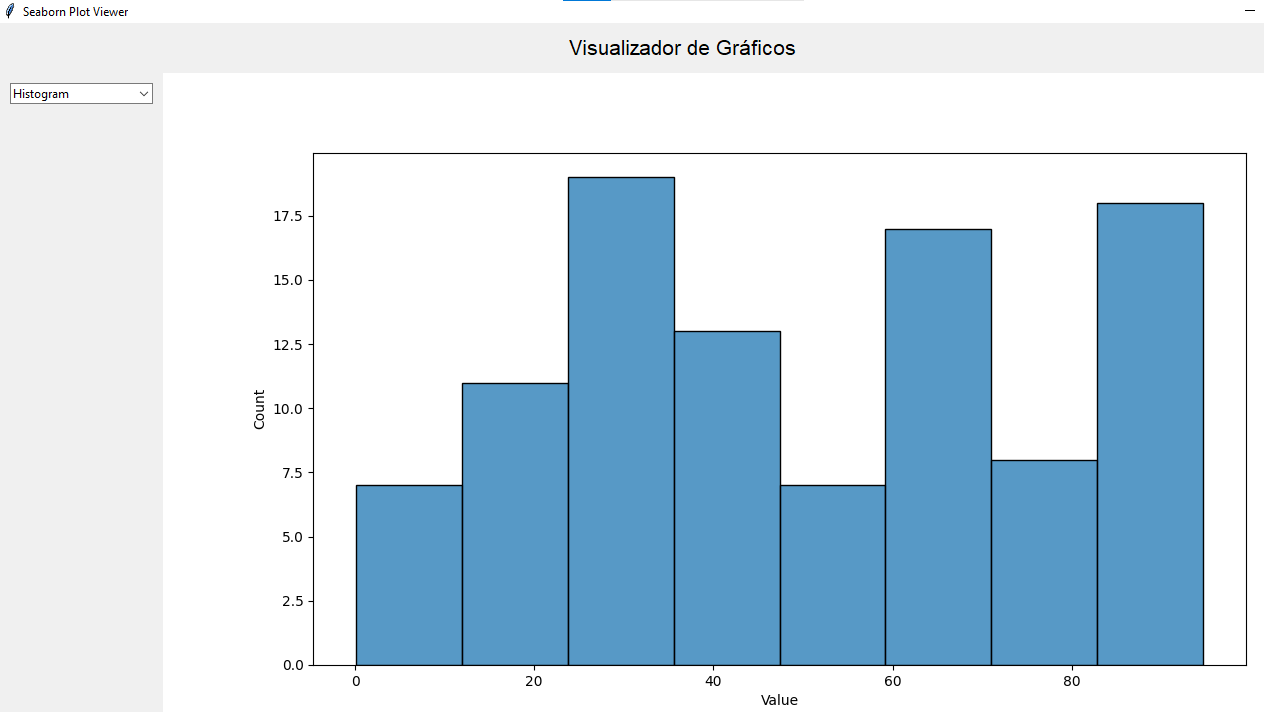
FAP 2024 – UFRN – Softex – Prof José Alfredo Costa

**Interfaces Gráficas em Python com Tkinter – Usando seaborn e plotly**

Outro exemplo de interface com tkinter Python, com um menu ao lado esquerdo com tipos de gráficos, comuns em seaborn. Temos um título central na parte de cima, uma caixa de seleção (menu) na parte superior esquerda, e abaixo do título o container com o gráfico a ser mostrado.



Interface gráfica usando o tkinter em Python para permitir a visualização de gráficos com a biblioteca seaborn.

Código inicial:

# interface

import tkinter as tk

from tkinter import ttk

import seaborn as sns

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import pandas as pd

import numpy as np

# Gerar um dataframe de exemplo

def generate\_dataframe():

    data = {

        'Category': np.random.choice(['A', 'B', 'C'], size=100),

        'Value': np.random.rand(100) \* 100

    }

    return pd.DataFrame(data)

df = generate\_dataframe()

# Função para atualizar o gráfico

def update\_plot(plot\_type):

    plt.clf()

    if plot\_type == 'Histogram':

        sns.histplot(df['Value'])

    elif plot\_type == 'Boxplot':

        sns.boxplot(x='Category', y='Value', data=df)

    elif plot\_type == 'Barplot':

        sns.barplot(x='Category', y='Value', data=df)

    elif plot\_type == 'Scatterplot':

        sns.scatterplot(x='Category', y='Value', data=df)

    canvas.draw()

# Criar a janela principal

root = tk.Tk()

root.title("Seaborn Plot Viewer")

# Título

title = tk.Label(root, text="Visualizador de Gráficos", font=("Helvetica", 16))

title.pack(pady=10)

# Criar o menu lateral

menu\_frame = tk.Frame(root)

menu\_frame.pack(side=tk.LEFT, fill=tk.Y)

# Opções de gráficos

plot\_types = ['Histogram', 'Boxplot', 'Barplot', 'Scatterplot']

plot\_menu = tk.StringVar(value=plot\_types[0])

plot\_dropdown = ttk.Combobox(menu\_frame, textvariable=plot\_menu, values=plot\_types)

plot\_dropdown.pack(padx=10, pady=10)

# Configurar o gráfico

fig, ax = plt.subplots(figsize=(5, 4))

canvas = FigureCanvasTkAgg(fig, master=root)

canvas.get\_tk\_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

# Atualizar gráfico quando a seleção mudar

def on\_selection(event):

    update\_plot(plot\_dropdown.get())

plot\_dropdown.bind('<<ComboboxSelected>>', on\_selection)

# Inicializar com o gráfico padrão

update\_plot(plot\_dropdown.get())

root.mainloop()

**Comentários:**

1. **Bibliotecas**: Utilizamos tkinter para a interface gráfica, seaborn para os gráficos e matplotlib para integrar o gráfico com o tkinter.
2. **Dataframe**: O generate\_dataframe cria um dataframe simples de exemplo. Você pode substituir isso pelos seus dados reais.
3. **Função update\_plot**: Atualiza o gráfico com base no tipo selecionado no menu. Limpa o gráfico atual (plt.clf()) e plota o novo gráfico.
4. **Interface tkinter**:
   * O título é exibido no topo da janela.
   * O menu com as opções de gráficos é exibido na lateral esquerda.
   * O gráfico é exibido na área à direita da janela.
5. **Atualização do gráfico**: O gráfico é atualizado sempre que uma nova opção é selecionada no menu suspenso.

Obs: Esse código é apenas um ponto de partida e pode ser expandido conforme necessário para atender a requisitos mais específicos ou para incluir mais tipos de gráficos.

Expansões para interfaces melhores - gráficos usando Python - podem incluir diversos aspectos e técnicas para enriquecer a experiência de aprendizado e a compreensão dos alunos. Aqui estão algumas sugestões:

**1. Exploração de Dados e Visualização**

* **Exploração Interativa**: Utilizar bibliotecas como Plotly ou Bokeh para criar gráficos interativos que permitem zoom, hover e outras interações.
* **Dashboard**: Criar dashboards com Dash ou Streamlit para integrar múltiplos gráficos e análises em uma interface interativa.

**2. Gráficos Avançados**

* **Gráficos de Séries Temporais**: Usar seaborn e matplotlib para visualizar dados de séries temporais com gráficos como linhas e áreas.
* **Heatmaps**: Utilizar seaborn para criar heatmaps para visualizar matrizes de dados e correlações.
* **Pairplots**: Criar pairplots para visualizar relações entre múltiplas variáveis de forma simultânea.

**3. Análise Estatística e Gráficos**

* **Distribuições**: Visualizar distribuições de dados com gráficos de densidade e histogramas.
* **Boxplots e Violin Plots**: Usar para identificar outliers e comparar distribuições entre diferentes categorias.

**4. Visualização de Dados Geoespaciais**

* **Mapas**: Utilizar bibliotecas como folium ou geopandas para criar mapas e visualizar dados geoespaciais.
* **Mapas de Calor**: Criar mapas de calor para visualizar a concentração de dados geoespaciais.

**5. Personalização e Estética**

* **Customização de Gráficos**: Ensinar como personalizar gráficos com cores, estilos de linha, anotações e etiquetas.
* **Tema e Design**: Explorar temas e estilos em seaborn e matplotlib para melhorar a estética dos gráficos.

**6. Integração com Outros Dados**

* **Leitura e Limpeza de Dados**: Mostrar como importar e limpar dados de diversas fontes (CSV, Excel, SQL).
* **Atualização Dinâmica**: Criar gráficos que atualizam automaticamente com base em novas entradas de dados ou alterações.

**7. Desafios e Projetos Práticos**

* **Desafios de Visualização**: Criar desafios para que os alunos construam gráficos que respondam a perguntas específicas sobre conjuntos de dados.
* **Projetos Finais**: Permitir que os alunos escolham um conjunto de dados e criem um projeto completo de visualização, incluindo análise e interpretação dos resultados.

**8. Integração com Machine Learning**

* **Visualização de Resultados**: Mostrar como visualizar os resultados de modelos de machine learning, como previsões e clusters.
* **Feature Importance**: Visualizar a importância das características com gráficos de barras ou árvores de decisão.

**9. Teoria de Gráficos**

* **Conceitos de Visualização**: Discutir a teoria por trás de diferentes tipos de gráficos e quando usá-los.
* **Análise Crítica**: Ensinar os alunos a criticar e interpretar gráficos com base em práticas recomendadas e erros comuns.

**Melhorias:**

Vamos adicionar algumas ideias – expansões – 1 e 2 - podendo abrir dados bem conhecidos como a íris data ou wine classification data (disponíveis na sckitlearn)?

Vamos expandir o código para incluir:

1. **Exploração Interativa com Plotly**.
2. **Gráficos Avançados com seaborn**, usando conjuntos de dados conhecidos como o iris e wine.

**Código Expandido:**

# Código Expandido: interface tkinter e seaborn

import tkinter as tk

from tkinter import ttk

import seaborn as sns

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import pandas as pd

from sklearn.datasets import load\_iris, load\_wine

import matplotlib

# Configurar backend do matplotlib para usar o TkAgg

matplotlib.use('TkAgg')

# Função para carregar dados

def load\_data(dataset):

    if dataset == 'Iris':

        iris = load\_iris()

        df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

        df['Species'] = iris.target

        df['Species'] = df['Species'].map({i: species for i, species in enumerate(iris.target\_names)})

    elif dataset == 'Wine':

        wine = load\_wine()

        df = pd.DataFrame(data=wine.data, columns=wine.feature\_names)

        df['Class'] = wine.target

        df['Class'] = df['Class'].map({i: cls for i, cls in enumerate(wine.target\_names)})

    return df

# Função para atualizar o gráfico

def update\_plot(plot\_type):

    plt.clf()

    if plot\_type == 'Histogram':

        if 'sepal length (cm)' in df.columns:

            sns.histplot(df['sepal length (cm)'])

        else:

            sns.histplot(df[df.columns[0]])  # Usar a primeira coluna se a coluna específica não existir

    elif plot\_type == 'Boxplot':

        if 'Class' in df.columns:

            # Agrupar variáveis numéricas

            num\_columns = df.select\_dtypes(include='number').columns

            if len(num\_columns) > 1:

                sns.boxplot(x='Class', y=num\_columns[0], data=df)

            else:

                sns.boxplot(x='Class', y=num\_columns[0], data=df)

    elif plot\_type == 'Barplot':

        if 'Class' in df.columns:

            # Agrupar variáveis numéricas

            num\_columns = df.select\_dtypes(include='number').columns

            if len(num\_columns) > 1:

                sns.barplot(x='Class', y=num\_columns[0], data=df)

            else:

                sns.barplot(x='Class', y=num\_columns[0], data=df)

    elif plot\_type == 'Scatterplot':

        if len(df.columns) >= 2:

            sns.scatterplot(x=df.columns[0], y=df.columns[1], hue=df.columns[2] if len(df.columns) > 2 else None, data=df)

    elif plot\_type == 'Heatmap':

        corr = df.select\_dtypes(include='number').corr()  # Calcular a correlação entre variáveis numéricas

        sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

    canvas.draw()

# Função para atualizar o conjunto de dados

def on\_data\_change(event):

    global df

    df = load\_data(data\_menu.get())

    # Atualizar o menu de gráficos para refletir que um novo conjunto de dados foi carregado

    plot\_menu.set(plot\_types[0])

    update\_plot(plot\_menu.get())

# Função para atualizar o gráfico ao selecionar um novo tipo de gráfico

def on\_selection(event):

    update\_plot(plot\_menu.get())

# Função para fechar a aplicação

def close\_app():

    root.destroy()

# Criar a janela principal

root = tk.Tk()

root.title("Seaborn Plot Viewer")

# Título

title = tk.Label(root, text="Visualizador de Gráficos", font=("Helvetica", 16))

title.pack(pady=10)

# Criar o menu lateral

menu\_frame = tk.Frame(root)

menu\_frame.pack(side=tk.LEFT, fill=tk.Y)

# Menu para escolher o conjunto de dados

data\_menu = tk.StringVar(value='Iris')

data\_dropdown = ttk.Combobox(menu\_frame, textvariable=data\_menu, values=['Iris', 'Wine'])

data\_dropdown.pack(padx=10, pady=10)

# Opções de gráficos

plot\_types = ['Histogram', 'Boxplot', 'Barplot', 'Scatterplot', 'Heatmap']

plot\_menu = tk.StringVar(value=plot\_types[0])

plot\_dropdown = ttk.Combobox(menu\_frame, textvariable=plot\_menu, values=plot\_types)

plot\_dropdown.pack(padx=10, pady=10)

# Botão Sair

exit\_button = tk.Button(menu\_frame, text="Sair", command=close\_app)

exit\_button.pack(padx=10, pady=10)

# Configurar o gráfico

fig, ax = plt.subplots(figsize=(6, 6))  # Ajuste o tamanho do gráfico para o heatmap

canvas = FigureCanvasTkAgg(fig, master=root)

canvas.get\_tk\_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

# Inicializar o dataframe com a base de dados padrão

df = load\_data(data\_menu.get())

update\_plot(plot\_menu.get())

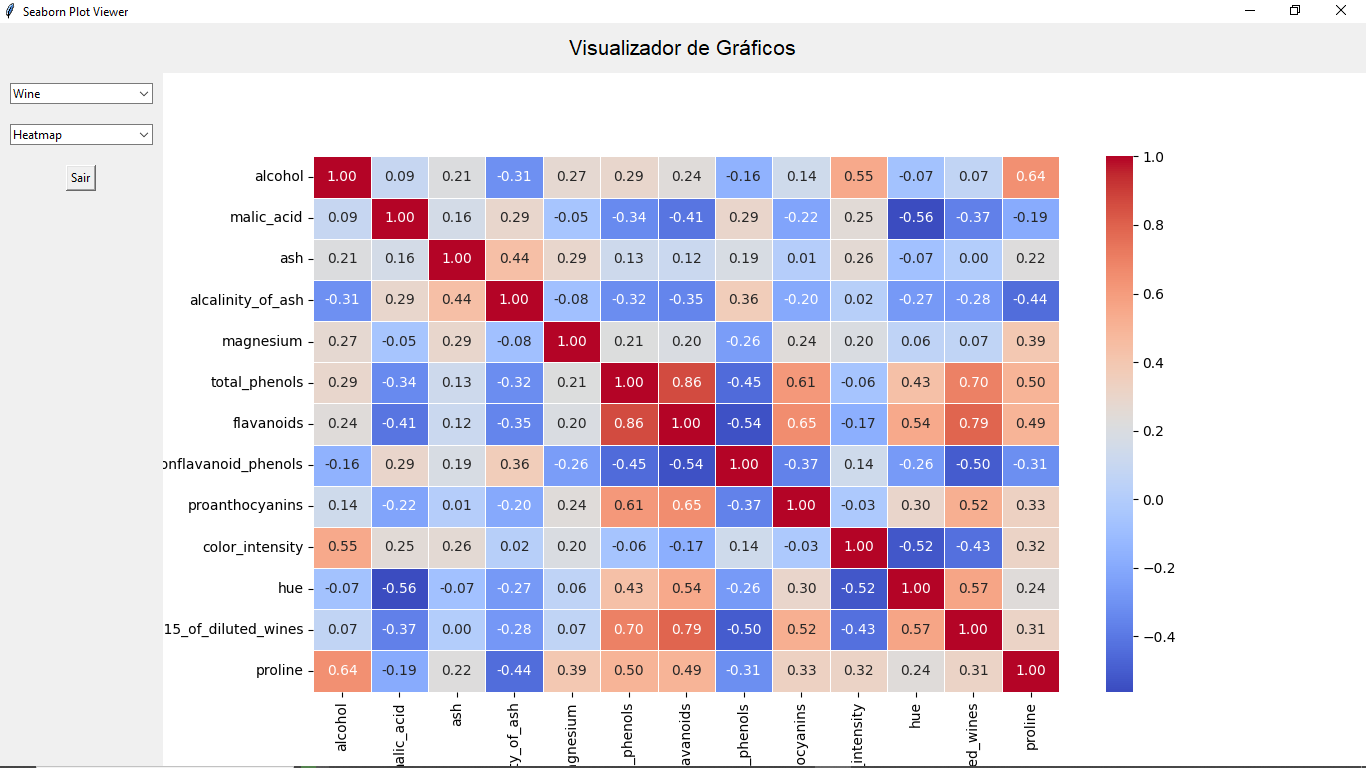
# Eventos

data\_dropdown.bind('<<ComboboxSelected>>', on\_data\_change)

plot\_dropdown.bind('<<ComboboxSelected>>', on\_selection)

root.mainloop()

**A interface ficará da forma:**



**Comentários sobre as Expansões:**

1. **Exploração Interativa com Plotly**:
   * Incluímos a biblioteca plotly para gerar gráficos interativos.
   * Converti os gráficos do Plotly em imagens que podem ser exibidas no Tkinter.
2. **Gráficos Avançados com seaborn**:
   * Adicionamos a opção de gráficos avançados como Pairplot, que é útil para visualizar múltiplas variáveis simultaneamente.
   * Utilizamos os conjuntos de dados Iris e Wine, que são bem conhecidos e úteis para demonstrações de visualização.

**Notas:**

* **Plotly**: A integração com Plotly é feita convertendo os gráficos para imagens e exibindo-as com matplotlib dentro do Tkinter. Essa abordagem permite visualização interativa no ambiente Tkinter.
* **Interatividade**: Atualize o gráfico com base na seleção do usuário para diferentes tipos de gráficos e conjuntos de dados.

Essa expansão proporciona uma visão mais rica e variada dos gráficos e melhora a interação com diferentes tipos de dados.