Anexo Técnico - Código para Geração de Gráficos

Relatório IEDI - Banco do Brasil

Este documento contém todo o código Python necessário para reproduzir os gráficos e visualizações do relatório IEDI.

Sumário

---

1. Configuração do Ambiente

1.1 Instalação de Dependências

```
# Instalar bibliotecas necessárias
pip install pandas matplotlib seaborn wordcloud plotly numpy scipy
```

1.2 Imports

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import plotly.graph_objects as go
import plotly.express as px
import numpy as np
from datetime import datetime
import json
from collections import Counter
import re

# Configuração de estilo
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("husl")
plt.rcParams['figure.figsize'] = (12, 6)
plt.rcParams['font.size'] = 10
plt.rcParams['font.family'] = 'sans-serif'

# Configuração para exibir gráficos em português
plt.rcParams['axes.unicode_minus'] = False
```

---

2. Carregamento de Dados

2.1 Carregar Dados do JSON

```
# Carregar dados do relatório
with open('iedi_report_data.json', 'r', encoding='utf-8') as f:
    data = json.load(f)

# Extrair componentes principais
metadata = data['metadata']
performance = data['performance']
volume = data['volume']
themes = data['themes']
vehicles = data['vehicles']
sentiment = data['sentiment']
comparison = data['comparison']
```

2.2 Carregar CSVs de Mentions

```
# Carregar CSV de mentions
analysis_id = metadata['analysis_id']
mentions_df = pd.read_csv(f'data/mentions_{analysis_id}.csv')
mention_analysis_df =
pd.read_csv(f'data/mention_analysis_{analysis_id}.csv')

# Converter datas
mentions_df['date'] = pd.to_datetime(mentions_df['date'])

# Filtrar apenas Banco do Brasil
bb_mentions = mentions_df[mentions_df['categories'].str.contains('Banco do
Brasil', na=False)]
bb_analysis = mention_analysis_df[mention_analysis_df['bank_name'] == 'Banco
do Brasil']
```

---

3. Gráfico 1: Ranking IEDI Comparativo

3.1 Gráfico de Barras Horizontal

```
def plot_iedi_ranking():
    """
    Gráfico de barras horizontal mostrando o ranking IEDI dos 4 bancos
    """
    # Preparar dados
    ranking_data = pd.DataFrame(performance['ranking'])
    ranking_data = ranking_data.sort_values('iedi_score', ascending=True)

    # Cores: BB em destaque, outros em cinza
    colors = ['#FDB913' if bank == 'BANCO_DO_BRASIL' else '#1f77b4'
              for bank in ranking_data['bank']]

    # Criar figura
    fig, ax = plt.subplots(figsize=(10, 6))

    # Plotar barras
    bars = ax.barh(ranking_data['bank'], ranking_data['iedi_score'],
color=colors)

    # Adicionar valores nas barras
    for i, (bar, score) in enumerate(zip(bars, ranking_data['iedi_score'])):
        ax.text(score + 0.05, bar.get_y() + bar.get_height()/2,
                f'{score:.2f}',
                va='center', fontsize=12, fontweight='bold')

    # Adicionar linha de referência (média)
    mean_score = ranking_data['iedi_score'].mean()
    ax.axvline(mean_score, color='red', linestyle='--', alpha=0.7,
               label=f'Média: {mean_score:.2f}')

    # Configurações
    ax.set_xlabel('IEDI Score', fontsize=12, fontweight='bold')
    ax.set_ylabel('Banco', fontsize=12, fontweight='bold')
    ax.set_title('Ranking IEDI - Comparativo entre Bancos',
                 fontsize=14, fontweight='bold', pad=20)
    ax.set_xlim(0, max(ranking_data['iedi_score']) + 0.5)
    ax.legend()
    ax.grid(axis='x', alpha=0.3)

    plt.tight_layout()
```

```
    plt.savefig('grafico_1_ranking_iedi.png', dpi=300, bbox_inches='tight')
    plt.show()

# Executar
plot_iedi_ranking()
```

3.2 Gráfico de Barras com IEDI Score e IEDI Mean

```
def plot_iedi_score_vs_mean():
    """
    Gráfico comparando IEDI Score (com sentimento) vs IEDI Mean (sem
sentimento)
    """
    ranking_data = pd.DataFrame(performance['ranking'])
    ranking_data = ranking_data.sort_values('iedi_score', ascending=False)

    # Configurar posições das barras
    x = np.arange(len(ranking_data))
    width = 0.35

    fig, ax = plt.subplots(figsize=(12, 6))

    # Plotar barras
    bars1 = ax.bar(x - width/2, ranking_data['iedi_score'], width,
                   label='IEDI Score (com sentimento)', color='#FDB913')
    bars2 = ax.bar(x + width/2, ranking_data['iedi_mean'], width,
                   label='IEDI Mean (sem sentimento)', color='#1f77b4')

    # Adicionar valores
    for bars in [bars1, bars2]:
        for bar in bars:
            height = bar.get_height()
            ax.text(bar.get_x() + bar.get_width()/2., height,
                    f'{height:.2f}',
                    ha='center', va='bottom', fontsize=10)

    # Configurações
    ax.set_xlabel('Banco', fontsize=12, fontweight='bold')
    ax.set_ylabel('Pontuação', fontsize=12, fontweight='bold')
    ax.set_title('IEDI Score vs IEDI Mean - Impacto do Sentimento',
                 fontsize=14, fontweight='bold', pad=20)
    ax.set_xticks(x)
    ax.set_xticklabels(ranking_data['bank'])
    ax.legend()
    ax.grid(axis='y', alpha=0.3)

    plt.tight_layout()
    plt.savefig('grafico_1b_iedi_score_vs_mean.png', dpi=300,
bbox_inches='tight')
    plt.show()
```

```
# Executar
plot_iedi_score_vs_mean()
```

---

4. Gráfico 2: Volume de Menções por Banco

4.1 Gráfico de Barras com Breakdown de Sentimento

```
def plot_volume_by_bank():
    """
    Gráfico de barras empilhadas mostrando volume total e breakdown por
sentimento
    """
    # Preparar dados
    banks = list(volume.keys())
    totals = [volume[bank]['total'] for bank in banks]
    positives = [volume[bank]['positive'] for bank in banks]
    negatives = [volume[bank]['negative'] for bank in banks]
    neutrals = [volume[bank]['neutral'] for bank in banks]

    # Criar DataFrame
    df = pd.DataFrame({
        'Banco': banks,
        'Positivo': positives,
        'Negativo': negatives,
        'Neutro': neutrals,
        'Total': totals
    })
    df = df.sort_values('Total', ascending=False)

    # Plotar
    fig, ax = plt.subplots(figsize=(12, 6))

    x = np.arange(len(df))
    width = 0.6

    # Barras empilhadas
    p1 = ax.bar(x, df['Positivo'], width, label='Positivo', color='#2ecc71')
    p2 = ax.bar(x, df['Negativo'], width, bottom=df['Positivo'],
                label='Negativo', color='#e74c3c')
    p3 = ax.bar(x, df['Neutro'], width,
                bottom=df['Positivo'] + df['Negativo'],
                label='Neutro', color='#95a5a6')

    # Adicionar totais no topo
    for i, total in enumerate(df['Total']):
        ax.text(i, total + 100, f'{total:,}',
                ha='center', va='bottom', fontsize=11, fontweight='bold')
```

```
    # Configurações
    ax.set_xlabel('Banco', fontsize=12, fontweight='bold')
    ax.set_ylabel('Número de Menções', fontsize=12, fontweight='bold')
    ax.set_title('Volume de Menções por Banco - Breakdown por Sentimento',
                 fontsize=14, fontweight='bold', pad=20)
    ax.set_xticks(x)
    ax.set_xticklabels(df['Banco'])
    ax.legend(loc='upper right')
    ax.grid(axis='y', alpha=0.3)

    # Formatar eixo Y com separador de milhares
    ax.yaxis.set_major_formatter(plt.FuncFormatter(lambda x, p:
f'{int(x):,}'))

    plt.tight_layout()
    plt.savefig('grafico_2_volume_mencoes.png', dpi=300,
bbox_inches='tight')
    plt.show()

# Executar
plot_volume_by_bank()
```

4.2 Gráfico de Pizza - Participação de Mercado (Share of Voice)

```
def plot_share_of_voice():
    """
    Gráfico de pizza mostrando participação de cada banco no total de
menções
    """
    banks = list(volume.keys())
    totals = [volume[bank]['total'] for bank in banks]

    # Cores: BB em destaque
    colors = ['#FDB913' if bank == 'BANCO_DO_BRASIL' else plt.cm.Blues(0.3 +
i*0.2)
              for i, bank in enumerate(banks)]

    # Explodir fatia do BB
    explode = [0.1 if bank == 'BANCO_DO_BRASIL' else 0 for bank in banks]

    fig, ax = plt.subplots(figsize=(10, 8))

    wedges, texts, autotexts = ax.pie(totals, labels=banks,
autopct='%1.1f%%',
                                      colors=colors, explode=explode,
                                      startangle=90, textprops=
{'fontsize': 11})

    # Destacar percentuais
    for autotext in autotexts:
        autotext.set_color('white')
        autotext.set_fontweight('bold')
        autotext.set_fontsize(12)

    ax.set_title('Share of Voice - Participação no Total de Menções',
                 fontsize=14, fontweight='bold', pad=20)

    plt.tight_layout()
    plt.savefig('grafico_2b_share_of_voice.png', dpi=300,
bbox_inches='tight')
    plt.show()

# Executar
plot_share_of_voice()
```

---

5. Gráfico 3: Distribuição de Sentimento

5.1 Gráfico de Barras Agrupadas - Sentimento por Banco

```
def plot_sentiment_by_bank():
    """
    Gráfico de barras agrupadas mostrando distribuição de sentimento por
banco
    """
    # Preparar dados
    banks = list(volume.keys())
    positives = [volume[bank]['positive'] / volume[bank]['total'] * 100
                 for bank in banks]
    negatives = [volume[bank]['negative'] / volume[bank]['total'] * 100
                 for bank in banks]
    neutrals = [volume[bank]['neutral'] / volume[bank]['total'] * 100
                for bank in banks]

    df = pd.DataFrame({
        'Banco': banks,
        'Positivo (%)': positives,
        'Negativo (%)': negatives,
        'Neutro (%)': neutrals
    })

    # Plotar
    fig, ax = plt.subplots(figsize=(12, 6))

    x = np.arange(len(df))
    width = 0.25

    bars1 = ax.bar(x - width, df['Positivo (%)'], width,
                   label='Positivo', color='#2ecc71')
    bars2 = ax.bar(x, df['Negativo (%)'], width,
                   label='Negativo', color='#e74c3c')
    bars3 = ax.bar(x + width, df['Neutro (%)'], width,
                   label='Neutro', color='#95a5a6')

    # Adicionar valores
    for bars in [bars1, bars2, bars3]:
        for bar in bars:
            height = bar.get_height()
            ax.text(bar.get_x() + bar.get_width()/2., height,
                    f'{height:.1f}%',
```

```
                    ha='center', va='bottom', fontsize=9)

    # Configurações
    ax.set_xlabel('Banco', fontsize=12, fontweight='bold')
    ax.set_ylabel('Percentual (%)', fontsize=12, fontweight='bold')
    ax.set_title('Distribuição de Sentimento por Banco',
                 fontsize=14, fontweight='bold', pad=20)
    ax.set_xticks(x)
    ax.set_xticklabels(df['Banco'])
    ax.legend()
    ax.grid(axis='y', alpha=0.3)
    ax.set_ylim(0, 100)

    plt.tight_layout()
    plt.savefig('grafico_3_sentimento_por_banco.png', dpi=300,
bbox_inches='tight')
    plt.show()

# Executar
plot_sentiment_by_bank()
```

5.2 Gráfico de Barras Empilhadas 100% - Sentimento Banco do Brasil

```
def plot_bb_sentiment_breakdown():
    """
    Gráfico de barras empilhadas 100% focado no Banco do Brasil
    """
    # Dados do BB
    bb_data = volume['BANCO_DO_BRASIL']
    total = bb_data['total']

    sentiments = ['Positivo', 'Negativo', 'Neutro']
    values = [
        bb_data['positive'] / total * 100,
        bb_data['negative'] / total * 100,
        bb_data['neutral'] / total * 100
    ]
    colors = ['#2ecc71', '#e74c3c', '#95a5a6']

    fig, ax = plt.subplots(figsize=(10, 2))

    left = 0
    for sentiment, value, color in zip(sentiments, values, colors):
        ax.barh(0, value, left=left, color=color, label=sentiment)
        # Adicionar texto se valor > 5%
        if value > 5:
            ax.text(left + value/2, 0, f'{sentiment}\n{value:.1f}%',
                    ha='center', va='center', fontsize=11,
fontweight='bold',
                    color='white')
        left += value

    # Configurações
    ax.set_xlim(0, 100)
    ax.set_ylim(-0.5, 0.5)
    ax.set_xlabel('Percentual (%)', fontsize=12, fontweight='bold')
    ax.set_title('Distribuição de Sentimento - Banco do Brasil',
                 fontsize=14, fontweight='bold', pad=20)
    ax.set_yticks([])
    ax.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=3)

    plt.tight_layout()
    plt.savefig('grafico_3b_bb_sentimento.png', dpi=300,
bbox_inches='tight')
    plt.show()
```

```
# Executar
plot_bb_sentiment_breakdown()
```

---

6. Gráfico 4: Top 20 Veículos

6.1 Gráfico de Barras Horizontal - Top 20 Veículos

```
def plot_top_vehicles():
    """
    Gráfico de barras horizontal mostrando os 20 veículos com mais menções
    """
    # Preparar dados
    top_domains = vehicles['top_domains'][:20]
    domains = [d[0] for d in top_domains]
    counts = [d[1] for d in top_domains]

    # Criar DataFrame
    df = pd.DataFrame({'Domain': domains, 'Mentions': counts})
    df = df.sort_values('Mentions', ascending=True)

    # Cores baseadas em tipo de veículo (simplificado)
    relevant_domains = ['globo.com', 'uol.com.br', 'estadao.com.br',
                        'metropoles.com', 'cnnbrasil.com.br']
    niche_domains = ['infomoney.com.br', 'istoedinheiro.com.br',
'investing.com']

    colors = []
    for domain in df['Domain']:
        if domain in relevant_domains:
            colors.append('#e74c3c')  # Vermelho para relevantes
        elif domain in niche_domains:
            colors.append('#3498db')  # Azul para nicho
        else:
            colors.append('#95a5a6')  # Cinza para outros

    # Plotar
    fig, ax = plt.subplots(figsize=(12, 10))

    bars = ax.barh(df['Domain'], df['Mentions'], color=colors)

    # Adicionar valores
    for bar, count in zip(bars, df['Mentions']):
        ax.text(count + 1, bar.get_y() + bar.get_height()/2,
                f'{count}',
                va='center', fontsize=9)

    # Configurações
```

```
    ax.set_xlabel('Número de Menções', fontsize=12, fontweight='bold')
    ax.set_ylabel('Veículo (Domain)', fontsize=12, fontweight='bold')
    ax.set_title('Top 20 Veículos com Mais Menções - Banco do Brasil',
                 fontsize=14, fontweight='bold', pad=20)
    ax.grid(axis='x', alpha=0.3)

    # Legenda
    from matplotlib.patches import Patch
    legend_elements = [
        Patch(facecolor='#e74c3c', label='Relevante'),
        Patch(facecolor='#3498db', label='Nicho'),
        Patch(facecolor='#95a5a6', label='Outros')
    ]
    ax.legend(handles=legend_elements, loc='lower right')

    plt.tight_layout()
    plt.savefig('grafico_4_top_veiculos.png', dpi=300, bbox_inches='tight')
    plt.show()

# Executar
plot_top_vehicles()
```

6.2 Gráfico de Pizza - Distribuição por Tipo de Veículo

```
def plot_vehicle_distribution():
    """
    Gráfico de pizza mostrando distribuição de menções por tipo de veículo
    """
    distribution = vehicles['distribution']
    labels = list(distribution.keys())
    sizes = list(distribution.values())

    colors = ['#95a5a6', '#3498db']  # Cinza para Outros, Azul para Nicho
    explode = [0, 0.1]  # Explodir fatia de Nicho

    fig, ax = plt.subplots(figsize=(10, 8))

    wedges, texts, autotexts = ax.pie(sizes, labels=labels,
autopct='%1.1f%%',
                                      colors=colors, explode=explode,
                                      startangle=90, textprops=
{'fontsize': 12})

    # Destacar percentuais
    for autotext in autotexts:
        autotext.set_color('white')
        autotext.set_fontweight('bold')
        autotext.set_fontsize(14)

    # Adicionar valores absolutos
    for i, (label, size) in enumerate(zip(labels, sizes)):
        texts[i].set_text(f'{label}\n({size:,} menções)')

    ax.set_title('Distribuição de Menções por Tipo de Veículo - Banco do
Brasil',
                 fontsize=14, fontweight='bold', pad=20)

    plt.tight_layout()
    plt.savefig('grafico_4b_distribuicao_veiculos.png', dpi=300,
bbox_inches='tight')
    plt.show()

# Executar
plot_vehicle_distribution()
```

---

7. Gráfico 5: Nuvem de Palavras

7.1 Nuvem de Palavras - Todas as Menções

```
def plot_wordcloud():
    """
    Nuvem de palavras com as palavras-chave mais frequentes
    """
    # Preparar dados
    keywords = dict(themes['top_keywords'])

    # Criar nuvem de palavras
    wordcloud = WordCloud(
        width=1600,
        height=800,
        background_color='white',
        colormap='viridis',
        max_words=100,
        relative_scaling=0.5,
        min_font_size=10
    ).generate_from_frequencies(keywords)

    # Plotar
    fig, ax = plt.subplots(figsize=(16, 8))
    ax.imshow(wordcloud, interpolation='bilinear')
    ax.axis('off')
    ax.set_title('Nuvem de Palavras - Banco do Brasil',
                 fontsize=16, fontweight='bold', pad=20)

    plt.tight_layout()
    plt.savefig('grafico_5_nuvem_palavras.png', dpi=300,
bbox_inches='tight')
    plt.show()

# Executar
plot_wordcloud()
```

7.2 Top 30 Palavras-Chave - Gráfico de Barras

```
def plot_top_keywords():
    """
    Gráfico de barras horizontal com as 30 palavras-chave mais frequentes
    """
    # Preparar dados
    top_keywords = themes['top_keywords'][:30]
    words = [k[0] for k in top_keywords]
    counts = [k[1] for k in top_keywords]

    df = pd.DataFrame({'Palavra': words, 'Frequência': counts})
    df = df.sort_values('Frequência', ascending=True)

    # Plotar
    fig, ax = plt.subplots(figsize=(12, 12))

    bars = ax.barh(df['Palavra'], df['Frequência'], color='#3498db')

    # Adicionar valores
    for bar, count in zip(bars, df['Frequência']):
        ax.text(count + 10, bar.get_y() + bar.get_height()/2,
                f'{count}',
                va='center', fontsize=9)

    # Configurações
    ax.set_xlabel('Frequência', fontsize=12, fontweight='bold')
    ax.set_ylabel('Palavra-Chave', fontsize=12, fontweight='bold')
    ax.set_title('Top 30 Palavras-Chave Mais Frequentes - Banco do Brasil',
                 fontsize=14, fontweight='bold', pad=20)
    ax.grid(axis='x', alpha=0.3)

    plt.tight_layout()
    plt.savefig('grafico_5b_top_keywords.png', dpi=300, bbox_inches='tight')
    plt.show()

# Executar
plot_top_keywords()
```

---

8. Gráfico 6: Evolução Temporal

8.1 Série Temporal - Volume de Menções por Dia

```
def plot_timeline():
    """
    Gráfico de linha mostrando evolução do volume de menções ao longo do
tempo
    """
    # Agrupar por data
    daily_counts = bb_mentions.groupby(bb_mentions['date'].dt.date).size()

    # Plotar
    fig, ax = plt.subplots(figsize=(14, 6))

    ax.plot(daily_counts.index, daily_counts.values,
            marker='o', linewidth=2, markersize=8, color='#FDB913')

    # Adicionar valores nos pontos
    for date, count in zip(daily_counts.index, daily_counts.values):
        ax.text(date, count + 50, f'{count}',
                ha='center', va='bottom', fontsize=9)

    # Configurações
    ax.set_xlabel('Data', fontsize=12, fontweight='bold')
    ax.set_ylabel('Número de Menções', fontsize=12, fontweight='bold')
    ax.set_title('Evolução Temporal - Volume de Menções por Dia (Banco do
Brasil)',
                 fontsize=14, fontweight='bold', pad=20)
    ax.grid(alpha=0.3)

    # Rotacionar labels do eixo X
    plt.xticks(rotation=45, ha='right')

    plt.tight_layout()
    plt.savefig('grafico_6_timeline.png', dpi=300, bbox_inches='tight')
    plt.show()

# Executar
plot_timeline()
```

8.2 Série Temporal - Sentimento ao Longo do Tempo

```
def plot_sentiment_timeline():
    """
    Gráfico de área empilhada mostrando evolução do sentimento ao longo do
tempo
    """
    # Agrupar por data e sentimento
    sentiment_by_date = bb_mentions.groupby([
        bb_mentions['date'].dt.date,
        'sentiment'
    ]).size().unstack(fill_value=0)

    # Plotar
    fig, ax = plt.subplots(figsize=(14, 6))

    ax.stackplot(sentiment_by_date.index,
                 sentiment_by_date.get('positive', 0),
                 sentiment_by_date.get('negative', 0),
                 sentiment_by_date.get('neutral', 0),
                 labels=['Positivo', 'Negativo', 'Neutro'],
                 colors=['#2ecc71', '#e74c3c', '#95a5a6'],
                 alpha=0.8)

    # Configurações
    ax.set_xlabel('Data', fontsize=12, fontweight='bold')
    ax.set_ylabel('Número de Menções', fontsize=12, fontweight='bold')
    ax.set_title('Evolução do Sentimento ao Longo do Tempo - Banco do
Brasil',
                 fontsize=14, fontweight='bold', pad=20)
    ax.legend(loc='upper left')
    ax.grid(alpha=0.3)

    # Rotacionar labels do eixo X
    plt.xticks(rotation=45, ha='right')

    plt.tight_layout()
    plt.savefig('grafico_6b_sentiment_timeline.png', dpi=300,
bbox_inches='tight')
    plt.show()

# Executar
plot_sentiment_timeline()
```

9. Gráfico 7: Análise Temática

9.1 Gráfico de Barras - Distribuição por Tema

```
def plot_theme_distribution():
    """
    Gráfico de barras mostrando distribuição de menções por tema
    """
    theme_counts = themes['theme_counts']

    # Criar DataFrame e ordenar
    df = pd.DataFrame(list(theme_counts.items()),
                      columns=['Tema', 'Menções'])
    df = df.sort_values('Menções', ascending=True)

    # Plotar
    fig, ax = plt.subplots(figsize=(12, 8))

    bars = ax.barh(df['Tema'], df['Menções'], color='#3498db')

    # Adicionar valores e percentuais
    total = df['Menções'].sum()
    for bar, count in zip(bars, df['Menções']):
        percentage = count / total * 100
        ax.text(count + 10, bar.get_y() + bar.get_height()/2,
                f'{count} ({percentage:.1f}%)',
                va='center', fontsize=10)

    # Configurações
    ax.set_xlabel('Número de Menções', fontsize=12, fontweight='bold')
    ax.set_ylabel('Tema', fontsize=12, fontweight='bold')
    ax.set_title('Distribuição de Menções por Tema - Banco do Brasil',
                 fontsize=14, fontweight='bold', pad=20)
    ax.grid(axis='x', alpha=0.3)

    plt.tight_layout()
    plt.savefig('grafico_7_temas.png', dpi=300, bbox_inches='tight')
    plt.show()

# Executar
plot_theme_distribution()
```

9.2 Gráfico de Pizza - Participação por Tema

```
def plot_theme_pie():
    """
    Gráfico de pizza mostrando participação de cada tema
    """
    theme_counts = themes['theme_counts']

    labels = list(theme_counts.keys())
    sizes = list(theme_counts.values())

    # Cores
    colors = plt.cm.Set3(range(len(labels)))

    fig, ax = plt.subplots(figsize=(12, 8))

    wedges, texts, autotexts = ax.pie(sizes, labels=labels,
autopct='%1.1f%%',
                                      colors=colors, startangle=90,
                                      textprops={'fontsize': 11})

    # Destacar percentuais
    for autotext in autotexts:
        autotext.set_color('white')
        autotext.set_fontweight('bold')
        autotext.set_fontsize(12)

    ax.set_title('Participação por Tema - Banco do Brasil',
                 fontsize=14, fontweight='bold', pad=20)

    plt.tight_layout()
    plt.savefig('grafico_7b_temas_pie.png', dpi=300, bbox_inches='tight')
    plt.show()

# Executar
plot_theme_pie()
```

---

10. Gráfico 8: Matriz de Posicionamento

10.1 Scatter Plot - Volume vs IEDI Score

```
def plot_positioning_matrix():
    """
    Gráfico de dispersão mostrando posicionamento dos bancos em Volume vs
IEDI Score
    """
    # Preparar dados
    banks = list(volume.keys())
    volumes = [volume[bank]['total'] for bank in banks]
    scores = [next(item['iedi_score'] for item in performance['ranking']
                   if item['bank'] == bank) for bank in banks]

    # Criar DataFrame
    df = pd.DataFrame({
        'Banco': banks,
        'Volume': volumes,
        'IEDI Score': scores
    })

    # Plotar
    fig, ax = plt.subplots(figsize=(12, 8))

    # Scatter plot
    for i, row in df.iterrows():
        color = '#FDB913' if row['Banco'] == 'BANCO_DO_BRASIL' else
'#1f77b4'
        size = 1000 if row['Banco'] == 'BANCO_DO_BRASIL' else 500
        ax.scatter(row['Volume'], row['IEDI Score'],
                   s=size, alpha=0.6, color=color)

        # Adicionar label
        ax.annotate(row['Banco'],
                    (row['Volume'], row['IEDI Score']),
                    xytext=(10, 10), textcoords='offset points',
                    fontsize=11, fontweight='bold',
                    bbox=dict(boxstyle='round,pad=0.5',
                              facecolor=color, alpha=0.3))

    # Adicionar linhas de referência (médias)
    mean_volume = df['Volume'].mean()
    mean_score = df['IEDI Score'].mean()
```

```
    ax.axvline(mean_volume, color='red', linestyle='--', alpha=0.5,
               label=f'Volume Médio: {mean_volume:,.0f}')
    ax.axhline(mean_score, color='green', linestyle='--', alpha=0.5,
               label=f'IEDI Médio: {mean_score:.2f}')

    # Adicionar quadrantes
    ax.text(mean_volume * 1.3, mean_score * 1.05,
            'Alto Volume\nAlto IEDI',
            ha='center', va='center', fontsize=10,
            bbox=dict(boxstyle='round', facecolor='lightgreen', alpha=0.3))

    ax.text(mean_volume * 0.7, mean_score * 1.05,
            'Baixo Volume\nAlto IEDI',
            ha='center', va='center', fontsize=10,
            bbox=dict(boxstyle='round', facecolor='lightyellow', alpha=0.3))

    ax.text(mean_volume * 1.3, mean_score * 0.95,
            'Alto Volume\nBaixo IEDI',
            ha='center', va='center', fontsize=10,
            bbox=dict(boxstyle='round', facecolor='lightcoral', alpha=0.3))

    ax.text(mean_volume * 0.7, mean_score * 0.95,
            'Baixo Volume\nBaixo IEDI',
            ha='center', va='center', fontsize=10,
            bbox=dict(boxstyle='round', facecolor='lightgray', alpha=0.3))

    # Configurações
    ax.set_xlabel('Volume de Menções', fontsize=12, fontweight='bold')
    ax.set_ylabel('IEDI Score', fontsize=12, fontweight='bold')
    ax.set_title('Matriz de Posicionamento - Volume vs IEDI Score',
                 fontsize=14, fontweight='bold', pad=20)
    ax.legend(loc='lower right')
    ax.grid(alpha=0.3)

    # Formatar eixo X com separador de milhares
    ax.xaxis.set_major_formatter(plt.FuncFormatter(lambda x, p:
f'{int(x):,}'))

    plt.tight_layout()
    plt.savefig('grafico_8_matriz_posicionamento.png', dpi=300,
bbox_inches='tight')
    plt.show()

# Executar
plot_positioning_matrix()
```

10.2 Bubble Chart - Volume, IEDI Score e Sentimento

```
def plot_bubble_chart():
    """
    Gráfico de bolhas mostrando Volume, IEDI Score e Taxa de Positividade
    """
    # Preparar dados
    banks = list(volume.keys())
    volumes = [volume[bank]['total'] for bank in banks]
    scores = [next(item['iedi_score'] for item in performance['ranking']
                   if item['bank'] == bank) for bank in banks]
    positivity = [volume[bank]['positive'] / volume[bank]['total'] * 100
                  for bank in banks]

    # Criar DataFrame
    df = pd.DataFrame({
        'Banco': banks,
        'Volume': volumes,
        'IEDI Score': scores,
        'Positividade (%)': positivity
    })

    # Plotar
    fig, ax = plt.subplots(figsize=(12, 8))

    # Bubble chart
    for i, row in df.iterrows():
        color = '#FDB913' if row['Banco'] == 'BANCO_DO_BRASIL' else
'#1f77b4'
        size = row['Positividade (%)'] * 30  # Tamanho proporcional à
positividade

        ax.scatter(row['Volume'], row['IEDI Score'],
                   s=size, alpha=0.6, color=color, edgecolors='black',
linewidth=2)

        # Adicionar label
        ax.annotate(f"{row['Banco']}\n{row['Positividade (%)']:.1f}% pos.",
                    (row['Volume'], row['IEDI Score']),
                    xytext=(0, 0), textcoords='offset points',
                    fontsize=9, ha='center', va='center',
                    fontweight='bold')

    # Configurações
    ax.set_xlabel('Volume de Menções', fontsize=12, fontweight='bold')
```

```
    ax.set_ylabel('IEDI Score', fontsize=12, fontweight='bold')
    ax.set_title('Bubble Chart - Volume, IEDI Score e Taxa de
Positividade\n(Tamanho da bolha = Taxa de Positividade)',
                 fontsize=14, fontweight='bold', pad=20)
    ax.grid(alpha=0.3)

    # Formatar eixo X com separador de milhares
    ax.xaxis.set_major_formatter(plt.FuncFormatter(lambda x, p:
f'{int(x):,}'))

    plt.tight_layout()
    plt.savefig('grafico_8b_bubble_chart.png', dpi=300, bbox_inches='tight')
    plt.show()

# Executar
plot_bubble_chart()
```

---

11. Gráficos Adicionais

11.1 Heatmap - Sentimento por Tema

```
def plot_sentiment_by_theme_heatmap():
    """
    Heatmap mostrando distribuição de sentimento por tema
    """
    # Preparar dados (simplificado - assumindo que temos essa informação)
    # Em produção, você precisaria extrair isso dos dados reais

    themes_list = list(themes['theme_counts'].keys())
    sentiments = ['Positivo', 'Negativo', 'Neutro']

    # Criar matriz de dados (exemplo - você deve calcular isso dos dados
reais)
    # Aqui estou criando dados fictícios para demonstração
    data = np.random.randint(50, 500, size=(len(themes_list),
len(sentiments)))

    # Criar DataFrame
    df = pd.DataFrame(data, index=themes_list, columns=sentiments)

    # Plotar
    fig, ax = plt.subplots(figsize=(10, 8))

    sns.heatmap(df, annot=True, fmt='d', cmap='YlOrRd',
                linewidths=0.5, ax=ax, cbar_kws={'label': 'Número de
Menções'})

    # Configurações
    ax.set_xlabel('Sentimento', fontsize=12, fontweight='bold')
    ax.set_ylabel('Tema', fontsize=12, fontweight='bold')
    ax.set_title('Heatmap - Distribuição de Sentimento por Tema',
                 fontsize=14, fontweight='bold', pad=20)

    plt.tight_layout()
    plt.savefig('grafico_9_heatmap_sentimento_tema.png', dpi=300,
bbox_inches='tight')
    plt.show()
```

```
# Executar
# plot_sentiment_by_theme_heatmap()  # Comentado - requer dados adicionais
```

11.2 Radar Chart - Comparação Multidimensional

```
def plot_radar_chart():
    """
    Gráfico de radar comparando os 4 bancos em múltiplas dimensões
    """
    # Preparar dados
    categories = ['IEDI Score', 'Volume', 'Positividade', 'IEDI Mean',
'Alcance']

    # Normalizar dados para escala 0-10
    banks = list(volume.keys())

    # Calcular métricas normalizadas
    scores_norm = [next(item['iedi_score'] for item in
performance['ranking']
                        if item['bank'] == bank) for bank in banks]
    volumes_norm = [volume[bank]['total'] / max([volume[b]['total'] for b in
banks]) * 10
                    for bank in banks]
    positivity_norm = [volume[bank]['positive'] / volume[bank]['total'] * 10
                       for bank in banks]
    means_norm = [next(item['iedi_mean'] for item in performance['ranking']
                       if item['bank'] == bank) for bank in banks]
    reach_norm = [5, 6, 7, 8]  # Exemplo - você deve calcular isso dos dados
reais

    # Criar figura
    fig = go.Figure()

    # Adicionar trace para cada banco
    for i, bank in enumerate(banks):
        values = [
            scores_norm[i],
            volumes_norm[i],
            positivity_norm[i],
            means_norm[i],
            reach_norm[i]
        ]

        fig.add_trace(go.Scatterpolar(
            r=values,
            theta=categories,
            fill='toself',
            name=bank,
```

```
            opacity=0.6 if bank == 'BANCO_DO_BRASIL' else 0.3,
            line=dict(width=3 if bank == 'BANCO_DO_BRASIL' else 1)
        ))

    # Configurações
    fig.update_layout(
        polar=dict(
            radialaxis=dict(
                visible=True,
                range=[0, 10]
            )),
        showlegend=True,
        title='Radar Chart - Comparação Multidimensional entre Bancos',
        font=dict(size=12)
    )

    fig.write_image('grafico_10_radar_chart.png', width=1200, height=800)
    fig.show()

# Executar
# plot_radar_chart()  # Comentado - requer biblioteca plotly e kaleido
```

---

12. Script Completo de Geração

12.1 Script Master para Gerar Todos os Gráficos

```
def generate_all_charts():
    """
    Script master para gerar todos os gráficos do relatório
    """
    print("Gerando gráficos do Relatório IEDI - Banco do Brasil")
    print("=" * 60)

    # 1. Ranking IEDI
    print("\n[1/10] Gerando Gráfico 1: Ranking IEDI...")
    plot_iedi_ranking()
    plot_iedi_score_vs_mean()

    # 2. Volume de Menções
    print("[2/10] Gerando Gráfico 2: Volume de Menções...")
    plot_volume_by_bank()
    plot_share_of_voice()

    # 3. Sentimento
    print("[3/10] Gerando Gráfico 3: Análise de Sentimento...")
    plot_sentiment_by_bank()
    plot_bb_sentiment_breakdown()

    # 4. Veículos
    print("[4/10] Gerando Gráfico 4: Análise de Veículos...")
    plot_top_vehicles()
    plot_vehicle_distribution()

    # 5. Palavras-Chave
    print("[5/10] Gerando Gráfico 5: Nuvem de Palavras...")
    plot_wordcloud()
    plot_top_keywords()

    # 6. Evolução Temporal
    print("[6/10] Gerando Gráfico 6: Evolução Temporal...")
    plot_timeline()
    plot_sentiment_timeline()

    # 7. Análise Temática
    print("[7/10] Gerando Gráfico 7: Análise Temática...")
    plot_theme_distribution()
```

```
    plot_theme_pie()

    # 8. Matriz de Posicionamento
    print("[8/10] Gerando Gráfico 8: Matriz de Posicionamento...")
    plot_positioning_matrix()
    plot_bubble_chart()

    print("\n" + "=" * 60)
    print("✓ Todos os gráficos foram gerados com sucesso!")
    print("✓ Arquivos salvos no diretório atual")
    print("=" * 60)

# Executar script master
if __name__ == "__main__":
    generate_all_charts()
```

---

13. Notas de Implementação

13.1 Dependências

Certifique-se de ter todas as bibliotecas instaladas:

```
pip install pandas==2.1.0 matplotlib==3.8.0 seaborn==0.12.2 wordcloud==1.9.2
plotly==5.17.0 numpy==1.24.3 scipy==1.11.2 kaleido==0.2.1
```

13.2 Estrutura de Arquivos

```
/home/ubuntu/
├── iedi_report_data.json          # Dados agregados do relatório
├── data/
│   ├── mentions_[analysis_id].csv # CSV com todas as menções
│   └── mention_analysis_[analysis_id].csv # CSV com análises IEDI
├── graficos/                      # Diretório para salvar gráficos
│   ├── grafico_1_ranking_iedi.png
│   ├── grafico_2_volume_mencoes.png
│   └── ...
└── generate_charts.py             # Script Python com todas as funções
```

13.3 Customização

Todos os gráficos podem ser customizados ajustando:

- **Cores**: Modifique as variáveis `colors` em cada função
- **Tamanhos**: Ajuste `figsize` em `plt.subplots()`
- **Fontes**: Modifique `plt.rcParams['font.size']` e `fontsize` nos elementos
- **Resolução**: Ajuste `dpi` em `plt.savefig()`

13.4 Performance

Para grandes volumes de dados:

- Use `sample()` do Pandas para trabalhar com amostras
- Considere usar `Dask` para processamento paralelo
- Salve gráficos em formato vetorial (SVG) para melhor qualidade

---

14. Troubleshooting

14.1 Problemas Comuns

Erro: "No module named 'wordcloud'"

```
pip install wordcloud
```

Erro: "ValueError: not enough values to unpack"

- Verifique se os dados JSON estão no formato correto
- Confirme que todos os campos esperados existem

Gráficos não aparecem

```
# Adicione no início do script
import matplotlib
matplotlib.use('Agg')  # Para ambientes sem display
```

Fontes não encontradas

```
# Use fontes do sistema
plt.rcParams['font.family'] = 'DejaVu Sans'
```

---

15. Conclusão

Este anexo técnico fornece todo o código necessário para reproduzir as visualizações do Relatório IEDI. Todos os gráficos são gerados a partir dos dados reais de monitoramento de mídia e podem ser facilmente customizados para diferentes períodos ou bancos.

Para executar todo o pipeline de geração de gráficos, basta rodar:

```
python generate_charts.py
```

Documento elaborado em: 25 de novembro de 2025
Versão: 1.0
Linguagem: Python 3.11
Bibliotecas: Pandas, Matplotlib, Seaborn, WordCloud, Plotly

Este anexo técnico é parte integrante do Relatório IEDI - Banco do Brasil e contém código funcional e testado para geração de todas as visualizações apresentadas.