Para saber mais: alterando estilos

Nós personalizamos as visualizações no vídeo anterior, alterando o tamanho das fontes, mudando a posição do título e também adicionando elementos como marcadores e grades às figuras. Mas além disso, segundo a [documentação](https://matplotlib.org/stable/gallery/style_sheets/index.html) da biblioteca Matplotlib podemos aplicar diferentes estilos para permitir que nós adaptemos as visualizações de acordo com as nossas necessidades. Para saber quais estilos estão disponíveis, podemos imprimir **style.available**, onde style é um submódulo da biblioteca Matplotlib utilizado para definir estilos de visualização:

print(plt.style.available)

O resultado é uma lista contendo todos os estilos disponíveis:

['Solarize\_Light2', '\_classic\_test\_patch', '\_mpl-gallery', '\_mpl-gallery-nogrid', 'bmh', 'classic', 'dark\_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn-v0\_8', 'seaborn-v0\_8-bright', 'seaborn-v0\_8-colorblind', 'seaborn-v0\_8-dark', 'seaborn-v0\_8-dark-palette', 'seaborn-v0\_8-darkgrid', 'seaborn-v0\_8-deep', 'seaborn-v0\_8-muted', 'seaborn-v0\_8-notebook', 'seaborn-v0\_8-paper', 'seaborn-v0\_8-pastel', 'seaborn-v0\_8-poster', 'seaborn-v0\_8-talk', 'seaborn-v0\_8-ticks', 'seaborn-v0\_8-white', 'seaborn-v0\_8-whitegrid', 'tableau-colorblind10']

Podemos explorar um desses estilos para melhorar ainda mais nossas visualizações!

Antes de aplicar um estilo, é importante saber que cada vez que a biblioteca Matplotlib é importada em um notebook, ela define uma configuração de tempo de execução que inclui os estilos padrão para cada elemento de plotagem criado.

Portanto, para evitar que o estilo seja aplicado a todos os gráficos plotados no mesmo notebook, podemos utilizar um código que cria uma cópia das configurações padrão de plotagem da biblioteca Matplotlib e as atribui à variável IPython\_default. Isso pode ser útil para armazenar e reutilizar as configurações padrão de plotagem ou para restaurá-las depois de terem sido modificadas:

IPython\_default = plt.rcParams.copy()

Um dos estilos disponíveis é baseado em um site de notícias e análises de dados chamado [FiveThirtyEight](https://fivethirtyeight.com/" \t "_blank), que cobre assuntos como política, economia, cultura, ciência e esportes.

Para utilizar o estilo **'fivethirtyeight'** usamos o código abaixo:

plt.style.**use**('fivethirtyeight')

Em seguida, podemos criar a figura, que terá esse novo estilo incorporado:

fig, ax = plt.**subplots**(figsize=(8, 4))

ax.**plot**(dados\_brasil['ano'], dados\_brasil['imigrantes'])

ax.**set\_title**('Imigração **do** Brasil para o Canadá\n1980 a 2013', fontsize=20, loc='left')

ax.**set\_ylabel**('Número de imigrantes', fontsize=14)

ax.**set\_xlabel**('Ano', fontsize=14)

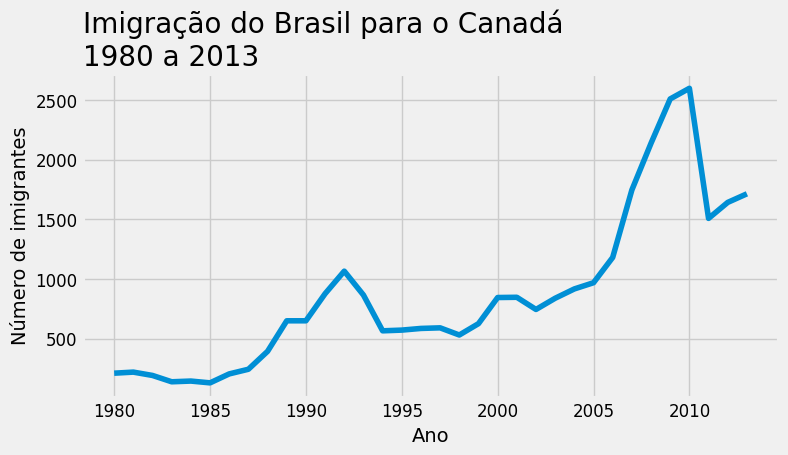
ax.yaxis.**set\_tick\_params**(labelsize=12)

ax.xaxis.**set\_tick\_params**(labelsize=12)

ax.xaxis.**set\_major\_locator**(plt.**MultipleLocator**(5))

plt.**show**()

O resultado será o gráfico exibido abaixo:



Os gráficos criados com o estilo FiveThirtyEight são lindos e possuem estética limpa e minimalista, com linhas mais grossas e cores vibrantes.

Bom, mas e se a gente quiser plotar gráficos sem esse estilo depois de ter definido ele no notebook? Podemos redefinir os parâmetros utilizando **rcParams.update** e passando a ele a variável **IPython\_default** que criamos anteriormente com as configurações padrão:

plt.rcParams.update(IPython\_default);

Além disso, podemos aplicar um estilo apenas a um bloco de código específico, pois o pacote de estilo fornece um gerenciador de contexto para limitar suas alterações a um escopo específico. Para isolar suas alterações de estilo, você pode escrever o código dentro de um contexto **with** da seguinte forma:

with plt.style.context('fivethirtyeight'):

fig, ax = plt.subplots(figsize=(8, 4))

ax.plot(dados\_brasil['ano'], dados\_brasil['imigrantes'], lw=3)

ax.set\_title('Imigração do Brasil para o Canadá\n1980 a 2013', fontsize=20, loc='left')

ax.set\_ylabel('Número de imigrantes', fontsize=14)

ax.set\_xlabel('Ano', fontsize=14)

ax.yaxis.set\_tick\_params(labelsize=12)

ax.xaxis.set\_tick\_params(labelsize=12)

ax.xaxis.set\_major\_locator(plt.MultipleLocator(5))

plt.show()

Gostou desse novo estilo? A biblioteca Matplotlib oferece uma variedade de estilos, além do FiveThirtyEight, que podemos testar e aplicar aos nossos dados. Com essas opções, podemos tornar nossas visualizações ainda mais atraentes, melhorando a apresentação das informações que queremos transmitir.