UNIVERSIDADE FEDERAL DA PARAÍBA - UFPB Centro de Ciências Exatas e da Natureza (CCEN) Departamento de Estatística

Trabalho Datasets Kaggle - COVID e Foodpanda

Disciplina: Introdução aos Softwares Estatísticos 2025.1 Professor: Pedro Rafael Marinho Aluna: Kaline de Almeida Felipe

João Pessoa - PB Outubro de 2025

Introdução

O presente trabalho foi desenvolvido como parte da terceira avaliação da disciplina **Introdução aos Softwares Estatísticos**.
O objetivo é demonstrar a utilização das bibliotecas **Matplotlib** e **Plotnine** para a criação de gráficos no Python, comentando as diferenças de sintaxe, estilo e visualização entre elas, assim como o uso da ferramenta **Numba**.

Essas bibliotecas são amplamente utilizadas por cientistas de dados e estatísticos para análise visual de informações.
O **Matplotlib** é a base de muitas outras ferramentas gráficas em Python, enquanto o **Plotnine** segue a filosofia do **ggplot2** do R, sendo mais declarativo e conciso.

Importação dos dados

Os conjuntos de dados utilizados neste trabalho foram obtidos do repositório **Kaggle**:

- Post-COVID Conditions Dataset
- Foodpanda Delivery Dataset

Importar bibliotecas

import pandas as pd import matplotlib.pyplot as plt from plotnine import ggplot, aes, geom_bar, theme_minimal, labs import random import time from numba import jit

Carregar os arquivos CSV

covid = pd.read_csv("Post-COVID_Conditions.csv") food = pd.read_csv("Food-panda Analysis Dataset.csv")

Mostrar as primeiras linhas de cada conjunto

print("POST-COVID CONDITIONS DATASET") display(covid.head())

Parte 1 – Plotagem em Python

Nesta seção são apresentados exemplos de gráficos desenvolvidos com as bibliotecas **Matplotlib** e **Plotnine**. Foram utilizados conjuntos de dados extraídos de repositórios conhecidos, como o Kaggle, de modo a demonstrar as diferenças de sintaxe, estilo e visualização.

1.1 Gráficos com Matplotlib

A biblioteca **Matplotlib**, desenvolvida por John D. Hunter, é uma das ferramentas mais conhecidas para visualização em Python. Ela permite a criação de gráficos personalizados e de alta qualidade. Considerada uma das bibliotecas mais populares, é atualmente mantida por uma comunidade ativa.

Dataset Post-COVID Conditions

Neste dataset, vamos analisar o número de casos registrados por país e, em seguida, comparar os sintomas mais relatados.

Filtrar e contar número de casos por país

casos_pais = covid[“Country”].value_counts().reset_index() casos_pais.columns = [“País”, “Total de Casos”]

Selecionar apenas os 10 países com mais registros

top_paises = casos_pais.head(10)

————- Gráfico Matplotlib ————-

plt.figure(figsize=(8,5)) plt.bar(top_paises[“País”], top_paises[“Total de Casos”], color=“skyblue”, edgecolor=“black”) plt.title(“Número de casos pós-COVID por país (Top 10)”) plt.xlabel(“País”) plt.ylabel(“Quantidade de casos”) plt.xticks(rotation=45) plt.grid(axis=‘y’, linestyle=‘–’, alpha=0.6) plt.tight_layout() plt.show()

Dessa forma, o **Matplotlib** usa uma sintaxe procedural, com funções diretas (plt.bar, plt.title, etc.). É mais flexível, porém o código tende a ser mais detalhado.

Comparando sintomas mais frequentes

Agora, analisaremos os sintomas mais relatados e sua frequência total, para comparação com o gráfico anterior.

Contar sintomas mais comuns

sintomas = covid[“Symptoms”].value_counts().reset_index() sintomas.columns = [“Sintoma”, “Frequência”] top_sintomas = sintomas.head(10)

import pandas as pd from plotnine import ggplot, aes, geom_bar, coord_flip, labs, theme_minimal

— Dados dos sintomas (exemplo) —

dados_sintomas = { “Sintoma”: [“Fadiga”, “Falta de ar”, “Dor de cabeça”, “Perda de olfato”, “Dificuldade de concentração”, “Tosse persistente”], “Percentual”: [65, 45, 30, 25, 20, 15] } df_sintomas = pd.DataFrame(dados_sintomas)

— Gráfico com Plotnine —

grafico_sintomas = (ggplot(df_sintomas, aes(x=“Sintoma”, y=“Percentual”, fill=“Sintoma”)) + geom_bar(stat=“identity”, show_legend=False) + coord_flip() + labs(title=“Sintomas mais frequentes relatados – Pós-COVID”, x=“Sintomas”, y=“Percentual (%)”) + theme_minimal())

— Salvar em PNG —

grafico_sintomas.save(“sintomas_pos_covid_plotnine.png”, dpi=300)

print(” Gráfico salvo como ‘sintomas_pos_covid_plotnine.png’ “)

———— Plotnine (ggplot) ————-

plot_sintomas = (ggplot(top_sintomas, aes(x=“Sintoma”, y=“Frequência”, fill=“Sintoma”)) + geom_bar(stat=“identity”) + theme_minimal() + labs(title=“Sintomas mais frequentes relatados - Pós-COVID”, x=“Sintoma”, y=“Frequência”)) print(plot_sintomas)

O **Plotnine** segue a filosofia do ggplot2 (em R): a construção é declarativa — você descreve o que quer (dados + estética + geometria) em camadas. O resultado é geralmente mais limpo e elegante, mas exige um pouco mais de conhecimento.

Foodpanda Dataset

Neste segundo dataset, o objetivo é visualizar o total de pedidos por cidade, comparando os estilos de codificação do **Matplotlib** e **Plotnine**.

Exemplo simples: total de pedidos por cidade

pedidos_cidade = food[“City”].value_counts().reset_index() pedidos_cidade.columns = [“Cidade”, “Total”]

Matplotlib – Total de pedidos por cidade

plt.figure(figsize=(8,4)) plt.bar(pedidos_cidade[“Cidade”], pedidos_cidade[“Total”], color=“lightcoral”, edgecolor=“black”) plt.title(“Total de pedidos por cidade - Foodpanda”) plt.xlabel(“Cidade”) plt.ylabel(“Quantidade de pedidos”) plt.xticks(rotation=45) plt.tight_layout() plt.show()

Plotnine – Total de pedidos por cidade

plot_food = (ggplot(pedidos_cidade, aes(x="Cidade", y="Total", fill="Cidade")) + geom_bar(stat="identity") + theme_minimal() + labs(title="Total de pedidos por cidade - Foodpanda", x="Cidade", y="Quantidade de pedidos")) print(plot_food)

No **Plotnine**, o estilo é mais automático — com cores e temas prontos — enquanto o Matplotlib oferece controle manual detalhado de cada elemento visual. O **Plotnine** utiliza o operador + para adicionar camadas de informações ao gráfico, tornando a sintaxe mais limpa e descritiva, ideal para análises exploratórias rápidas e relatórios automatizados.

Parte 2 – Aceleração de Código com Numba

A biblioteca **Numba** permite acelerar a execução de códigos Python numéricos, compilando funções para código de máquina (muito mais rápido). Usaremos o Método de Monte Carlo para calcular uma aproximação de (Pi), um problema que envolve muitas repetições.

2.1 Implementação em Python Puro

import random import time

def calcular_pi_puro(num_pontos): """Calcula pi usando Monte Carlo em Python 'normal'.""" pontos_dentro_circulo = 0 for _ in range(num_pontos): x = random.random() y = random.random() if x**2 + y**2 <= 1.0: pontos_dentro_circulo += 1 return 4 * (pontos_dentro_circulo / num_pontos)

Define 10 milhões de repetições

N = 10_000_000

Mede o tempo da versão pura

tempo_inicial_puro = time.time() pi_puro = calcular_pi_puro(N) tempo_final_puro = time.time()

tempo_puro = tempo_final_puro - tempo_inicial_puro

print(f"Versão Python Puro:") print(f"Valor aproximado de pi: {pi_puro:.6f}") print(f"Tempo de execução: {tempo_puro:.4f} segundos")

2.2 Aceleração usando o Numba

from numba import jit

@jit(nopython=True) def calcular_pi_numba(num_pontos): """Calcula pi com Numba (código acelerado)."""" pontos_dentro_circulo = 0 for _ in

range(num_pontos): x = random.random() y = random.random() if x**2 + y**2 <= 1.0: pontos_dentro_circulo += 1 return 4 * (pontos_dentro_circulo / num_pontos)

Compilação inicial

calcular_pi_numba(100)

Mede o tempo da versão acelerada

tempo_inicial_numba = time.time() pi_numba = calcular_pi_numba(N) tempo_final_numba = time.time()

tempo_numba = tempo_final_numba - tempo_inicial_numba

print(f"Versão Acelerada com Numba:“) print(f"Valor aproximado de pi: {pi_numba:.6f}“) print(f"Tempo de execução: {tempo_numba:.4f} segundos")

Assim, a função para calcular usa um loop simples com milhões de repetições. A versão com Numba executa em uma fração do tempo, demonstrando sua eficiência para cálculos intensivos.

Considerações Finais

Neste trabalho, foram apresentadas três ferramentas fundamentais para análise e comunicação de dados em Python:

Matplotlib – maior flexibilidade e controle detalhado dos gráficos;

Plotnine – sintaxe mais declarativa e elegante, inspirada no ggplot2;

Numba – aceleração significativa de códigos numéricos.

Com o uso dessas ferramentas, o Python se consolida como uma linguagem completa para análise estatística, comunicação de dados e computação de alta performance.

Referências

MCKINNEY, Wes. Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter. 3. ed. Sebastopol: O'Reilly Media, 2022.

MÜLLER, M. Numba Essentials. Birmingham: Packt Publishing, 2015. MCKINNEY, W. Python for Data Analysis. 3. ed. Sebastopol: O'Reilly Media, 2022.

YLMAZ, ASEL. Post-COVID Conditions Dataset. Kaggle, 2023. Disponível em: https://www.kaggle.com/datasets/ylmzasel/post-covid-conditions-dataset . Acesso em: 1 out. 2025.

IMRAN, Ayesha. Foodpanda Order and Delivery Trends Dataset. Kaggle, 2023. Disponível em: https://www.kaggle.com/datasets/ayeshaimran123/foodpanda-order-and-delivery-trends . Acesso em: 1 out. 2025.

PLOTNINE Documentation. Disponível em: https://plotnine.org . Acesso em: 1 out. 2025. MATPLOTLIB Documentation. Disponível em: https://matplotlib.org . Acesso em: 1 out. 2025.