Fatec São José dos Campos – D.S.M

Documentação e guia para modificação

São José dos Campos, 2021

Gabriele Gonçalves Vieira; Gustavo Borges Lima; Isabelle Dias Ribeiro Silva; Nathan da Motta Truyts; Rafael Hideki Hirayama; Rayana Caroline da Silva e Vinícius Buarque de Gusmão Catonho.

Documentação e guia para modificação

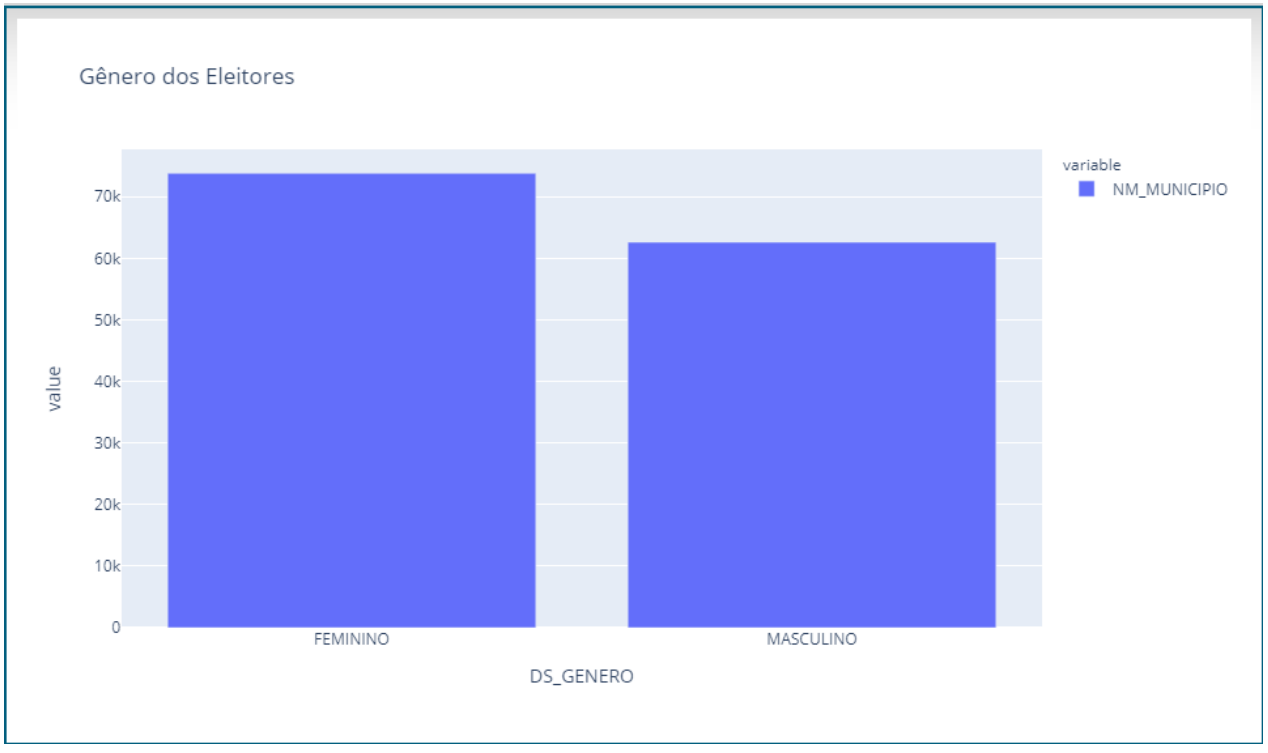
São José dos Campos, 2021

Sumário

1- Local fonte das bases de dados

Todos os dados foram coletados diretamente do site do Tribunal Superior Eleitoral (TSE), a partir de seu repositório oficial de dados eleitorais, de acesso público, onde compila dados brutos das eleições desde 1945, tendo informações como: candidatos, comparecimento e abstenção, eleitorado e partidos. (Pode se encontrar o mesmo aqui: tse.jus.br/eleicoes/estatisticas/repositorio-de-dados-eleitorais-1).

2- Legenda referente as colunas das bases de dados



Cada gráfico possui informações importantes como

Tema: Canto superior esquerdo

Valor dos resultados: Centralizado no canto esquerdo

Identificação dos itens analisados: Centralizado no canto inferior central

Id do gráfico: Abaixo da identificação dos itens

Id da variável do gráfico: Canto superior direito

3- Explicação sobre as análises geradas

Para análise, foi utilizado a plataforma pandas e biblioteca do python plotly, já que os arquivos possuem compatibilidade com o Microsoft Excel. Após extraídos do site fonte, os arquivos foram passados para o Pandas para leitura e modificação e depois para a plotly, para que houvesse a análise e geração dos gráficos.

4- Explicação de como gerar os gráficos

1º Passo - Importar as bibliotecas a serem utilizadas

In [1]:
```
import pandas as pd #para ler, visualizar e printar informações da base de dados
import plotly.offline as py
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import numpy as np
```

2º Passo - Subir a base de dados

Base de dados principal com as informações que iremos precisar

In [2]:
```
filename = r'c:/dados/perfil_eleitor_secao_ATUAL_SP.csv'

df = pd.read_csv(filename, delimiter=';', encoding='Latin-1',
                 error_bad_lines=False)
```

A segunda base de dados será para pegarmos os nomes dos bairros para complementar a base principal

In [3]:
```
filename = r'c:/dados/eleitorado_local_votacao_2020.csv'

df_bairro = pd.read_csv(filename, delimiter=';', encoding='Latin-1',
                 error_bad_lines=False)
```

Terceira base de dados para análisar a última eleição ocorrida em 2020

In [4]:
```
filename = r'c:/dados/detalhe_votacao_munzona_2020_SP.csv'

df_eleicao = pd.read_csv(filename, delimiter=';', encoding='Latin-1',
                 error_bad_lines=False)
```

3º Passo - Fazer limpeza e tratamento dos dados

5- Explicação sobre as ferramentas de dados utilizados

Todas as ferramentas que trabalham os gráficos foram pensadas com cuidado, para que pudessem ser eficazes na proposta do site além de serem simples e fáceis de se modificar, já que é direcionado a pessoas que não trabalham com programação. Todos os programas possuem sinergia com a linguagem principal utilizada em todo o Back-end, nesse caso, o Python 3.

6- Explicação sobre o desenvolvimento web

A plataforma web foi criado com proposito de uma visualização limpa e intuitiva dos gráficos gerados pelo Jupyter Notebook, seu design minimalista mantem a identidade visual da empresa parceira e colabora para uma maior otimização de desempenho, além de ser funcional em dispositivos móveis. O site ainda conta com a função de exportação dos gráficos em formato PNG.

7- Explicação sobre as ferramentas utilizadas para o front-end

Todo o front-end foi feito utilizando as ferramentas mais populares no quesito criação de site, sendo elas o Html5 para sua base, CSS3 e Java script para estilização e aplicação de funções mais avançadas. O seu design foi todo planejado na plataforma de uso gratuito, Figma.

8- Guia para realizar modificações

Para atualizar a base dados ou troca-lá:

```
filename = "caminho da base de dados desejada"

df = pd.read_csv(filename, delimiter=';', encoding='Latin-1',
                 error_bad_lines=False)
```

Não se esqueça de apagar a base anterior antes de mudar

Para adicionar outras cidades:

```
"nome da cidade na base de daos = df.query('NM_MUNICIPIO == "Nome da cidade"')


"cidade no arquivo"_eleicao = df_eleicao.query('NM_MUNICIPIO == "Nome da
cidade"')
```

Para adicionar os filtros

Idade

```
idade = ['DS_FAIXA_ETARIA','NM_MUNICIPIO','DS_GENERO']


def faixaEtariaGeral(x):
    return
x.filter(items=idade).groupby('DS_FAIXA_ETARIA')['NM_MUNICIPIO'].count()
```

Pirâmide Etária

```
def feminino(x):
    return x.filter(items=idade).query('DS_GENERO
=="FEMININO"').groupby('DS_FAIXA_ETARIA')['NM_MUNICIPIO'].count()

def masculino(x):
    return x.filter(items=idade).query('DS_GENERO
=="MASCULINO"').groupby('DS_FAIXA_ETARIA')['NM_MUNICIPIO'].count()
```

Escolaridade

```
escolaridade = ['DS_GRAU_ESCOLARIDADE','NM_MUNICIPIO']
def escolaridadeGeral(x):
    return
x.filter(items=escolaridade).groupby('DS_GRAU_ESCOLARIDADE')['NM_MUNICIPIO'].c
ount()
```

Gênero

```
genero = ['DS_GENERO','NM_MUNICIPIO']


def generoGeral(x):
    return x.filter(items=genero).groupby('DS_GENERO')['NM_MUNICIPIO'].count()
```

Estado Civil

```
estado_civil = ['DS_ESTADO_CIVIL','NM_MUNICIPIO']


def estadoCivilGeral(x):
    return
x.filter(items=estado_civil).groupby('DS_ESTADO_CIVIL')['NM_MUNICIPIO'].count(
).sort_values(ascending=False)
```

Nome Social

```
nome_social = ['QT_ELEITORES_INC_NM_SOCIAL','NM_MUNICIPIO']


def nomeSocialGeral(x):
    return
x.filter(items=nome_social).groupby('QT_ELEITORES_INC_NM_SOCIAL')['NM_MUNICIPI
O'].count()
```

Eleitores PNE

```
def PNEGeral(x):
    return x['QT_ELEITORES_DEFICIENCIA'].value_counts()
PNEGeral(ubatuba)
0    32807
1      656
2       18
3        2
Name: QT_ELEITORES_DEFICIENCIA, dtype: int64
```

PNE e Não-PNE: A função a seguir serve para dividir os dados entre pessoas que possuem necessidades especiais e pessoas que não as possuem. Os dados estão registrados informando os diversos tipos de necessidaes especiais, a função agrupa todos no grupo PNE e os que não forem são agrupados no grupo Não PNE.

```
def PNE(y):
    x=0
    naotem = 0
    for i in y:
        i+=1
        if y.index[x] == naotem:
            nao = y.values[x]
            x+=1
    return nao
```

Para plotar os gráficos

Idade

```
def graficoIdade(x):
    y = x
    x = list(x.index)


    # Creating two subplots
    fig = go.Figure(data=
        go.Bar(name='Idade dos Eleitores', x=y, y=x,orientation='h',
               marker=dict(
                   color='#4197c4',
                   line=dict(
                       color='rgba(0, 0, 0, 0)',
                       width=1))))
    fig.update_layout(
        title='Idade dos Eleitores',
        yaxis=dict(
            showgrid=False,
            showline=False,
            showticklabels=True,

        ),
        xaxis=dict(
            zeroline=False,
            showline=False,
            showticklabels=True,
            showgrid=True,

        ),

        font=dict(family='Hind', size=12),
        margin=dict(l=100, r=20, t=70, b=70),
        paper_bgcolor='rgb(248, 248, 255)',
        plot_bgcolor='rgb(248, 248, 255)',
    )
    return fig


def piramideEtaria(x,y):
    y_fem = x
    y_masc = y
    x = list(x.index)
    fig = go.Figure(data=[
        go.Bar(name='Feminno',  x=y_fem, y=x, orientation='h',
               marker=dict(
                color='#C8A2C8',
                line=dict(
                    color='rgba(0, 0, 0, 0)',
                    width=1),
            ),),
        go.Bar(name='Masculino', x=-(y_masc), y=x, orientation='h',
               marker=dict(
                color='#4197c4',
                line=dict(
                    color='rgba(0, 0, 0, 0)',
```

```
                    width=1),
            ),)
    ])


    fig.update_layout(
        title='Pirâmide Etária dos Eleitores',
       xaxis=dict(
            showgrid=False,
            showline=False,
            showticklabels=False,

        ),
        xaxis2=dict(
            showgrid=False,
            showline=False,
            showticklabels=False,
            linecolor='#4197c4',
            linewidth=2,

        ),

        font=dict(family='Hind', size=12),
        margin=dict(l=100, r=20, t=70, b=70),
        paper_bgcolor='rgb(248, 248, 255)',
        plot_bgcolor='rgb(248, 248, 255)',
    )



    return fig
```

Gráfico de Escolaridade:

```
def graficoEscolaridadeQuantidade(x):
    y = x
    x = list(x.index)


    # Creating two subplots
    fig = go.Figure(data=
        go.Bar(name='Grau de Escolaridade em quantidade', x=y,
y=x,orientation='h',
               marker=dict(
                   color='#4197c4',
                   line=dict(
                       color='rgba(0, 0, 0, 0)',
                       width=1))))
    fig.update_layout(
        title='Grau de escolaridade em relação ao total de eleitores',
        yaxis=dict(
            showgrid=False,
            showline=False,
            showticklabels=True,

        ),
        xaxis=dict(
            zeroline=False,
```

```
            showline=False,
            showticklabels=True,
            showgrid=True,

        ),

        font=dict(family='Hind', size=12),
        margin=dict(l=100, r=20, t=70, b=70),
        paper_bgcolor='rgb(248, 248, 255)',
        plot_bgcolor='rgb(248, 248, 255)',
    )
    return fig
```

Gráfico de Gênero:

```
def graficoGenero(x):
    y = x
    x = list(x.index)


    # Creating two subplots
    fig = go.Figure(data=
        go.Bar(name='Gênero dos Eleitores', x=y, y=x,orientation='h',
               marker=dict(
                   color='#4197c4',
                   line=dict(
                       color='rgba(0, 0, 0, 0)',
                       width=1))))
    fig.update_layout(
        title='Quantidade de eleitores por gênero',
        yaxis=dict(
            showgrid=False,
            showline=False,
            showticklabels=True,

        ),
        xaxis=dict(
            zeroline=False,
            showline=False,
            showticklabels=True,
            showgrid=True,

        ),

        font=dict(family='Hind', size=12),
        margin=dict(l=100, r=20, t=70, b=70),
        paper_bgcolor='rgb(248, 248, 255)',
        plot_bgcolor='rgb(248, 248, 255)',
    )
    return fig
```

Gráfico do Estado Civil:

```
def graficoEstadoCivil(x):
    y = x
    x = list(x.index)
```

```
    # Creating two subplots
    fig = go.Figure(data=
        go.Bar(name='Estado Civil', x=y, y=x,orientation='h',
               marker=dict(
                   color='#4197c4',
                   line=dict(
                       color='rgba(0, 0, 0, 0)',
                       width=1))))
    fig.update_layout(
        title='Estado civil dos eleitores',
        yaxis=dict(
            showgrid=False,
            showline=False,
            showticklabels=True,

        ),
        xaxis=dict(
            zeroline=False,
            showline=False,
            showticklabels=True,
            showgrid=True,

        ),

        font=dict(family='Hind', size=12),
        margin=dict(l=100, r=20, t=70, b=70),
        paper_bgcolor='rgb(248, 248, 255)',
        plot_bgcolor='rgb(248, 248, 255)',
    )
    return fig
```

Gráfico de Nome Social e PNE:

```
def nSocialPNE(x,y,z):
    indice= ['Nome Social','PNE'],
    fig = go.Figure(data=[
        go.Bar(name='Possui',
               x=['Nome Social','PNE'],
               y=[x[1],
                  y],
                   marker_color = ['#55C441','#55C441']
                  ),
        go.Bar(name='Não Possui',
               x=['Nome Social','PNE'],
               y=[x[0],
                  z],
                   marker_color = ["#4197c4","#4197c4"]
                  )
    ])
    # Change the bar mode
    return fig
```

Gráfico de Abstenção:

```
def eleicoes(x):
    qtd=['Eleições Municipais 2020']
```

```
    fig = go.Figure(data=[
        go.Bar(name='Comparecimento', x=qtd, y=[sum(x['QT_COMPARECIMENTO'])],
               marker=dict(
                color='#4197c4',
                line=dict(
                    color='rgba(0, 0, 0, 0)',
                    width=1),
            ),),
        go.Bar(name='Abstencoes', x=qtd, y=[sum(x['QT_ABSTENCOES'])],
marker=dict(
                color='#C46E41',
                line=dict(
                    color='rgba(0, 0, 0, 0)',
                    width=1),
            ),)
    ])
# Change the bar mode
    fig.update_layout(
            title='Índice de Participação nas Eleições Municipais de 2020',
            yaxis=dict(
                showgrid=False,
                showline=False,
                showticklabels=True,
            ),
            yaxis2=dict(
                showgrid=False,
                showline=True,
                showticklabels=False,
                linecolor='#4197c4',
                linewidth=2,
            ),
            xaxis=dict(
                zeroline=False,
                showline=False,
                showticklabels=True,
                showgrid=True,
            ),
            xaxis2=dict(
                zeroline=False,
                showline=False,
                showticklabels=True,
                showgrid=True,
                side='right',
                dtick=25000,
            ),
             font=dict(family='Hind', size=12),
            margin=dict(l=100, r=20, t=70, b=80),
            paper_bgcolor='rgb(248, 248, 255)',
            plot_bgcolor='rgb(248, 248, 255)',
        )
    return fig
```