Lab 11: Análise de Dados de Renda

AUTHOR
Felipe Pedroso Popic

Setup do Ambiente

O primeiro passo em qualquer relatório reproduzível é carregar as bibliotecas necessárias.

```
# Importa todas as bibliotecas Python que vamos usar no relatório
import polars as pl
import matplotlib.pyplot as plt
import numpy as np

print("Bibliotecas Python carregadas.")
```

```
Bibliotecas Python carregadas.
```

1. Carga e Preparação dos Dados

Este é o passo mais importante. Criamos o nosso DataFrame principal `df_renda_adulta` aqui. Todos os blocos de código seguintes dependerão da execução bem-sucedida deste.

```
# Passo 1 e 2: Definições (sem alteração)
colunas = [
    'age', 'workclass', 'fnlwgt', 'education', 'education-num',
    'marital-status', 'occupation', 'relationship', 'race', 'sex',
    'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
    'income'
]
tipos_colunas = {
    'age': pl.Int64, 'workclass': pl.Categorical, 'fnlwgt': pl.Int64,
    'education': pl.Categorical, 'education-num': pl.Int64,
    'marital-status': pl.Categorical, 'occupation': pl.Categorical,
    'relationship': pl.Categorical, 'race': pl.Categorical, 'sex': pl.Categorical,
    'capital-gain': pl.Int64, 'capital-loss': pl.Int64,
    'hours-per-week': pl.Int64, 'native-country': pl.Categorical,
    'income': pl.Categorical
}

# Passo 3: Importar o arquivo CSV (ESTRUTURA CORRIGIDA)
try:
    # A linha abaixo inicia o bloco 'try'. TUDO dentro dele deve ser recuado.
    print("Tentando carregar o arquivo usando o caminho seguro...")

    # Usamos a variável 'caminho_do_arquivo' que criamos no chunk setup do R.
    # O objeto `r` é a ponte do Python para acessar objetos do R.
    df_renda_adulta = pl.read_csv(
        r.caminho_do_arquivo, # Caminho seguro vindo do R
        has_header=False,
```

```
        new_columns=colunas,
        dtypes=tipos_colunas,
        null_values="?"
    )
    print("Dados importados com sucesso usando o caminho completo!")
    df_renda_adulta.head()

except Exception as e:
    # Este bloco 'except' corresponde ao 'try' acima.
    print(f"ERRO CRÍTICO DURANTE A CARGA DOS DADOS: {e}")
    raise
```

shape: (5, 15)

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain
i64	cat	i64	cat	i64	cat	cat	cat	cat	cat	i64
39	"State-gov"	77516	"Bachelors"	13	"Never-married"	"Adm-clerical"	"Not-in-family"	"White"	"Male"	2174
50	"Self-emp-not-inc"	83311	"Bachelors"	13	"Married-civ-spouse"	"Exec-managerial"	"Husband"	"White"	"Male"	0
38	"Private"	215646	"HS-grad"	9	"Divorced"	"Handlers-cleaners"	"Not-in-family"	"White"	"Male"	0
53	"Private"	234721	"11th"	7	"Married-civ-spouse"	"Handlers-cleaners"	"Husband"	"Black"	"Male"	0
28	"Private"	338409	"Bachelors"	13	"Married-civ-spouse"	"Prof-specialty"	"Wife"	"Black"	"Female"	0

2. Apresente os tipos de cada uma das colunas

```
df_renda_adulta.schema
```

```
Schema({'age': Int64, 'workclass': Categorical, 'fnlwgt': Int64, 'education': Categorical,
'education-num': Int64, 'marital-status': Categorical, 'occupation': Categorical,
'relationship': Categorical, 'race': Categorical, 'sex': Categorical, 'capital-gain': Int64,
'capital-loss': Int64, 'hours-per-week': Int64, 'native-country': Categorical, 'income':
Categorical})
```

3. Apresente as dimensões da tabela de dados

```
print(f"A tabela tem {df_renda_adulta.height} linhas e {df_renda_adulta.width} colunas.")
```

```
A tabela tem 32561 linhas e 15 colunas.
```

4. Contagem de pessoas por faixa salarial

```
df_renda_adulta.group_by("income").count()
```

shape: (2, 2)

income	count
cat	u32
"<=50K"	24720
">50K"	7841

5. Transformação para formato Longo (capital-gain/loss)

```
colunas_id = [col for col in df_renda_adulta.columns if col not in ['capital-gain', 'capital-l

renda_longo = df_renda_adulta.melt(
    id_vars=colunas_id,
    value_vars=['capital-gain', 'capital-loss'],
    variable_name='tipo',
    value_name='Valor'
)
```

```
<string>:2: DeprecationWarning: `DataFrame.melt` is deprecated; use `DataFrame.unpivot`
instead, with `index` instead of `id_vars` and `on` instead of `value_vars`
```

```
print("Dimensões da nova tabela:", renda_longo.shape)
```

```
Dimensões da nova tabela: (65122, 15)
```

```
renda_longo.head()
```

shape: (5, 15)

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	hours-per-week	c
i64	cat	i64	cat	i64	cat	cat	cat	cat	cat	i64	
39	"State-gov"	77516	"Bachelors"	13	"Never-married"	"Adm-clerical"	"Not-in-family"	"White"	"Male"	40	"I
50	"Self-emp-not-inc"	83311	"Bachelors"	13	"Married-civ-spouse"	"Exec-managerial"	"Husband"	"White"	"Male"	13	"I

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	hours-per-week	c
i64	cat	i64	cat	i64	cat	cat	cat	cat	cat	i64	
38	"Private"	215646	"HS-grad"	9	"Divorced"	"Handlers-cleaners"	"Not-in-family"	"White"	"Male"	40	"I
53	"Private"	234721	"11th"	7	"Married-civ-spouse"	"Handlers-cleaners"	"Husband"	"Black"	"Male"	40	"I
28	"Private"	338409	"Bachelors"	13	"Married-civ-spouse"	"Prof-specialty"	"Wife"	"Black"	"Female"	40	

6. Média de horas trabalhadas por classe salarial

```
df_renda_adulta.group_by("income").agg(
    pl.col("hours-per-week").mean().alias("media_horas_semanais")
)
```

shape: (2, 2)

income	media_horas_semanais
cat	f64
">50K"	45.473026
"<=50K"	38.84021

7. Contagem de pessoas por profissão

```
df_renda_adulta.group_by("occupation").count().sort("count", descending=True)
```

shape: (15, 2)

occupation	count
cat	u32
"Prof-specialty"	4140
"Craft-repair"	4099
"Exec-managerial"	4066
"Adm-clerical"	3770
"Sales"	3650
…	…
"Farming-fishing"	994
"Tech-support"	928

occupation	count
cat	u32
"Protective-serv"	649
"Priv-house-serv"	149
"Armed-Forces"	9

8. Gráfico: Média de horas por nível salarial

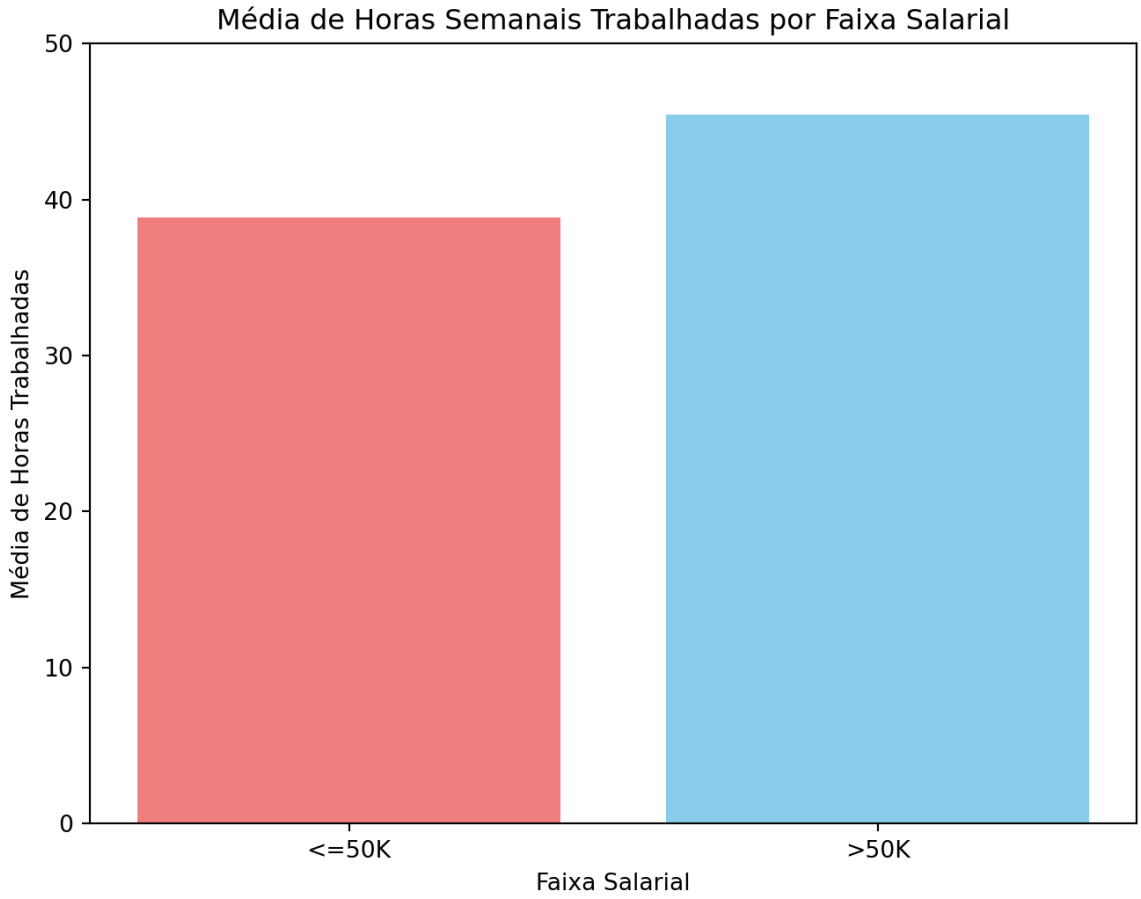
```
# 1. Calculamos a média
media_horas_por_renda = df_renda_adulta.group_by("income").agg(
    pl.col("hours-per-week").mean().alias("media_horas_semanais")
)

# 2. Extraímos as colunas do Polars como listas
categorias = media_horas_por_renda.get_column("income").to_list()
valores = media_horas_por_renda.get_column("media_horas_semanais").to_list()

# 3. Criamos o gráfico com matplotlib
fig, ax = plt.subplots(figsize=(8, 6))
ax.bar(categorias, valores, color=['lightcoral', 'skyblue'])
ax.set_title("Média de Horas Semanais Trabalhadas por Faixa Salarial")
ax.set_xlabel("Faixa Salarial")
ax.set_ylabel("Média de Horas Trabalhadas")
ax.set_ylim(0, 50)
```

```
(0.0, 50.0)
```

```
plt.show()
```



9. Desafio: Evidência de discriminação salarial por gênero

```
analise_fatores = df_renda_adulta.group_by("sex").agg(
    pl.col("hours-per-week").mean().alias("media_horas_semanais"),
    pl.col("education-num").mean().alias("media_anos_educacao")
)

analise_fatores
```

shape: (2, 3)

sex	media_horas_semanais	media_anos_educacao
cat	f64	f64
"Female"	36.410361	10.035744
"Male"	42.428086	10.102891