# Supressão

## Uso de bibliotecas

Foram utilizadas as seguintes bibliotecas para resolução desse projeto

```
pip install pandas numpy matplotlib seaborn polars
pip install pyinstaller
```

> As lib acima requer o "install" via pip, além disso foram usadas outras libs no quais foram :
> * os , random, glob. Pyinstaller é para gerar o arquivo executavel.

> Ademais, cada função será explicada mais adiante.

## Código

Antes da resolução problema se faz necessario o import das libs para resolução desse problema

```
# Importando dados para realização do trabalho
import os
import glob
import random
import polars as pl
import seaborn as sea
import matplotlib.pyplot as plt
```

- glob e os : São libs utilizadas para criação de pastas (os). O glob usada para **localizar arquivos e pastas** com base em **padrões** de
- Random |: É usada para seleção de instancias aletórias para supressão
- Polars: Para leitura do csv. Ela é boa quando temos uma grande quantidade de dados
- seaborn : Para visualização
- matplotlib : Para visualização

```
def fn_supressao(subset, porcentagem=0.25):
    TAM = subset.shape[0]
    print("_"*100)
    print("FN_SUPRESSAO")
    print(f"Supressão para coluna de {subset.name} com {porcentagem*100}%
({int(TAM*porcentagem)})")

    print("* Escolhendo 25% de instancias aleatoriamente para supressão dos
respectivos valores")

    list_ = random.sample(range(TAM), k=int(TAM * porcentagem))
```

```
    print(f"* Instancias escolhidas\n   + ex: {', '.join(map(str,
list_[:5]))}....")


    array_ = subset.to_numpy()

    print('* Supressão dos valores')

    array_[list_] = ""
    return pl.Series(name='municipioCaso', values=array_.tolist(), dtype=str)
```

A função de supressão apagar uma parte dos dados de uma coluna, escolhendo aleatoriamente as posições a serem suprimidas(ela é feita usando o `random.sample` ).

```
def fn_supressao_to_csv(data:pl.DataFrame, porcentagens=[0.25, 0.50, 0.75]):
    series = data['municipioCaso']
    for por in porcentagens:

        # Modificando coluna
        data = data.with_columns(
            [fn_supressao(series, porcentagem=por)]
        )

        string = str(por).replace('.', '_')
        print('* Salvando resultados')
        print(f' =>dataset_result\\Salvando dados_covid-
ce_trab02_{string}.csv')

        os.makedirs(name='dataset_result', exist_ok=True)

        data.select(pl.col('municipioCaso')).write_csv(f'dataset_result\
\dados_covid-ce_trab02_{string}.csv')
        print('')

    print("Processo Supressão finalizado!")
```

A função aplica a supressão de dados na coluna `municipioCaso` em diferentes níveis (25%, 50%, 75%) e salva um arquivo CSV para cada versão do dataset.

```
print('Gerando frequencia dos resultados para :')
```

```
datas_sup = []

for path_name in glob.glob('dataset_result\\*.csv'):
    print(f' * {path_name}')

    character = path_name.split('_')[-1].split('.')[0]

    data_sup = pl.read_csv(path_name)

    data_sup = data_sup\
                    .group_by(['municipioCaso'])\
                    .len(name=f'freq_{character}')\
                    .sort(by=f'freq_{character}', descending=True)
    #
    datas_sup.append( data_sup )
```

O código busca todos os CSVs em `dataset_result` usando `glob`, lê cada arquivo com Polars e extrai parte do nome para usar na coluna de frequência, agrupa os dados por `municipioCaso`, conta as ocorrências e ordena pela frequência, e finalmente armazena cada DataFrame de frequência na lista `datas_sup`.

```
print("Junção das frequencias dos resultados")

datas_sub_final = datas_sup[0]

for e, df in enumerate(datas_sup[1:]):
    datas_sub_final = datas_sub_final.join(df, on='municipioCaso',
how='full',suffix=f"_dup_{e}")  # outer = une tudo

datas_sub_final = datas_sub_final.select([
    'municipioCaso', 'freq_25', 'freq_5', 'freq_75'
])
```

Com o `datas_sup` gera, é feito a junção de todos o resultados.

```
data = pl.read_csv('dataset\\dados_covid-ce_trab02.csv')
subset = data\
            .group_by(['municipioCaso'])\
```

```
            .len(name='Freq_Original')\
            .sort(by='Freq_Original', descending=True)
```

O mesmo processo do `datas_sup` é repetido, mas para o dados originais.

```
dataset_final = subset.join(
    datas_sub_final,
    on='municipioCaso', how='full',
).select(['municipioCaso', 'Freq_Original', 'freq_25', 'freq_5', 'freq_75'])
```

Dessa forma, feito a junção do datas_sup com o frequencia dos municipios com os dados originais. Assim, gerando o resultado desse trabalho

```
os.makedirs(name='dataset_result_final', exist_ok=True)

dataset_final.write_csv('dataset_result_final\\dataset_final.csv')
```

É criada a pasta dataset_result_final por meio do os e em seguida é salvo o resultado nessa pasta.

**Geração da visualização**

```
subset_vis = dataset_final.head(20)

subset_mel_vis = subset_vis.unpivot(
    on=["Freq_Original", "freq_25", "freq_5", 'freq_75'],
    index="municipioCaso",
    variable_name="Categoria",
    value_name="Frequencia"
)
```

É seleciona os 20 primeiros municípios do DataFrame. Em seguida, usa `unpivot` (melt) para transformar as colunas de frequência em formato longo. As colunas originais de frequência viram uma coluna `Categoria` indicando a origem. Isso é feito para gerção de uma melhor visualização no python.

```
plt.figure(figsize=(20, 22))
ax = sea.barplot(
    subset_mel_vis, y='municipioCaso', x='Frequencia', hue='Categoria'
)
plt.gca().spines[['top','bottom']].set_visible(False)
plt.gca().grid(axis='y')

for container in ax.containers:
    ax.bar_label(
        container,
        fmt='%.0f',
        label_type='edge',
        padding=3,
        fontsize=13,
        color='black'
    )
leg = ax.legend(
    title='Categoria',
    title_fontsize=14,
    fontsize=12,
    loc='upper right',
    bbox_to_anchor=(1.15, 1),
    frameon=True,
    shadow=True,
    borderpad=1.2
)

plt.title('Frequência dos Municípios por Categoria', fontsize=16)
plt.xlabel('Frequência')
plt.ylabel('Município')
plt.tight_layout()
plt.savefig(fname='plots\\plot_k20_total.pdf')
```

A visualização criada em um unico grafico

```
for freq in ["Freq_Original", "freq_25", "freq_5", "freq_75"]:
    plt.figure(figsize=(20, 10))
    sub = subset_mel_vis.filter(pl.col('Categoria')==freq)
    ax = sea.barplot(
        sub, y='municipioCaso', x='Frequencia', hue='Categoria'
    )
    plt.gca().spines[['top','bottom']].set_visible(False)
    plt.gca().grid(axis='y')
```

```
    for container in ax.containers:
        ax.bar_label(
            container,
            fmt='%.0f',
            label_type='edge',
            padding=3,
            fontsize=13,
            color='black'
        )
    leg = ax.legend(
        title='Categoria',
        title_fontsize=14,
        fontsize=12,
        loc='upper right',
        bbox_to_anchor=(1.15, 1),
        frameon=True,
        shadow=True,
        borderpad=1.2
    )

    plt.title('Frequência dos Municípios por Categoria', fontsize=16)
    plt.xlabel('Frequência')
    plt.ylabel('Município')
    plt.tight_layout()
    plt.savefig(fname=f'plots\\plot_k20_tota_{freq}.pdf')
```

Criando o histograma para cada frequencia