Incêndios Florestais no Brasil

CONSCIENTIZAÇÃO SOBRE OS IMPACTOS AMBIENTAIS DAS QUEIMADAS

Grupo:
Samuel
Victor César Sampaio Ferreira da Silva
Gilmar
Breno
Jonathan

Disciplina: Tópicos de big data em Python / Estácio
Professor: Davi barros

Introdução

- As **queimadas no Brasil** representam um dos maiores desafios ambientais do país, com consequências devastadoras para a biodiversidade, o clima e a saúde humana. Historicamente associadas à abertura de novas áreas para a agropecuária, ao manejo inadequado do solo e a fatores climáticos como secas prolongadas, as queimadas se intensificaram nos últimos anos, especialmente em biomas como a Amazônia, o Cerrado e o Pantanal.
- As florestas são pilares fundamentais para a manutenção do equilíbrio ambiental global. Elas atuam como verdadeiros "pulmões do mundo", absorvendo dióxido de carbono e liberando oxigênio, um processo vital para a regulação climática e a qualidade do ar que respiramos. Além disso, desempenham um papel crucial na regulação do ciclo hidrológico, influenciando regimes de chuva e prevenindo secas e inundações.

O fogo que destrói a floresta também ameaça nosso futuro.

Objetivo

- Nosso projeto busca **conscientizar** a população sobre os graves impactos ambientais das queimadas. Para isso, vamos além de apenas informar; queremos utilizar dados reais para demonstrar a verdadeira gravidade do problema, tornando os efeitos visíveis e inegáveis.
- Além disso, nosso objetivo é incentivar atitudes sustentáveis e apoiar políticas de preservação que promovam um futuro mais seguro e equilibrado para todos. Queremos transformar a preocupação em ação, mostrando que cada um pode fazer a diferença na proteção de nossas florestas.

Metodologia

- A coleta dos dados foram feitas no site do INPE (Instituto Nacional de Pesquisas Espaciais), ano 2021 a 2024.

- Usamos Python para limpar e tratar os dados usando a biblioteca **Pandas**
- Usamos análise estatísticas e identificações de padrões
- Utilizamos Matplotlib e Seaborn para geração de gráficos e visualizações dos dados
- Ferramentas:
  - Linguagem: Python
  - Bibliotecas: Pandas, Matplotlib, Seaborn
  - Fonte de dados: https://terrabrasilis.dpi.inpe.br/queimadas/portal/dados-abertos/#da-rf
  - https://dataserver-coids.inpe.br/queimadas/queimadas/focos/csv/anual/Brasil_todos_sats/
  - https://terrabrasilis.dpi.inpe.br/app/dashboard/fires/biomes/aggregated/
  - https://www.gov.br/cemaden/pt-br/assuntos/monitoramento/monitoramento-de-seca-para-o-brasil
  - https://www.kaggle.com/datasets/gnomows/dados-metereologicos-2018-2024-inmet/data
  - https://bdmep.inmet.gov.br/

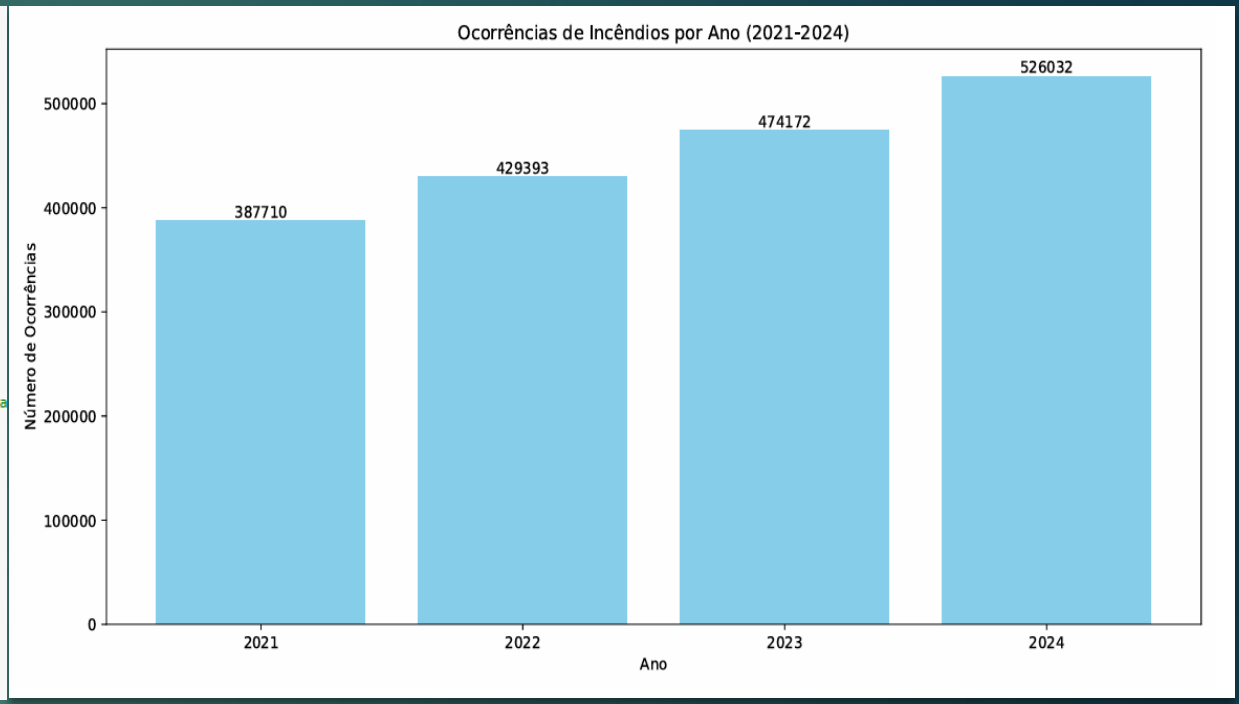
- Nosso foco foi identificar as regiões e períodos mais críticos, buscando relacionar dados com os efeitos ambientais das queimadas

Dados utilizados no projeto

- **Período Analisado:** de 2021 a 2024
- Fonte de dados: Banco de dados do INPE (Instituto Nacional de Pesquisas Espaciais).
- Informações coletadas:
  - Quantidade de focos de queimadas por estado e por mês
  - Distribuição ao longo dos anos
  - Regiões mais afetadas
  - Picos de ocorrência por período

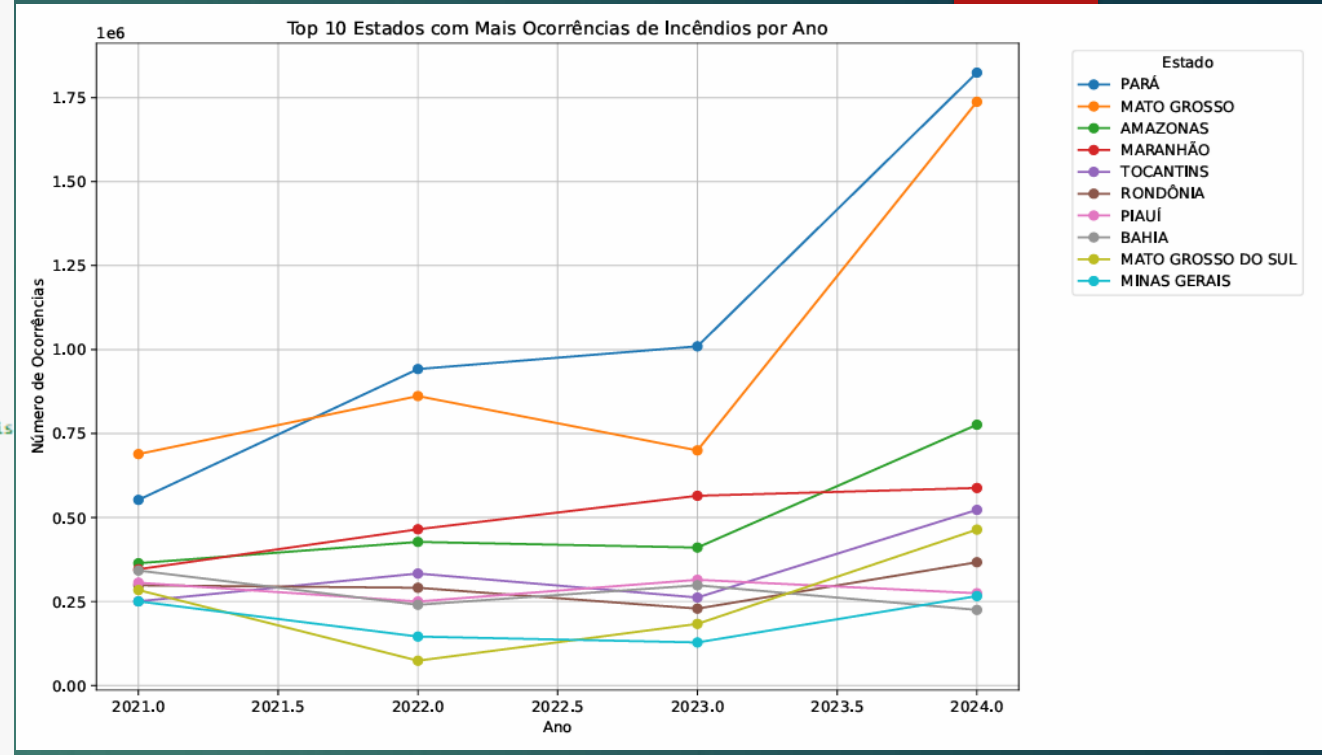
- Os dados são públicos e atualizados constantemente
- A base de dados nos permitiu identificar padrões e tendências importantes para nossa análise.

```python
anos = [2021, 2022, 2023, 2024]    # Lista de anos a serem analisados

def contar_ocorrencias_por_ano(anos):
    ocorrencias_por_ano = {}          # Inicializa um dicionário vazio para armazenar resultados

    for ano in anos:                  # Itera sobre cada ano da lista
        try:
            # Carrega apenas a coluna 'data_pas' do CSV específico do ano (torna a leitura mais leve)
            df = pd.read_csv(f'data/focos_br_todos-sats_{ano}.csv', usecols=['data_pas'])  # Lê somente a coluna de datas do arquivo CSV daquele ano
            df['data_pas'] = pd.to_datetime(df['data_pas'], errors='coerce') # Converte a coluna de datas para o tipo datetime, tratando erros como NaT
            df = df.drop_duplicates(subset='data_pas')                       # Remove datas duplicadas, mantendo apenas as únicas
            ocorrencias_por_ano[ano] = df.shape[0]                           # Conta o número de linhas (datas únicas) e armazena no dicionário
        except Exception as e:                                               # Caso ocorra algum erro ao processar o arquivo
            print(f"Erro ao processar {ano}: {e}")                           # Exibe mensagem de erro para o ano correspondente
            ocorrencias_por_ano[ano] = 0                                     # Define zero ocorrências para o ano com erro

    return ocorrencias_por_ano                                               # Retorna o dicionário com as contagens por ano

def gerar_grafico_barras(ocorrencias_dict):
    plt.figure(figsize=(12, 6))                                              # Cria uma nova figura para o gráfico, tamanho 12x6 polegadas
    plt.bar(ocorrencias_dict.keys(), ocorrencias_dict.values(), color='skyblue')  # Gera gráfico de barras com os anos no eixo x e ocorrências no eixo y, cor azul clara
    plt.title('Ocorrências de Incêndios por Ano (2021-2024)')                # Define o título do gráfico
    plt.xlabel('Ano')                                                        # Define o rótulo do eixo x
    plt.ylabel('Número de Ocorrências')                                      # Define o rótulo do eixo y
    plt.xticks(list(ocorrencias_dict.keys()))                                # Garante que todos os anos apareçam no eixo x

    for ano, qtd in ocorrencias_dict.items():                                # Para cada barra (ano e quantidade)
        plt.text(ano, qtd + 100, str(qtd), ha='center', va='bottom')         # Adiciona o valor numérico acima da barra

    plt.tight_layout()                                                       # Ajusta o layout para evitar sobreposição
    plt.savefig('graficos/Variacao_anual.png')                               # Salva o gráfico como imagem PNG na pasta 'graficos'
    plt.close()                                                              # Fecha a figura para liberar memória

# Executa todo o fluxo:
ocorrencias = contar_ocorrencias_por_ano(anos)       # Chama a função para contar ocorrências por ano
gerar_grafico_barras(ocorrencias)                    # Gera e salva o gráfico de barras com os resultados
```



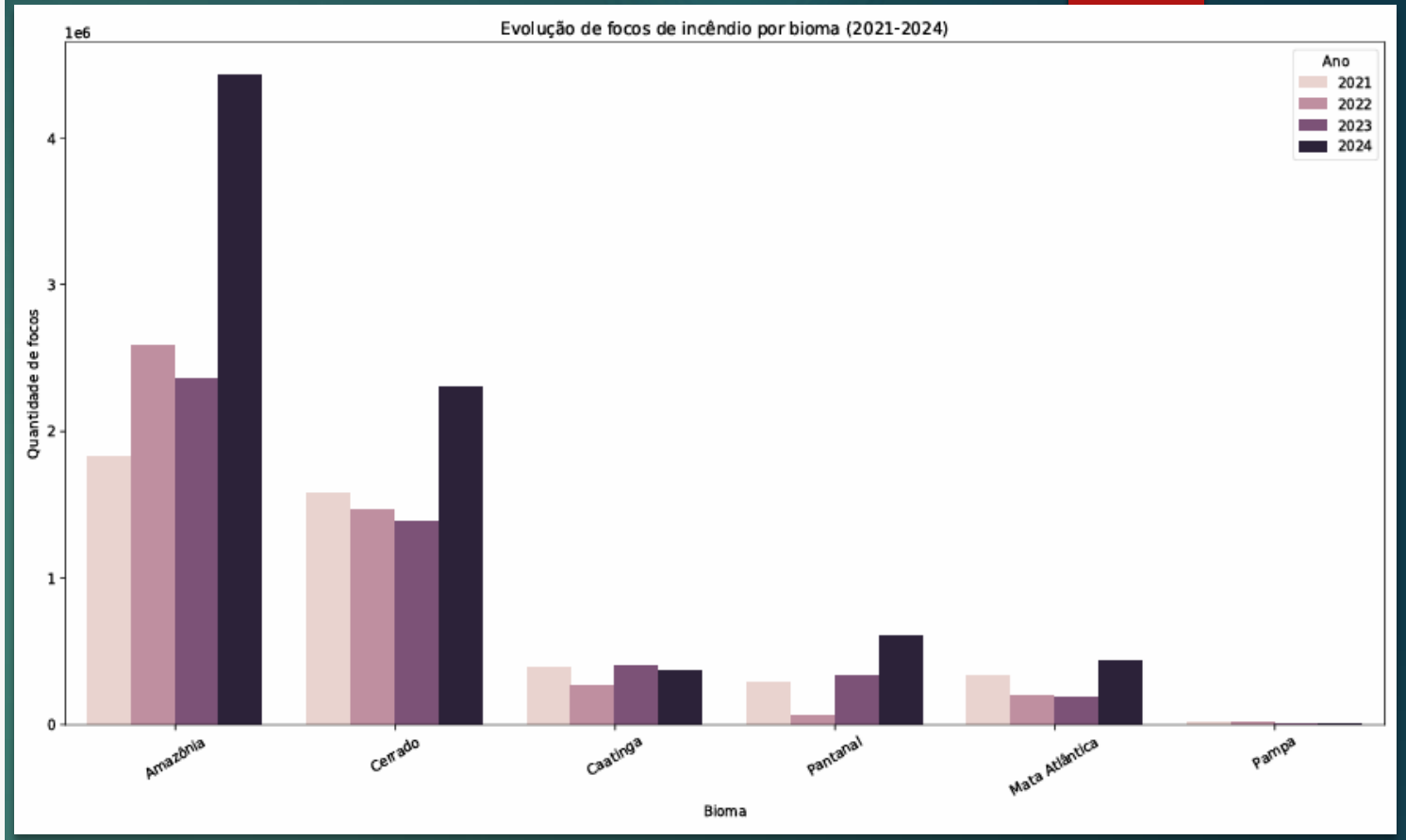
Evolução das queimadas no Brasil

```python
# 2. Processamento de arquivos por ano
for ano in anos:                        # Itera sobre cada ano na lista
    try:
        # Carrega apenas a coluna 'estado' do arquivo CSV do ano atual
        df = pd.read_csv(f'data/focos_br_todos-sats_{ano}.csv', usecols=['estado'])  # Lê coluna específica do CSV

        # Conta ocorrências por estado usando value_counts()
        contagem = df['estado'].value_counts()  # Retorna série com contagem por estado

        # Preenche o dicionário de dados
        for estado, qtd in contagem.items():    # Itera sobre cada par estado-contagem
            if estado not in dados:             # Cria entrada no dicionário se o estado não existir
                dados[estado] = {}
            dados[estado][ano] = qtd            # Armazena contagem para o ano específico

    except Exception as e:                      # Trata erros na leitura/processamento
        print(f"Erro no ano {ano}: {e}")        # Exibe mensagem de erro detalhada

# 3. Transformação e limpeza dos dados
df_estados = pd.DataFrame(dados).fillna(0).astype(int)  # Converte dicionário para DataFrame, preenche missing com 0
df_estados = df_estados.T                               # Transpõe o DataFrame (estados nas linhas, anos nas colunas)

# 4. Seleção dos top 10 estados
total_por_estado = df_estados.sum(axis=1)               # Soma ocorrências por estado
top_10_estados = total_por_estado.sort_values(ascending=False).head(10).index  # Índices dos 10 estados com maiores totais
df_top10 = df_estados.loc[top_10_estados]               # Filtra apenas os top 10 estados

# 5. Preparação para plotagem
df_plot = df_top10.T                                    # Transpõe novamente para anos nas linhas e estados nas colunas

# 6. Geração do gráfico de linhas
plt.figure(figsize=(12, 7))                             # Cria figura com tamanho personalizado

# Plota uma linha para cada estado
for estado in df_plot.columns:                          # Itera sobre cada estado no DataFrame
    plt.plot(df_plot.index, df_plot[estado],            # Cria linha com marcadores circulares
             marker='o',
             label=estado)

# Customização do gráfico
plt.title('Top 10 Estados com Mais Ocorrências de Incêndios por Ano')  # Título principal
plt.xlabel('Ano')                                       # Rótulo do eixo X
plt.ylabel('Número de Ocorrências')                     # Rótulo do eixo Y
plt.legend(title='Estado',                              # Legenda com título
           bbox_to_anchor=(1.05, 1),                    # Posiciona legenda fora do gráfico à direita
           loc='upper left')
plt.grid(True, alpha=0.3)                               # Adiciona grade semi-transparente
plt.tight_layout()                                      # Ajuste automático do layout
plt.savefig('graficos/Top10_ocorrencias_por_estado_ano.png', dpi=300)  # Salva gráfico em alta resolução
plt.close()                                             # Fecha a figura
```



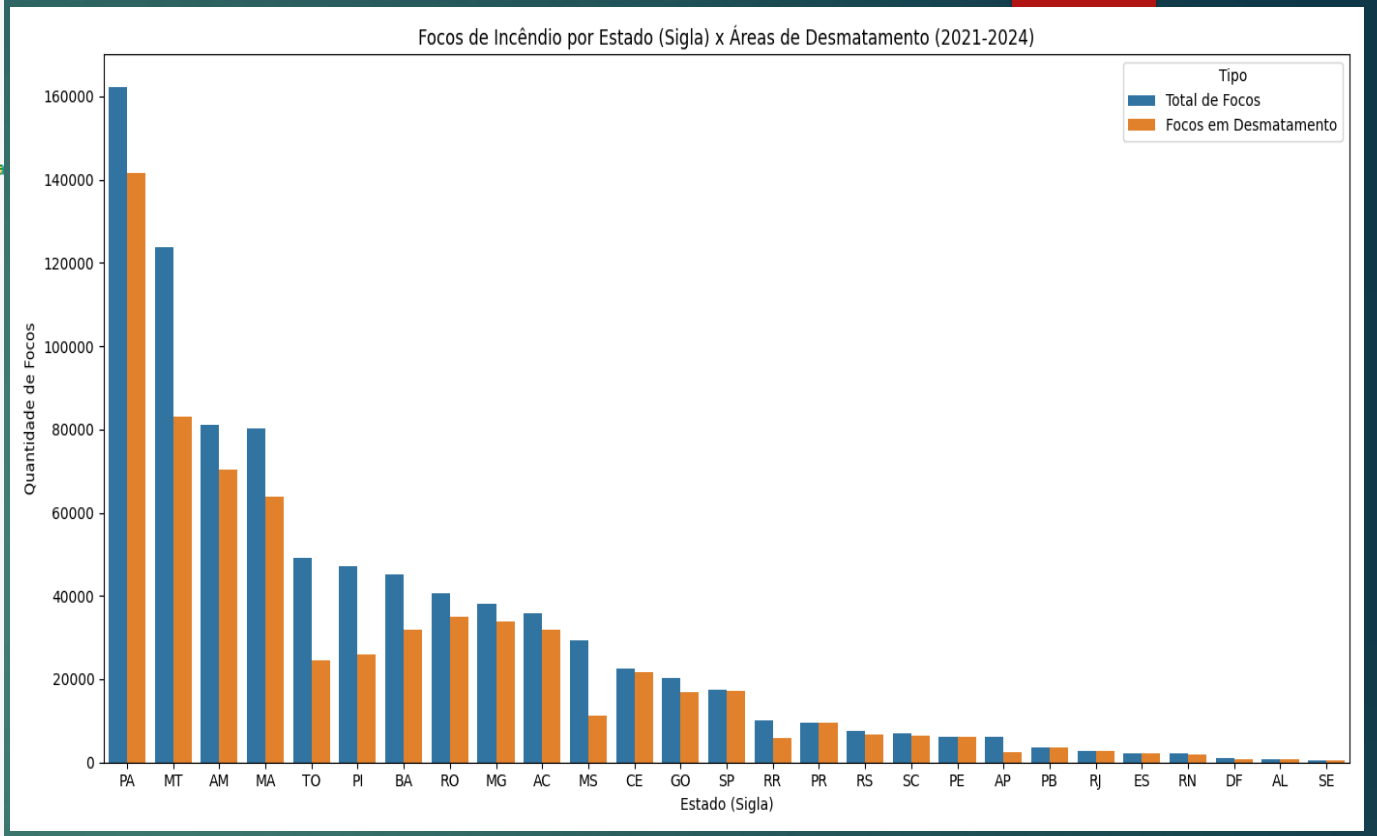
Evolução das queimadas no Brasil por estado

```python
# 2. Processamento de cada arquivo anual
for ano in anos:                # Itera sobre cada ano na lista de entrada
    try:
        # Carrega apenas a coluna 'bioma' do arquivo CSV correspondente
        df = pd.read_csv(
            f'data/focos_br_todos-sats_{ano}.csv',
            usecols=['bioma']   # Seleciona apenas coluna relevante para economia de memória
        )
        df = df.dropna(subset=['bioma'])  # Remove registros com bioma não identificado
        df['ano'] = ano                   # Adiciona coluna com ano atual para análise temporal
        lista_dfs.append(df)              # Adiciona DataFrame processado à lista

    except Exception as e:
        print(f"Erro ao ler {ano}: {e}")  # Exibe erro detalhado se houver falha na leitura

# 3. Verificação de dados carregados
if not lista_dfs:               # Se nenhum DataFrame foi carregado com sucesso
    print("Nenhum dado carregado. Verifique os arquivos CSV.")
    return

# 4. Consolidação de dados
df_completo = pd.concat(lista_dfs, ignore_index=True)  # Combina todos os DataFrames em um só

# 5. Ordenação de biomas por relevância
ordem_biomas = df_completo['bioma'].value_counts().index  # Ordena biomas pela frequência total

# 6. Configuração do gráfico
plt.figure(figsize=(14, 8))    # Cria figura com tamanho personalizado (largura x altura)

# Cria gráfico de barras agrupadas por ano
sns.countplot(
    data=df_completo,
    x='bioma',                 # Variável no eixo X (biomas)
    hue='ano',                 # Cores representam anos diferentes
    order=ordem_biomas         # Ordena biomas pela frequência total
)

# 7. Personalização do gráfico
plt.title('Evolução de focos de incêndio por bioma (2021-2024)')  # Título descritivo
plt.ylabel('Quantidade de focos')     # Rótulo eixo Y
plt.xlabel('Bioma')                   # Rótulo eixo X
plt.xticks(rotation=30)               # Rotaciona labels do eixo X para melhor legibilidade
plt.legend(title='Ano')               # Título da legenda
plt.tight_layout()                    # Ajuste automático do layout
plt.savefig('graficos/evolucao_biomas_anos_2021_-_2024.png', dpi=300)  # Salva em alta resolução
plt.close()                           # Fecha a figura
```



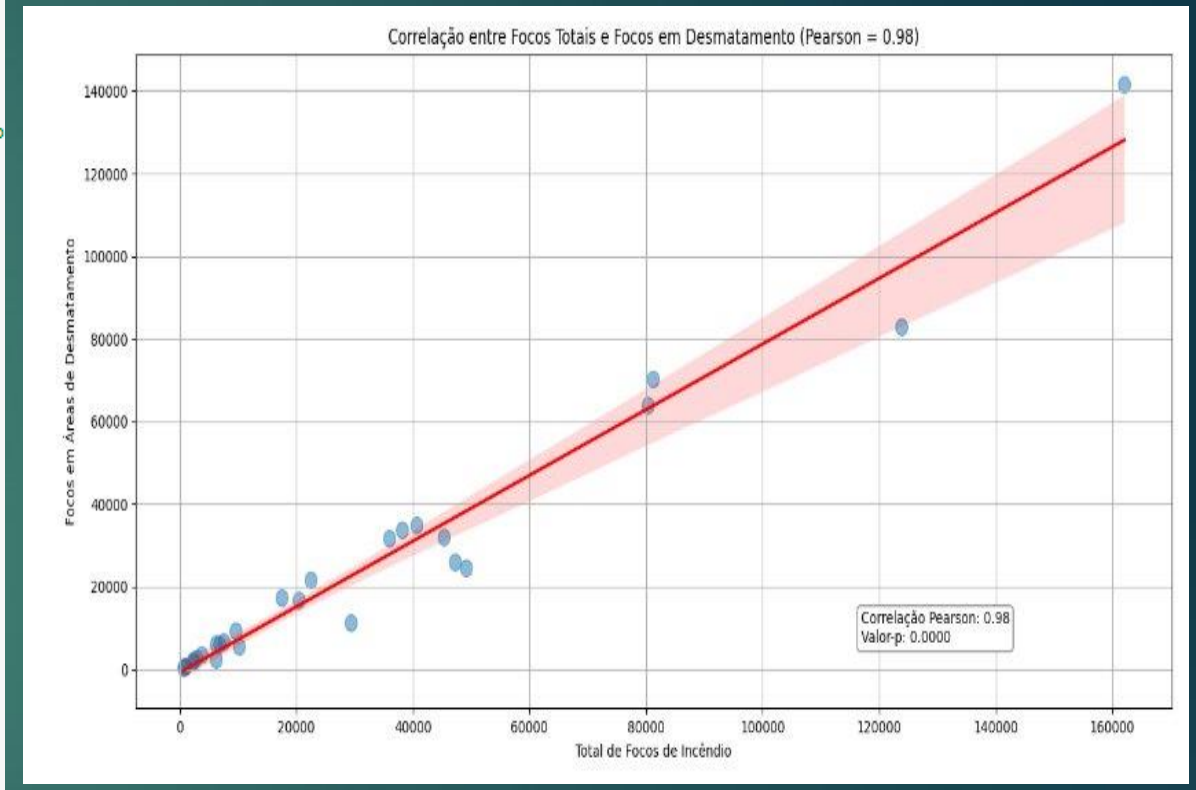
Evolução das queimadas no Brasil por bioma

```python
# 7. Agregação de dados por estado
# Soma total de focos por estado
focos_total_estado = df.groupby('uf_sigla')['focuses'].sum().reset_index()
# Soma focos em desmatamento
focos_desmatamento_estado = desmatamento.groupby('uf_sigla')['focuses'].sum().reset_index()

# 8. Renomeação de colunas para clareza
focos_total_estado = focos_total_estado.rename(columns={'focuses':'Total de Focos'})# Renomeia coluna
focos_desmatamento_estado = focos_desmatamento_estado.rename(columns={'focuses':'Focos em Desmatamento'})# Renomeia coluna

# 9. Combinação e preparação dos dados
comparativo = pd.merge(focos_total_estado, focos_desmatamento_estado, on='uf_sigla', how='left').fillna(0)# Combina dados
comparativo = comparativo.sort_values('Total de Focos', ascending=False)# Ordena por total de focos

# 10. Reformatação para plotagem
comparativo_melt = comparativo.melt(
    id_vars='uf_sigla',                                  # Mantém siglas como identificador
    value_vars=['Total de Focos', 'Focos em Desmatamento'],  # Colunas para combinar
    var_name='Tipo',                                     # Nome da nova coluna de categorias
    value_name='Focos'                                   # Nome da coluna de valores
)

# 11. Geração do gráfico comparativo
plt.figure(figsize=(14, 7))                              # Define tamanho da figura
sns.barplot(
    data=comparativo_melt,                               # DataFrame formatado
    x='uf_sigla',                                        # Eixo X: siglas dos estados
    y='Focos',                                           # Eixo Y: quantidade de focos
    hue='Tipo'                                           # Cores diferentes para cada categoria
)
plt.title('Focos de Incêndio por Estado (Sigla) x Áreas de Desmatamento (2021-2024)')  # Título descritivo
plt.xlabel('Estado (Sigla)')                             # Rótulo do eixo X
plt.ylabel('Quantidade de Focos')                        # Rótulo do eixo Y
plt.legend(title='Tipo')                                 # Legenda com título
plt.tight_layout()                                       # Ajuste automático do layout
plt.savefig('graficos/Focos de Incêndio por Estado (Sigla) x Áreas de Desmatamento.png', dpi=300)  # Salva gráfico
plt.close()                                              # Fecha figura
```



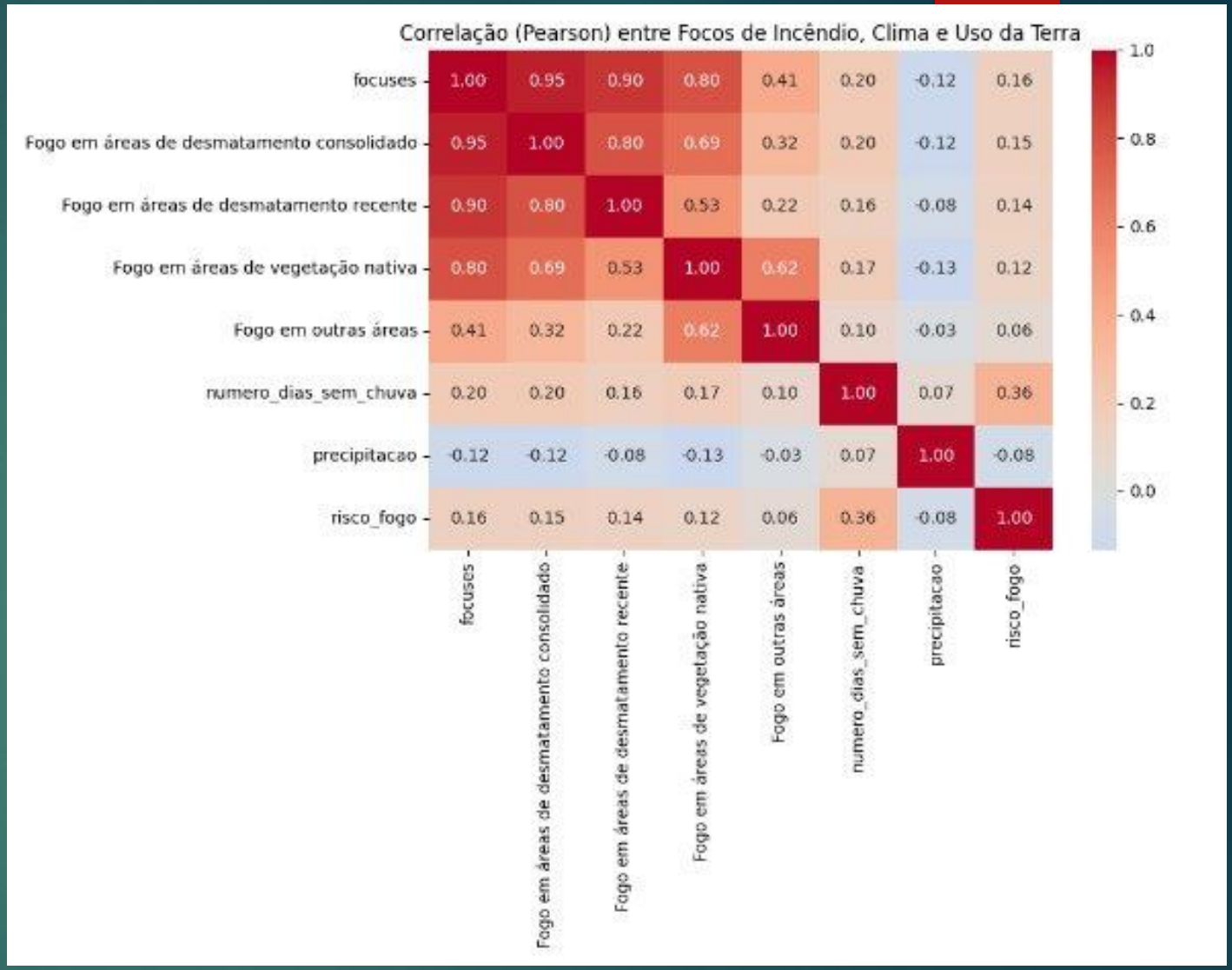
Focos de incêndios por estado x áreas desmatadas

```python
df = pd.read_csv(arquivo_csv, sep=';')            # Carrega arquivo CSV com separador ;
df.columns = df.columns.str.strip().str.lower()   # Padroniza nomes das colunas: remove espaços e converte para minúsculas
df['uf'] = df['uf'].str.upper()                   # Converte nomes de estados para maiúsculas
df['uf_sigla'] = df['uf'].map(estado_para_sigla).fillna(df['uf'])  # Cria coluna com siglas dos estados
df['focuses'] = pd.to_numeric(df['focuses'], errors='coerce')  # Converte coluna de focos para numérico
df['date'] = pd.to_datetime(df['date'], format='%Y/%m', errors='coerce')  # Converte datas para formato datetime

# 2. Filtragem de focos de desmatamento
desmatamento = df[df['class'].str.contains('desmatamento', case=False)]  # Filtra registros com menção a desmatamento

# 3. Agregação de dados por estado
total_focos = df.groupby('uf_sigla')['focuses'].sum().reset_index(name='Total de Focos')  # Soma total de focos por estado
focos_desmat = desmatamento.groupby('uf_sigla')['focuses'].sum().reset_index(name='Focos em Desmatamento')  # Soma focos em desmatamento

# 4. Combinação dos datasets
comparativo = pd.merge(total_focos, focos_desmat, on='uf_sigla', how='left').fillna(0)  # Junta dados totais e de desmatamento

# 5. Análise estatística
correlacao, p_valor = pearsonr(comparativo['Total de Focos'], comparativo['Focos em Desmatamento'])  # Calcula correlação de Pearson

# 6. Visualização gráfica
plt.figure(figsize=(10, 6))               # Define tamanho da figura
sns.regplot(
    data=comparativo,
    x='Total de Focos',
    y='Focos em Desmatamento',
    scatter_kws={'s': 80, 'alpha': 0.6},  # Configura pontos: tamanho 80, transparência 60%
    line_kws={'color': 'red'}             # Configura linha de regressão vermelha
)
plt.title(f'Relação entre Focos Totais e em Desmatamento (r = {correlacao:.2f})')  # Título dinâmico
plt.xlabel('Total de Focos Registrados')          # Rótulo eixo X
plt.ylabel('Focos em Áreas de Desmatamento')      # Rótulo eixo Y
plt.grid(True, alpha=0.3)                         # Adiciona grade semi-transparente

# Anotação com métricas estatísticas
plt.annotate(
    f'Correlação Pearson: {correlacao:.2f}\nValor-p: {p_valor:.4f}',  # Texto com resultados
    xy=(0.68, 0.15), xycoords='axes fraction',    # Posição no gráfico (68% da largura, 15% da altura)
    bbox=dict(boxstyle="round", fc="white", ec="gray", alpha=0.8)  # Caixa de texto estilizada
)

plt.tight_layout()                                # Ajuste automático do layout
plt.savefig('correlacao_focos_desmatamento.png', dpi=300)  # Salva imagem em alta resolução
plt.close()                                       # Fecha a figura para liberar memória
```



Focos de incêndios x desmatamentos

```python
# Junta os dados de incêndios com os dados de clima, pela chave (estado, ano_mes)
df = pd.merge(incendios_agg, clima_agg, on=['estado', 'ano_mes'], how='inner')

# Junta o resultado anterior com os dados de desmatamento
df = pd.merge(df, desmatamento_pivot, on=['estado', 'ano_mes'], how='left')

# Remove registros com valores nulos após as junções
df = df.dropna()

# Lista de variáveis que serão incluídas na análise de correlação
variaveis_relevantes = [
    'focuses',  # Total de focos de incêndio
    'Fogo em áreas de desmatamento consolidado',
    'Fogo em áreas de desmatamento recente',
    'Fogo em áreas de vegetação nativa',
    'Fogo em outras áreas',
    'numero_dias_sem_chuva',
    'precipitacao',
    'risco_fogo'
]

# Calcula a matriz de correlação de Pearson entre as variáveis selecionadas
df_corr_relevantes = df[variaveis_relevantes].corr(method='pearson')

# Criação do heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(
    df_corr_relevantes, annot=True, cmap='coolwarm', fmt=".2f", center=0
)
plt.title('Correlação (Pearson) entre Focos de Incêndio, Clima e Uso da Terra')
plt.tight_layout()

# Salva o gráfico como imagem
plt.savefig('graficos/Correlação entre Focos de Incêndio, Clima e Uso da Terra.png')
plt.close()
```



Correlação entre focos x desmatamento x fatores climaticos

Conclusão

- A análise dos dados entre 2021 e 2024 revelou que os focos de incêndio no Brasil continuam fortemente associados ao desmatamento, especialmente em biomas como a Amazônia e o Cerrado. Identificamos que os estados com maior número de focos coincidem com regiões críticas de desmatamento, e fatores climáticos como seca prolongada e aumento do risco de fogo também contribuem significativamente para a intensificação das queimadas.
- Nossos gráficos e análises estatísticas demonstraram correlações relevantes entre variáveis ambientais e os incêndios, evidenciando que esse problema é multifatorial e exige ações integradas.
- Portanto, é urgente que políticas públicas sejam fortalecidas, a fiscalização ambiental ampliada e a sociedade mobilizada para preservar nossas florestas. Mais do que números, os dados revelam uma ameaça concreta ao meio ambiente, à biodiversidade e à qualidade de vida das próximas gerações.