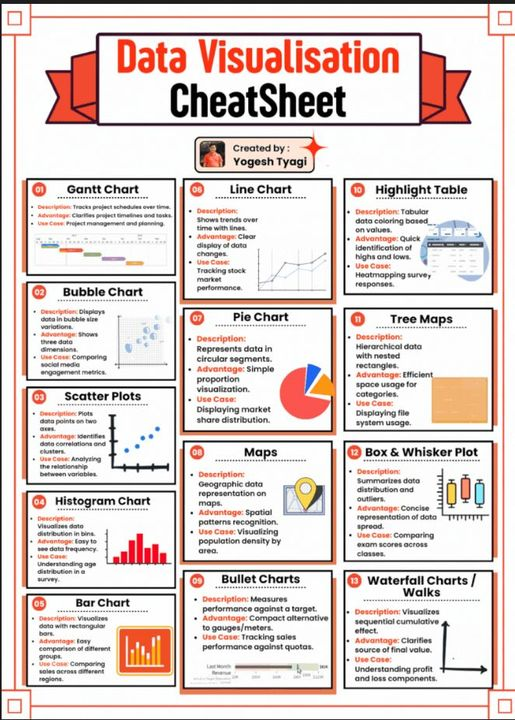
**FAP UFRN Softex 2024 – Prof. José Alfredo Costa**

**Visualização de dados (simples) em Python**

**Descrições dos Gráficos de Visualização de Dados**

A imagem abaixo apresenta um guia rápido ("cheat sheet") sobre diversos tipos de gráficos usados em visualização de dados, juntamente com suas descrições e casos de uso. A seguir, farei a tradução e expansão dessas descrições, incluindo exemplos em Python usando bibliotecas como Matplotlib e Seaborn.



Disp. em <https://x.com/tut_ml/status/1820662040173691260/photo/1>

**1. Gantt Chart (Gráfico de Gantt)**

**Descrição**: Acompanha os marcos do projeto ao longo do tempo.

**Vantagens**: Mostra a duração das tarefas do projeto e suas dependências.

**Uso**: Planejamento e gestão de projetos.

**Expansão**:

* Visualiza a programação e o progresso de tarefas em um projeto.
* Permite identificar dependências e possíveis atrasos.
* Útil para comunicação com a equipe e stakeholders.

**Exemplo em Python**:

import matplotlib.pyplot as plt

import pandas as pd

import plotly.express as px

# Dados de exemplo

df = pd.DataFrame({

'Tarefa': ['Iniciar projeto', 'Requisitos', 'Desenvolvimento', 'Testes', 'Entrega'],

'Início': ['2024-01-01', '2024-01-10', '2024-02-01', '2024-03-01', '2024-04-01'],

'Fim': ['2024-01-09', '2024-01-31', '2024-02-28', '2024-03-31', '2024-04-10']

})

# Conversão para formato datetime

df['Início'] = pd.to\_datetime(df['Início'])

df['Fim'] = pd.to\_datetime(df['Fim'])

# Criação do gráfico de Gantt

fig = px.timeline(df, x\_start="Início", x\_end="Fim", y="Tarefa", title='Gráfico de Gantt')

fig.update\_yaxes(categoryorder='total ascending')

fig.show()

**2. Bubble Chart (Gráfico de Bolhas)**

**Descrição**: Exibe três dimensões de dados em um gráfico de dispersão.

**Vantagens**: Visualiza a correlação entre variáveis.

**Uso**: Comparação de métricas de engajamento.

**Expansão**:

* A terceira dimensão é representada pelo tamanho das bolhas.
* Útil para comparar conjuntos de dados onde um fator adicional (tamanho) é relevante.
* Pode mostrar clusters e outliers de forma intuitiva.

**Exemplo em Python**:

import matplotlib.pyplot as plt

# Dados de exemplo

x = [5, 10, 15, 20, 25]

y = [10, 20, 25, 40, 50]

sizes = [50, 200, 300, 400, 500]

plt.scatter(x, y, s=sizes, alpha=0.5)

plt.xlabel('Variável X')

plt.ylabel('Variável Y')

plt.title('Gráfico de Bolhas')

plt.show()

**3. Scatter Plots (Gráficos de Dispersão)**

**Descrição**: Plota pontos de dados em dois eixos.

**Vantagens**: Identifica a relação entre duas variáveis.

**Uso**: Análise de correlação entre variáveis.

**Expansão**:

* Útil para detectar padrões, tendências e possíveis relações.
* Permite a visualização de outliers e agrupamentos.
* Frequente em análises estatísticas e modelagem preditiva.

**Exemplo em Python**:

import seaborn as sns

import matplotlib.pyplot as plt

# Dados de exemplo

tips = sns.load\_dataset('tips')

sns.scatterplot(data=tips, x='total\_bill', y='tip', hue='day', size='size')

plt.title('Gráfico de Dispersão')

plt.show()

**4. Histogram Chart (Gráfico de Histograma)**

**Descrição**: Exibe a distribuição dos dados.

**Vantagens**: Mostra a frequência de valores diferentes.

**Uso**: Análise de distribuição de dados.

**Expansão**:

* Útil para entender a distribuição e variabilidade dos dados.
* Ajuda a identificar padrões, como normalidade ou assimetria.
* Pode ser usado para análises exploratórias e estatísticas.

**Exemplo em Python**:

import matplotlib.pyplot as plt

import seaborn as sns

# Dados de exemplo

data = sns.load\_dataset('iris')

sns.histplot(data['sepal\_length'], kde=True)

plt.title('Histograma de Comprimento das Sépalas')

plt.xlabel('Comprimento da Sépala')

plt.ylabel('Frequência')

plt.show()

**5. Bar Chart (Gráfico de Barras)**

**Descrição**: Exibe dados categóricos com barras retangulares.

**Vantagens**: Comparação fácil entre categorias.

**Uso**: Comparação de vendas, receita, etc.

**Expansão**:

* Utilizado para comparar quantidades em diferentes categorias.
* As barras podem ser verticais ou horizontais.
* Fácil de entender e amplamente utilizado em apresentações e relatórios.

**Exemplo em Python**:

import matplotlib.pyplot as plt

# Dados de exemplo

categories = ['A', 'B', 'C', 'D']

values = [4, 7, 1, 8]

plt.bar(categories, values)

plt.xlabel('Categoria')

plt.ylabel('Valor')

plt.title('Gráfico de Barras')

plt.show()

**6. Line Chart (Gráfico de Linhas)**

**Descrição**: Mostra tendências ao longo do tempo.

**Vantagens**: Clareza na identificação de mudanças.

**Uso**: Acompanhamento de desempenho de ações, métricas de desempenho.

**Expansão**:

* Útil para visualizar tendências e padrões em séries temporais.
* Pode exibir múltiplas linhas para comparação entre diferentes séries de dados.
* Comum em finanças, economia e monitoramento de desempenho.

**Exemplo em Python**:

import matplotlib.pyplot as plt

# Dados de exemplo

x = [1, 2, 3, 4, 5]

y = [2, 3, 5, 7, 11]

plt.plot(x, y, marker='o')

plt.xlabel('Tempo')

plt.ylabel('Valor')

plt.title('Gráfico de Linhas')

plt.show()

**7. Pie Chart (Gráfico de Pizza)**

**Descrição**: Representa dados em segmentos circulares.

**Vantagens**: Simples visualização de proporções.

**Uso**: Exibição da participação de mercado, distribuição de recursos.

**Expansão**:

* Cada segmento representa uma proporção do total.
* Útil para mostrar a composição de um todo.
* Deve ser usado com moderação para evitar confusão em gráficos com muitos segmentos.

**Exemplo em Python**:

import matplotlib.pyplot as plt

# Dados de exemplo

labels = ['A', 'B', 'C', 'D']

sizes = [15, 30, 45, 10]

plt.pie(sizes, labels=labels, autopct='%1.1f%%')

plt.title('Gráfico de Pizza')

plt.show()

**8. Maps (Mapas)**

**Descrição**: Representa dados geográficos em mapas.

**Vantagens**: Visualização espacial.

**Uso**: População, densidade, etc.

**Expansão**:

* Útil para análise de dados geoespaciais.
* Pode mostrar a distribuição geográfica de variáveis.
* Comum em estudos de demografia, epidemiologia e logística.

**Exemplo em Python**:

import geopandas as gpd

import matplotlib.pyplot as plt

# Dados de exemplo

world = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))

# Criação do mapa

world.plot()

plt.title('Mapa Mundial')

plt.show()

**9. Bullet Charts (Gráficos de Bala)**

**Descrição**: Mede o desempenho em comparação a uma meta.

**Vantagens**: Compacto e claro.

**Uso**: Acompanhamento de desempenho em relação a metas.

**Expansão**:

* Útil para visualização de métricas de desempenho.
* Pode mostrar o desempenho atual, a meta e faixas de desempenho.
* Comum em dashboards e relatórios de KPI.

**Exemplo em Python**:

import matplotlib.pyplot as plt

# Dados de exemplo

performance = [75]

target = 100

threshold = [50, 75, 100]

fig, ax = plt.subplots()

ax.barh(['Desempenho'], performance, color='blue')

ax.axvline(target, color='red', linestyle='--')

for t in threshold:

ax.axvline(t, color='gray', linestyle='-')

plt.xlabel('Valor')

plt.title('Gráfico de Bala')

plt.show()

**10. Highlight Table (Tabela Destacada)**

**Descrição**: Tabela de dados com destaque em coloração.

**Vantagens**: Identificação rápida de valores.

**Uso**: Respostas de pesquisa, desempenho.

**Expansão**:

* Utilizada para destacar valores importantes em uma tabela.
* Facilita a comparação e a identificação de extremos.
* Comum em relatórios de desempenho e análise de resultados.

**Exemplo em Python**:

import seaborn as sns

import pandas as pd

# Dados de exemplo

data = pd.DataFrame({

'Categoria': ['A', 'B', 'C', 'D'],

'Valor1': [10, 20, 30, 40],

'Valor2': [20, 30, 40, 50]

})

# Criação da tabela destacada

sns.heatmap(data.set\_index('Categoria'), annot=True, cmap='YlGnBu')

plt.title('Tabela Destacada')

plt.show()

**11. Tree Maps (Mapas de Árvore)**

**Descrição**: Representa dados hierárquicos.

**Vantagens**: Uso eficiente de espaço.

**Uso**: Visualização de estrutura de hierarquias, como diretórios de arquivos ou participação de mercado.

**Expansão**:

* Útil para mostrar a composição e a estrutura de dados hierárquicos.
* Pode representar proporcionalmente grandes conjuntos de dados.
* Comum em análises financeiras e de sistemas de arquivos.

**Exemplo em Python**:

import plotly.express as px

# Dados de exemplo

data = {

'labels': ['A', 'B', 'C', 'D', 'E'],

'parents': ['', '', '', 'A', 'A'],

'values': [10, 20, 30, 15, 25]

}

fig = px.treemap(data, path=['labels', 'parents'], values='values')

fig.update\_traces(root\_color="lightgrey")

fig.update\_layout(margin = dict(t=50, l=25, r=25, b=25))

fig.show()

**12. Box & Whisker Plot (Gráfico de Caixa e Bigodes)**

**Descrição**: Exibe a distribuição dos dados através de quartis.

**Vantagens**: Representação concisa de distribuição.

**Uso**: Comparação de conjuntos de dados.

**Expansão**:

* Útil para identificar outliers e a variabilidade dos dados.
* Mostra a mediana, quartis e possíveis valores extremos.
* Comum em estatísticas descritivas e análises comparativas.

**Exemplo em Python**:

import seaborn as sns

import matplotlib.pyplot as plt

# Dados de exemplo

tips = sns.load\_dataset('tips')

sns.boxplot(x='day', y='total\_bill', data=tips)

plt.title('Gráfico de Caixa e Bigodes')

plt.xlabel('Dia da Semana')

plt.ylabel('Total da Conta')

plt.show()

**13. Waterfall Charts / Walks (Gráficos de Cascata)**

**Descrição**: Visualiza mudanças sequenciais.

**Vantagens**: Clareza nas mudanças de valor.

**Uso**: Compreensão de lucros e perdas, componentes de variação.

**Expansão**:

* Mostra como uma série de aumentos e diminuições levam a um valor final.
* Útil para análise de finanças e contabilidade.
* Ajuda a decompor a variação total em contribuições individuais.

**Exemplo em Python**:

import matplotlib.pyplot as plt

import waterfall\_chart

# Dados de exemplo

labels = ['Início', 'Aumento 1', 'Diminuição 1', 'Aumento 2', 'Diminuição 2', 'Fim']

values = [500, 300, -200, 400, -100, 900]

fig, ax = plt.subplots()

waterfall\_chart.plot(labels, values, ax=ax)

plt.title('Gráfico de Cascata')

plt.show()

Com esses exemplos e explicações, temos uma visão abrangente de diversos tipos de gráficos e como implementá-los em Python.

Cada gráfico tem suas próprias vantagens e aplicações específicas, ajudando a comunicar diferentes aspectos dos dados de forma eficaz.