Avaliação Sumativa

UFCD 10809_1/l

Este documento consiste no enunciado da ficha de avaliação sumativa do módulo.

A mesma deve ser realizada na aula do dia 17/04/2025.

Todos os ficheiros gerados devem ser compactados num ficheiro PrimeirNome_UltimoNome.zip e o mesmo submetido na secção correspondente da plataforma moodle.

Os exercícios orientados e não orientados a resolver encontram-se no ponto 4 deste enunciado. Devem, contudo, ler atentamente as primeiras secções pois através das mesmas irão adquirir novos conhecimentos e facilmente podem clarificar dúvidas que surjam ao longo da resolução dos exercícios.

Capítulo 1 - Biblioteca Plotly

1. Bibliografia e Webgrafia

Bibliografia

• "Interactive Data Visualization with Python" – Abha Belorkar e Thomas Nield.
• "Python Data Science Handbook" – Jake VanderPlas.

Webgrafia

• Documentação Oficial do Plotly para Python: https://plotly.com/python/
• API Reference do Plotly: https://plotly.com/python-api-reference/
• Documentação do Plotly Express: https://plotly.com/python/plotly-express/
• Documentação do Graph Objects: https://plotly.com/python/graph-objects/
• Cheat Sheet do Plotly Express: https://www.datacamp.com/cheat-sheet/plotly-express-cheat-sheet

2. Datasets que vêm com o Plotly Express

Podes usar estes datasets diretamente com px.data.nome():

gapminder() – Dados demográficos globais.

iris() – Medidas de flores para classificação.

tips() – Conta de restaurante, sexo, gorjetas.

carshare() – Partilha de carros por hora.

wind() – Direção e velocidade do vento.

3. Documentação Biblioteca Plotly Express

Plotly Express é a interface de alto nível do Plotly para criar gráficos de forma rápida e consistente. Abaixo listamos suas principais funções de plotagem, explicando a sintaxe de cada uma (em texto com fonte

monoespaçada) e o papel de seus principais parâmetros. Em seguida, apresentamos detalhes sobre a função query() do pandas.

Funções Básicas (Gráficos Cartesianos 2D)

px.scatter – Cria um gráfico de dispersão (scatter plot) bidimensional.

Sintaxe: px.scatter(data_frame, x, y, color=None, symbol=None, size=None, hover_name=None, ...)

Parâmetros:

- data_frame: DataFrame de entrada
- x, y: colunas para os eixos
- color: coloração por categoria ou valor
- symbol: símbolos distintos por categoria
- size: tamanho dos pontos
- hover_name: nome exibido ao passar o rato
- facet_row, facet_col: subplots por categoria
- animation_frame, animation_group: animações
- marginal_x, marginal_y: histogramas, box ou violin nas margens
- trendline: adiciona linha de tendência (ex. 'ols')

px.line – Gráfico de linhas conectando pontos em sequência.

Sintaxe: px.line(data_frame, x, y, color=None, line_group=None, symbol=None, ...)

Parâmetros similares a px.scatter. Adiciona line_dash, line_shape para estilo e forma da linha.

px.area – Gráfico de área preenchida sob a linha.

Sintaxe: px.area(data_frame, x, y, color=None, line_group=None, ...)

px.bar – Gráfico de barras verticais ou horizontais.

Sintaxe: px.bar(data_frame, x=None, y=None, color=None, barmode='relative', ...)

px.funnel – Gráfico de funil com retângulos horizontais ou verticais.

Sintaxe: px.funnel(data_frame, x=None, y=None, color=None, ...)

px.timeline – Gráfico de Gantt com início/fim por item.

Sintaxe: px.timeline(data_frame, x_start, x_end, y, color=None, ...)

Gráficos Parte-do-Todo (Hierárquicos)

px.pie – Gráfico de pizza.

Sintaxe: px.pie(data_frame, names, values, color=None, hole=None, ...)

px.sunburst, **px.treemap**, **px.icicle** – Hierarquias visuais.

Sintaxe (geral): px.sunburst(data_frame, path=['nivel1','nivel2'], values='valor')

px.funnel_area – Gráfico de funil com áreas trapezoidais.

Gráficos de Distribuição 1D

px.histogram – Histograma.

Sintaxe: px.histogram(data_frame, x=None, color=None, histfunc='count', ...)

px.box – Box plot (mediana, quartis, outliers).

Sintaxe: px.box(data_frame, x=None, y=None, color=None, points='outliers', ...)

px.violin – Distribuição com forma simétrica de densidade.

px.strip – Pontos individuais sobre categorias.

px.ecdf – Função de distribuição acumulada (ECDF).

Distribuições 2D

px.density_heatmap – Mapa de calor com bins 2D.

px.density_contour – Curvas de nível de densidade.

Imagens e Matrizes

px.imshow – Visualização de matrizes ou imagens (ex. correlação).

Gráficos 3D

px.scatter_3d – Dispersão em 3D.

px.line_3d – Linhas em 3D.

Matrizes Multivariadas

px.scatter_matrix – Matriz de dispersão entre variáveis.

px.parallel_coordinates – Linhas sobre eixos verticais.

px.parallel_categories – Fluxo entre categorias.

Mapas com Telas (Map Tiles)

px.scatter_map – Pontos geográficos sobre mapa.

px.line_map – Linhas sobre mapa.

px.choropleth_map – Regiões coloridas sobre mapa.

px.density_map – Mapa de densidade geográfica.

Mapas com Projeções

px.scatter_geo, **px.line_geo**, **px.choropleth** – Sem necessidade de tiles, com projeção geográfica.

Gráficos Polares

px.scatter_polar, **px.line_polar**, **px.bar_polar** – Eixos circulares.

Gráficos Ternários

px.scatter_ternary, **px.line_ternary** – Para composições de 3 componentes.

A função query() do pandas

Permite filtrar dados de forma legível com strings de condição:

Sintaxe: df.query("condição")

Exemplo: df.query("idade > 30 and sexo == 'F'")

Parâmetros e uso:

- Pode usar and, or, not
- Para nomes de colunas com espaços, use crases: coluna com espaço
- Use @variavel para inserir valores externos
- Suporta comparações encadeadas: @limite_inf <= nota <= @limite_sup

Vantagens:

- Mais legível que indexação booleana
- Mais eficiente com numexpr (interno do pandas)

Em resumo, query() melhora a clareza de filtragens complexas e evita redundância de código.

Exemplos:

Exemplos com Dataset Titanic

1. Importar o dataset:

```
import plotly.express as px
import seaborn as sns
df = sns.load_dataset("https://raw.githubusercontent.com/plotly/datasets/master/titanic.csv")
```

2. Gráfico de barras por classe e sobrevivência:

```
px.bar(df, x="class", color="survived", barmode="group")
```

3. Dispersão da idade vs tarifa por sexo:

```
px.scatter(df, x="age", y="fare", color="sex", size="fare", hover_name="embark_town")
```

4. Box plot da idade por classe:

```
px.box(df, x="class", y="age", color="sex")
```

5. Histogramas da idade por sobrevivência:

```
px.histogram(df, x="age", color="survived", barmode="overlay")
```

6. Matrix de dispersão:

```
px.scatter_matrix(df, dimensions=["age", "fare"], color="survived")
```

7. ECDF da tarifa:

```
px.ecdf(df, x="fare", color="survived")
```

8. Violin plot da idade por sexo:

```
px.violin(df, y="age", x="sex", color="survived", box=True, points="all")
```

9. Sunburst de classe e sexo com contagem de sobreviventes:

```
px.sunburst(df, path=["Pclass", "Sex"], values=None, color="Survived")
```

10. Gráfico de densidade de tarifa e idade:

```
px.density_heatmap(df, x="Age", y="Fare", marginal_x="histogram",
marginal_y="histogram")
```

11. Gráfico de coordenadas paralelas:

```
px.parallel_coordinates(df.dropna(), dimensions=["Age", "Fare"], color="Survived")
```

12. Comparação de classes no tempo de embarque:

```
df["Embarked"] = df["Embarked"].fillna("Unknown")
```

```
px.histogram(df, x="Embarked", color="Pclass", barmode="group")
```

4. Exercícios

4.1. Reproduz cada um dos exemplos num ficheiro jupyter notebook.

Não te esqueças de colocar `import plotly.express as px`

px.scatter

Descrição: Gráfico de dispersão

Exemplo de uso:

```
df = px.data.iris()
fig = px.scatter(df, x='sepal_width', y='sepal_length', color='species',
size='petal_length', hover_name='species')
fig.show()
```

px.line

Descrição: Gráfico de linhas

Exemplo de uso:

```
df = px.data.gapminder().query("country == 'Portugal'")

# Filtra os dados para mostrar apenas as linhas onde o país é Portugal com o query

fig = px.line(df, x='year', y='gdpPercap')
fig.show()
```

px.bar

Descrição: Gráfico de barras

Exemplo de uso:

```
df = px.data.tips()
fig = px.bar(df, x='day', y='total_bill', color='sex', barmode='group')
fig.show()
```

px.pie

Descrição: Gráfico de pizza

Exemplo de uso:

```
df = px.data.tips()
fig = px.pie(df, names='day', values='total_bill')
fig.show()
```

px.box

Descrição: Boxplot

Exemplo de uso:

```
df = px.data.tips()
fig = px.box(df, x='day', y='total_bill', color='sex', points='all')
fig.show()
```

px.histogram

Descrição: Histograma

Exemplo de uso:

```
df = px.data.tips()
fig = px.histogram(df, x='total_bill', nbins=20, color='sex')
fig.show()
```

px.density_contour

Descrição: Contorno de densidade

Exemplo de uso:

```
df = px.data.tips()
fig = px.density_contour(df, x='total_bill', y='tip', color='sex')
fig.show()
```

px.density_heatmap

Descrição: Mapa de calor de densidade

Exemplo de uso:

```
df = px.data.tips()
fig = px.density_heatmap(df, x='total_bill', y='tip', marginal_x='histogram',
marginal_y='histogram')
fig.show()
```

px.violin

Descrição: Gráfico de violino

Exemplo de uso:

```
df = px.data.tips()
fig = px.violin(df, y='total_bill', x='day', color='sex', box=True, points='all')
fig.show()
```

px.scatter_matrix

Descrição: Matriz de dispersão

Exemplo de uso:

```
df = px.data.iris()
fig = px.scatter_matrix(df, dimensions=['sepal_length', 'sepal_width', 'petal_length', 'petal_width'], color='species')
fig.show()
```

px.treemap

Descrição: Treemap

Exemplo de uso:

```
df = px.data.gapminder().query("year == 2007")
fig = px.treemap(df, path=['continent', 'country'], values='pop', color='lifeExp')
fig.show()
```

px.sunburst

Descrição: Sunburst

Exemplo de uso:

```
df = px.data.gapminder().query("year == 2007")
fig = px.sunburst(df, path=['continent', 'country'], values='pop', color='lifeExp')
fig.show()
```

px.funnel

Descrição: Gráfico de funil

Exemplo de uso:

```
import pandas as pd
data = pd.DataFrame({'etapas': ['Visitou site', 'Adicionou ao carrinho', 'Comprou'], 'n': [1000, 300, 100]})
fig = px.funnel(data, x='n', y='etapas')
fig.show()
```

px.bar_polar

Descrição: Gráfico de barras polares

Exemplo de uso:

```
df = px.data.tips()
fig = px.bar_polar(df, r='total_bill', theta='day', color='sex')
fig.show()
```

px.imshow

Descrição: Matriz/Imagem

Exemplo de uso:

```
import numpy as np
matrix = np.random.rand(10,10)
fig = px.imshow(matrix)
fig.show()
```

4.2. Exercício Dash + Plotly – para reproduzir – num ficheiro .py

Vamos brincar agora com o Dash e o plotly pra criarmos um dashboard interativo.

No seguinte código:

app.layout = html.Div([...])

Define a estrutura do dashboard como um conjunto de componentes:

- html.H1: Título principal
- dcc.Dropdown: Menu suspenso para escolher o ano
- dcc.Graph: Área onde os gráficos são desenhados

@app.callback([...])

Função **reativa** que atualiza os gráficos sempre que o ano muda.

- Entrada: valor do dropdown (Input('dropdown-ano', 'value'))
- Saída: os dois gráficos (Output(..., 'figure'))
- Dentro da função, o DataFrame é filtrado pelo ano escolhido.

Resultado

Um dashboard interativo onde:

- O utilizador escolhe o ano.
- O gráfico de dispersão mostra a relação entre PIB per capita e esperança de vida.
- O histograma mostra a distribuição da população.

Reproduz o seguinte exemplo num ficheiro de nome **app.py**:

```
import dash
from dash import html, dcc, Input, Output
import plotly.express as px
import pandas as pd

# Carregar dataset
df = px.data.gapminder()

# Inicializar a aplicação Dash
app = dash.Dash(__name__)

# Layout da aplicação
app.layout = html.Div([
  html.H1("🌍 Dashboard Interativo - Gapminder", style={"textAlign": "center"}),

  # Dropdown para escolher o ano
  html.Label("Escolhe o ano:"),
  dcc.Dropdown(
    id='dropdown-ano',
    options=[{'label': ano, 'value': ano} for ano in sorted(df['year'].unique())],
    value=2007,
    clearable=False
  ),

  # Gráfico de dispersão
  dcc.Graph(id='grafico-dispersao'),

  # Histograma de população
  dcc.Graph(id='histograma-populacao')
])

# Callback para atualizar gráficos com base no dropdown
@app.callback(
  Output('grafico-dispersao', 'figure'),
  Output('histograma-populacao', 'figure'),
  Input('dropdown-ano', 'value')
```

```
)
def atualizar_graficos(ano_selecionado):
    dados_filtrados = df[df['year'] == ano_selecionado]

    fig1 = px.scatter(
        dados_filtrados,
        x="gdpPercap", y="lifeExp",
        size="pop", color="continent",
        hover_name="country", log_x=True,
        title=f"Esperança de vida vs PIB per capita ({ano_selecionado})"
    )

    fig2 = px.histogram(
        dados_filtrados,
        x="pop", nbins=20, color="continent",
        title=f"Distribuição Populacional por País ({ano_selecionado})"
    )

    return fig1, fig2

# Executar a aplicação
if __name__ == '__main__':
    app.run(debug=True)
```

Acrescenta um gráfico à tua escolha na função atualizar_gráficos.

4.3. Criar uma aplicação para correr localmente utilizando a script anterior (para poderes por exemplo partilhar dentro da empresa e qualquer um poder executar)

Passo 0: Adiciona ao ficheiro app.py

```
import webbrowser #inicio

webbrowser.open("http://127.0.0.1:8050") #antes do main
```

Passo 1: Instala a biblioteca pyinstaller – pip install pyinstaller

Passo 2: Cria uma pasta com o nome dashProject e coloca o ficheiro app.py dentro dessa pasta.

Passo 3: Na Linha de Comandos coloca-te na pasta que criaste no passo anterior e corre o seguinte comando:

- se usas ambientes virtuais:

pyinstaller --onefile --add-data "venv\Lib\site-packages\plotly;plotly" app.py

- caso não utilizes ambientes virtuais:

Em Windows: **pyinstaller --onefile app.py**

Em Linux Ou Mac: **pyinstaller --onefile app.py**

Passo 4: Entra na pasta criada e de seguida na pasta dist. **Dá duplo clique no ficheiro .exe**

Passo 5: Agora é só distribuíres o .exe 😊

Tarefa Extra:

Podemos distribuir a aplicação como um instalador executável (.msi ou .exe). Para tal podemos usar uma das ferramentas:

1. Inno Setup (Windows)
2. NSIS

Passo 1: Instala a ferramenta Inno Setup: **https://jrsoftware.org/isinfo.php**

Passo 2: Dentro da pasta do projeto cria um ficheiro com o nome e extensão **instalador_dashboard.iss.** Deverás obter a seguinte estrutura dentro da pasta do teu projeto:

```
├── dist/
│   └── app.exe          ← gerado com PyInstaller
├── instalador_dashboard.iss
```

Passo 3: Coloca a seguinte informação no ficheiro .iss

```
; Script de Instalação para o Dashboard Gapminder
[Setup]
AppName=Dashboard Gapminder
AppVersion=1.0
DefaultDirName={pf}\DashboardGapminder
DefaultGroupName=Dashboard Gapminder
UninstallDisplayIcon={app}\app.exe
```

```
OutputDir=dist_instalador
OutputBaseFilename=DashboardGapminderSetup
Compression=lzma
SolidCompression=yes
ArchitecturesInstallIn64BitMode=x64

[Files]
Source: "dist\app.exe"; DestDir: "{app}"; Flags: ignoreversion

[Icons]
Name: "{group}\Dashboard Gapminder"; Filename: "{app}\app.exe"
Name: "{commondesktop}\Dashboard Gapminder"; Filename: "{app}\app.exe"; Tasks:
desktopicon

[Tasks]
Name: "desktopicon"; Description: "Criar atalho no ambiente de trabalho";
GroupDescription: "Opções adicionais"

[Run]
Filename: "{app}\app.exe"; Description: "Executar o Dashboard Gapminder"; Flags:
nowait postinstall skipifsilent
```

Passo 4: Abre o ficheiro .iss com o **Inno Setup Compiler**

Passo 5: Clica em Compile. O instalador final será gerado na pasta dist_instalador

Clica no instalador 😊

O QUE O INSTALADOR FAZ:

- Instala o executável app.exe em C:\Program Files\DashboardGapminder
- Cria atalho no Menu Iniciar
- Cria ícone no ambiente de trabalho (se o utilizador quiser)
- Abre o dashboard no browser automaticamente após instalação

Se quiseres incluir um **ícone personalizado**, **ficheiro README** ou **atalhos adicionais. Para tal investiga como o fazer** 😊.