### Bem-vindo, estudante!

Ao trabalhar com dados, você frequentemente irá se deparar com dados categóricos. Ou seja, dados que representam classes, grupos ou qualidades. Esses dados são fundamentais para representar informações do mundo real. Assim, para entender conjuntos de dados com variáveis categóricas, precisamos visualizar quantidades em cada categoria. Por exemplo, em dados de uma empresa, você pode querer responder a perguntas como: quantos funcionários existem em cada departamento de uma empresa? Quantos produtos de cada tipo foram vendidos em um mês? Qual o lucro total que cada tipo de produto rendeu? Essas e outras perguntas podem ser respondidas através de visualização de dados.

Neste conteúdo, você irá conhecer duas das bibliotecas mais utilizadas em Python para visualização de dados, [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) e [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn), e aprenderá como utilizar essas bibliotecas para gerar as visualizações mais utilizadas para representar dados categóricos.

**Objetivos**

* Representar dados categóricos com gráficos de barras verticais e horizontais;
* Representar dados categóricos com gráficos de barras agrupadas.

## Tópico 1 – Gráficos de Barras Horizontais e Verticais

**OBJETIVOS**

* Aprender a exibir visualizações com as bibliotecas [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) e [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) no Jupyter Notebook;
* Aprender a criar gráficos de barras horizontais e verticais;
* Entender como se altera o design de gráficos de barra pelas cores e tamanho.

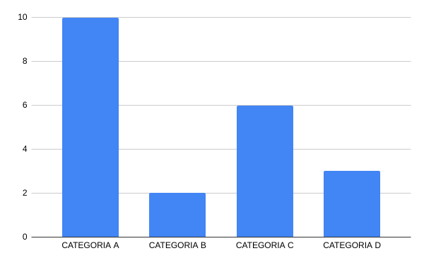
No mundo real, costumamos classificar itens por categorias. Por exemplo, carros podem ser classificados pela cor, modelo e marca. Atributos categóricos são atributos em conjuntos de dados que representam uma categoria, classe, conjunto ou qualidade. Algo que não pode ser medido quantitativamente. Você já deve ter percebido que não é possível realizar operações matemáticas em atributos categóricos. Ainda assim, comumente existem quantidades associadas às categorias. Essas quantidades podem ser visualizadas para melhor compreender os dados. Por exemplo, em um conjunto de dados de uma concessionária, você pode estar interessado em saber quantos carros foram vendidos de cada marca, ou ainda, o total de dinheiro recebido pela venda de carros de cada marca.

É com base nisso que, neste tópico, você entenderá como deve utilizar gráficos de barras para visualizar dados classificados por categorias. Você irá aprender a interpretar e criar gráficos de barras utilizando as principais bibliotecas de visualização de dados em Python e entenderá como alterar o design dos gráficos, alterando cores e tamanho, para transmitir a informação da melhor maneira possível. Vamos lá!

## 1.1 Gráficos de Barras com matplotlib

Você já se deparou alguma vez com gráficos de barras antes? Gráficos desse tipo são uma das abordagens mais utilizadas para representar quantidades relacionadas a atributos categóricos. São gráficos claros, simples de entender e muito flexíveis, podendo ser utilizados de várias formas que melhor se adequem ao tipo de informação que se deseja passar.

Um gráfico de barras padrão é formado por barras que têm a mesma espessura e que são alinhadas em um eixo. Cada barra representa uma categoria; e o comprimento da barra representa uma quantidade associada a essa categoria, como mostra a figura a seguir.

Figura 1 – Exemplo de um gráfico de barras

Provavelmente, você já deve saber como esse tipo de gráfico funciona, certo? Agora, o que você acha de aprender a criá-los? Para isso, vamos utilizar a linguagem **Python** e as bibliotecas [**matplotlib**](https://leadfortaleza.com.br/ead/glossary/matplotlib) e [**seaborn**](https://leadfortaleza.com.br/ead/glossary/seaborn), que são poderosas ferramentas que permitem criar e customizar diversos tipos de gráficos facilmente. Vamos lá!

[Matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) é uma das primeiras e mais famosas bibliotecas de visualização de dados de Python. Algumas das vantagens dela diz respeito à facilidade de criação de visualizações simples, à criação de imagens de alta qualidade em diferentes formatos e à alta customização. Já a biblioteca [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) foi construída com base na [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib), utilizando muitos de seus recursos. Porém, [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) oferece algumas outras facilidades, como novos tipos de gráficos, opções de estilo e uma integração natural com a biblioteca de manipulação de dados (**pandas**), que você já deve conhecer. Pode-se pensar que [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) é a biblioteca mãe do [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) e de outras bibliotecas de visualização de dados em Python. Agora, talvez você se pergunte qual a necessidade de conhecer as duas bibliotecas, já que [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) é uma “versão avançada” de [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib). De fato, a maior parte das visualizações neste conteúdo serão feitas utilizando a [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn), mas é útil compreender o funcionamento básico da [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) pela sua praticidade e por auxiliar no entendimento e uso do [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn).

Então, vamos lá!

Os exemplos serão executados no já conhecido ambiente Jupyter Notebooks e necessitam das bibliotecas pandas, [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) e [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) devidamente instaladas. Considerando que você tenha instalado o Python através da distribuição Anaconda, que é fortemente recomendado, todas essas bibliotecas já vêm instaladas por padrão. Para acompanhar o conteúdo, **crie um notebook e execute cada exemplo em uma célula**. Caso você não lembre como deve fazer isso ou já saiba, siga estas dicas: se você utiliza Windows, pode **inicializar o jupyter selecionando o menu Iniciar**. Se você utiliza macOX ou Linux, **abra uma janela do terminal, digite o comando jupyter notebook e tecle Enter.** Uma nova aba deverá surgir no seu navegador web onde você encontrará os diretórios do seu computador. Para criar um novo notebook, **selecione o diretório onde você quer criar o novo notebook, selecione a opção New e, em seguida, a opção Python 3.** Agora vamos começar!

O primeiro passo é importar as bibliotecas. Confira como isso pode ser feito no código a seguir:

1 import pandas as pd

2 import matplotlib.pyplot as plt

3 import seaborn as sns

4 %matplotlib inline

Download do código sem numeração no link a seguir: [Download código](https://leadfortaleza.com.br/ead2pcd/conteudo/tmp/myopenolat_1_103154023698583/aula/code/codet1p4s6.txt)

Analisando o que foi escrito, perceba que, das linhas 1 a 3, são importadas as bibliotecas que serão utilizadas, atribuindo a cada uma um pseudônimo padrão. Na linha 1, é importada a biblioteca pandas para manipular dados, sob o pseudônimo pd. Na linha 2 é importado o módulo pyplot da biblioteca [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) sob o pseudônimo plt; e na linha 3 é importada a biblioteca [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) sob o pseudônimo sns. Sempre que você for utilizar uma função dessas bibliotecas, deverá se referir a elas pelos seus pseudônimos. Lembre-se que utilizar pseudônimos não é obrigatório para o código funcionar, mas é uma boa prática, pois torna o código mais legível e padronizado de acordo com a comunidade de cientistas de dados.

A linha 4 apresenta um tipo de comando específico do ambiente jupyter, chamado comando mágico, e que é um tipo de comando utilizado para configurar notebook. Esse comando específico diz que, quando imagens forem geradas pela [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib), elas devem ser exibidas no notebook. Esse mesmo comando também vale para imagens geradas pela [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn).

Agora que você importou as bibliotecas, podemos começar a utilizá-las.

Inicialmente, você irá utilizar o módulo pyplot da biblioteca [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib" \t "_blank) (referenciado em nosso código pelo pseudônimo plt). Esse módulo é uma coleção de funções que criam e alteram figuras a partir de dados de forma programática. Para criar gráficos de barras, existe o método .bar(). Esta função recebe como argumentos uma sequência de categorias e uma sequência de valores associados a estas categorias que serão mapeados no comprimento das barras. Confira, no exemplo de código a seguir, a visualização de dados fictícios, armazenados em listas e, em seguida, verifique o resultado da execução do código.

1 categorias = ['A','B','C','D']

2 valores = [10, 20, 15, 5]

3 plt.bar(categorias,valores)

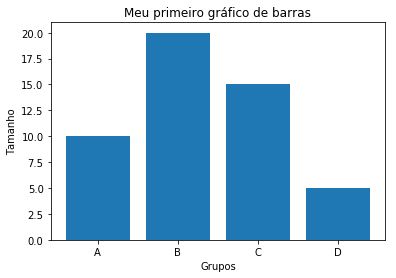
4 plt.xlabel('Grupos')

5 plt.ylabel('Tamanho')

6 plt.title('Meu primeiro gráfico de barras')

7 plt.show()

Download do código sem numeração no link a seguir: [Download código](https://leadfortaleza.com.br/ead2pcd/conteudo/tmp/myopenolat_1_103154023698583/aula/code/codet1p5s7.txt)

Figura 2 – Criando um gráfico de barras

As listas de categorias e valores nas linhas 1 e 2 armazenam os dados. Na linha 3 é chamada a função bar() a partir do pseudônimo plt, passando como argumentos as categorias e valores, indicando que esses dados serão exibidos em gráfico de barras. Nas linhas 4 e 5, a função xlabel() informa o rótulo usado no eixo horizontal (x), e a função ylabel informa o rótulo utilizado no eixo vertical (y). A função title() na linha 6 configura o título do gráfico. A função plt.show() na linha 4 é utilizada para exibir a visualização gerada. Ao executar o código, o gráfico é gerado de forma rápida, e o resultado é bem simples.

Existem várias formas de alterar o design de uma visualização no pyplot. Vamos estudar algumas delas a seguir.

Inicialmente, você irá utilizar o módulo pyplot da biblioteca [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib" \t "_blank) (referenciado em nosso código pelo pseudônimo plt). Esse módulo é uma coleção de funções que criam e alteram figuras a partir de dados de forma programática. Para criar gráficos de barras, existe o método .bar(). Esta função recebe como argumentos uma sequência de categorias e uma sequência de valores associados a estas categorias que serão mapeados no comprimento das barras. Confira, no exemplo de código a seguir, a visualização de dados fictícios, armazenados em listas e, em seguida, verifique o resultado da execução do código.

1 categorias = ['A','B','C','D']

2 valores = [10, 20, 15, 5]

3 plt.bar(categorias,valores)

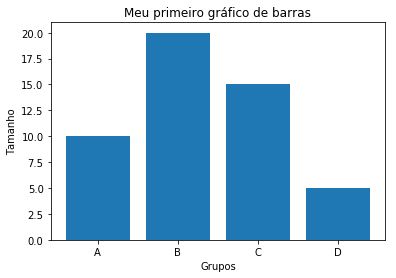
4 plt.xlabel('Grupos')

5 plt.ylabel('Tamanho')

6 plt.title('Meu primeiro gráfico de barras')

7 plt.show()

Download do código sem numeração no link a seguir: [Download código](https://leadfortaleza.com.br/ead2pcd/conteudo/tmp/myopenolat_1_103154023698583/aula/code/codet1p5s7.txt)

Figura 2 – Criando um gráfico de barras

As listas de categorias e valores nas linhas 1 e 2 armazenam os dados. Na linha 3 é chamada a função bar() a partir do pseudônimo plt, passando como argumentos as categorias e valores, indicando que esses dados serão exibidos em gráfico de barras. Nas linhas 4 e 5, a função xlabel() informa o rótulo usado no eixo horizontal (x), e a função ylabel informa o rótulo utilizado no eixo vertical (y). A função title() na linha 6 configura o título do gráfico. A função plt.show() na linha 4 é utilizada para exibir a visualização gerada. Ao executar o código, o gráfico é gerado de forma rápida, e o resultado é bem simples.

Existem várias formas de alterar o design de uma visualização no pyplot. Vamos estudar algumas delas a seguir.

Inicie pela cor das barras, que pode ser alterada através do argumento color, para o qual é passado um código de cor. A lista dos códigos possíveis para fazer essa alteração pode ser encontrada na página da documentação oficial, neste link: [https://](https://matplotlib.org/3.1.0/gallery/color/named_colors.html)[matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib).org/3.1.0/gallery/color/named\_colors.html. Além disso, o tamanho do gráfico pode ser alterado com a função figure() através do argumento figsize, que recebe um par de valores na forma de uma tupla de Python. Esse par de valores representa largura e altura da figura em polegadas. Confira isso no código de exemplo a seguir, depois verifique a figura que representa como ficou o gráfico após a execução do código.

1 plt.figure(figsize=(9, 3))

2 plt.bar(categorias,valores,color='darkred')

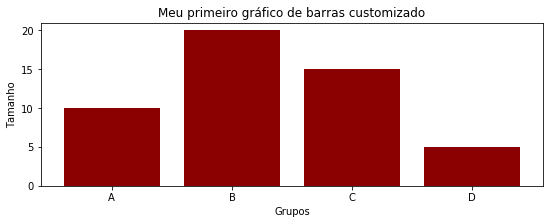
3 plt.xlabel('Grupos')

4 plt.ylabel('Tamanho')

5 plt.title('Meu primeiro gráfico de barras')

6 plt.show()

Download do código sem numeração no link a seguir: [Download código](https://leadfortaleza.com.br/ead2pcd/conteudo/tmp/myopenolat_1_103154023698583/aula/code/codet1p6s8.txt)

Figura 3 – Alterando a cor do gráfico de barras

Na linha 1, foi utilizada a função plt.figure(figsize=(9, 3)) para definir que a visualização terá 9 polegadas de comprimento e 3 de altura.

Na linha 2, é adicionado o argumento ‘darkred’ ao parâmetro color na função bar(), ou seja, vermelho escuro. As funções do [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) em geral possuem diversos parâmetros que podem ser utilizados para controlar a estética das visualizações.

Note que cada nova função chamada altera detalhes da visualização de forma incremental. Ou seja, você pode utilizar várias funções, uma após a outra, e alterar incrementalmente a estética da visualização.

Para conhecer as diversas opções de costomização que a biblioteca [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) oferece, você pode consultar a rica e detalhada documentação oficial, em inglês, neste link: [https://](https://matplotlib.org/contents.html)[matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib).org/contents.html.

A seguir, você irá trabalhar com um conjunto de dados reais para exercitar conceitos de visualização.

Você gosta de cinema? Para exemplificar os conceitos estudados neste conteúdo, você irá analisar um conjunto de dados de bilheterias de filmes famosos lançados entre 2007 e 2011. Os dados estão disponibilizados gratuitamente na Web e podem ser acessados pela URI: [https://raw.githubusercontent.com/emanueles/datavis-course/master/assets/files/observable/movies.](https://raw.githubusercontent.com/emanueles/datavis-course/master/assets/files/observable/movies.json)[json](https://leadfortaleza.com.br/ead/glossary/json).

Você conferiu os dados? Note que o formato deles é diferente das tradicionais tabelas, mas como seguem uma estrutura padronizada também são considerados dados estruturados. O nome desse tipo de formato é JavaScript Object Notation, mais conhecido como JSON, e é um formato de troca de dados compacto, de padrão aberto, muito utilizado por sistemas computacionais, principalmente na Web. Por seguir uma estrutura definida, é fácil converter JSON para uma tabela. De fato, o pandas disponibiliza um método para criar os familiares DataFrames de pandas a partir de um arquivo JSON. A seguir, confira um exemplo de código e, em seguida, verifique o resultado de sua execução.

1 filmes\_uri = 'https://raw.githubusercontent.com/emanueles/datavis-course/master/assets/files/observable/movies.[json](https://leadfortaleza.com.br/ead/glossary/json)'

2 filmes = pd.read\_json(filmes\_uri)

3 filmes.head()

Out:

|  | **Budget\_M** | **Film** | **Genre** | **Worldwide\_Gross\_M** | **Year** |
| --- | --- | --- | --- | --- | --- |
| 0 | 125 | Harry Potter and the Deathly Hallows Part 2 | Fantasy | 1328 | 2011 |
| 1 | 145 | Mission Impossible 4 | Action | 689 | 2011 |
| 2 | 200 | Toy Story 3 | Animation | 1063 | 2010 |
| 3 | 200 | 2012 | Action | 768 | 2009 |
| 4 | 200 | Alice in Wonderland | Adventure | 1024 | 2010 |

Agora, vamos analisar o código e a saída.

Na linha 1, o endereço dos dados na web é atribuído à variável filmes\_uri, ou seja, o endereço é armazenado na variável. Na linha 2, chama-se o pseudônimo pd para utilizar o método de pandas read\_json. O método recebe como argumento o endereço no qual os dados estão armazenados e retorna um DataFrame com os dados organizados em um formato de tabela que é armazenado na variável filmes. Como filmes é um DataFrame, podemos utilizar alguns dos métodos existentes dessa estrutura para analisar os dados. O método head() na linha 3 tem como saída as colunas e as 5 primeiras linhas da tabela. Note que o conjunto de dados possui 5 atributos, 2 quantitativos e 3 categóricos. Os quantitativos são **Budget\_M** e **Worldwide\_Gross\_M**, que correspondem ao orçamento do filme e a bilheteria mundial que o filme obteve em milhões de dólares, respectivamente. Os atributos categóricos são **Film**, **Genre** e **Year**, que representam o nome do filme, o gênero e o ano em que foi lançado, respectivamente. Note que apesar de os valores de Year serem números, ele é um atributo categórico, e não quantitativo! Isso porque ele é o ano em que o filme foi lançado, logo não representa uma quantidade, e sim um grupo. Você compreende o porquê? Confira novamente o código de exemplo e o resultado de sua execução após compreender os detalhes apresentados até aqui.

1 filmes\_uri = 'https://raw.githubusercontent.com/emanueles/datavis-course/master/assets/files/observable/movies.[json](https://leadfortaleza.com.br/ead/glossary/json)'

2 filmes = pd.read\_json(filmes\_uri)

3 filmes.head()

Out:

|  | **Budget\_M** | **Film** | **Genre** | **Worldwide\_Gross\_M** | **Year** |
| --- | --- | --- | --- | --- | --- |
| 0 | 125 | Harry Potter and the Deathly Hallows Part 2 | Fantasy | 1328 | 2011 |
| 1 | 145 | Mission Impossible 4 | Action | 689 | 2011 |
| 2 | 200 | Toy Story 3 | Animation | 1063 | 2010 |
| 3 | 200 | 2012 | Action | 768 | 2009 |
| 4 | 200 | Alice in Wonderland | Adventure | 1024 | 2010 |

Agora que você já entendeu que tipos de dados estão armazenados nesse conjunto, podemos começar a analisá-los.

Como análise preliminar, você pode utilizar o conhecido método describe() para obter medidas estatísticas sobre os atributos quantitativos. Confira no exemplo a seguir:

1 filmes.describe()

Out:

|  | **Budget\_M** | **Worldwide\_Gross\_M** | **Year** |
| --- | --- | --- | --- |
| **count** | 30.0 | 30.0 | 30.0 |
| **mean** | 168.93 | 889.13 | 2009.1 |
| **std** | 54.59 | 397.20 | 1.42 |
| **min** | 50.0 | 621.0 | 2007.0 |
| **25%** | 146.25 | 696.0 | 2008.0 |
| **50%** | 162.5 | 793.0 | 2009.0 |
| **75%** | 200.0 | 951.25 | 2010.0 |
| **max** | 300.0 | 2784.0 | 2011.0 |

Agora você possui algumas informações sobre os dados. Por exemplo, você sabe que nesse conjunto há dados sobre 30 filmes (**count**), que o valor máximo de bilheteria que um filme obteve nesse conjunto foi de 2.784 milhões de dólares (**max**), que o orçamento mínimo entre os filmes foi de 50 milhões de dólares (**min**). Também sabe sobre a média (**mean**) e desvio padrão (**std**) dessas variáveis. Observe que o pandas também realizou essas medidas para o atributo Year, pois como ele é representado por números, o pandas o interpretou como um atributo quantitativo. Note também que essas informações não dizem nada a respeito das outras categorias. Como você já deve saber, não é possível calcular a média ou desvio padrão de atributos categóricos. Porém, você pode estar interessado em questões como: quantos filmes foram feitos em cada gênero? E em cada ano? Quais gêneros obtiveram maior bilheteria no total? E quais tiveram mais orçamento? Essas perguntas podem ser respondidas através de gráficos de barras, o que você irá conferir a seguir.

Vamos começar por esta pergunta: quantos filmes existem de cada gênero? Como você pode visualizar esses dados com o matplotlib? Note que é necessário passar uma sequência de dados (seja uma lista ou uma Series de pandas) contendo o nome dos gêneros e outra sequência contendo as quantidades de filmes de cada respectivo gênero. Note que os dados originais não estão nesse formato. Então, como fazer?

Uma alternativa é calcular essas sequências utilizando o pandas. O DataFrame possui o método groupby(), que permite agrupar os dados em relação a um atributo e aplicar uma operação matemática sobre eles. No caso, desejamos agrupar os dados pelos gêneros e contá-los. Confira como é feito no exemplo de código a seguir, depois verifique o resultado de sua execução.

1 contagens = filmes.groupby(by='Genre').count()

2 contagens

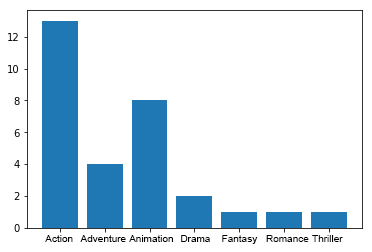
|  | **Film** | **Budget\_M** | **Worldwide\_Gross\_M** | **Year** |
| --- | --- | --- | --- | --- |
| **Genre** |  |  |  |  |
| **Action** | 13 | 13 | 13 | 13 |
| **Adventure** | 4 | 4 | 4 | 4 |
| **Animation** | 8 | 8 | 8 | 8 |
| **Drama** | 2 | 2 | 2 | 2 |
| **Fantasy** | 1 | 1 | 1 | 1 |
| **Romance** | 1 | 1 | 1 | 1 |
| **Thriller** | 1 | 1 | 1 | 1 |

Na variável filmes, que contém o DataFrame dos dados analisados, foi aplicado o método groupby(), recebendo como argumento o atributo ‘Genre’ e, por fim, aplicando o método count(), que conta a quantidade de elementos em cada gênero. O resultado desse método é um DataFrame, cujo index são os gêneros, e as colunas são os atributos do DataFrame original, cujo valor é o resultado da função aplicada, nesse caso, a contagem.

Agora você já possui os dados necessários para gerar a visualização utilizando o matplotlib. Basta selecionar o index e qualquer coluna do DataFrame contagens. Confira a seguir:

1 plt.bar(contagens.index, contagens['Worldwide\_Gross\_M'])

2 plt.show()

Figura 4 – Gerando gráfico com [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib" \t "_blank)

Muito bem! Agora você já deve saber fazer uma visualização de dados reais utilizando o [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib)!

Note que essa visualização indica informações importantes sobre os dados. Observa-se, por exemplo, que a quantidade de filmes não está bem distribuída entre os gêneros nesses dados. Existem muito mais filmes de ação do que todos os outros gêneros. De fato, existe apenas um exemplar de filme de fantasia, romance e suspense. É necessário ter em mente que esses dados são apenas uma amostra e não representam todos os filmes feitos nesse período. Apesar de informativa, a visualização possui alguns problemas funcionais. O nome dos gêneros no eixo horizontal está se sobrepondo, dificultando a leitura e não há linhas guias do eixo horizontal, o que pode dificultar quando se quiser obter o valor exato de cada barra. Além disso, para gerar essa visualização, é necessário realizar algumas manipulações nos dados. A seguir, iremos utilizar a biblioteca [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) para tentar solucionar esses problemas.

## 1.2 Gráficos de Barras com seaborn

Como já mencionado, [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) é uma biblioteca de visualização de dados, construída a partir do [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib), mas com algumas vantagens. Uma dessas vantagens é que enquanto o [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) é orientado a sequências de dados, o [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) é orientado a conjuntos de dados com múltiplas variáveis. Ele se integra, naturalmente, com os DataFrames de pandas e possui suporte para visualizar agregações de variáveis categóricas.

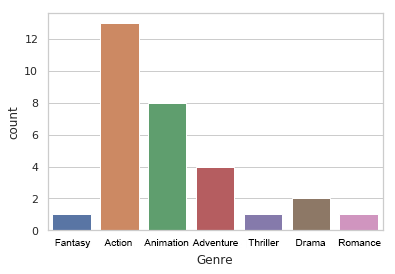
O que isso significa? Significa que podemos gerar a visualização anterior a partir do DataFrame original diretamente, sem a necessidade de agrupar os dados explicitamente. Basta indicar o conjunto de dados e a coluna que se deseja visualizar. O [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) se encarrega de agrupar e calcular os valores dos dados, utilizando apenas uma função. Outras características apreciadas no [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) é a sua simplicidade e o design atraente das visualizações geradas.

A função para gerar gráficos de barras de contagem de elementos é countplot(). Confira o exemplo a seguir para entender como essa função é aplicada e como são apresentados os mesmos dados da visualização anterior, a contagem de filmes por gênero.

1 sns.set(style="whitegrid")

2 sns.countplot(x="Genre", data=filmes)

3 plt.show()

Figura 5 – Criando gráfico com [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn" \t "_blank)

O que você achou do resultado? Vamos analisar cada linha deste código. Na linha 1, usamos o pseudônimo sns que atribuímos ao seaborn para chamar a função set(), recebendo como argumento ao parâmetro style o valor ‘whitegrid’. Essa função pode receber diferentes argumentos que controlam a estética das visualizações do [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn). O parâmetro style pode receber um conjunto de códigos de estilos pré-definidos. O estilo escolhido, ‘whitegrid’, utiliza um fundo branco com linhas guias que auxiliam a comparar valores. Outros estilos possíveis são ‘darkgrid’, ‘whitegrid’, ‘dark’, ‘white’ e ‘ticks. Encorajo você a experimentar esses outros estilos, que podem ser mais ou menos adequados a depender do objetivo da visualização. Uma vez utilizada a função set(), ele configura todas as futuras visualizações criadas no código.

Na linha 2 é chamada a função countplot(), que gera a visualização em si, contando o número de elementos de um determinado grupo. Neste exemplo, ela recebe dois argumentos. Um é o parâmetro x, ou seja, o eixo horizontal, que recebe o valor ‘Genre’, que corresponde a coluna de gêneros cinematográficos dos nossos dados. Isso significa dizer que cada barra, alinhadas no eixo x, irão corresponder a um gênero. O outro parâmetro é data (do inglês, dados), que recebe a variável onde está armazenado o conjunto de dados. Como já mencionado, essa é uma diferença conceitual entre o [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) e o [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn). Lembre que no [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) você passou como argumentos sequências de dados (Series, listas), uma sequência contendo as classes e outra contendo os valores. No seaborn, você passa o conjunto de dados (tabela, DataFrame) e indica qual coluna você deseja visualizar.

Por último é chamada a função show() do [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib), que exibe a visualização criada. Notou como o [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) necessita do [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib)?

Uma falha desta visualização é que o nome das categorias, indicados no eixo x, ficaram tão apertados que acabaram por se sobrepor, o que dificulta a leitura. Existem algumas formas de resolver isso, uma deles é modificar o tamanho da visualização. Para fazer isso, utilizamos a o método figure() do pyplot. Você já notou que o [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) compartilha muitas das funções do [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib), não é? Por isso é interessante se trabalhar com as duas. Também iremos usar a método title() do pylplot para exibir um título no gráfico. Vamos também mudar o estilo de gráfico para “darkgrid”. Confira no exemplo o resultado:

1 sns.set(style="darkgrid")

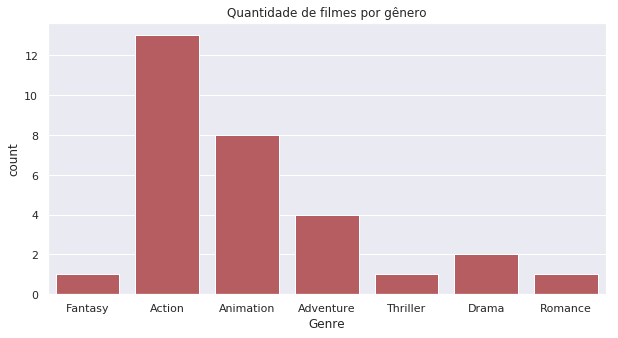
2 plt.figure(figsize=(10,5))

3 plt.title('Quantidade de filmes por gênero')

4 sns.countplot(x="Genre", color = 'r', data=filmes)

5 plt.show()

Download do código sem numeração no link a seguir: [Download código](https://leadfortaleza.com.br/ead2pcd/conteudo/tmp/myopenolat_1_103154023698583/aula/code/codet1p14s16.txt)

Figura 6 – Quantidade de filmes por gênero

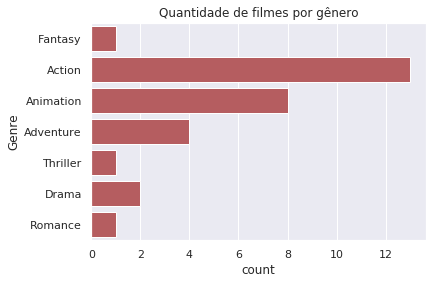
Agora a visualização está bem melhor. Uma outra solução, principalmente quando existem muitas classes, é criar um gráfico de barras horizontais, agrupando as barras no eixo y e mapeando os comprimentos no eixo x. Confira como isso é feito!

Alterar o gráfico já criado para um de barras horizontais é muito simples no [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn). Basta que o valor da categoria, nesse caso ‘Genre’, seja passado ao parâmetro y, ao invés de x. Isso significa dizer a função que as barras devem ficar alinhadas no eixo vertical. Confira no exemplo a seguir:

1 plt.title('Quantidade de filmes por gênero')

2 sns.countplot(y="Genre", color = 'r', data=filmes)

3 plt.show()

Figura 7 – Quantidade de filmes por gênero em barras horizontais

Logicamente, se você tentar atribuir, simultaneamente, valores aos parâmetros x e y, a função retornará um erro.

Com essa função você já pode explorar os dados com mais facilidade e obter uma série de informações a respeito de contagens dos filmes em grupos. Por exemplo, você pode usar essa mesma função para saber quantos filmes foram feitos em cada ano. Mas e se você estiver interessado em outros valores além de contagem? Lembre-se que cada filme, além de gênero e ano, possui um valor de orçamento e de bilheteria mundial. Pode ser útil, por exemplo, saber a **média** de orçamento por gênero, ou o **somatório** de bilheteria total. Enquanto no [matplotlib](https://leadfortaleza.com.br/ead/glossary/matplotlib) seria necessário manipular o dataset para calcular esses valores e utilizá-los nas funções, o [seaborn](https://leadfortaleza.com.br/ead/glossary/seaborn) permite fazer isso diretamente através da função barplot(). Acompanhe como ela é aplicada.

O resultado da função barplot() é muito semelhante ao da countplot(). Ou seja, um gráfico de barras. A diferença é que countplot() mostra uma contagem de elementos de grupos, enquanto barplot() mostra medidas associadas a esses grupos. Por padrão, essa medida é a média aritmética. No exemplo a seguir, você visualizará a média de bilheteria por gênero. Confira o exemplo de código e o resultado de sua aplicação.

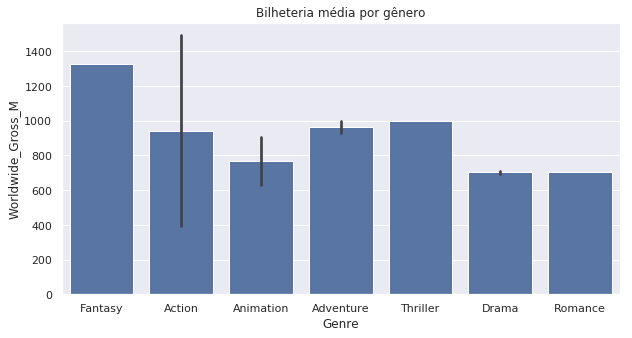
1 plt.figure(figsize=(10,5))

2 plt.title('Bilheteria média por gênero')

3 sns.barplot(x="Genre", y="Worldwide\_Gross\_M", color = 'b', ci = ‘sd’, data=filmes)

4 plt.show()

Download do código sem numeração no link a seguir: [Download código](https://leadfortaleza.com.br/ead2pcd/conteudo/tmp/myopenolat_1_103154023698583/aula/code/codet1p16s18.txt)

Figura 8 – Bilheteria média por gênero

Vamos analisar os parâmetros da função barplot(), chamada na linha 3. O parâmetro x recebe a categoria que cada barra irá representar, enquanto o parâmetro y recebe o valor de bilheteria. Será então calculada a média das bilheterias por gênero, e esse valor será mapeado no eixo y. Note que a cor utilizada foi ‘b’, ou seja, blue (azul).

Por fim, um novo parâmetro chamado ci (confidence interval) foi utilizado, recebendo o valor 'sd' (standard deviation, o desvio padrão). Por padrão, o parâmetro ci é usado para estimar intervalos de confiança, um intervalo estatístico onde provavelmente se encontra um valor desconhecido. Neste caso, passando o valor ‘sd’, ao invés de intervalo de confiança, é calculado o já conhecido desvio padrão, que indica como os dados se dispersam ao redor da média. O desvio padrão é representado pela linha escura no centro das barras.

Essa visualização diz algo interessante! Apesar de mais filmes de ação terem sido feitos nesses anos, os filmes de fantasia tiveram, em média, maior bilheteria. Além disso, os filmes de ação tiveram grande variação de bilheteria. Por que você acha que isso ocorreu? Vamos conferir?

E se, ao invés de querer saber a média, você precise do somatório dos valores associados a cada categoria? Basta alterar o parâmetro estimator, que deve receber a função que se deseja calcular. Nesse caso, a função sum, que soma todos os valores. Confira o exemplo de código e o resultado de sua execução:

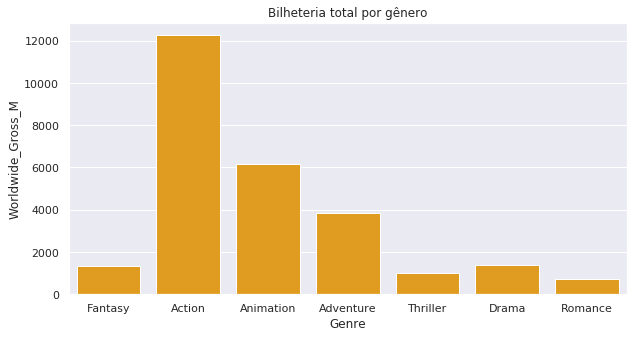
1 plt.figure(figsize=(10,5))

2 plt.title('Bilheteria total por gênero')

3 sns.barplot(x="Genre", y="Worldwide\_Gross\_M", color = 'orange', estimator = sum, ci = None, data=filmes)

4 plt.show()

Download do código sem numeração no link a seguir: [Download código](https://leadfortaleza.com.br/ead2pcd/conteudo/tmp/myopenolat_1_103154023698583/aula/code/codet1p17s19.txt)

Figura 9 – Bilheteria total por gênero

Note que houveram duas diferenças significativas nos parâmetros da função: a passagem do argumento sum para o parâmetro estimator e a passagem do argumento None (vazio) para o parâmetro ci, para que não sejam exibidos intervalos de confiança.

Dessa vez, o resultado ficou mais similar à contagem. Mas é preciso ter atenção para não tomar conclusões precipitadas. Esse gráfico pode dar a entender que os filmes de ação têm maior bilheteria, mas isso acontece porque existem mais filmes de ação nesse conjunto de dados. Relembre que, na média, os filmes de fantasia obtiveram maior bilheteria.

Até o momento você aprendeu a gerar gráficos de barras que representam um valor para uma categoria. Porém, gráficos de barra são muito flexíveis e existem formas de representar mais de uma categoria com barras agrupadas. Você poderá entender como criá-las no próximo tópico.