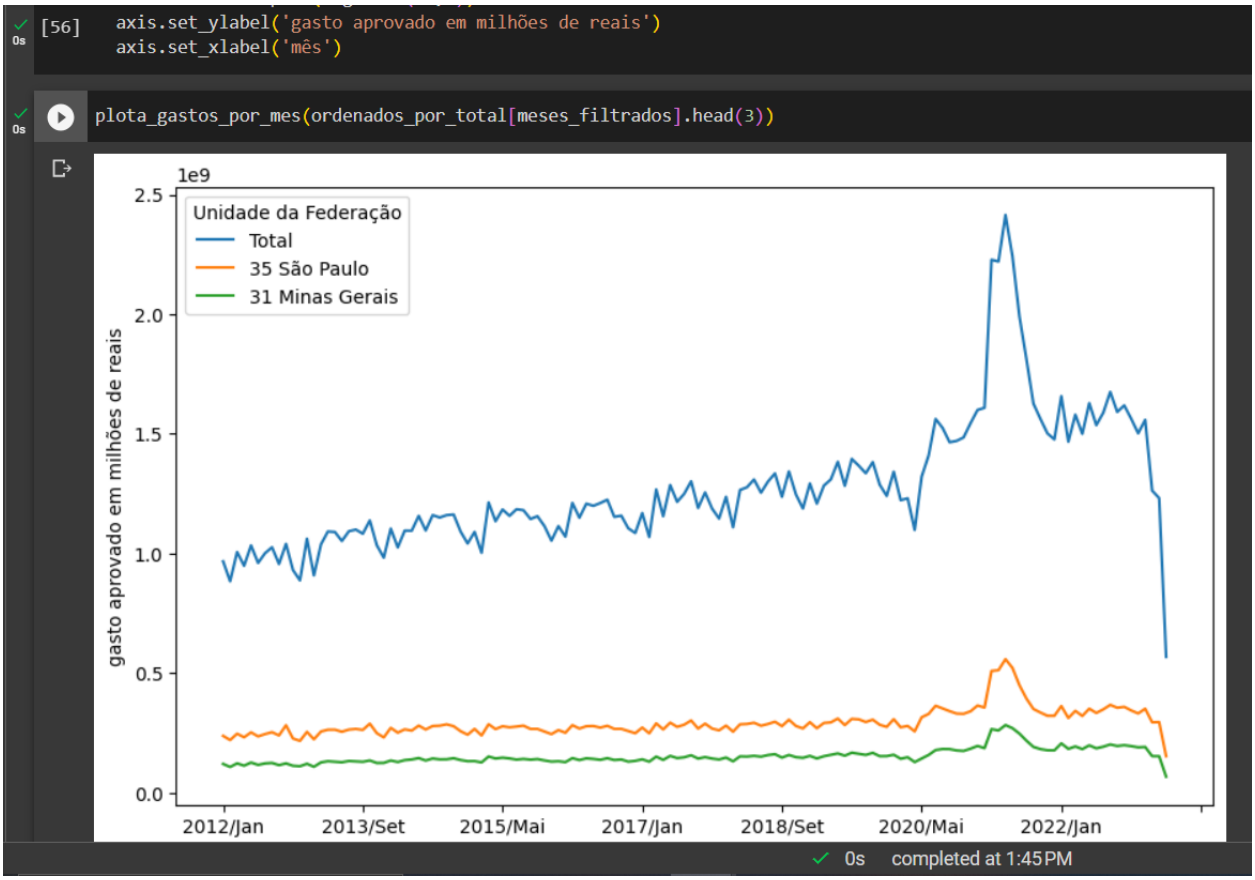


0s completed at 1:39 PM

Agora, posso chamar uma função em python para fazer a plotagem dos dados:

```
def plota_gastos_por_mes(dados):
    axis = dados.T.plot(figsize=(10,6))
    axis.set_ylabel(”gasto aprovado em milhões de reais”)
    axis.set_xlabel(”mês”)
```

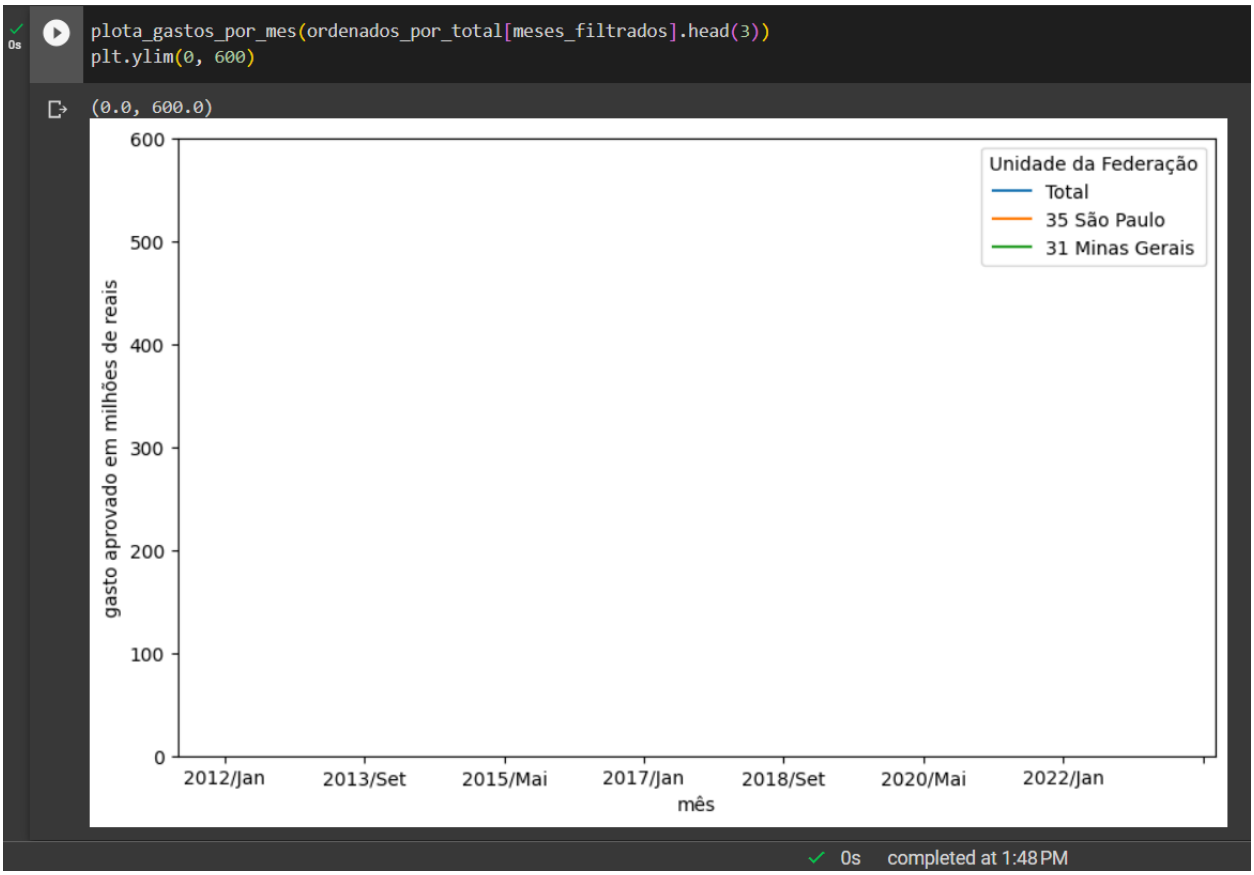
☐ `plota_gastos_por_mes(ordenados_por_total[meses_filtrados].head(3))` # adjust the bottom leaving top



Agora, podemos “settar” o limite do nosso gráfico.

- [ ] `plota_gastos_por_mes(ordenados_por_total[meses_filtrados].head(3))` # adjust the bottom leaving top

- [ ] `plt.ylim(0, 600)`

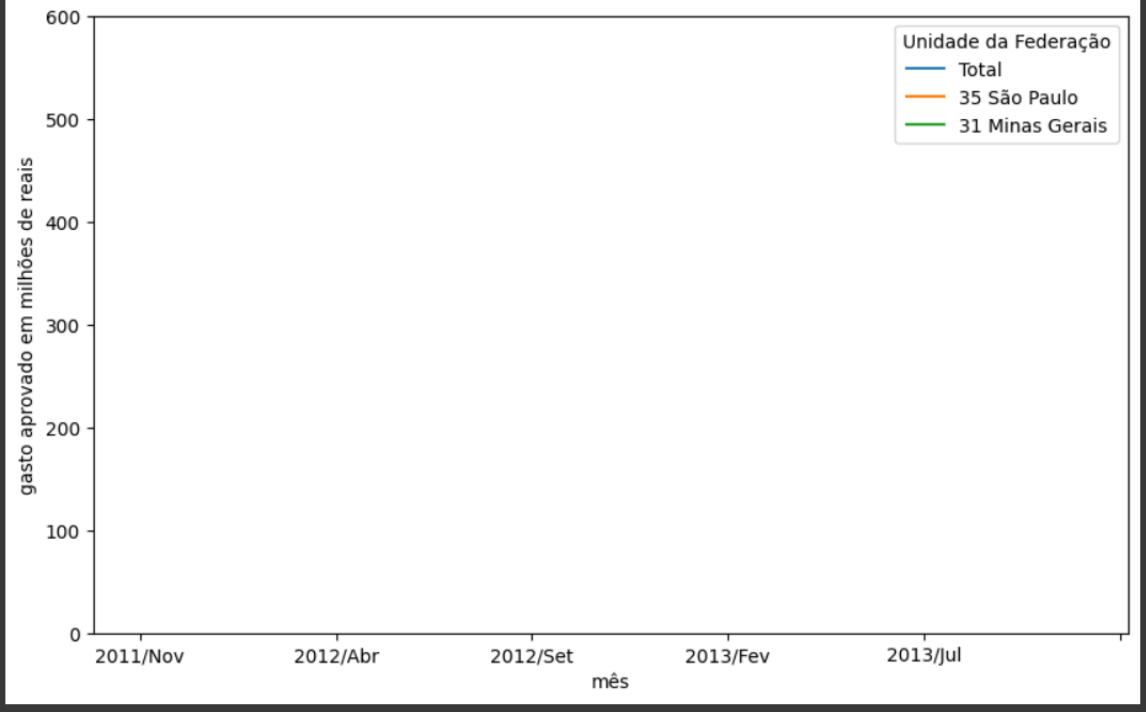


- [ ] `plota_gastos_por_mes(ordenados_por_total[ordenados_por_total.columns[45:70]].head(3))` # adjust the bottom leaving top

- [ ] `plt.ylim(0, 600)`

```
plota_gastos_por_mes(ordenados_por_total[ordenados_por_total.columns[45:70]].head(3))
plt.ylim(0, 600)
```

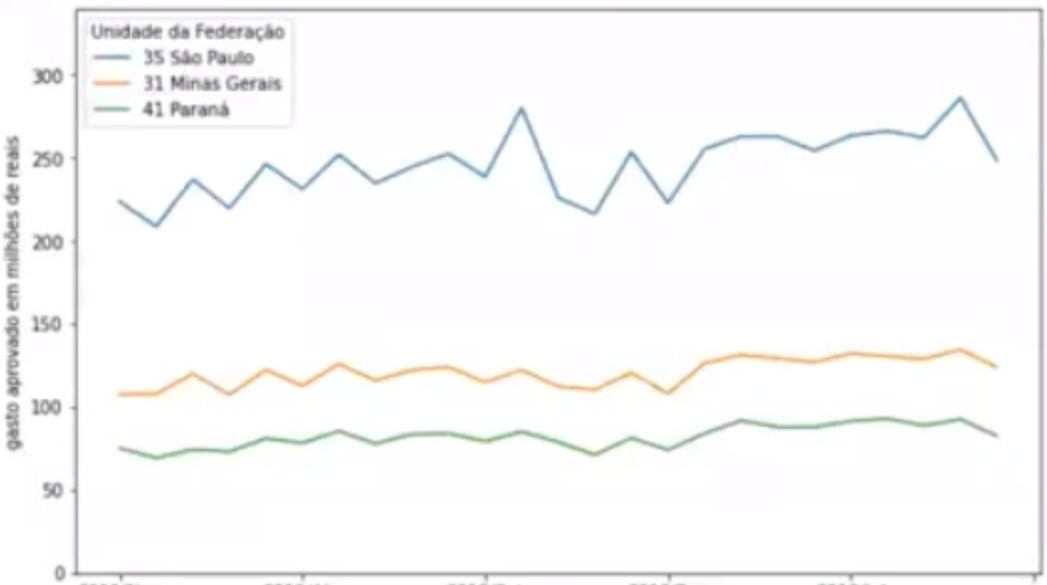
```
(0.0, 600.0)
```



0s completed at 1:50 PM

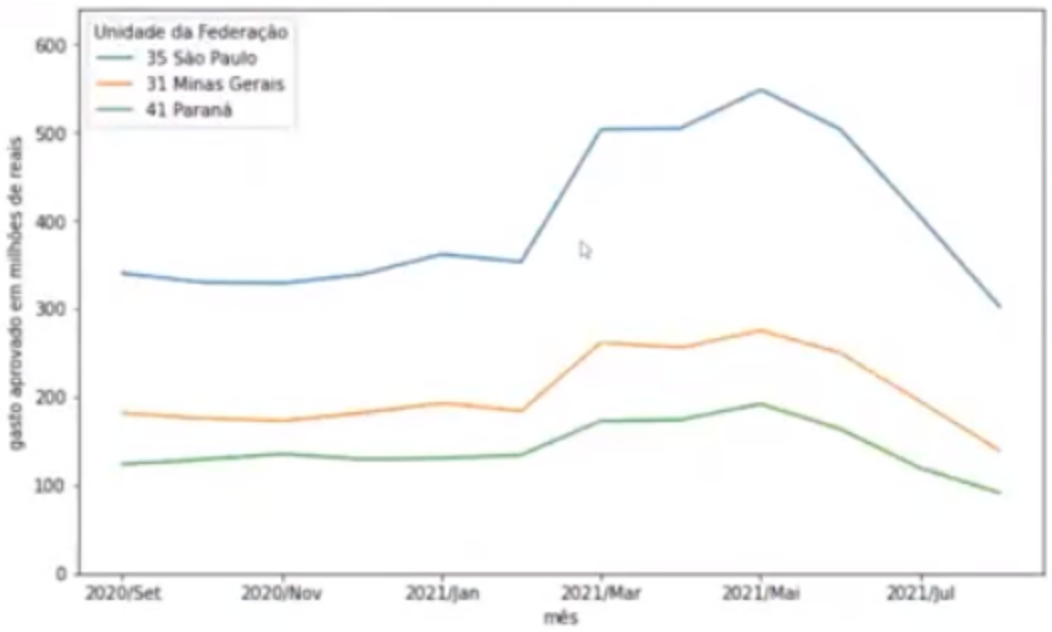
```
1 plota_gastos_por_mes(ordenados_por_total[ordenados_por_total.columns[45:70]].head
2 plt.ylim(0, 340)
```

```
(0.0, 340.0)
```



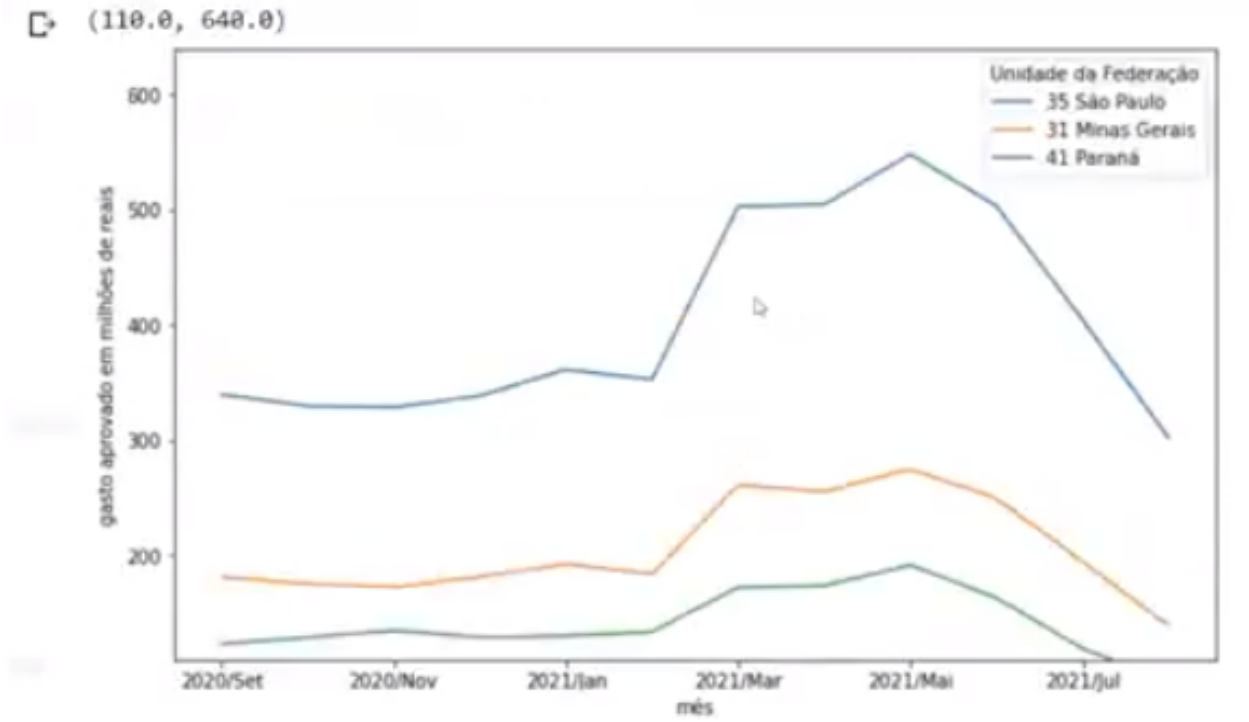
O problema de brincar com limite é que, pegando os últimos 12 meses, em vez de pegar [45:70], vou começar com a coluna -12 até o fim: [-12:]. Vou colocar um limite também de 640:

- [ ] `plota_gastos_por_mes(ordenados_por_total[ordenados_por_total.columns[-12:]].head(3))` # adjust the bottom leaving top
- [ ] `plt.ylim(0, 640)`



Se eu colocar um limite em y de 80 à 640…

- [ ] `plota_gastos_por_mes(ordenados_por_total[ordenados_por_total.columns[-12:]].head(3))` # adjust the bottom leaving top
- [ ] `plt.ylim(80, 640)`



É bem sutil a diferença, mas parece que São Paulo gasta muito muito muito mais que Paraná, com relação ao gráfico acima. Eu estou mentindo através de um gráfico… Sem querer, eu acabei passando uma mensagem errada com relação aos dados que estão contidos no gráfico. A informação é criada pelos dados, mas a gente pode transformar ela.

Portanto, a prática correta é o eixo y começar do zero, principalmente quando estamos fazendo comparações.

Quando você vê um retrato parcial, por exemplo, de um ano até outro ano específico, se pergunte o motivo de ser assim. Por que escolheu esse recorte temporal? Por que escolheu justamente esses 3 estados? Tudo tem um motivo. De onde vem esse contexto?

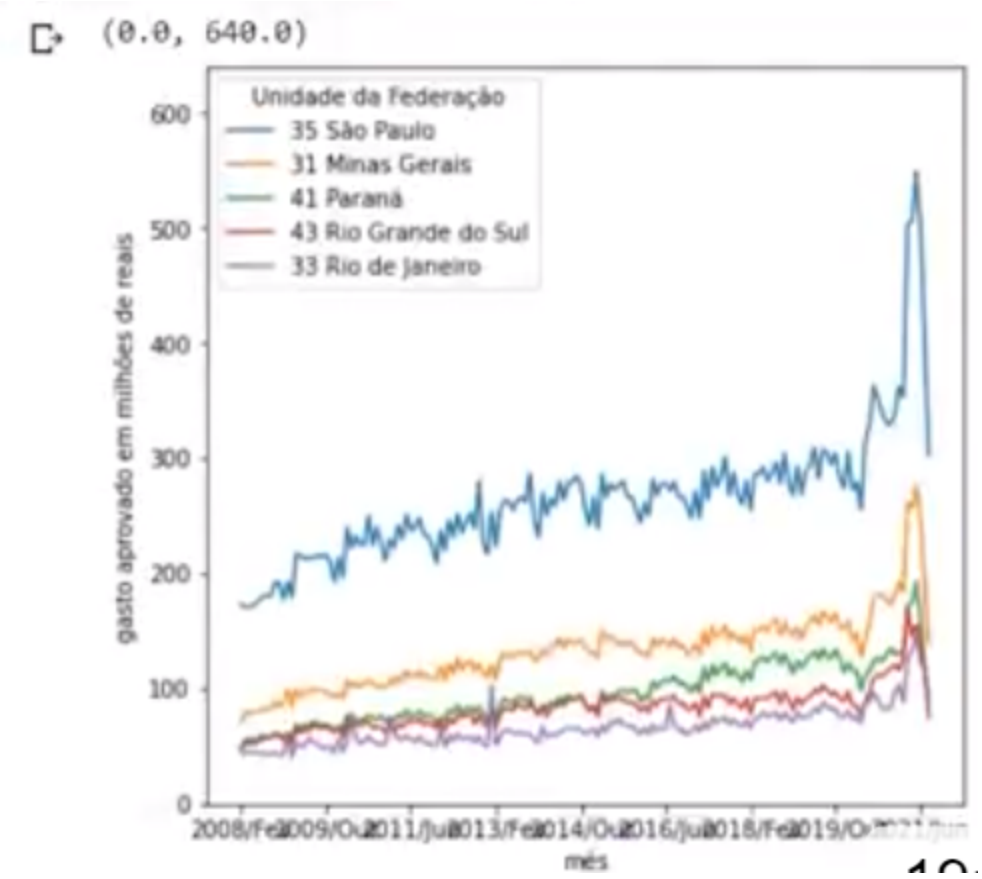
Sempre tomar cuidado, portanto, com a escolha do eixo x, do eixo y e dos elementos no gráfico. A mensagem condiz com a análise dos dados? 🙂

Agora, olha como o figsize faz diferença na percepção dos dados:

```
def plota_gastos_por_mes(dados, figsize=(10,6)):
    axis = dados.T.plot(figsize=figsize)
    axis.set_ylabel(”gasto aprovado em milhões de reais”)
    axis.set_xlabel(”mês”)
```

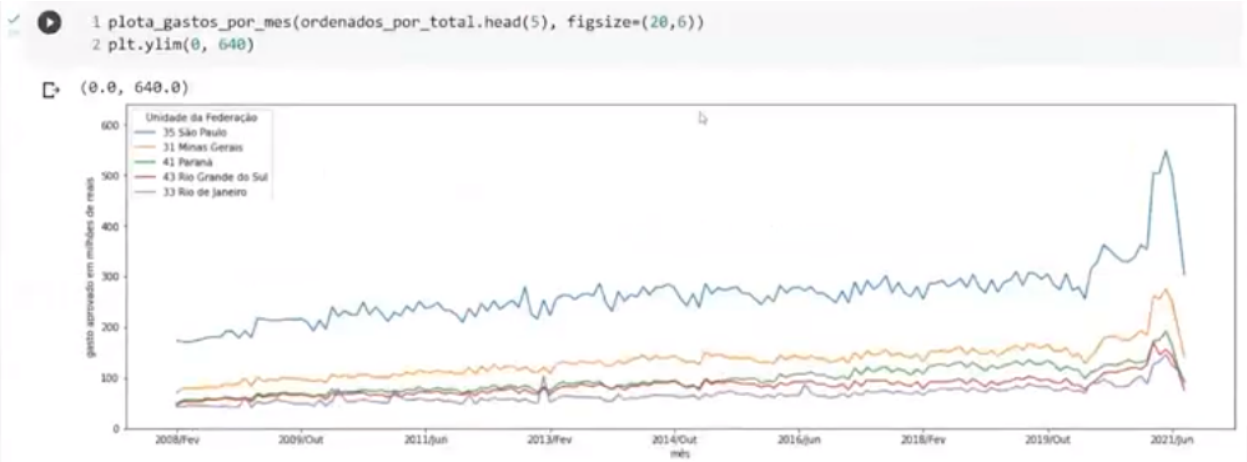
No meu último gráfico, vou passar um parâmetro a mais:

- [ ] `plota_gastos_por_mes(ordenados_por_total.head(5), figsize=(6,6))`
- [ ] `plt.ylim(0,640)`



Agora, 2021 está mais perto de 2008.

- [ ] `plota_gastos_por_mes(ordenados_por_total.head(5), figsize=(20,6))`
- [ ] `plt.ylim(0,640)`



Aqui temos a mesma coisa, mas a sensação humana ao interpretar esses 3 gráficos é distinta. Com o gráfico mais quadrado (6,6), parece que o crescimento é mais acelerado. Com um gráfico mais longo, parece que as coisas são mais vagarosas. Através de uma proporção, podemos observar uma grande diferença.

Pergunta: quanto tempo está passando entre o lado direito e esquerdo? Quantos períodos de um ano temos? Qual o crescimento dentro de um ano?

Desafio 01: padronizar os ticks verticais para espaçamento de 12 em 12 unidades (uma vez ao ano).

Desafio 02: trabalhar melhor as cores desse gráfico.

Desafio 03: colocar uma grade (grid) horizontal e vertical que não seja intrusivo (procurar na função plot

do pandas ou na matplotlib). Cuidado para não ficar poluído! 😊