**Para saber mais: quando saber que um gráfico de pizza é eficaz?**

Muitos(as) especialistas debatem sobre a praticidade dos gráficos de setores, como o gráfico de pizza e rosca, como ferramenta de visualização de dados.

Existem estudiosos(as) que se opõem a sua utilização, como [Edward Tufte](https://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=00018S), um renomado teórico considerado pioneiro de visualização de dados e autor de vários trabalhos sobre o assunto; e [Cole Nussbaumer Knaflic](https://www.storytellingwithdata.com/blog/2011/07/death-to-pie-charts), escritora do "Storytelling com dados", uma das obras mais destacadas na área.

Apesar disso, esses gráficos são incorporados em nosso cotidiano devido à sua capacidade direta de apresentar informações e à sua facilidade de criação em diversas ferramentas de análise de dados.

É importante destacar que muitas pessoas também apontam aplicações práticas dos gráficos de pizza e rosca para situações específicas. O artigo, [When Pie Charts Are Okay (Seriously): Guidelines for Using Pie and Donut Charts](https://depictdatastudio.com/when-pie-charts-are-okay-seriously-guidelines-for-using-pie-and-donut-charts/" \t "_blank), escrito por Ann K. Emery, destaca alguns cenários nos quais os gráficos de pizza e rosca podem ser utilizados:

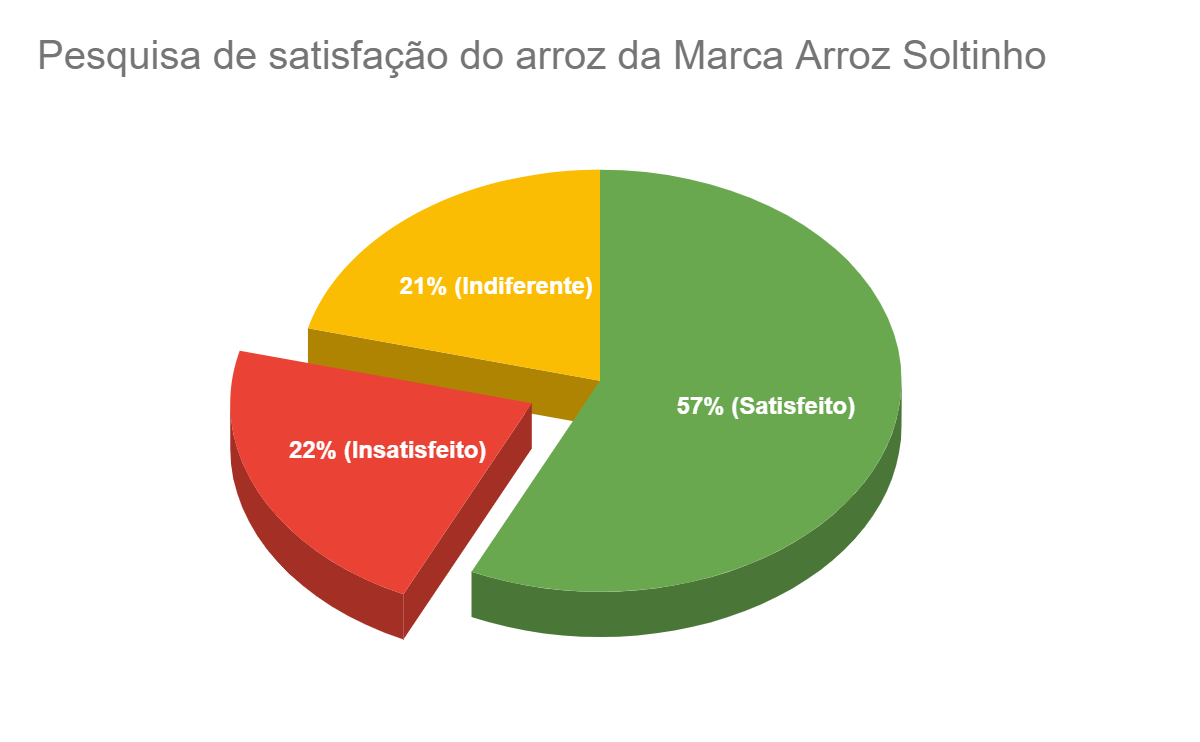
* Dados de categorias de gêneros homem/mulher/etc;
* Quantidade ou porcentagem de respostas de pesquisas de sim/não;
* Estudantes que concluíram o ensino médio a tempo ou não;
* Outros dados binários/dicotômicos.

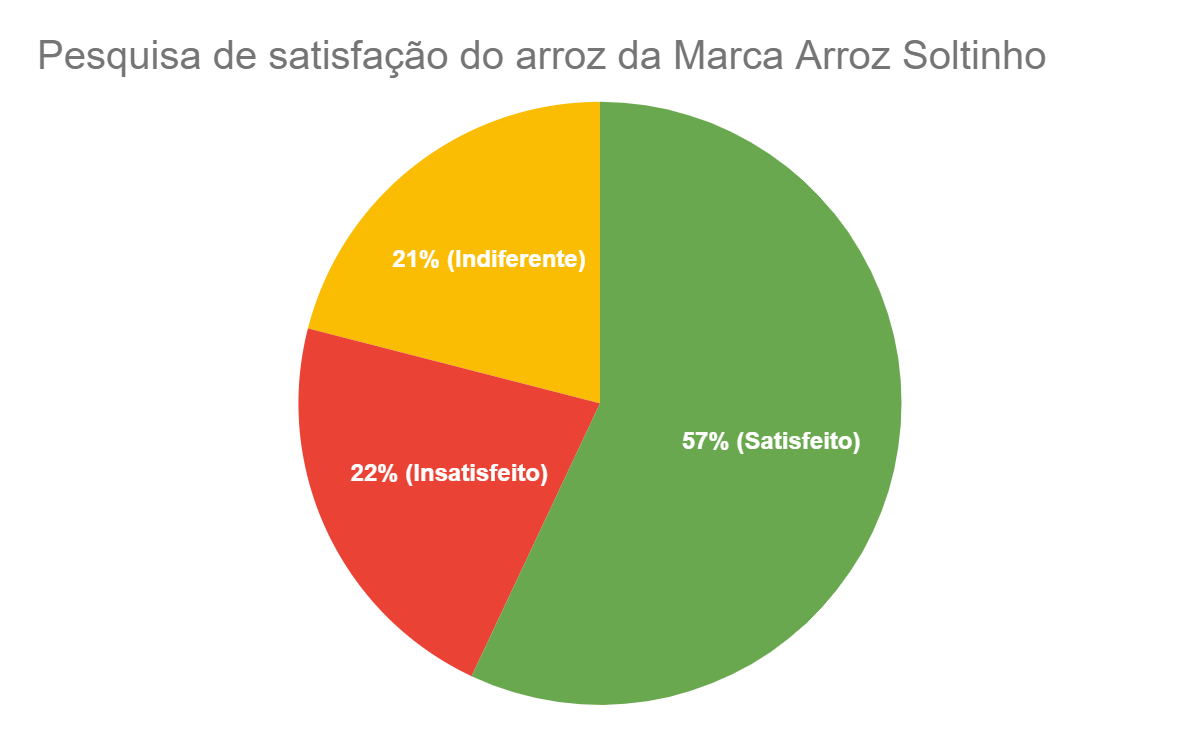
É evidente que a adequação de um gráfico de pizza depende muito do tipo de dado ou da maneira como desejamos comunicar nossas análises ao público. Assim, apresentaremos três pontos de atenção para evitar que nossa visualização de dados cause confusão na interpretação pelo público-alvo.

### Nunca utilize gráficos 3D e explosão de fatias

Quanto mais **simples** e **direta** for a nossa visualização, menor será o esforço cognitivo exigido de nosso público para compreender os dados apresentados.

Os gráficos de pizza já possuem certa complexidade ao relacionar partes com o todo por meio de áreas de setores circulares. A inserção de elementos 3D adiciona uma camada de complexidade ao representar dados em volumes, que não são nada triviais para nossa análise. Além disso, a explosão de fatias desloca as fatias do centro, tornando ainda mais desafiadora a comparação entre elas, como ilustrado nos gráficos a seguir.

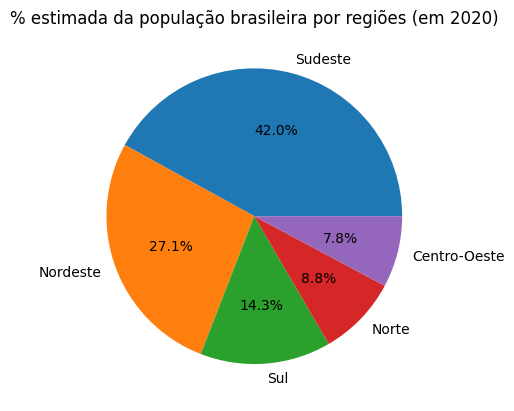


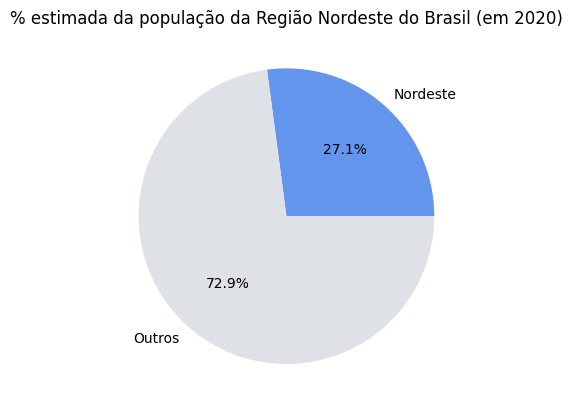


### Quanto menos fatias, melhor

Não há um número ideal de fatias para uma pizza. No entanto, para garantir a máxima clareza, é suficiente utilizar entre duas ou três fatias para comunicar certos pontos ao nosso público.

À medida que o número de fatias aumenta, a visualização dos dados se torna mais complexa, levantando questionamentos sobre a adequação do uso de gráficos de setores. Frequentemente, a resposta é negativa, exigindo que consideremos outras formas de visualização, como gráficos de barras, colunas, linhas ou outras representações. Os exemplos abaixo demonstram os mesmos dados dispostos de maneiras distintas.

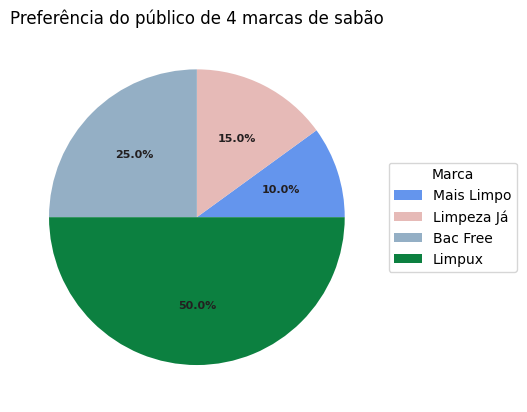


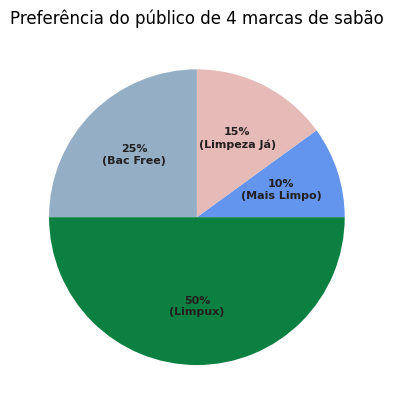


Fonte: IBGE. Diretoria de Pesquisas - DPE - Coordenação de População e Indicadores Sociais - COPIS (apenas os dados).

### Posicionamento dos rótulos

É ideal posicionar os rótulos próximos aos dados, evitando distrações que prejudiquem a leitura e interpretação dos dados. Um exemplo disso é o posicionamento da legenda, cuja distância pode afetar diretamente a interpretação da visualização. Quanto mais distante a legenda estiver do gráfico, mais difícil será a compreensão do visual, pois exigirá uma divisão da atenção entre a legenda e o gráfico, como mostrado nos exemplos abaixo:





Para concluir, os gráficos de setores tendem a ser mais compreensíveis quando utilizamos frações comuns, como um quarto e três quartos (25% vs. 75%) ou um terço e dois terços (33% vs 67%).

**Para saber mais: Biblioteca Seaborn**

Como estamos vendo ao longo deste curso, Seaborn é uma biblioteca de visualização de dados em Python que é baseada no Matplotlib. Ela fornece uma interface de alto nível para criar gráficos mais atrativos e com menor necessidade de ajustar os parâmetros de visualização. Seu principal objetivo é tornar a visualização de dados mais simples e esteticamente agradável, oferecendo uma ampla variedade de gráficos pré-construídos e personalizáveis.

Com apenas algumas linhas de código, é possível criar gráficos complexos, como histogramas e boxplots, como aprendemos ao longo deste curso, e também gráficos de dispersão, gráficos de linhas, mapas de calor e muito mais. Além disso, a biblioteca Seaborn integra-se perfeitamente com Pandas, outra biblioteca essencial em ciência de dados, facilitando a visualização de conjuntos de dados.

Vamos trazer um exemplo de código explorando algumas de suas possibilidades. Logo abaixo, utilizamos a base do TMDB, que exploramos durante a aula, e filtramos os filmes nas línguas francesa, espanhola e alemã com valores válidos de receita (*revenue*) e orçamento (*budget*). Em sequência, criamos 3 gráficos de dispersão com retas de regressão linear (lmplot()) para cada língua:

**import** pandas **as** pd

**import** seaborn **as** sns

# Importando a base de dados

tmdb = pd.read\_csv("https://raw.githubusercontent.com/alura-cursos/data-science-analise-exploratoria/main/Aula\_0/tmdb\_5000\_movies.csv")

# Filtrando os dados para as colunas desejadas

dados = tmdb[["title", "original\_language", "budget", "revenue"]]

linguas = ["fr", "es", "de"]

# Mantendo dados com valores de receita e orçamento válidos e nas línguas desejadas (fr, es, de)

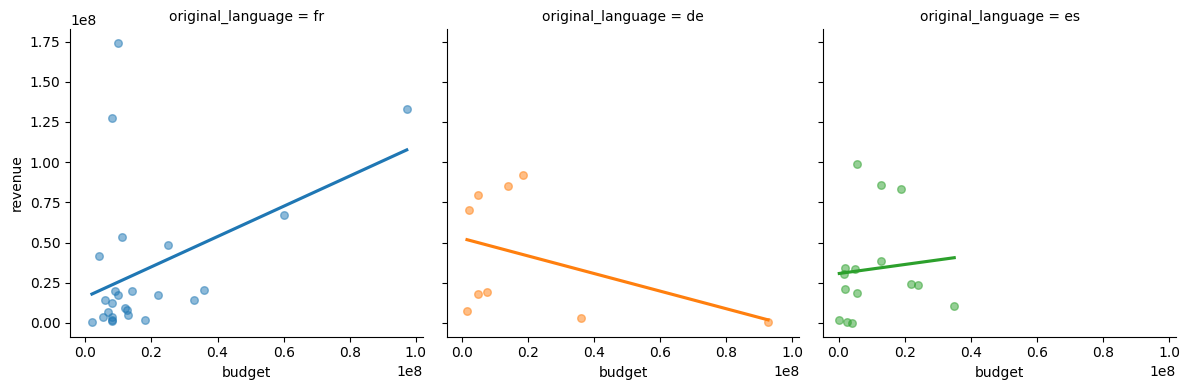
fr\_es\_de\_com\_revenue\_e\_budget = dados.query("revenue > 0 and budget > 0 and original\_language == @linguas")

# Construindo o gráfico de dispersão mais a reta de regressão linear para cada caso

sns.lmplot(data=fr\_es\_de\_com\_revenue\_e\_budget, x="budget", y="revenue", col="original\_language",

hue="original\_language", ci=**None**, height=4, scatter\_kws={"s": 30, "alpha": 0.5})

O gráfico gerado é o seguinte:



Podemos notar diferenças entre as produções em cada uma das línguas, mostrando as tendências da relação entre o orçamento utilizado e a receita gerada pelos filmes. Seria necessário um número maior de filmes para cada uma delas para gerarmos melhores interpretações com os dados.

Sinta-se livre para observar e testar mais exemplos disponíveis na [documentação do Seaborn](https://seaborn.pydata.org/examples/index.html), e não deixe de continuar os estudos dentro deste curso e nos próximos para aprender mais sobre o potencial da visualização de dados em nossas análises.

**Para saber mais: histograma e boxplot**

Falamos bastante sobre histograma e boxplot durante as nossas aulas e construímos estes visuais utilizando as bibliotecas do Python. Mas afinal o que são essas representações visuais?

### Histograma

Um histograma é uma representação visual que mostra a distribuição dos dados em um dado intervalo por meio de colunas. Os dados são divididos em intervalos, chamados de “bins”, que apresentam quantas observações estão presentes naquela faixa de dados. Ele é amplamente utilizado para visualizar a distribuição de frequências ou contagens em um conjunto de dados.

Este tipo de gráfico é eficaz para mostrar a distribuição de variáveis discretas ou categorias em que a altura de cada barra indica a contagem ou proporção de observações naquela categoria. Isso facilita a identificação de tendências, como a concentração de observações em determinados intervalos, e ajuda a compreender também a distribuição geral dos dados.

Como vimos na aula, é importante termos atenção no equilíbrio entre o número apropriado de “bins” e/ou a largura dos “bins” para representação dos dados, pois bins estreitos podem gerar ruídos por poucas observações em determinados intervalos e bins largos podem descaracterizar a curva e reduzir a sensibilidade na variação das faixas.

### Boxplot

O Boxplot, também conhecido como “diagrama de caixa”, é uma representação visual que apresenta a distribuição de um conjunto de dados por meio de uma série de estatísticas descritivas: quartis, mediana, mínimo e máximo.

Este tipo de gráfico auxilia na visualização da dispersão e centralidade dos dados, identificando assimetrias, outliers e variações nos dados.

Você pode aprender um pouco mais sobre estas visualizações de dados no curso [Estatística com Python: frequências e medidas](https://cursos.alura.com.br/course/estatistica-distribuicoes-e-medidas).

**Faça como eu fiz: refinando o gráfico das línguas**

Vamos prosseguir para a criação de um novo visual. Para isso, criaremos uma nova variável  total\_de\_outros\_filmes\_por\_lingua, que receberá uma  query() retornando todos os valores das categorias diferentes de  en:

total\_de\_outros\_filmes\_por\_lingua = tmdb.query("original\_language != 'en'")["original\_language"].value\_counts()

total\_de\_outros\_filmes\_por\_lingua.head()

Observamos os 5 valores mais frequentes logo acima. Agora, vamos construir um countplot() que receba a consulta de todos os filmes excetos em língua inglesa, passando a coluna original\_language para a contagem de ocorrências:

sns.countplot(data=tmdb.query("original\_language != 'en'"), x="original\_language")

Pronto! Temos a primeira versão do visual ainda com os dados desordenados, mas conseguimos notar todas as línguas possíveis, exceto inglês, em um gráfico de barras. Vamos ordenar os dados e adicionar cores?

Para isso, utilizaremos 2 parâmetros no countplot():

* order: como ordenar os dados. Aqui vamos ordenar de acordo com o índice em total\_de\_outros\_filmes\_por\_lingua;
* hue: determina qual coluna devemos utilizar para colorir o gráfico. Aqui, vamos passar o mesmo valor do eixo x original\_language.

sns.countplot(data=tmdb.query("original\_language != 'en'"),

order=total\_de\_outros\_filmes\_por\_lingua.index,

hue="original\_language",

x="original\_language")

Nada mal! Já conseguimos perceber a ordenação e notar diferentes cores nos gráficos, mas as cores não seguem um padrão que ajudariam a informar. Para isso, você pode acessar o [tutorial do Seaborn com a escolha da paleta de cores](https://seaborn.pydata.org/tutorial/color_palettes.html) e definir qual é o ideal para sua visualização.

Aqui utilizamos a “mako” para palette, que sai de um tom de azul mais escuro para um tom verde mais claro. Além disso, passamos para hue\_order, responsável por ordenar as cores, a mesma ordenação dos dados (order) e ajustamos o tamanho da figura para (16, 8) pelo figsize do plt.figure().

plt.figure(figsize=(16, 8))

sns.countplot(data=tmdb.query("original\_language != 'en'"),

order=total\_de\_outros\_filmes\_por\_lingua.index,

palette="mako",

hue="original\_language",

hue\_order=total\_de\_outros\_filmes\_por\_lingua.index,

x="original\_language")

Legal, né? E agora, como conseguimos alterar esse visual para ao invés de representar a frequência de ocorrência dos idiomas nos filmes nós contássemos a distribuição percentual dos idiomas exceto inglês? Coloque também um título no visual.