Desafio11

Luciano Floriano

2025-10-07

```
# Instalar e configurar reticulate
library(reticulate)
reticulate::py_install("pandas")
```

```
## Using virtual environment "C:/Users/ra277195/AppData/Local/R/cache/R/reticulate/uv/cache/archive-v0/
```

```
reticulate::py_install("matplotlib")
```

```
## Using virtual environment "C:/Users/ra277195/AppData/Local/R/cache/R/reticulate/uv/cache/archive-v0/
```

```
reticulate::py_install("scipy")
```

```
## Using virtual environment "C:/Users/ra277195/AppData/Local/R/cache/R/reticulate/uv/cache/archive-v0/
```

```
py_install("seaborn")
```

```
## Using virtual environment "C:/Users/ra277195/AppData/Local/R/cache/R/reticulate/uv/cache/archive-v0/
```

```
py_install("matplotlib")
```

```
## Using virtual environment "C:/Users/ra277195/AppData/Local/R/cache/R/reticulate/uv/cache/archive-v0/
```

1. Utilizando o arquivo renda_adulta.csve sabendo que ele não possui cabeçalho, faça a importação do banco de dados utilizando os nomes das colunas conforme apresentado acima e na sequência ali indicada. No momento da importação do arquivo, você deve, também, indicar os tipos de cada uma das colunas. Utilize o fato de que o símbolo ?representa valores faltantes.

```
import pandas as pd

# Definindo os nomes das colunas conforme a descrição
colunas = [
    "age", "workclass", "fnlwgt", "education", "education-num",
    "marital-status", "occupation", "relationship", "race", "sex",
    "capital-gain", "capital-loss", "hours-per-week", "native-country", "income"
]

# Definindo os tipos das colunas
tipos = {
    "age": "int64",
    "workclass": "category",
    "fnlwgt": "int64",
    "education": "category",
    "education-num": "int64",
    "marital-status": "category",
    "occupation": "category",
    "relationship": "category",
    "race": "category",
```

```
    "sex": "category",
    "capital-gain": "int64",
    "capital-loss": "int64",
    "hours-per-week": "int64",
    "native-country": "category",
    "income": "category"
}

# Importando o arquivo compactado (.csv.gz), sem cabeçalho, e tratando "?" como valor faltante
df = pd.read_csv(
    "renda_adulta.csv.gz",
    header=None,            # sem cabeçalho
    names=colunas,          # nomes definidos acima
    dtype=tipos,            # tipos de dados
    na_values="?"           # tratar "?" como valor ausente
)

# Verificando as primeiras linhas e tipos
print(df.head())
```

```
##    age         workclass  fnlwgt  ... hours-per-week  native-country income
## 0   39         State-gov   77516  ...             40   United-States  <=50K
## 1   50  Self-emp-not-inc   83311  ...             13   United-States  <=50K
## 2   38           Private  215646  ...             40   United-States  <=50K
## 3   53           Private  234721  ...             40   United-States  <=50K
## 4   28           Private  338409  ...             40            Cuba  <=50K
##
## [5 rows x 15 columns]
```

```
print(df.info())
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 32561 entries, 0 to 32560
## Data columns (total 15 columns):
##  #   Column          Non-Null Count  Dtype
## ---  ------          --------------  -----
##  0   age             32561 non-null  int64
##  1   workclass       30725 non-null  category
##  2   fnlwgt          32561 non-null  int64
##  3   education       32561 non-null  category
##  4   education-num   32561 non-null  int64
##  5   marital-status  32561 non-null  category
##  6   occupation      30718 non-null  category
##  7   relationship    32561 non-null  category
##  8   race            32561 non-null  category
##  9   sex             32561 non-null  category
##  10  capital-gain    32561 non-null  int64
##  11  capital-loss    32561 non-null  int64
##  12  hours-per-week  32561 non-null  int64
##  13  native-country  31978 non-null  category
##  14  income          32561 non-null  category
## dtypes: category(9), int64(6)
## memory usage: 1.8 MB
## None
```

2. Apresente os tipos de cada uma das colunas.

```
print(df.dtypes)
```

```
## age                int64
## workclass         category
## fnlwgt               int64
## education         category
## education-num        int64
## marital-status    category
## occupation        category
## relationship      category
## race              category
## sex               category
## capital-gain         int64
## capital-loss         int64
## hours-per-week       int64
## native-country    category
## income            category
## dtype: object
```

3. Apresente as dimensões da tabela de dados.

```
print(df.shape)
```

```
## (32561, 15)
```

4, Quantas pessoas recebem acima de $50.000 e quantas pessoas recebem abaixo desses limites?

```
print(df['income'].value_counts())
```

```
## income
## <=50K    24720
## >50K      7841
## Name: count, dtype: int64
```

5. Crie um objeto chamado renda_longo, no qual você transforma as colunas capital-gaine capital-loss(formato largo) para formato longo. Os valores dessas variáveis devem ser armazenados em uma nova coluna de chamada Valore os tipos de valores (ganho e perda) devem ser armazenados em uma coluna de chamada tipo.

```
import pandas as pd

# Criar o dataframe em formato longo
renda_longo = pd.melt(
    df,
    id_vars=[col for col in df.columns if col not in ['capital-gain', 'capital-loss']],
    value_vars=['capital-gain', 'capital-loss'],
    var_name='tipo',
    value_name='Valor'
)

# Substituir nomes técnicos por "ganho" e "perda"
renda_longo['tipo'] = renda_longo['tipo'].replace({
    'capital-gain': 'ganho',
    'capital-loss': 'perda'
})
```

```
# Exibir as primeiras linhas
print(renda_longo.head())
```

```
##    age         workclass  fnlwgt  education  ...  native-country income   tipo Valor
## 0   39         State-gov   77516  Bachelors  ...   United-States  <=50K  ganho  2174
## 1   50  Self-emp-not-inc   83311  Bachelors  ...   United-States  <=50K  ganho     0
## 2   38           Private  215646    HS-grad  ...   United-States  <=50K  ganho     0
## 3   53           Private  234721       11th  ...   United-States  <=50K  ganho     0
## 4   28           Private  338409  Bachelors  ...            Cuba  <=50K  ganho     0
##
## [5 rows x 15 columns]
```

6. Quais são as médias de horas trabalhadas por classe salarial?

```
media_horas = (
    df.groupby('income', observed=True)['hours-per-week']
      .mean()
      .reset_index()
      .rename(columns={'income': 'Classe Salarial', 'hours-per-week': 'Média de Horas/Semana'})
)

print(media_horas)
```

```
##   Classe Salarial  Média de Horas/Semana
## 0           <=50K              38.840210
## 1            >50K              45.473026
```

7. Se cada linha representa uma pessoa, quantas pessoas foram amostradas em cada profissão?

```
# Contar quantas pessoas há em cada profissão
contagem_profissoes = df['occupation'].value_counts(dropna=False)

print(contagem_profissoes)
```

```
## occupation
## Prof-specialty       4140
## Craft-repair         4099
## Exec-managerial      4066
## Adm-clerical         3770
## Sales                3650
## Other-service        3295
## Machine-op-inspct    2002
## NaN                  1843
## Transport-moving     1597
## Handlers-cleaners    1370
## Farming-fishing       994
## Tech-support          928
## Protective-serv       649
## Priv-house-serv       149
## Armed-Forces            9
## Name: count, dtype: int64
```

8. Crie um gráfico de barras que apresenta o número médio de horas trabalhadas semanalmente em função do nível salarial.

```
# Análise estatística para evidência de discriminação salarial entre gêneros
```

```python
# 1. Tabela de contingência com porcentagens por linha
tabela_porcentagem = pd.crosstab(df['sex'], df['income'], normalize='index') * 100
print("Distribuição percentual da renda por gênero:")
```

```
## Distribuição percentual da renda por gênero:
```

```python
print(tabela_porcentagem)
```

```
## income      <=50K       >50K
## sex
## Female  89.053941  10.946059
## Male    69.426342  30.573658
```

```python
print("\n")
```

```python
# 2. Tabela de contingência com contagens absolutas
tabela_absoluta = pd.crosstab(df['sex'], df['income'])
print("Distribuição absoluta da renda por gênero:")
```

```
## Distribuição absoluta da renda por gênero:
```

```python
print(tabela_absoluta)
```

```
## income  <=50K  >50K
## sex
## Female   9592  1179
## Male    15128  6662
```

```python
print("\n")
```

```python
# 3. Teste Qui-Quadrado para significância estatística
from scipy.stats import chi2_contingency

chi2, p_valor, dof, expected = chi2_contingency(tabela_absoluta)
print(f"Teste Qui-Quadrado de Independência:")
```

```
## Teste Qui-Quadrado de Independência:
```

```python
print(f"Estatística Qui-Quadrado: {chi2:.4f}")
```

```
## Estatística Qui-Quadrado: 1517.8134
```

```python
print(f"Valor-p: {p_valor:.4f}")
```

```
## Valor-p: 0.0000
```

```python
print(f"Graus de liberdade: {dof}")
```

```
## Graus de liberdade: 1
```

```python
# 4. Interpretação do valor-p
if p_valor < 0.05:
    print("\n Há evidências estatísticas de associação entre gênero e renda (p < 0.05)")
    print("Isso sugere possível discriminação salarial por gênero.")
else:
    print("\n Não há evidências estatísticas suficientes de associação (p >= 0.05)")
```

```
##
##  Há evidências estatísticas de associação entre gênero e renda (p < 0.05)
## Isso sugere possível discriminação salarial por gênero.
```

```python
# 5. Diferença percentual entre gêneros
diferenca_percentual = (tabela_porcentagem.loc['Male', '>50K'] -
                        tabela_porcentagem.loc['Female', '>50K'])
print(f"\nDiferença percentual na faixa >50K: {diferenca_percentual:.2f}%")
```

```
##
## Diferença percentual na faixa >50K: 19.63%
```

```python
print(f"Homens com >50K: {tabela_porcentagem.loc['Male', '>50K']:.2f}%")
```

```
## Homens com >50K: 30.57%
```

```python
print(f"Mulheres com >50K: {tabela_porcentagem.loc['Female', '>50K']:.2f}%")
```

```
## Mulheres com >50K: 10.95%
```