2 - Data Analysis

April 17, 2025

[ ]:
```
from src.libs.lib import *
import pandas as pd
import matplotlib.pyplot as plt
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
df_original = pd.read_csv("src/data/
 ↪tabela_ocorrencias_dbpx_com_a_pontuação_academia.csv")

#2.6s
```

[ ]:
```
import pandas as pd

# 1. Carregar dados (ajuste o caminho conforme necessário)
# df = pd.read_csv("src/data/tabela_ocorrencias_dbpx_com_a_pontuação_academia.
 ↪csv",
#                  parse_dates=['occurence_create', 'contract_start'])
# Para este exemplo, assume-se que o DataFrame já está em `df_original`
df = df_original.copy()

# 2. Identificar variáveis categóricas e numéricas
categorical_vars = df.select_dtypes(include=['object', 'category']).columns.
 ↪tolist()
numeric_vars = df.select_dtypes(include=['number']).columns.tolist()

print("Variáveis categóricas:", categorical_vars)
print("Variáveis numéricas:", numeric_vars)

# 3. Agregar estatísticas para cada variável categórica
agg_list = []
for cat in categorical_vars:
    stats = df.groupby(cat)[numeric_vars].agg(['count', 'mean', 'std', 'min',␣
 ↪'max'])
    # Flatten MultiIndex nas colunas
    stats.columns = [f"{num}_{stat}" for num, stat in stats.columns]
    stats = stats.reset_index().rename(columns={cat: 'Categoria'})
    stats['Variável_Categórica'] = cat
    agg_list.append(stats)
```

```
# 4. Concatenar todos os resultados num único DataFrame
agg_df = pd.concat(agg_list, ignore_index=True)

# 5. Reordenar colunas para melhor leitura
cols = ['Variável_Categórica', 'Categoria'] + \
       [c for c in agg_df.columns if c not in ['Variável_Categórica',␣
 ↪'Categoria']]
agg_df = agg_df[cols]

# 6. Exibir tabela agregada
print(agg_df)
```

[ ]:
```
df_original.head(50)
```

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import Normalize
from matplotlib import cm

# seu DataFrame
df = df_original.copy()

# contagem de ocorrências por descrição, em ordem decrescente
counts = df['description'].value_counts()

# prepara colormap
norm = Normalize(vmin=counts.min(), vmax=counts.max())
cmap = cm.get_cmap('Spectral')
colors = cmap(norm(counts.values))

# cria figura e eixos
fig, ax = plt.subplots(figsize=(12, 8))

# plota barras coloridas
bars = ax.bar(counts.index, counts.values, color=colors)

# define escala logarítmica no eixo Y
ax.set_yscale('log')

# labels e título
ax.set_xlabel('Descrição')
ax.set_ylabel('Contagem de Ocorrências (escala log)')
ax.set_title('Contagem de Ocorrências por Descrição')

# rotaciona ticks
plt.xticks(rotation=45, ha='right')
```

```
# adiciona barra de cor para referência de valores
sm = cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])
cbar = fig.colorbar(sm, ax=ax)
cbar.set_label('Número de Ocorrências')

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

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import Normalize
from matplotlib import cm

df = df_original.copy()

# Garante que a pontuação seja numérica
df['Pontuação'] = pd.to_numeric(df['Pontuação'], errors='coerce').fillna(0)

# Soma dos pontos por descrição, ordenado decrescente
sums = df.groupby('description')['Pontuação'].sum().sort_values(ascending=False)

# Prepara o colormap Spectral com normalização dos valores de soma
norm   = Normalize(vmin=sums.min(), vmax=sums.max())
cmap   = cm.get_cmap('Spectral')
colors = cmap(norm(sums.values))

# Cria figura e eixos
fig, ax = plt.subplots(figsize=(12, 8))

# Plota barras coloridas conforme o total de pontos
bars = ax.bar(sums.index, sums.values, color=colors)

# Define escala logarítmica no eixo Y
ax.set_yscale('log')

# Labels e título
ax.set_xlabel('Descrição')
ax.set_ylabel('Total de Pontos (escala log)')
ax.set_title('Total de Pontos por Descrição')

# Rotaciona e alinha os ticks do eixo x
plt.xticks(rotation=45, ha='right')

# Adiciona colorbar para referência de valores
sm = cm.ScalarMappable(cmap=cmap, norm=norm)
```

```
sm.set_array([])
cbar = fig.colorbar(sm, ax=ax)
cbar.set_label('Total de Pontos')

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

[ ]:
```
import matplotlib.pyplot as plt
from matplotlib.colors import Normalize
from matplotlib import cm

# Copia do DataFrame original (já contendo as linhas "Motorista enviado para atualização")
df = df_original.copy()

# Filtra somente as ocorrências de atualização
updates = df[df['description'] == 'Motorista enviado para atualização']

# Conta quantas atualizações por driver_id e pega os top 50
counts = updates['driver_id'].value_counts().sort_values(ascending=False).head(50)

# Prepara colormap Spectral para colorir conforme a contagem
norm = Normalize(vmin=counts.min(), vmax=counts.max())
cmap = cm.get_cmap('Spectral')
colors = cmap(norm(counts.values))

# Cria figura e eixos
fig, ax = plt.subplots(figsize=(15, 10))

# Plota barras coloridas
bars = ax.bar(counts.index.astype(str), counts.values, color=colors)

# Labels e título
ax.set_xlabel('driver_id')
ax.set_ylabel('Contagem de Envios para Atualização')
ax.set_title('Top 50 Motoristas por Envios para Atualização')

# Rotaciona os ticks do eixo x para melhor visualização
plt.xticks(rotation=90)

# Adiciona colorbar
sm = cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])
cbar = fig.colorbar(sm, ax=ax)
cbar.set_label('Número de Envios')
```

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

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import Normalize
from matplotlib import cm

# Copia do DataFrame original
df = df_original.copy()

# Garante que a pontuação seja numérica
df['Pontuação'] = pd.to_numeric(df['Pontuação'], errors='coerce').fillna(0)

# Soma dos pontos por motorista e seleciona os top 50
points_by_driver = (
    df.groupby('driver_id')['Pontuação']
      .sum()
      .sort_values(ascending=False)
      .head(50)
)

# Prepara colormap Spectral
norm = Normalize(vmin=points_by_driver.min(), vmax=points_by_driver.max())
cmap = cm.get_cmap('Spectral')
colors = cmap(norm(points_by_driver.values))

# Cria figura e eixos
fig, ax = plt.subplots(figsize=(15, 10))

# Plota barras coloridas
bars = ax.bar(points_by_driver.index.astype(str), points_by_driver.values,␣
 ↪color=colors)

# Labels e título
ax.set_xlabel('driver_id')
ax.set_ylabel('Total de Pontos')
ax.set_title('Top 50 Motoristas por Total de Pontos')

# Rotaciona ticks
plt.xticks(rotation=90)

# Adiciona colorbar para referência dos valores
sm = cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])
cbar = fig.colorbar(sm, ax=ax)
cbar.set_label('Total de Pontos')
```

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

[ ]:
```
import matplotlib.pyplot as plt
from matplotlib.colors import Normalize
from matplotlib import cm

# Copia do DataFrame original (já contendo as linhas "Motorista enviado para atualização")
df = df_original.copy()

# Filtra somente as ocorrências de atualização
updates = df[df['description'] == 'Motorista enviado para comitê']

# Conta quantas atualizações por driver_id e pega os top 50
counts = updates['driver_id'].value_counts().sort_values(ascending=False).head(50)

# Prepara colormap Spectral para colorir conforme a contagem
norm = Normalize(vmin=counts.min(), vmax=counts.max())
cmap = cm.get_cmap('Spectral')
colors = cmap(norm(counts.values))

# Cria figura e eixos
fig, ax = plt.subplots(figsize=(15, 10))

# Plota barras coloridas
bars = ax.bar(counts.index.astype(str), counts.values, color=colors)

# Labels e título
ax.set_xlabel('driver_id')
ax.set_ylabel('Contagem de Envios para comitê')
ax.set_title('Top 50 Motoristas por Envios para comitê')

# Rotaciona os ticks do eixo x para melhor visualização
plt.xticks(rotation=90)

# Adiciona colorbar
sm = cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])
cbar = fig.colorbar(sm, ax=ax)
cbar.set_label('Número de Envios')

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