

Introdução

Olá seja bem-vinda e bem-vindo ao notebook da **aula 02**, desça até o conteúdo da aula 02 e bons na seta antes do título Aula 01 ela comprime todo o conteúdo da aula 1, deixando o layout mais

Não esqueça de rodar todos as células de códigos da aula 01, antes de iniciar a aula 02

▼ Aula 01

Nós estaremos desenvolvendo nosso projeto aqui no google colaboratory, assim podemos mesclar textos em formato markdown e células de código, além disso você não precisar instalar nada na máquina que tal começar testando algumas linhas de código.

Nesta primeira célula estamos realizando um `print()`, lembre-se que esta função python imprime o que estamos passando como parâmetro, então o retorno é exibido logo abaixo da célula com código.

```
print("Guilherme Silveira")  
print("Paulo Silveira")
```



Guilherme Silveira
Paulo Silveira

Agora vamos analisar a proxima célula de código.

Aqui estamos fazendo uma atribuição de variável, conforme dito em aula, as atribuições não tem diferente da célula anterior não temos um *output* logo abaixo do código.

```
nome_do_filme = "Totoro, o filme"
```

Agora que criamos a variável `nome_do_filme`, podemos reutilizá-la, por exemplo na função `print`, imprimir a *string* "Totoro, o filme".

```
print(nome_do_filme)
```



Totoro, o filme

```
nome_do_filme
```



'Totoro, o filme'

▼ Lendo os dados do MovieLens

Overall, how satisfied are you with Colaboratory?

Very satisfied

Somewhat satisfied

Neither satisfied nor dissatisfied

Somewhat dissatisfied

Very dissatisfied

Google

Nosso primeiro passo foi conhecer e realizar um "hello-world" no colab, agora chegou a hora de ir para o notebook e começar as análises.

Vamos importar a biblioteca [pandas](#), um poderoso projeto open source para análise de manipulação. O primeiro passo é ler uma base de dados e podemos fazer isso com o comando `pd.read_csv()`.

Estamos lendo um arquivo **CSV** (Comma-separated values), neste tipo de arquivo os valores são separados por vírgulas e podem ser abertos em outras ferramentas como excel e google-sheet. CSV não é o único formato que o pandas suporta, temos o `pd.read_excel()` que lê arquivos **xlsx** entre diversos outros formatos, você pode encontrar mais informações na seção de [input/output da documentação](#).

Depois de ler o dataset, nós trocamos os nomes das colunas pelos termos em português, logo em seguida utilizamos o método `filmes.head()` para visualizar as primeiras 5 linhas do nosso dataframe. Ou seja, para visualizar as informações dos dados é utilizando o método `filmes.sample()`, se você tentar, vai retornar uma linha aleatória do seus dados. Para escolher aleatoriamente mais de 1 linha, por exemplo:

```
import pandas as pd
```

```
filmes = pd.read_csv("https://raw.githubusercontent.com/alura-cursos/introducao-a-data-science/master/02-Pandas/02-01-Importando-dados/filmes.csv")
# filmes é um DataFrame
filmes.columns = ["filmeId", "titulo", "generos"]
filmes.head()
```



	filmeId	titulo	generos
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
1	2	Jumanji (1995)	Adventure Children Fantasy
2	3	Grumpier Old Men (1995)	Comedy Romance
3	4	Waiting to Exhale (1995)	Comedy Drama Romance
4	5	Father of the Bride Part II (1995)	Comedy

Há pouco falamos para consultar a documentação para obter mais informações, mas será que é necessário sair do notebook para tirar algumas dúvidas mais simples?

Os notebooks facilitam a nossa vida podendo consultar o docstring das funções e métodos, rodando um `?` na frente da chamada, uma view é aberta com as informações resumidas. Veja a seguir alguns exemplos:

```
# lendo a documentação de um método/atributo
```

```
?filmes.head
```


```
# lendo a documentação do tipo (docstring)
```

```
?filmes
```

A base de dados que usamos até o momento contém o nome do filme, ano de lançamento e gênero. Mas também conta com outras informações que estão em bases separadas, uma delas é a de avaliações.

Agora vamos analisar um pouco melhor o dataset de avaliações.


```
avaliacoes = pd.read_csv("https://github.com/alura-cursos/introducao-a-data-science/blob/main/dados/avaliacoes.csv")
avaliacoes.head()
```




	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

Para visualizar algumas linhas estamos usando o `.head()`, como ela mostra apenas as 5 primeiras linhas, sabemos qual é a quantidade de linhas que temos. Para descobrir a "forma" dos nossos dados podemos usar `avaliacoes.shape`, retornando uma [tupla](#), onde o primeiro termo indica o número de linhas e o segundo o número de colunas.

```
avaliacoes.shape
```

 (100836, 4)

```
len(avaliacoes)
```

 100836

Vamos substituir os nomes das colunas de inglês para português e entender o que são essas colunas:

- userId => ID para o usuário que votou em determinado filme.

- movieId => ID para identificar um filme votado.

- rating => A nota dada pelo usuário para o respectivo filme.

- timestamp => A data da votação que não está formatada como data

Como cada linha contém um voto para o respectivo filme, é de se esperar que um filme tenha diversas avaliações. Olhando as primeiras 5 linhas, vemos que o filme **1, 3, 6, 47, 50** foram avaliados. Mas e se eu quiser analisar apenas o filme 1, como posso separar essa informação?

```
avaliacoes.columns = ["usuarioId", "filmeId", "nota", "momento"]
avaliacoes.head()
```



	usuarioId	filmeId	nota	momento
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

Uma forma para "separar" as informações apenas do **filmeId 1** é chamando o método `avaliacoes.query("filmeId==1")`, esse método retornará apenas as linhas para quais a expressão "filmeId==1", for verdadeira.

Tendo as informações do **filmeId 1** podemos chamar o `avaliacoes_do_filme_1.describe()`, para estatísticas gerais dos dados.

```
avaliacoes_do_filme_1 = avaliacoes.query("filmeId==1")
avaliacoes_do_filme_1.head()
```



	usuarioId	filmeId	nota	momento
0	1	1	4.0	964982703
516	5	1	4.0	847434962
874	7	1	4.5	1106635946
1434	15	1	2.5	1510577970
1667	17	1	4.5	1305696483

```
avaliacoes_do_filme_1.describe()
```



	usuarioId	filmeId	nota	momento
count	215.000000	215.0	215.000000	2.150000e+02
mean	306.530233	1.0	3.920930	1.129835e+09
std	180.419754	0.0	0.834859	2.393163e+08
min	1.000000	1.0	0.500000	8.293223e+08
25%	155.500000	1.0	3.500000	8.779224e+08
50%	290.000000	1.0	4.000000	1.106855e+09
75%	468.500000	1.0	4.500000	1.348523e+09
max	610.000000	1.0	5.000000	1.535710e+09

Caso queira uma estatística particular, podemos apenas chamar o método desejado, repare abaixo

```
avaliacoes_do_filme_1.mean()
```

```
usuarioId    3.065302e+02
filmeId      1.000000e+00
nota         3.920930e+00
momento      1.129835e+09
dtype: float64
```

Calculamos as estatísticas apenas para o **filmeId 1**, mas também podemos chamar o método `.describe()` de base completa (avaliações).

```
avaliacoes.describe()
```

	usuarioId	filmeId	nota	momento
count	100836.000000	100836.000000	100836.000000	1.008360e+05
mean	326.127564	19435.295718	3.501557	1.205946e+09
std	182.618491	35530.987199	1.042529	2.162610e+08
min	1.000000	1.000000	0.500000	8.281246e+08
25%	177.000000	1199.000000	3.000000	1.019124e+09
50%	325.000000	2991.000000	3.500000	1.186087e+09
75%	477.000000	8122.000000	4.000000	1.435994e+09
max	610.000000	193609.000000	5.000000	1.537799e+09

Ok, nós calculamos um tanto de coisa usando `.describe()` e `.mean()`, mas a informação que nos interessa é a média da nota. Então o ponto é, como calcular a média apenas das notas?

A primeira coisa que precisamos fazer é selecionar apenas as informações de notas. Usando uma sintaxe parecida com a de [chave-valor dos dicionários python](#).

Com o comando `avaliacoes["nota"]`, obtemos os valores da coluna nota (repare que o tipo retornado são pandas, por isso o index de cada nota é mantido). Para calcular a média de todas as notas executamos `avaliacoes["notas"].mean()`

```
avaliacoes["nota"]
```



```

0      4.0
1      4.0
2      4.0
3      5.0
4      5.0
...
100831  4.0
100832  5.0
100833  5.0
100834  5.0
100835  3.0
Name: nota, Length: 100836, dtype: float64


```

```
avaliacoes["nota"].mean()
```

 3.501556983616962


Podemos calcular também na nota média do **filmeId 1**, repare que o resultado é um pouco maior e com essa análise não dá para bater o martelo que o filme 1 é acima da média, mas apenas com e conseguimos formular uma primeira hipótese!

```
avaliacoes_do_filme_1["nota"].mean()
```

 3.9209302325581397

Nós calculamos uma média geral, uma média para o filmeId 1. Agora eu quero calcular a média de todos os filmes, podemos fazer isso usando o método `.groupby(filmeId)`, o parâmetro passado é para a coluna ele deve utilizar para "agrupar" os dados. Depois só calcular a média como fizemos anteriormente.

```
notas_medias_por_filme = avaliacoes.groupby("filmeId")["nota"].mean()
notas_medias_por_filme.head()
```

 filmeId

```

1      3.920930
2      3.431818
3      3.259615
4      2.357143
5      3.071429
Name: nota, dtype: float64

```

Temos as notas médias calculadas, mas agora precisamos juntar as informações de notas médias dos dados **filmes**.

Poderíamos criar uma nova coluna e atribuir a variável `notas_medias_por_filme`, de forma direta:

```
filmes["nota_media"] = notas_medias_por_filme
```

Como discutimos em aula, essa não é uma boa prática pois precisamos garantir que a nota média é por filme.

Para garantir essa condição vamos utilizar o `.join()`, criando um novo dataframe (`filmes_com_m`
`filmes.join(notas_medias_por_filme, on="filmeId")`).

filmes

	filmeId	titulo	gene
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
1	2	Jumanji (1995)	Adventure Children Fantasy
2	3	Grumpier Old Men (1995)	Comedy Romance
3	4	Waiting to Exhale (1995)	Comedy Drama Romance
4	5	Father of the Bride Part II (1995)	Comedy
...
9737	193581	Black Butler: Book of the Atlantic (2017)	Action Animation Comedy Fantasy
9738	193583	No Game No Life: Zero (2017)	Animation Comedy Fantasy
9739	193585	Flint (2017)	Drama
9740	193587	Bungo Stray Dogs: Dead Apple (2018)	Action Animation
9741	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy

9742 rows × 3 columns

notas_medias_por_filme

filmeId	nota
1	3.920930
2	3.431818
3	3.259615
4	2.357143
5	3.071429
...	...
193581	4.000000
193583	3.500000
193585	3.500000
193587	3.500000
193609	4.000000

Name: nota, Length: 9724, dtype: float64

```
filmes_com_media = filmes.join(notas_medias_por_filme, on="filmeId")
filmes_com_media.head()
```



	filmeId	titulo	generos	r
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	3.920
1	2	Jumanji (1995)	Adventure Children Fantasy	3.431
2	3	Grumpier Old Men (1995)	Comedy Romance	3.259
3	4	Waiting to Exhale (1995)	Comedy Drama Romance	2.357
4	5	Father of the Bride Part II (1995)	Comedy	3.071

Agora que temos as médias, que tal visualizar o nosso dataframe ordenado pela nota de forma de

```
filmes_com_media.sort_values("nota", ascending=False).head(15)
```



	filmeId	titulo	generos	nota
7656	88448	Paper Birds (Pájaros de papel) (2010)	Comedy Drama	
8107	100556	Act of Killing, The (2012)	Documentary	
9083	143031	Jump In! (2007)	Comedy Drama Romance	
9094	143511	Human (2015)	Documentary	
9096	143559	L.A. Slasher (2015)	Comedy Crime Fantasy	
4251	6201	Lady Jane (1986)	Drama Romance	
8154	102217	Bill Hicks: Revelations (1993)	Comedy	
8148	102084	Justice League: Doom (2012)	Action Animation Fantasy	
4246	6192	Open Hearts (Elsker dig for evigt) (2002)	Romance	
9122	145994	Formula of Love (1984)	Comedy	
8115	100906	Maniac Cop 2 (1990)	Action Horror Thriller	
9129	146662	Dragons: Gift of the Night Fury (2011)	Adventure Animation Comedy	
8074	99636	English Vinglish (2012)	Comedy Drama	
5785	31522	Marriage of Maria Braun, The (Ehe der Maria Br...	Drama	
9131	146684	Cosmic Scrat-tastrophe (2015)	Animation Children Comedy	

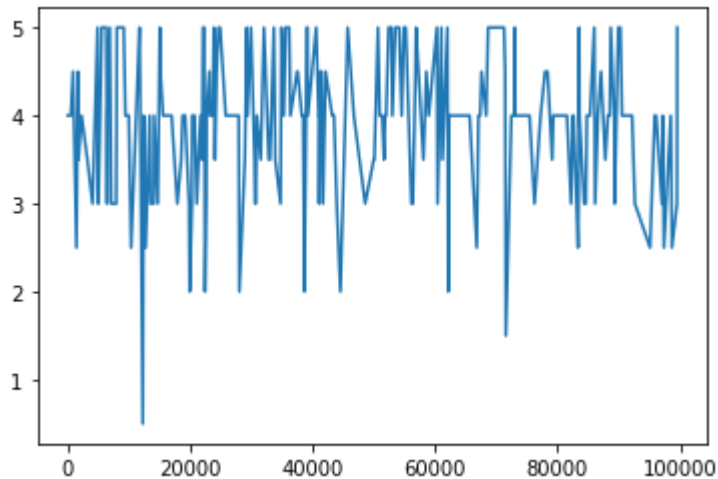
Fizemos um tanto de análise e manipulação de dados interessante, não é?

Mas diz a verdade, você está sentindo falta daquele gráfico que todo cientista de dados adora =D, nosso primeiro gráfico!

O pandas facilita muito o plot de alguns gráficos simples, apenas selecionamos a informação que visualizar e chamamos o método `.plot()`

```
avaliacoes.query("filmeId == 1")["nota"].plot()
```

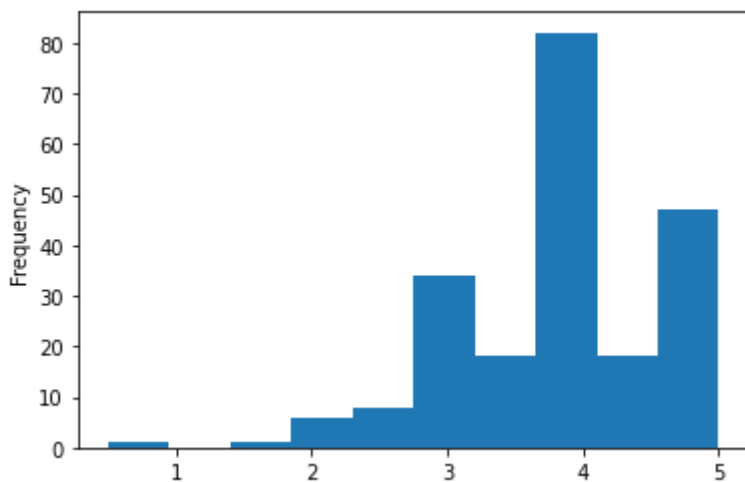

<matplotlib.axes._subplots.AxesSubplot at 0x7fe026757550>



Por padrão o método plotou um gráfico de linhas, o que não é adequado para os dados que estão. Precisamos mudar o tipo de gráfico para realizar uma análise mais adequada, para fazer isso apenas alteramos o parâmetro **kind** do método `.plot`. Vamos plotar um [histograma](#) rodando a célula a seguir.

```
avaliacoes.query("filmeId == 1")["nota"].plot(kind='hist')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fe02669ea58>



Legal, agora temos uma visualização muito mais agradável de analisar. Compare com o gráfico de linhas que você viu antes, qual você acha melhor para análise?

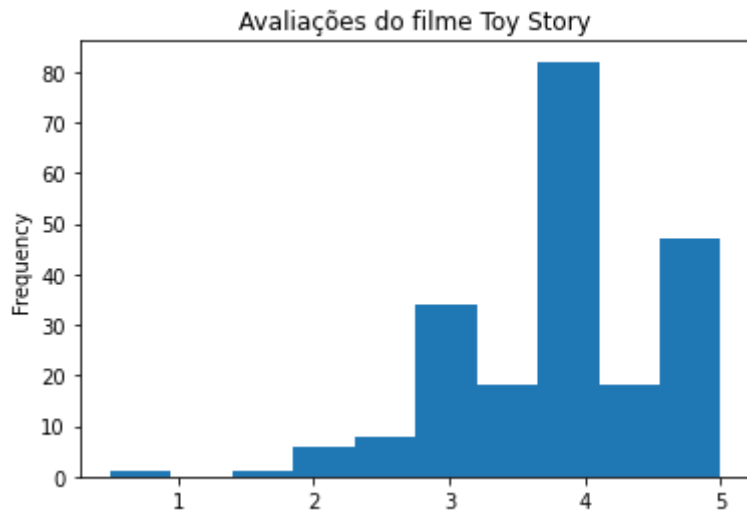
P.S: Deixar de usar o gráfico de linhas, não significa que seja uma visualização ruim. Apenas quando os dados não têm características ideais para serem visualizados como um *line plot*, agora pense em [temporal](#). **Você acha que o gráfico de linhas ainda seria uma má ideia?**

Antes de analisar o histograma de outros filmes, quero colocar um título na imagem. Vamos ver como fazer isso!

```
avaliacoes.query("filmeId == 1")["nota"].plot(kind='hist',
                                              title="Avaliações do filme Toy Story")
```



<matplotlib.axes._subplots.AxesSubplot at 0x7fe0261d8da0>

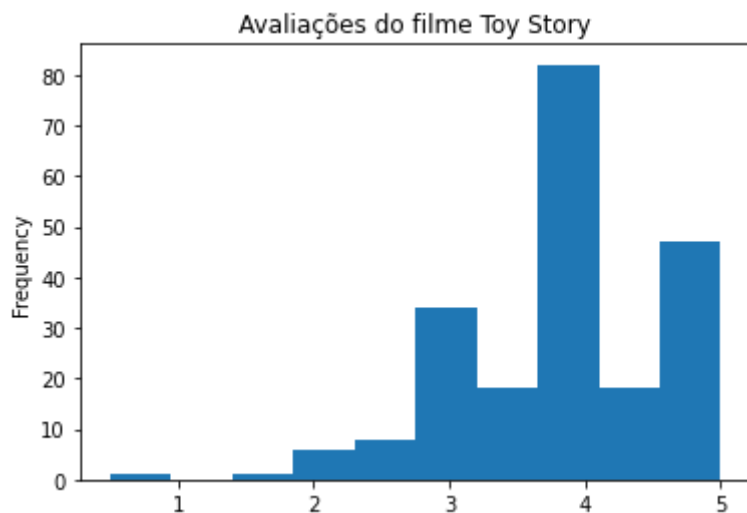


Claro que python tem outras ferramentas muito poderosas para manipular gráficos, uma delas é c
Que tal experimentar um pouquinho esta poderosa ferramenta?

Vamos importar a lib e adicionar título no gráfico usando o matplotlib, veja como fica na célula a s

```
import matplotlib.pyplot as plt
```

```
avaliacoes.query("filmeId == 1")["nota"].plot(kind='hist')
plt.title("Avaliações do filme Toy Story")
plt.show()
```



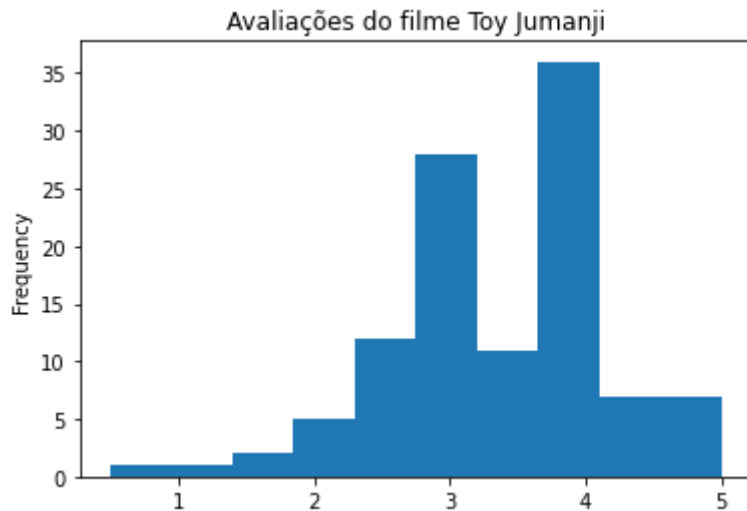
Agora que aprendemos a criar um histograma e manipular os gráficos, vamos plotar informações
realizar uma análise desses gráficos?

Vamos plotar o histograma do filme Jumanji e da animação Liga da justiça: Doom.

```
avaliacoes.query("filmeId == 2")["nota"].plot(kind='hist',
                                                title="Avaliações do filme Toy Jumanji")
```



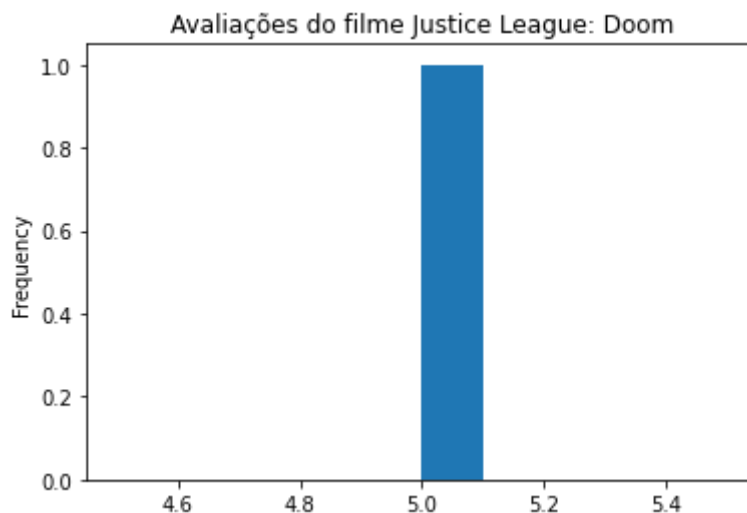
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0260d72b0>
```



```
avaliacoes.query("filmeId == 102084")["nota"].plot(kind='hist',
                                                    title="Avaliações do filme Justice Leag
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe02604b710>
```



Agora que temos os gráficos, chegou a hora de analisar.

A primeira coisa que preciso saber é o que cada eixo do meu gráfico significa. Então, eixo **x** mostra o eixo **y** a frequência das notas (quantas vezes determinada nota foi dada).

Entendido nosso gráfico, vamos contextualizar o cenário que estamos analisando:

- Temos 3 filmes, dois muito populares (Toy story e Jumanji) e outro que nenhuma pessoa pra da aula conhecia (animação da liga da justiça). O ponto que chamou a atenção, foi que a an média de nota maior que dois filmes, aparentemente mais popular, Jumaji e Toy Story. **Será um filme tão bom assim?**

Dado esse cenário a primeira coisa que me chama a atenção é a animação da liga da justiça ter n a 5. Ao analisar o histograma do respectivo filme, verificamos que ele só teve uma avaliação igual evidente que a **quantidade de votos é um aspecto importante na avaliação das médias**. Com aper não conseguimos garantir que o filme é realmente bom, tornando a avaliação muito "volátil". Imag

Justiça receba mais uma avaliação, com nota 0, assim a média seria 2.5. Apenas com mais essa passaria a ser considerada um "pior" que Jumanji e Toy Story.

Outro ponto interessante é comparar o histograma de Toy Story e Jumanji, ambos tem médias "re próximas". Mas repare que a distribuição de notas são diferentes, Toy Story recebe mais notas 5 e outra nota, enquanto Jumanji recebe mais notas 4 e 3, assim concluímos que a **distribuição das n** **fator importante na avaliação das médias**. (Se ficar alguma dúvida sobre esse tema reveja o exem apresenta no final na aula)

Com isso nós fechamos a nossa primeira aula do **#quarentenadados**, viu quanta coisa aprendem isso em prática?

Crie seu próprio notebook, reproduza nossa aula e resolva os desafios que deixamos para vocês

Até a próxima aula!

Desafio 1 do [Paulo Silveira](#)

O Paulo fez uma análise rápida e disse que tem 18 filmes sem avaliações, será que ele acertou?

Determine quantos filmes não tem avaliações e quais são esses filmes.

Desafio 2 do [Guilherme Silveira](#)

Mudar o nome da coluna nota do dataframe **filmes_com_media** para nota_média após o join.

Desafio 3 do [Guilherme Silveira](#)

Colocar o número de avaliações por filme, isto é, não só a média mas o TOTAL de votos por filme.

Desafio 4 do [Thiago Gonçalves](#)

Arredondar as médias (coluna de nota média) para duas casas decimais.

Desafio 5 do [Allan Spadini](#)

Descobrir os generos dos filmes (quais são eles, únicos). (esse aqui o bicho pega)

Desafio 6 da [Thais André](#)

Contar o número de aparições de cada genero.

Desafio 7 do [Guilherme Silveira](#)

Plotar o gráfico de aparições de cada genero. Pode ser um gráfico de tipo igual a barra.

▼ Aula 02

Nesta aula vamos estudar com mais profundidade as técnicas de centralidade, conhecer algumas visualização de dados e o famoso Boxplot.

Para inciar vamos precisar resolver alguns dos desafios deixados na **aula 01** (Caso não tenha tentado desafios, recomendo tentar algumas vezes antes de olhar as repostas). Começando pelo exercício precisamos segregar os gêneros de cada um dos filmes contidos na base de dados do **Movie Len**

Vamos lembrar como os dados estavam configurados.

```
filmes.head()
```



	filmeId	titulo	generos
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
1	2	Jumanji (1995)	Adventure Children Fantasy
2	3	Grumpier Old Men (1995)	Comedy Romance
3	4	Waiting to Exhale (1995)	Comedy Drama Romance
4	5	Father of the Bride Part II (1995)	Comedy

Temos os títulos e uma coluna com os respectivos gêneros, todos em uma única coluna, cada *label* um | (Adventure|Children|Fantasy) sendo do tipo *string*.

Para solucionar nosso problema precisamos separar cada um dos gêneros para então realizar a c várias formas de resolver este problema, por exemplo, desde métodos inputs das *strings* até as i estamos usando o pandas já temos algo para facilitar nosso processamento dos dados.

Vamos aplicar o método e logo em seguida explicar a saída geranda.

```
filmes["generos"].str.get_dummies('|')
```



Nossa, uma linha de código gerou essa tabelona cheia de linhas, colunas e números.

Como você percebeu a saída é um [DataFrame](#), cada linha corresponde a respectiva linha da coluna corresponde a um gênero (repare que cada gênero **único** virou uma coluna no DF). O que vem perguntando é como os valores **0/1** são preenchidos?.

Para explicar, vamos pegar os gêneros do filme **Jumanji, Adventure|Children|Fantasy**, na coluna de gêneros (dataframe gerado por `filmes["generos"].str.get_dummies('|')`) o valor será **1**, para todos os gêneros, que não são gêneros do filme Jumanji, vale **0**. Em suma, se o nome da coluna pertence a respectivo filme, o valor será **1** caso contrário 0 (Se ainda não ficou claro, pegue alguns filmes e compare com a tabela anterior).

Até aqui resolvemos uma parte do problema, agora precisamos somar quantos **1** cada coluna tem.

```
filmes["generos"].str.get_dummies('|').sum()
```



Ótimo, resolvemos o desafio e agora temos quantas vezes cada gênero aparece. Assim, fica fácil perguntar como, qual o gênero com mais filmes produzidos? Qual o menos? Qual o segundo? (Ler

dado está restrito as informações do movie lens)

Se você tentou reponder, deve ter notado que não foi tão fácil assim, as informações não estão or
hora você precisa percorrer a tabela para fazer comparações. Nós podemos melhor isso ordenan

```
filmes["generos"].str.get_dummies('|').sum().sort_values(ascending=False)
```



Maravilha, agora tudo ficou mais fácil!

Conseguimos responder as perguntas anterior sem grandes dificuldades. Mas ainda podemos me
de expor nossa informação, não acha?

Que tal uma imagem para visualizar? (Desafio 07 da aula 01)

```
filmes["generos"].str.get_dummies('|').sum().sort_values(ascending=False).plot()
```



Iniciamos com o plot padrão do pandas, e como percebemos não adianta só plotar uma imagem, sentido para a informação que queremos analisar, um gráfico de linhas não está fazendo muito se Temos um gráfico muito conhecido que sempre encontramos por aí, o famoso gráfico de pizza ou Já que ele é tão famoso talvez seja uma boa ideia tentar!

```
filmes["generos"].str.get_dummies('|').sum().sort_values(ascending=False).plot(
    kind='pie',
    title='Categorias de filmes e suas presenças relativas',
    figsize=(8,8))
plt.show()
```



E aí o que você achou?

Algo que fica evidente neste gráfico é que **Drama, Comedy, Thriller, e Action** tem proporções "grar qualquer outra análise fica complicada.

Primeiro, as cores começa a se repetir e isso não é o ideal.

Segundo, repare nos gêneros com menos filmes, consegue tirar alguma informação de lá? é muito

Quarto, vamos tentar comparar **thriller e Action**, qual está presente em mais filmes? Difícil respon estamos trabalhando com gráficos tipo esse fazemos comparações entre área, não somos bons i

Por fim, o importante de uma visualização é que ela seja "**transparente**" ao intuito de nossa análise querendo analisar as informações de quantidade, comparando as labels de forma geral e evidênci

clara as diferenças entre elas (proporções).

Portanto, o gráfico de pizza não torna as comparações claras, sendo assim uma má ideia.

Vamos construir juntos uma solução mais adequada!

```
filmes["generos"].str.get_dummies('|').sum().sort_values(ascending=False).plot(
    kind='bar',
    title='Filmes por categoria',
    figsize=(8,8))
plt.show()
```



Mudamos da pizza para a barra, alterando apenas o parâmetro kind do método.

Veja como o gráfico de barra torna a análise mais simples, logo de cara a diferença entre **Drama** e comparado aos demais gêneros fica evidente. No gráfico de pizza era super difícil comparar **Thrill** a comparação ficou fácil e conseguimos perceber o quão perto estão uma da outra.

A interpretação dos dados melhorou muito com essa visualização, mas podemos melhorar ainda queremos é tornar evidente os gêneros que tem a maior participação nos filmes em geral, ou seja através da imagem uma visão geral de proporcionalidade. Para tornar evidente essa informação v "semelhante" a um [mapa de calor](#).

```
import seaborn as sns
sns.set_style("whitegrid")

filmes_por_genero = filmes["generos"].str.get_dummies('|').sum().sort_values(ascending=False)
plt.figure(figsize=(16,8))
sns.barplot(x=filmes_por_genero.index,
            y=filmes_por_genero.values,
            palette=sns.color_palette("BuGn_r", n_colors=len(filmes_por_genero) + 4))
plt.show()
```



Já, já explicamos o que foi feito em toda imagem, por agora repare como a imagem passa muito mais informação. Conseguimos comparar de forma fácil entre os gêneros e através do **mapa de calor** (gêneros com um verde muito mais forte, gêneros com menor número é praticamente transparente) evidenciam labels com maior participação, médias e insignificantes. Toda essa informação em uma única imagem. Bom, agora vamos entender como foi o código.

Primeiro, não plotamos mais a imagem com o `.plot()` do pandas, vamos precisar de uma biblioteca de visualização mais poderosa para configurar nossa imagem, utilizamos o [seaborn](https://seaborn.pydata.org/).

Segundo, chamamos o barplot do **seaborn**, adicionando uma **paleta de cores** com efeito de mapa (parâmetro `palette`), no parâmetro `n_color` de `sns.color_palette()` adicionamos **+4** para que a imagem seja totalmente transparente.

Terceiro, também adicionamos o **sns.set_style("whitegrid")** para que todos os gráficos tenham a **linha X** evidente, facilitando a comparação entre as barras.

```
import seaborn as sns

filmes_por_genero = filmes["generos"].str.get_dummies('|').sum().sort_values(ascending=False)
plt.figure(figsize=(8,8))
sns.barplot(x=filmes_por_genero.index,
            y=filmes_por_genero.values,
            palette=sns.color_palette("BuGn_r", n_colors=len(filmes_por_genero) + 4))
plt.show()
```



Por fim, mudamos o tamanho da imagem com o **figsize** do método `plt.figure()`. Assim, temos uma imagem com muitas informações e agradável de analisar.

```
import seaborn as sns
sns.set_style("whitegrid")

filmes_por_genero = filmes["generos"].str.get_dummies('|').sum().sort_values(ascending=False)
plt.figure(figsize=(16,8))
sns.barplot(x=filmes_por_genero.index,
            y=filmes_por_genero.values,
            palette=sns.color_palette("BuGn_r", n_colors=len(filmes_por_genero) + 4))
plt.show()
```



Conseguimos analisar e tirar diversas conclusões trabalhando com a visualização dos gêneros. Se conseguimos utilizar visualizações para entender melhor as notas de um filme?

Vamos relembrar alguns pontos que já discutimos e nos aprofundar nas análises de notas para tirar conclusões mais sofisticadas.

Na **aula 01** calculamos as notas médias por filmes, vamos dar uma olhada no resultado.

```
filmes_com_media.head()
```



Como vimos, olhar apenas as médias pode ser um problema e para interpretar um pouco melhor o histograma das notas para comparar alguns filmes. Por exemplo, **Toy Story e Jumanji**

```
notas_do_filme_1 = avaliacoes.query("filmeId==1")["nota"]
print(notas_do_filme_1.mean())
notas_do_filme_1.plot(kind='hist')
```



```
notas_do_filme_1 = avaliacoes.query("filmeId==2")["nota"]  
print(notas_do_filme_1.mean())  
notas_do_filme_1.plot(kind='hist')
```



ToyStory e Jumanji tem médias relativamente próximas mas com comportamento de notas diferente. No nosso exemplo, as médias ajudam mas escondem informações importantes sobre os dados.

Lembra o exemplo que o Guilherme Silveira deu em aula comparando os salários de uma cidade? Para as médias dos salários não conseguimos evidenciar a desigualdade que havia entre as cidades.

```
# Cidade A  
populacao = 1000  
salario = 1100  
  
media = 1100  
  
# Cidade B  
populacao = 1000  
salario1 = 1000000
```

```
salario999 = 100
```

```
media = (salario1 * 1 + salario999 * 999) / 1000
```

```
media = 1099.00
```

P.S: Se tiver dúvidas reveja essa parte da aula e tente entender o problema da média.

Outras métrica que pode nos ajudar a interpretar melhor os dados são os quatis, principalmente a

Vamos buscar dois filmes com médias muito mais próximas que Toy Story e Jumanji, para análise além das médias.

```
filmes_com_media.sort_values("nota", ascending=False)[2450:2500]
```



Bom, ordenando os filmes pela nota médias e [fatiando](#) os dados entre 2450 e 2500, temos uma re
médias são semelhantes e provavelmente não tem apenas um único voto. Vamos comparar o filme
filmId=919 e **Little Miss Sunshine* **filmId=46578**.

Para não precisar copiar e colar toda hora o plot dos gráficos vamos criar nossa primeira função, i
apenas o FilmId e temos as informações desejadas.

```
def plot_filme(n):  
    notas_do_filme = avaliacoes.query(f"filmeId=={n}")["nota"]  
    notas_do_filme.plot(kind='hist')  
    return notas_do_filme.describe()
```

Definimos nossa [função plot em python](#) e repare que estamos usando **F-string** para fazer a interp
se tiver dúvida veja essa [explicação no fórum da alura](#).

Agora precisamos chamar a função!

```
#Mágico de Oz  
plot_filme(919)
```



A função `plot`, além de gerar o histograma também retorna algumas estatísticas. Vamos chamar a função para o filme *Little Miss Sunshine*.

```
plot_filme(46578)
```



Ótimo, agora com essas informações conseguimos comparar melhor ambos os filmes. Analisando os dados vemos que muitas pessoas realmente amam **Wizard of Oz** (notas 5), mas também temos pessoas que deram uma nota baixa (notas 1). Quando comparamos com a histograma temos um do **Little miss sunshine** que os resultados se concentra entre valores medianos (notas 2.5-4).

O que confirma nossa análise aqui é comparar os **25% 50% e 75%**. 50% é o valor da mediana, e analisando a mesma mediana, mas 25% e 75% são diferentes. Se você lembra lá da estatística esses são os **1º**

Olha, mesclar os gráficos com as estatísticas ajuda a interpretar melhor os dados. Mas o que precisamos é uma imagem que nos ajude a interpretar os dados ainda melhor, o gráfico que nos ajuda neste caso é o boxplot. Vamos adaptar nossa função para conseguir plotar o boxplot e interpretá-lo.

```
def plot_filme(n):
    notas_do_filme = avaliacoes.query(f"filmeId=={n}")["nota"]
    notas_do_filme.plot(kind='hist')
    plt.show()
    print('\n')
    notas_do_filme.plot.box()
    plt.show()
    return notas_do_filme.describe()
```



```
plot_filme(919)
```



E aí, viu como é simples criar um [boxplot com o pandas](#)?

Apenas chamamos o método `.plot.box()`, agora o que precisamos fazer é interpretar este gráfico. Vamos focar primeiro na "caixa": a linha verde que divide a caixa em dois é a mediana (compare com a gerada pelo `describe()`), a parte superior da caixa é o 3º Quartil (75%) e a parte inferior é o 1º Quartil. Agora repare nos limites inferior e superior, representados pelas extremidades em preto. Por coincidência, na imagem os limites inferior e superior são equivalentes ao ponto de máximo e mínimo, mas nem sempre, pois esses limites superior e inferior são calculados e dependem de Q1 e Q3. Algumas vezes os limites coincidem com os extremos das "caixas" e isso geralmente ocorre quando temos uma quantidade pequena de dados. Como tivemos sobreposição do limite superior, vamos calcular o boxplot de outro filme, para análise.

```
plot_filme(46578)
```



Olha que legal, diferente do primeiro boxplot, neste os limites superiores não se sobrepõe e temos mais, no caso temos essa bolinha localizada em $y=1$. A "bolinha" chamamos de valor discrepante, limites inferior e superior (chamamos na aula de [outliers](#), existem várias formas de calcular os outliers, no nosso caso esses são os outliers do boxplot).

Não vamos entrar em todos os detalhes do boxplot mas recomendo a explicação do [wikipedia](#), ela está cheia de exemplo e imagens para facilitar o entendimento.

Agora comparando os boxplot dos dois filmes deixa muito mais evidente as diferenças entre elas, complexo olhando só médias e outras informações separadas.

Embora melhoramos muito nossa qualidade de análise ainda temos mais um ponto. Estamos com o boxplot dos filmes, mas eles estão em imagens separadas, vamos juntar vários boxplot em uma única imagem.

como podemos fazer isso usando o **seaborn**, para aprendermos outra forma de plotar boxplot!

```
sns.boxplot(data = avaliacoes.query("filmeId in [1,2,919,46578]"), x = "filmeId", y = "nota")
```



Chamamos o `sns.boxplot()` passando três parâmetros. Parâmetro `dados` é um dataframe das notas com Toy Story, Jumanji, Wizard of Oz e Little miss sunshine (usamos o `.query()` para selecionar o filme e `y` as respectivas notas. Agora conseguimos comparar as notas dos filmes de forma muito mais fácil e tente realiza a análise aí na sua casa!

Com isso nós fechamos nossa segunda aula do **#quarentenadados**, viu quanta coisa aprendemos com isso em prática?

Crie seu próprio notebook, reproduza nossa aula e resolva os desafios que deixamos para vocês

Até a próxima aula!

Desafio 1 do [Guilherme Silveira](#)

Rotacionar os ticks (os nomes dos generos) do gráfico de barras verdes (o último), de forma a deixar o gráfico mais legíveis.

Desafio 2 do [Paulo Silveira](#)

Encontrar vários filmes com médias próximas e distribuições diferentes, use a função `plot_filmes()`

Desafio 3 do [Paulo Silveira](#)

Criar o boxplot dos 10 filmes com mais votos (não é com maior média, é com mais votos!). Não apenas, também analise e tente tirar conclusões.

Desafio 4 do [Guilherme Silveira](#)

Configurar a visualização do boxplot gerado pelo seaborn (último boxplot plotado na aula). Configurar o nome dos filmes nos ticks.

Desafio 5 do [Allan Spadini](#)

Calcular moda, média e mediana dos filmes. Explore filmes com notas mais próximas de 0.5, 3 e 5.

Desafio 6 da [Thais André](#)

Plotar o boxplot e o histograma um do lado do outro (na mesma figura ou em figuras distintas, mas no mesmo notebook).

Desafio 7 do [Thiago Gonçalves](#)

Criar um gráfico de notas médias por ano (média geral considerando todos os filmes lançados na

▼ Aula 3

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Não esqueça de compartilhar a solução dos seus desafios com instrutores, seja no twitter ou linkedin. Boa sorte!