Desafio_02

AUTHOR
Iuri Santos Oliveira

PUBLISHED
August 14, 2025

Versão do código em python

````
::: {.cell}

```{.r .cell-code}
# Configuração do ambiente Python integrado ao R
library(reticulate)
use_virtualenv("~/.virtualenvs/r-reticulate", required = T)
py_discover_config()

# Instalação dos pacotes Python necessários para análise de dados e visualização
reticulate::py_install("matplotlib")
reticulate::py_install("calmap")
reticulate::py_install("seaborn")
reticulate::py_install("numpy")
reticulate::py_install("pandas", envname = NULL, python = "C:/Program
Files/Python312/python.exe")
```
:::
````

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import calendar
from matplotlib.colors import LinearSegmentedColormap
```

```python
def getStats(chunk):
    """
    Função para calcular estatísticas suficientes de cada lote de dados
    """
    mask = (chunk["AIRLINE"].isin(["AA", "DL", "UA", "US"])) & (chunk["ARRIVAL_DELAY
    filtered = chunk.loc[mask].copy()

    # Calcular estatísticas por grupo
    grouped = filtered.groupby(["DAY", "MONTH", "AIRLINE"])
    result = pd.DataFrame({
        'n': grouped.size(),
        'atrasos': grouped['ARRIVAL_DELAY'].apply(lambda x: (x > 10).sum())
    }).reset_index()

    return result
```

```python
def computeStats(stats):
    """
```

```
    Função para combinar as estatísticas suficientes e calcular percentual de atrasc
    """
    result = stats.groupby(["DAY", "MONTH", "AIRLINE"]).agg({
        'n': 'sum',
        'atrasos': 'sum'
    }).reset_index()

    result['Perc'] = result['atrasos'] / result['n']
    result['Data'] = pd.to_datetime({
        'year': 2015,
        'month': result['MONTH'],
        'day': result['DAY']
    })

    return result[['AIRLINE', 'Data', 'Perc']].rename(columns={'AIRLINE': 'Cia'})
```

```
def baseCalendario(stats, cia):
    """
    Função para criar mapa de calor em formato de calendário
    """
    df = stats[stats['Cia'] == cia].copy()

    # Criar colormap personalizado
    colors = ['#4575b4', '#74add1', '#abd9e9', '#e0f3f8', '#ffffbf',
              '#fee090', '#fdae61', '#f46d43', '#d73027']
    cmap = LinearSegmentedColormap.from_list('custom', colors, N=100)

    # Criar figura com subplots para cada mês
    fig, axes = plt.subplots(3, 4, figsize=(16, 12))
    fig.suptitle(f'Percentual de Atrasos - {cia}', fontsize=16, fontweight='bold')

    for mes in range(1, 13):
        ax = axes[(mes-1)//4, (mes-1)%4]
        df_mes = df[df['Data'].dt.month == mes]

        if len(df_mes) > 0:
            # Criar matriz do calendário (6 semanas x 7 dias)
            matriz = np.full((6, 7), np.nan)

            for _, row in df_mes.iterrows():
                dia = int(row['Data'].day)

                # Descobrir que dia da semana é o dia 1 do mês
                primeiro_dia = pd.Timestamp(year=2015, month=mes, day=1)
                dia_semana_primeiro = (primeiro_dia.weekday() + 1) % 7  # Domingo =

                # Calcular posição do dia atual
                posicao_total = dia_semana_primeiro + dia - 1
                semana = posicao_total // 7
                dia_semana = posicao_total % 7

                if semana < 6:
```

```
                    matriz[semana, dia_semana] = row['Perc']

            # Plotar heatmap
            sns.heatmap(matriz,
                        cmap=cmap,
                        cbar=False,
                        vmin=0,
                        vmax=1,
                        square=True,
                        linewidths=0.5,
                        linecolor='white',
                        ax=ax)

        # Configurar labels
        ax.set_title(calendar.month_name[mes], fontsize=12, pad=10)
        ax.set_xticks(np.arange(7) + 0.5)
        ax.set_xticklabels(['D', 'S', 'T', 'Q', 'Q', 'S', 'S'])
        ax.set_yticks([])
        ax.tick_params(axis='x', length=0)

    plt.tight_layout()
    plt.show()
```

```
# Processamento principal dos dados
def processar_dados():
    """
    Função principal para processar os dados de voos
    """
    # Leitura por chunks
    chunks = pd.read_csv(
        "../dados/flights.csv.zip",
        usecols=["AIRLINE", "ARRIVAL_DELAY", "DAY", "MONTH"],
        chunksize=100000
    )

    # Processar cada chunk
    stats_list = []
    for chunk in chunks:
        stats_list.append(getStats(chunk))

    # Combinar todos os chunks
    voos = pd.concat(stats_list, ignore_index=True)

    # Calcular estatísticas finais
    voos_final = computeStats(voos)

    return voos_final
```

```
# Executar processamento
if __name__ == "__main__":
    # Processar dados
    voos_resultado = processar_dados()
```

```
baseCalendario(voos_resultado, "AA")
```