



# Visualizando o Sucesso: Análise Gráfica de Dados



# Objetivos da aula

- Visualização de dados de filmes com gráficos! Nesta aula vamos explorar dados de filmes por meio de gráficos, usando matplotlib e seaborn.
- Ao final da aula você será capaz de:
  - Criar scatterplots e gráficos de barras
  - Analisar relação entre orçamento e receita
  - Contar e visualizar frequência de filmes por gênero
  - Interpretar visualmente padrões de sucesso
  - Separar responsabilidades e organizar código para visualização

# Instalando bibliotecas

- Antes de criar gráficos, precisamos instalar as bibliotecas que permitem gerar visualizações e manipular dados:
  - **matplotlib**: gráficos básicos e personalizáveis
  - **seaborn**: gráficos estatísticos avançados
  - **pandas**: manipulação de tabelas e dados estruturados

# Instalando bibliotecas

- Criamos um ambiente virtual para isolar dependências do projeto.
- Instalamos matplotlib, seaborn e pandas dentro do ambiente virtual para gerar gráficos e manipular dados sem afetar o sistema global.

# Instalando bibliotecas

- No terminal do VS Code execute:

```
● ● ●  
# Criar e ativar um ambiente virtual  
python -m venv venv  
# No Windows  
venv\Scripts\activate  
# No Mac/Linux  
source venv/bin/activate  
  
# Instalação das bibliotecas necessárias  
pip install matplotlib seaborn pandas
```

# Importando bibliotecas

- Depois importamos as bibliotecas principais para criação de gráficos:
  - **matplotlib** (básico) e **seaborn** (visualizações estatísticas avançadas).



```
import matplotlib.pyplot as plt  
import seaborn as sns
```

# Preparando dados para gráfico

- Extraímos listas de dados numéricos para criar gráficos comparativos de orçamento e receita.



```
orcamentos = [f.orcamento for f in filmes]
receitas = [f.receita for f in filmes]
titulos = [f.titulo for f in filmes]
```

# Scatterplot básico

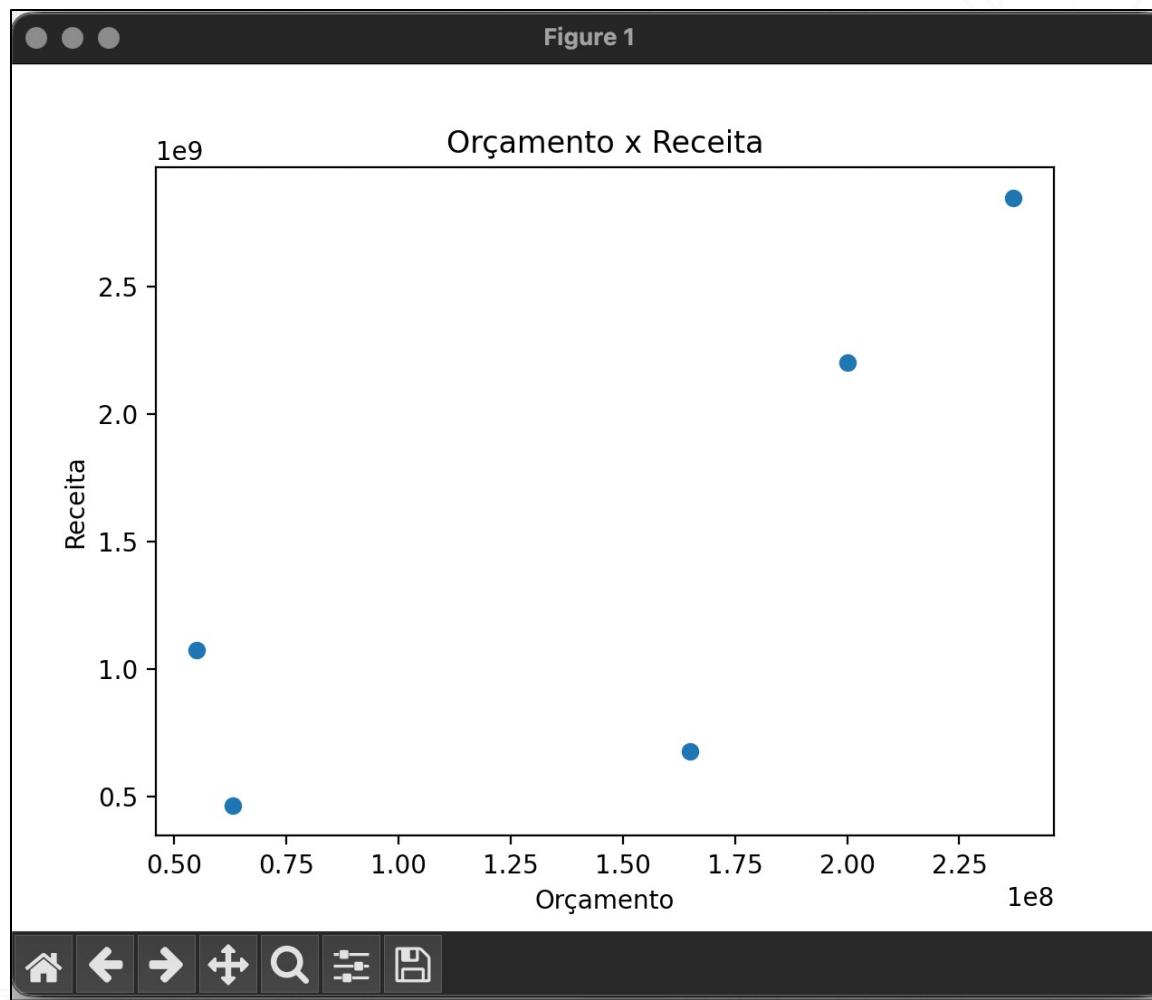
- Scatterplot permite visualizar a relação entre duas variáveis numéricas (orcamento e receita).



```
plt.scatter(orçamentos, receitas)
plt.xlabel("Orçamento")
plt.ylabel("Receita")
plt.title("Orçamento x Receita")
plt.show()
```

# Scatterplot básico

- Resultado



# Scatterplot básico

- Código completo



```
from filme import Filme
import matplotlib.pyplot as plt
import seaborn as sns

filmes =
    Filme("Matrix", 63000000, 465000000, "Ficção"),
    Filme("Titanic", 20000000, 2200000000, "Drama"),
    Filme("Avatar", 23700000, 2847000000, "Ficção"),
    Filme("Joker", 5500000, 1074000000, "Drama"),
    Filme("Interestelar", 16500000, 677000000, "Ficção")
]

# Preparando os dados para o scatterplot
orcamentos = [f.orcamento for f in filmes]
receitas = [f.receita for f in filmes]
titulos = [f.titulo for f in filmes]

# Criando o scatterplot
plt.scatter(orcamentos, receitas)
plt.xlabel("Orçamento")
plt.ylabel("Receita")
plt.title("Orçamento x Receita")
plt.show()
```

# Scatterplot com cores por gênero

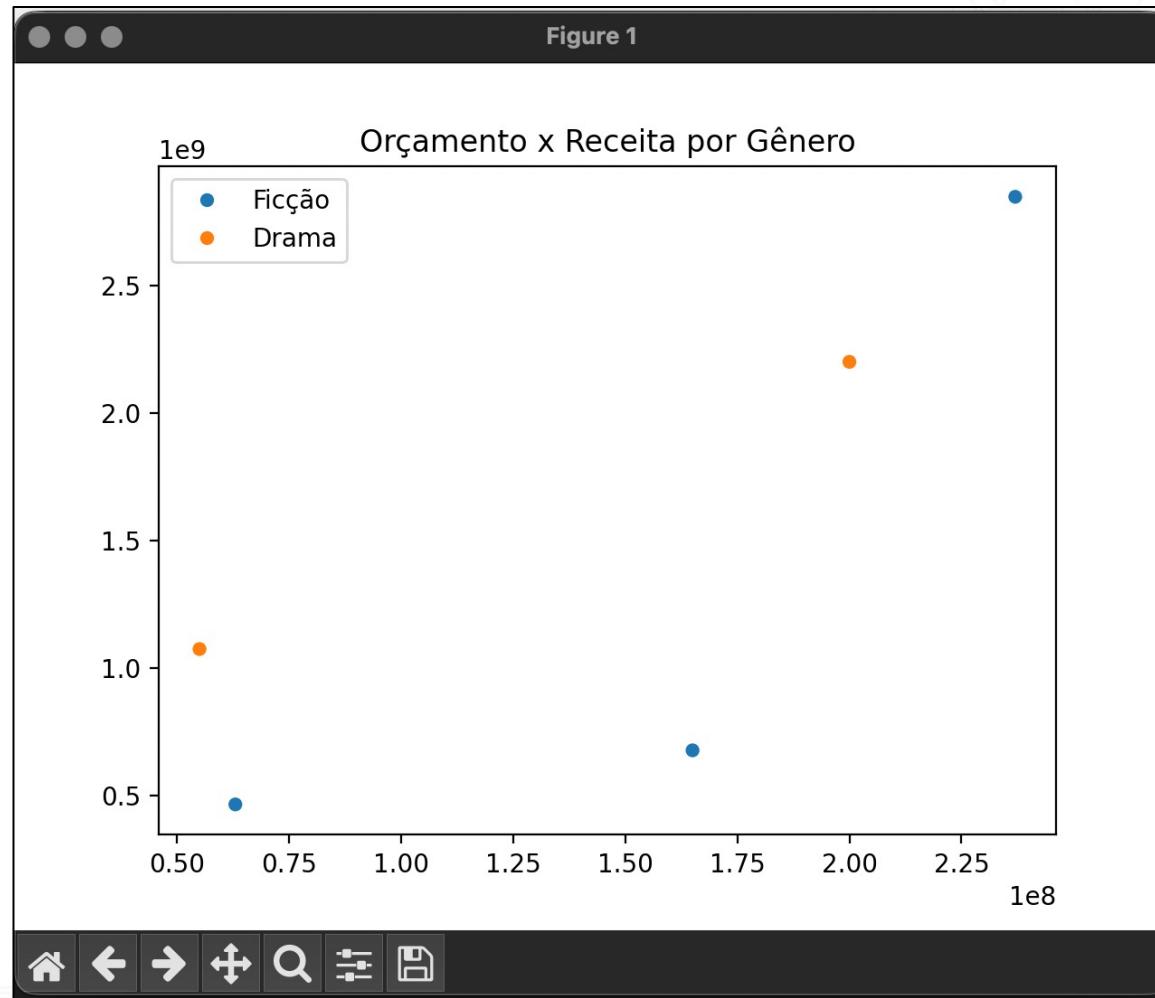
- Separamos os pontos por cores, usando o gênero como variável categórica.



```
generos = [ f.genero for f in filmes]
sns.scatterplot(x=orcamentos, y=receitas, hue=generos)
plt.title("Orçamento x Receita por Gênero")
plt.show()
```

# Scatterplot com cores por gênero

- Resultado



# Scatterplot com cores por gênero

- O parâmetro **hue** colore os pontos do gráfico de acordo com uma categoria, permitindo identificar padrões por grupo, como diferentes gêneros de filmes.
- Trata-se de um parâmetro do Seaborn usado em gráficos (como scatterplot, relplot, barplot) para colorir os pontos ou barras de acordo com uma categoria.
- 1e8 é notação científica e significa  $1 \times 10^8$ , ou seja, 100 milhões. É útil para trabalhar com números grandes de forma compacta.

# Gráfico de barras

- O Gráfico de barras a seguir mostra a frequência de filmes por categoria.

```
● ● ●  
# Preparando dados para o gráfico de barras  
generos_unicos = list(dicionario_genero.keys())  
contagem = [len(dicionario_genero[g]) for g in generos_unicos]
```

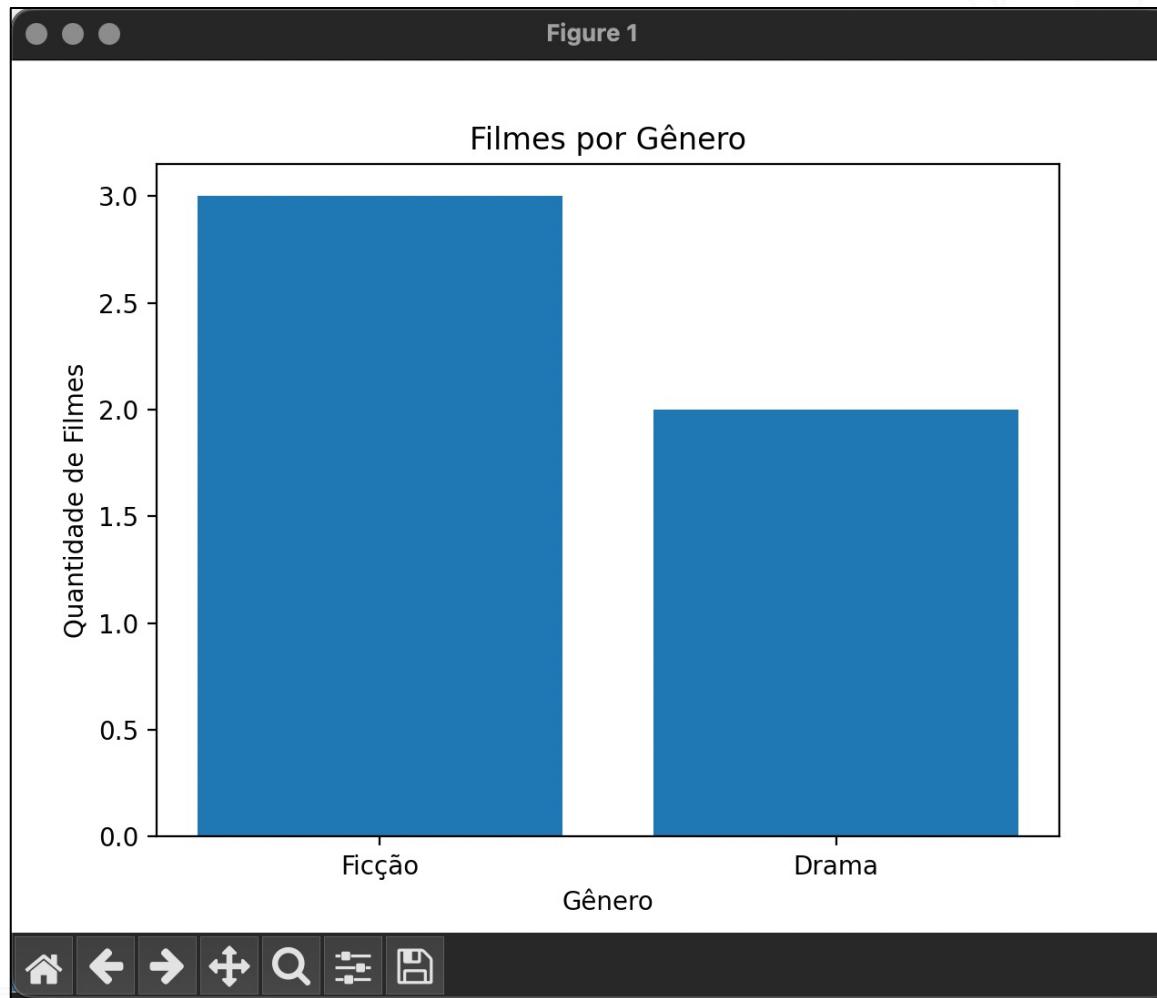
```
# Gráfico de barras  
plt.bar(generos_unicos, contagem)  
plt.xlabel("Gênero")  
plt.ylabel("Quantidade de Filmes")  
plt.title("Filmes por Gênero")  
plt.show()
```



```
# Criando dicionário de filmes por gênero  
dicionario_genero = {}  
for f in filmes:  
    if f.genero not in dicionario_genero:  
        dicionario_genero[f.genero] = []  
        dicionario_genero[f.genero].append(f)
```

# Gráfico de barras

- Resultado



# Histogram: distribuição de receitas

- Histogramas mostram como os valores de uma variável se distribuem em intervalos.

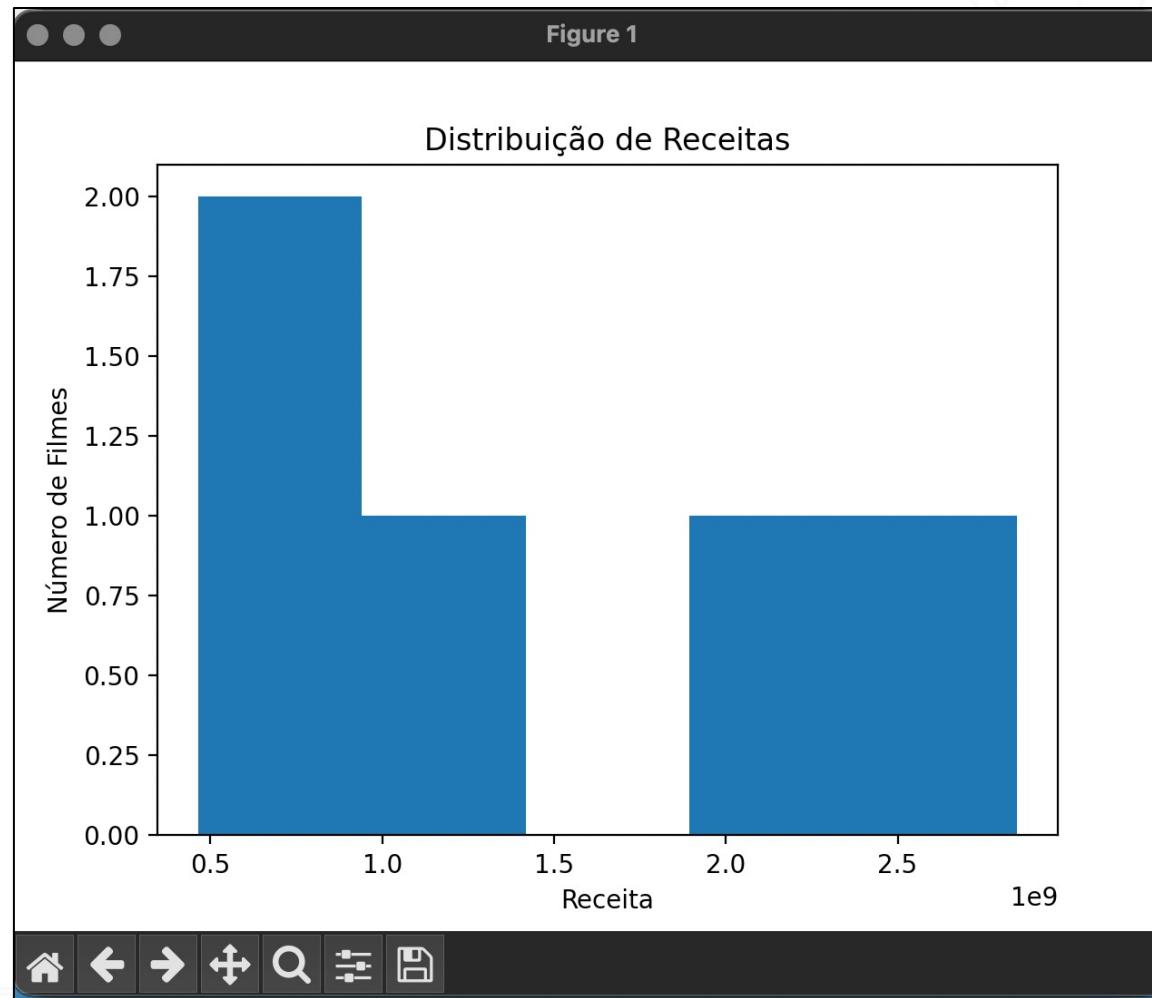


```
# Preparando listas de dados
receitas = [f.receita for f in filmes]

plt.hist(receitas, bins=5)
plt.xlabel("Receita")
plt.ylabel("Número de Filmes")
plt.title("Distribuição de Receitas")
plt.show()
```

# Histogram: distribuição de receitas

- Resultado



# Histogram: distribuição de receitas

- No contexto de histogramas no Matplotlib, o parâmetro bins define em quantos intervalos (ou “caixas”) os dados serão agrupados.
- Cada bin conta quantos valores do conjunto de dados caem naquele intervalo.
- Quanto mais bins, mais detalhado será o histograma; quanto menos, mais “grossa” a visualização.

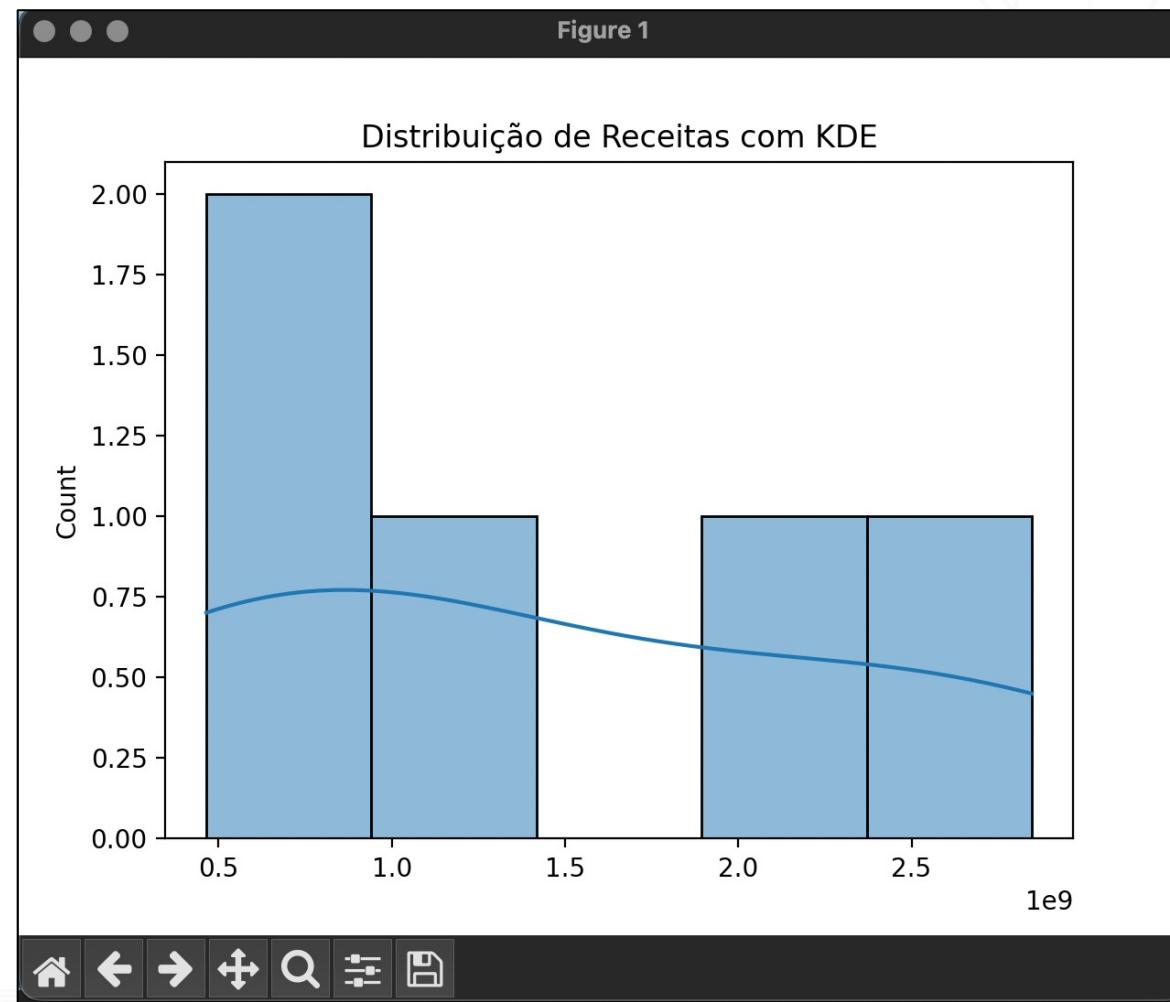
# Histograma com seaborn

- KDE adiciona uma curva suavizada sobre o histograma, mostrando densidade dos dados.

```
● ● ●  
sns.histplot(receitas, bins=5, kde=True)  
plt.title("Distribuição de Receitas com KDE")  
plt.show()
```

# Histograma com seaborn

- Resultado



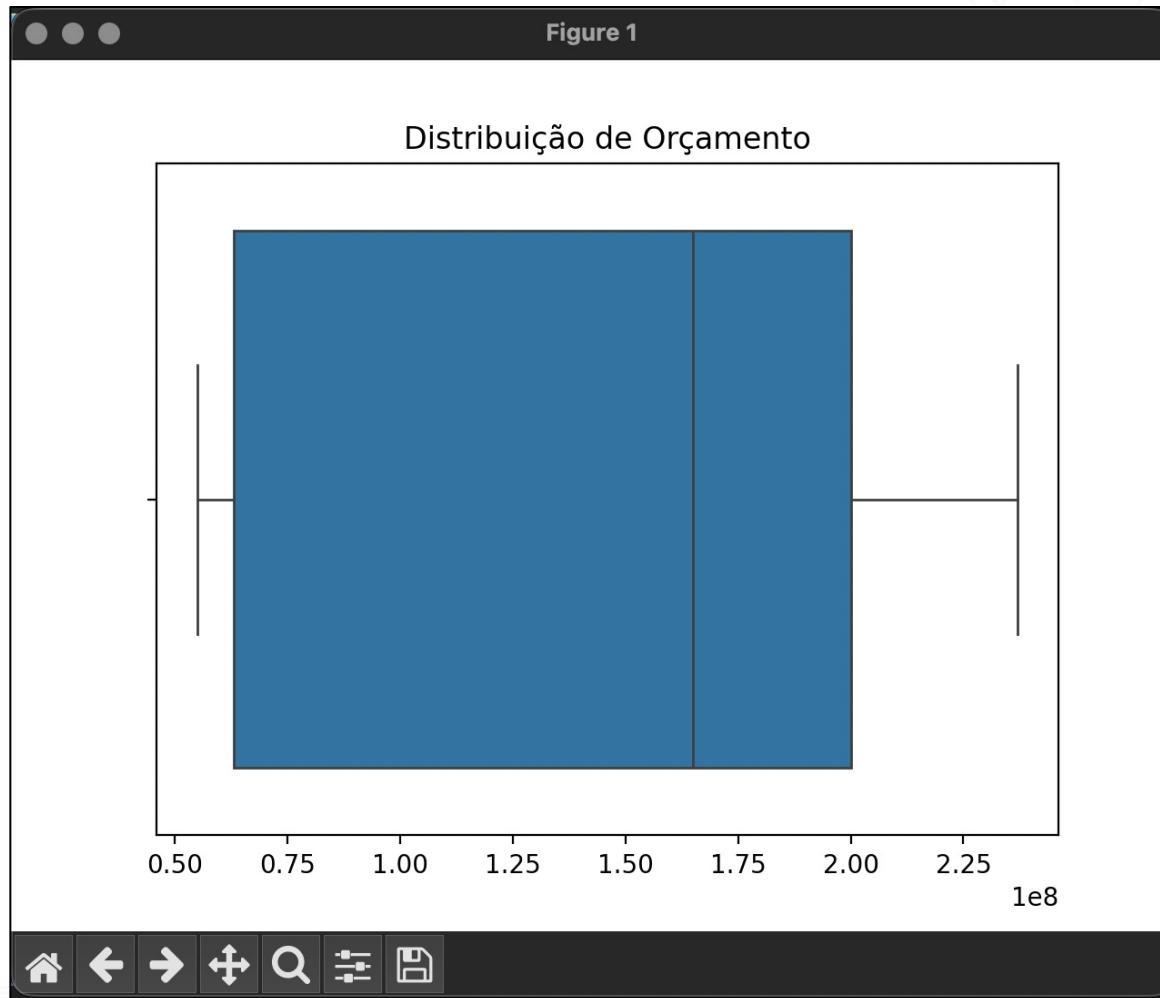
# Boxplot de orçamento

- Boxplots permitem identificar valores medianos, quartis e outliers.

```
● ● ●  
# Lista de orçamentos  
orcamentos = [f.orcamento for f in filmes]  
  
# Boxplot  
sns.boxplot(x=orcamentos)  
plt.title("Distribuição de Orçamento")  
plt.show()
```

# Boxplot de orçamento

- Resultado



# Boxplot por gênero

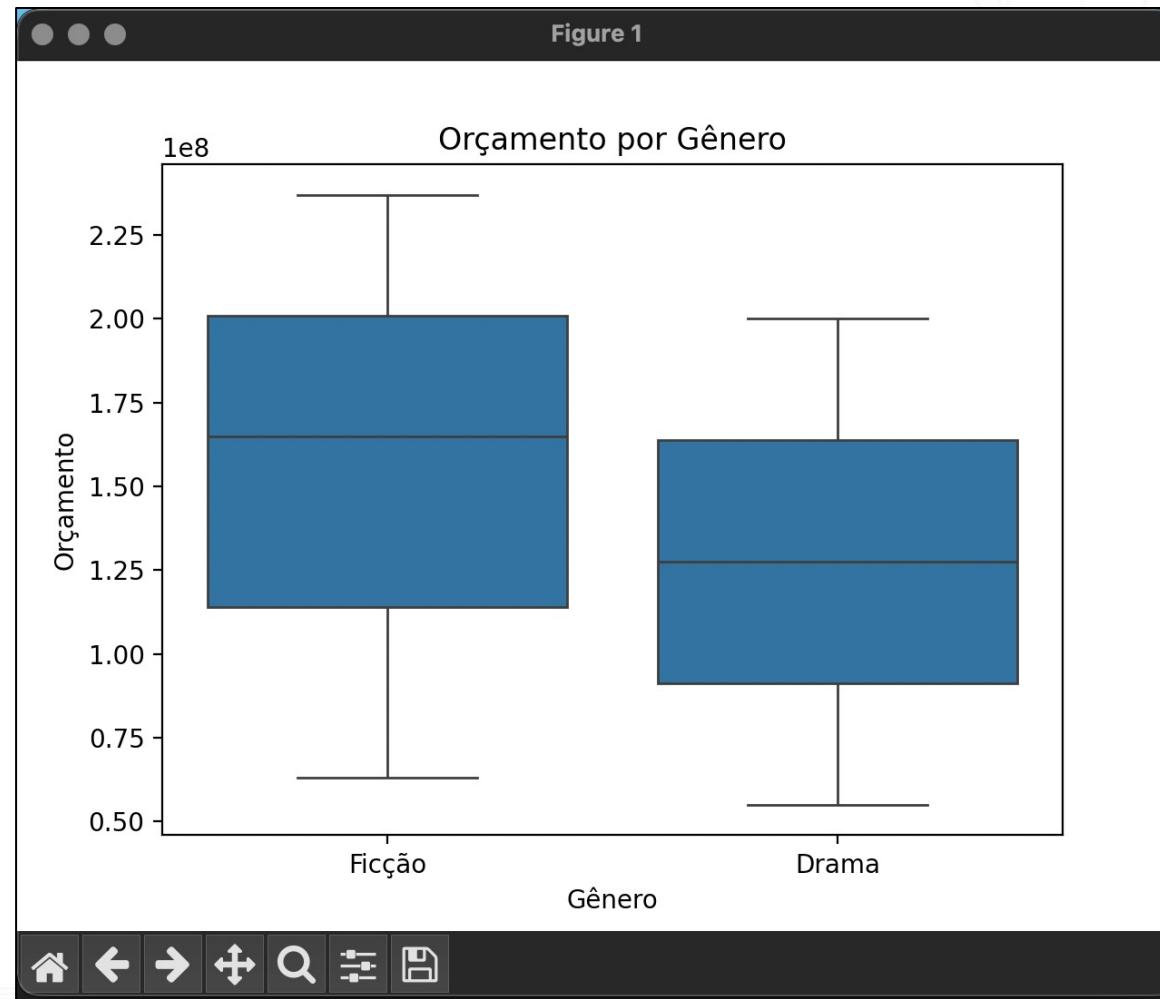
- Visualizamos a distribuição de orçamento para cada gênero.

```
# Listas de dados
generos = [f.gênero for f in filmes]
orcamentos = [f.orcamento for f in filmes]

# Boxplot de orçamento por gênero
sns.boxplot(x=generos, y=orcamentos)
plt.title("Orçamento por Gênero")
plt.xlabel("Gênero")
plt.ylabel("Orçamento")
plt.show()
```

# Boxplot por gênero

- Resultado



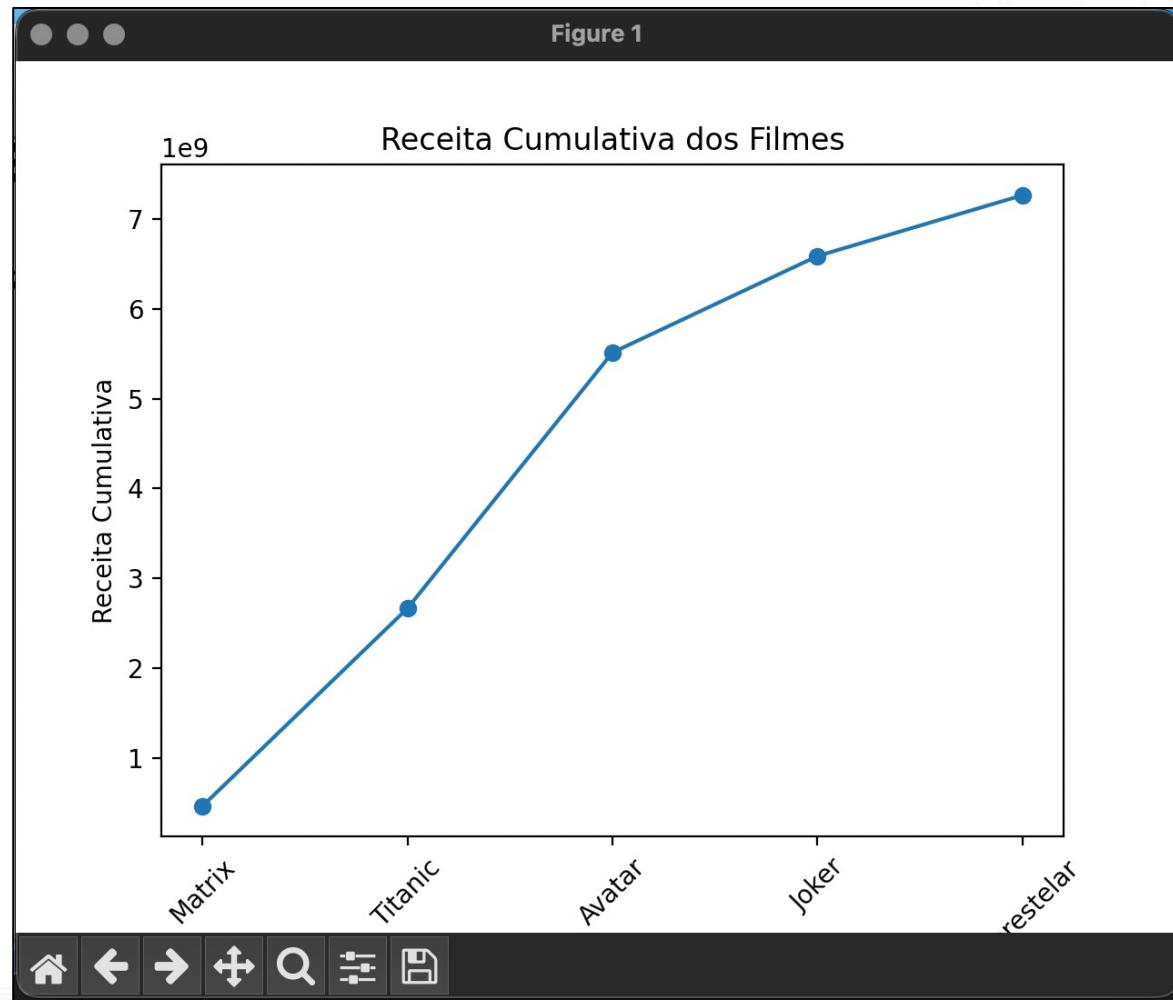
# Gráfico de linhas

- Gráfico de linhas mostra tendência ou evolução acumulada.

```
● ● ●  
# Listas de dados  
titulos = [f.titulo for f in filmes]  
receitas = [f.receita for f in filmes]  
  
# Receita cumulativa  
receita_cumulativa = [sum(receitas[:i+1]) for i in range(len(receitas))]  
  
# Gráfico de linha  
plt.plot(titulos, receita_cumulativa, marker='o')  
plt.xticks(rotation=45)  
plt.xlabel("Filmes")  
plt.ylabel("Receita Cumulativa")  
plt.title("Receita Cumulativa dos Filmes")  
plt.show()
```

# Gráfico de linhas

## ■ Resultado



# Pairplot com seaborn

- Pairplot mostra todas as combinações possíveis de variáveis numéricas e cores por categoria.



```
import pandas as pd

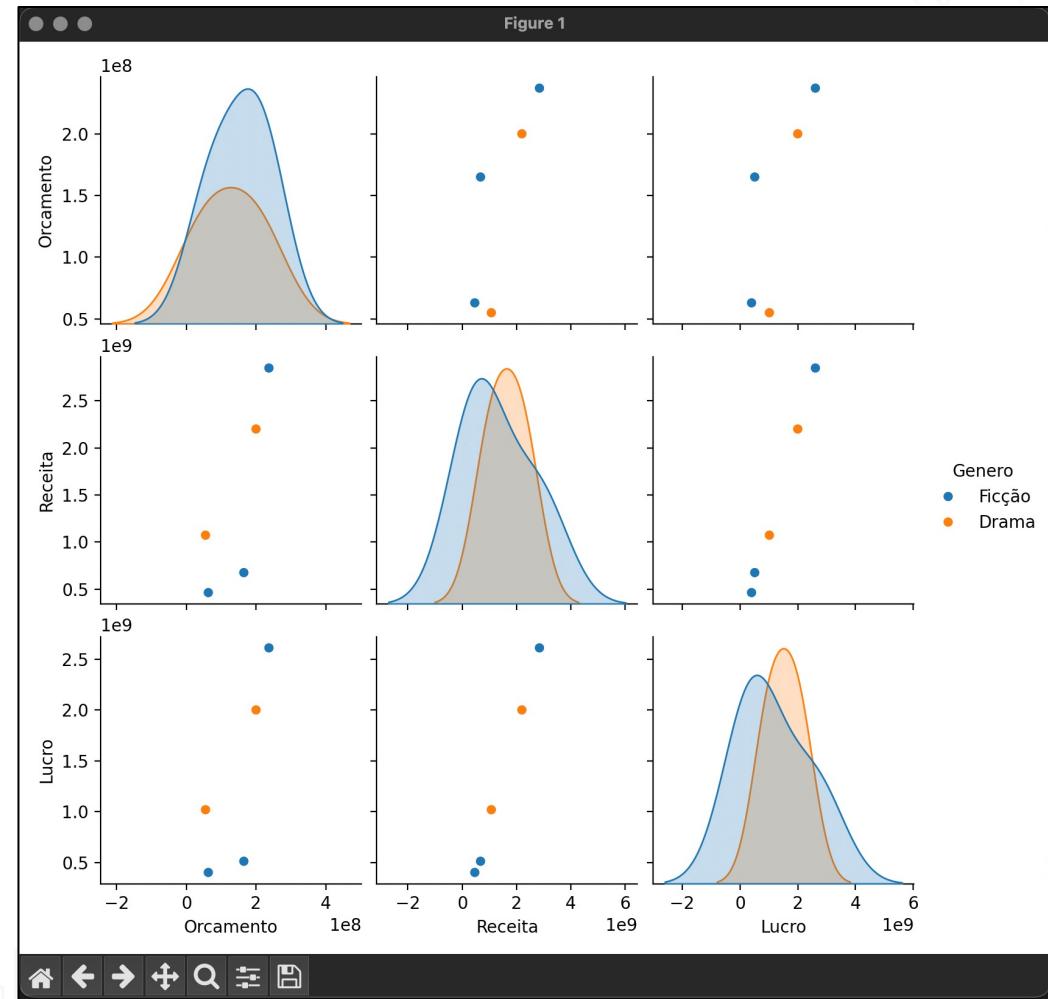
# Preparando listas de dados
orcamentos = [f.orcamento for f in filmes]
receitas = [f.receita for f in filmes]
lucros = [f.lucro() for f in filmes]
generos = [f.genero for f in filmes]

# Criando DataFrame
df = pd.DataFrame({
    "Orcamento": orcamentos,
    "Receita": receitas,
    "Lucro": lucros,
    "Genero": generos
})

# Pairplot colorido por gênero
sns.pairplot(df, hue="Genero")
plt.show()
```

# Pairplot com seaborn

- Resultado



# Gráfico de pizza: proporção de gêneros

- Gráficos de pizza mostram a participação percentual de cada categoria.

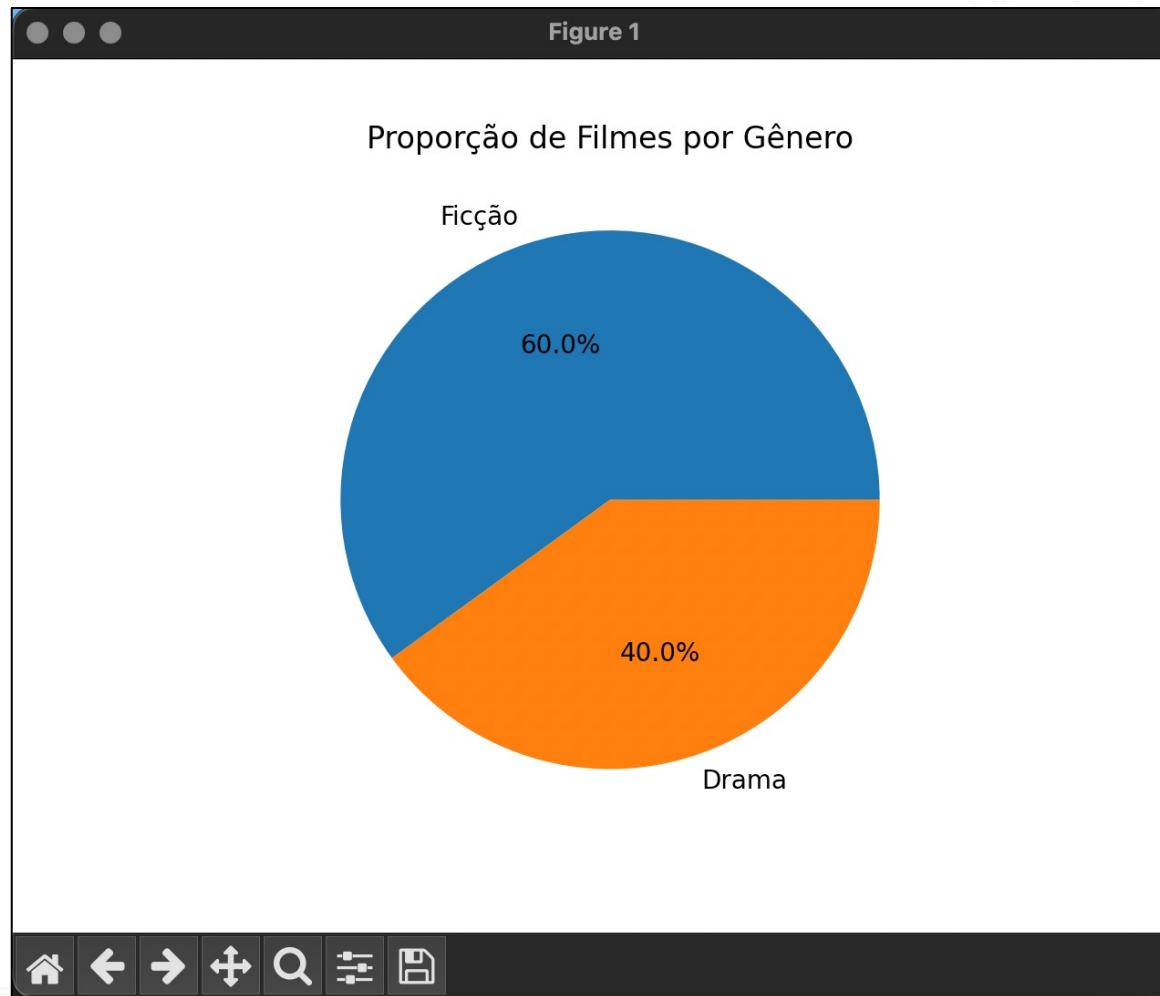


```
# Listas para o gráfico de pizza
generos_unicos = list(set(f.genero for f in filmes))
contagem = [sum(1 for f in filmes if f.genero == g) for g in generos_unicos]

# Gráfico de pizza
plt.pie(contagem, labels=generos_unicos, autopct="%1.1f%%")
plt.title("Proporção de Filmes por Gênero")
plt.show()
```

# Gráfico de pizza: proporção de gêneros

- Resultado



# Heatmap de correlação

- Heatmaps destacam visualmente relações entre variáveis numéricas.

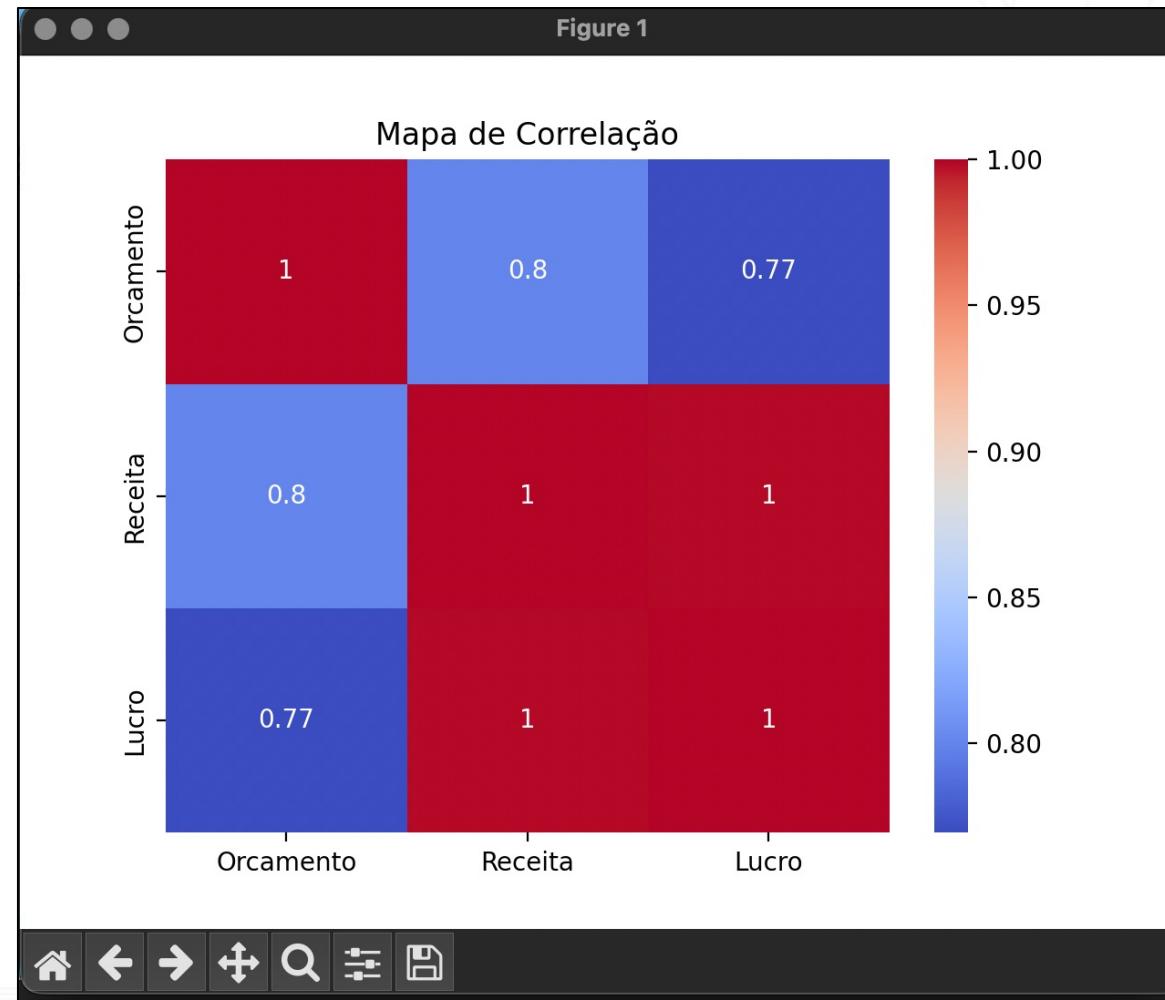


```
# Criando DataFrame
df = pd.DataFrame({
    "Orcamento": orcamundos,
    "Receita": receitas,
    "Lucro": lucros,
    "Genero": generos
})

# Heatmap de correlação
sns.heatmap(df[["Orcamento", "Receita", "Lucro"]].corr(), annot=True, cmap="coolwarm")
plt.title("Mapa de Correlação")
plt.show()
```

# Heatmap de correlação

- Resultado



# Subplots básicos

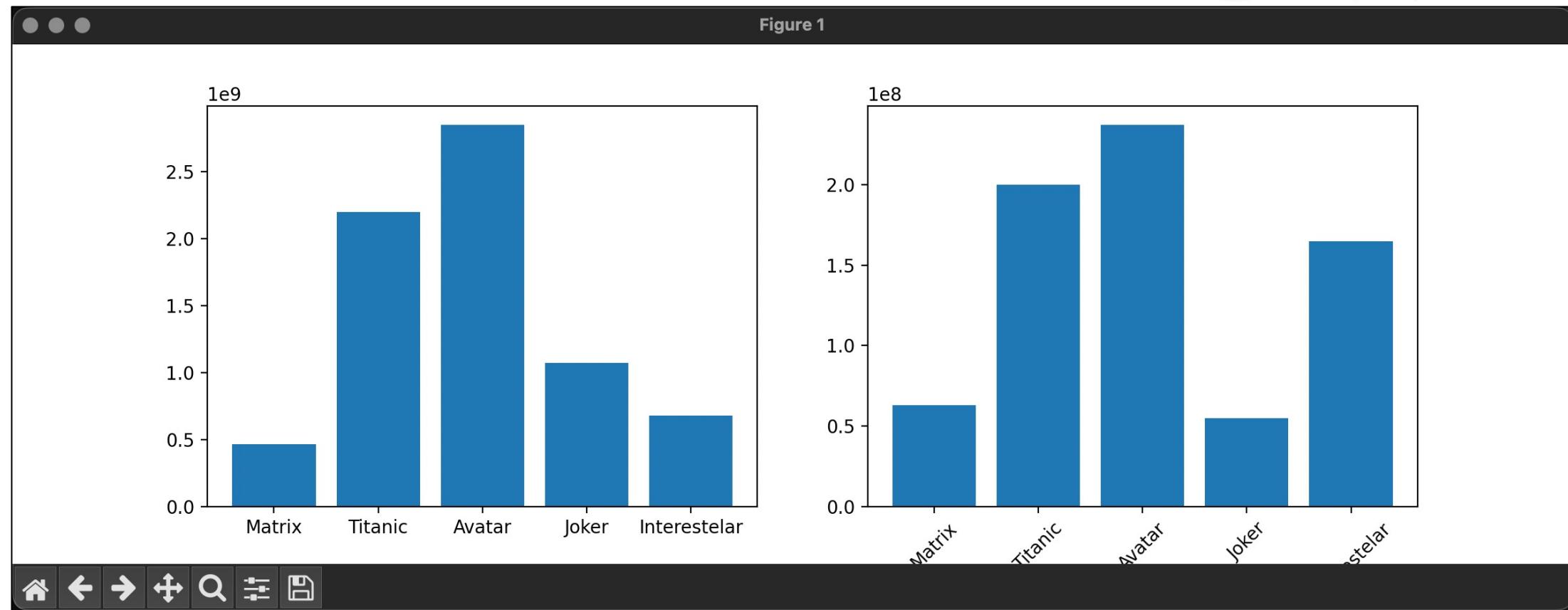
- Subplots permitem mostrar múltiplos gráficos lado a lado.



```
fig, ax = plt.subplots(1,2, figsize=(12,4))
ax[0].bar(titulos, receitas)
ax[1].bar(titulos, orcamentos)
plt.xticks(rotation=45)
plt.show()
```

# Subplots básicos

- Resultado



# Subplots com seaborn

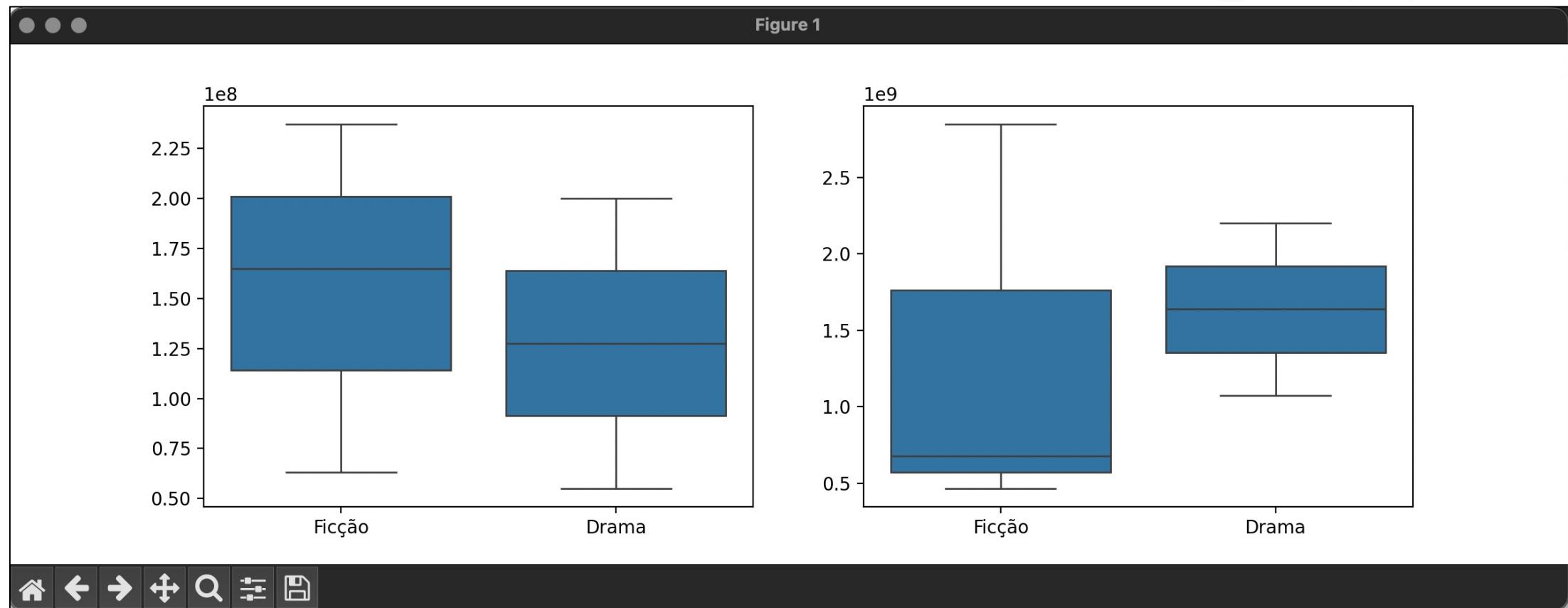
- Combina visualizações seaborn em múltiplos eixos para comparar métricas.



```
fig, ax = plt.subplots(1,2, figsize=(12,4))
sns.boxplot(x=generos, y=orcamentos, ax=ax[0])
sns.boxplot(x=generos, y=receitas, ax=ax[1])
plt.show()
```

# Subplots com seaborn

- Resultado



# Gráfico de densidade

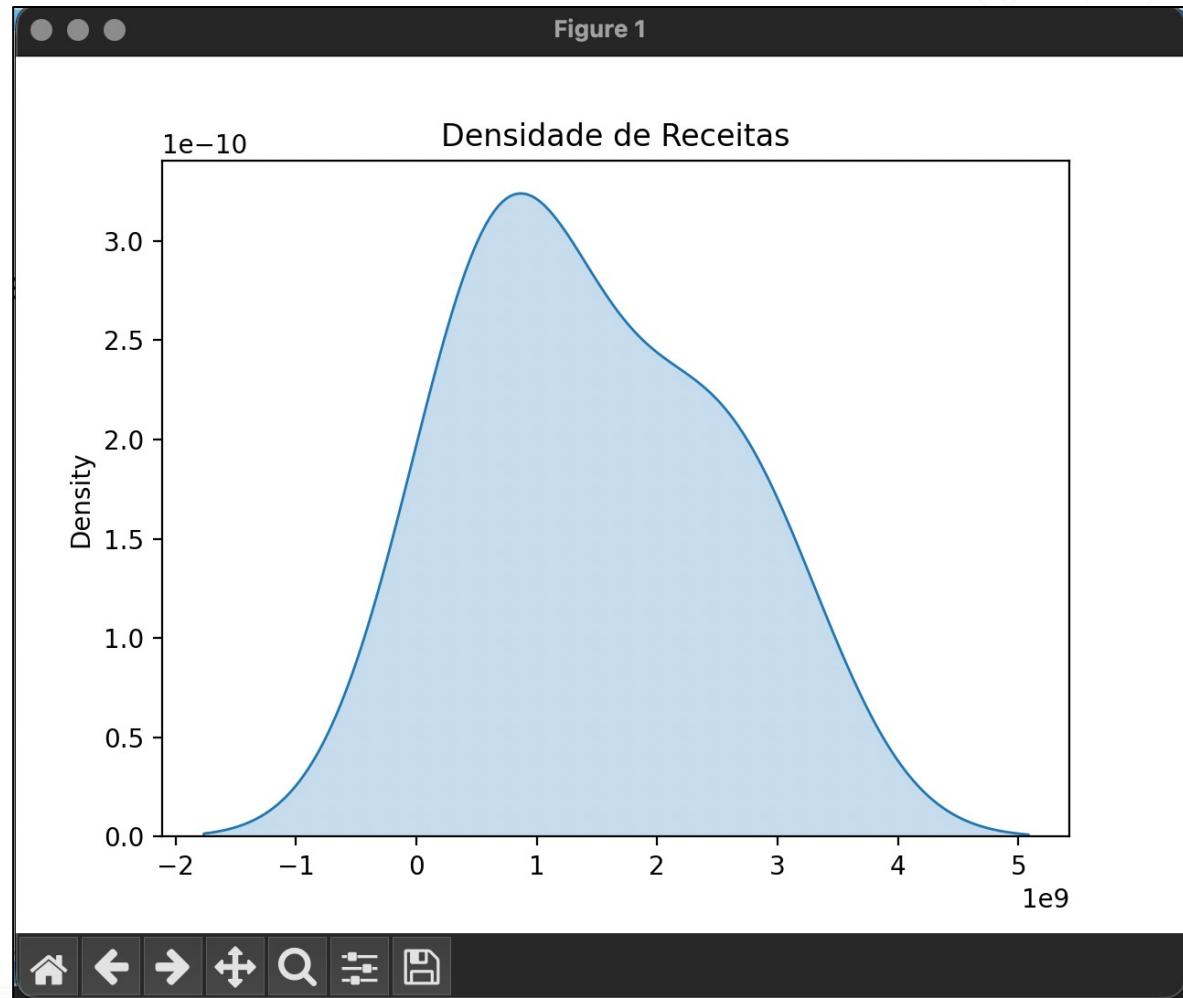
- KDE mostra a distribuição contínua das receitas, útil para análise estatística.



```
sns.kdeplot(receitas, shade=True)  
plt.title("Densidade de Receitas")  
plt.show()
```

# Gráfico de densidade

- Resultado



# Encerramento

- Você aprendeu:
  - Criar gráficos diversos com **matplotlib** e **seaborn**
  - Interpretar padrões entre orçamento, receita e lucro
  - Visualizar contagem e proporção de filmes por gênero
  - Usar cores, tamanhos e **subplots** para enriquecer a análise
  - Separar código de visualização do restante do projeto
  - Preparar os dados para análises e insights visuais



© PUC Minas • Todos os direitos reservados, de acordo com o art. 184 do Código Penal e com a lei 9.610 de 19 de fevereiro de 1998.  
Proibidas a reprodução, a distribuição, a difusão, a execução pública, a locação e quaisquer outras  
modalidades de utilização sem a devida autorização da Pontifícia Universidade Católica de Minas Gerais.