Análise de dados

Associação entre variáveis categóricas

Guilherme Prado

Carregamento do dataset

Para esta explicação, utilizaremos a base de dados do RMS Titanic, disponível na biblioteca Seaborn.

Segue a importação das bibliotecas e o carregamento da base de dados:

```
import seaborn as sns
import pandas as pd
import numpy as np
import statsmodels.api as sm
```

✓ 0.0s

```
# Carregue o conjunto de dados do Titanic
df = sns.load_dataset('titanic')
```

✓ 0.0s

Tabela de contingência

Quando queremos analisar o comportamento de duas variáveis *X1* e *X2*, podemos recorrer a uma tabela de contingência. Na tabela de contingência, cada elemento representa a frequência observada das realizações simultâneas de *X1* e *X2*.

Em Python, podemos usar a função *crosstab* da biblioteca pandas para criar uma tabela de contingência. Neste exemplo, criaremos uma tabela de contingência para análise das variáveis “survived” e “sex”.

```
# Crie a tabela de contingência
cont_table = pd.crosstab(
    df['survived'],
    df['sex'],
    margins=True,
    margins_name="total"
)
```

A tabela de contingência criada é mostrada na proxima página.

Tabela de contingência

```
# Mostre a tabela de contingência
display(cont_table)
```

✓ 0.0s

sex	female	male	total
survived			
0	81	468	549
1	233	109	342
total	314	577	891

Esta tabela nos motstra algumas informações interessantes:

- Dos 891 passageiros, 342 sobreviveram.
- Dos 549 mortos, 81 eram do sexo feminino;
- Dos 342 sobreviventes, 233 eram do sexo feminino.

Estes são apenas alguns exemplos de informações. A leitura

Tabela de contingência

foi feita no sentido das linhas, mas também poderia ser feita no sentido das colunas.

Agora removeremos a coluna e linha de total para as próximas operações que serão realizadas com a tabela. Também poderiamos não ter passado os parâmetros *margins* (default=False) e *margins_name* na função *crosstab*.

```
# Remova coluna e linha de total
cont_table.drop("total", axis=1, inplace=True)
cont_table.drop("total", axis=0, inplace=True)
```

✓ 0.0s

Pode ser muito mais interessante visualizar a tabela de contingência em porcentagem em relação ao total geral ou em relação ao total das linhas/colunas.

Tabela de contingência

Abaixo mostro uma forma de apresentar a tabela de contingência em porcentagem em relação ao total geral.

```
# Mostre a tabela de contingência normalizada pelo total geral
display(cont_table / cont_table.sum().sum() * 100)
```

✓ 0.0s

sex	female	male
survived		
0	9.090909	52.525253
1	26.150393	12.233446

Também podemos calcular uma tabela com os valores esperados caso não houvesse dependência entre as variáveis analisadas. Cada elemento da tabela de valores esperados é calculado pela equação:

$$E_{ij} = \frac{(\text{Total da linha } i) \times (\text{total da coluna } j)}{\text{Total geral}}$$

Valores esperados

A tabela de valores esperados pode ser criada pelo produto externo entre os vetores contendo as somas das colunas por linha e as somas das linhas por coluna, respectivamente, divido pelo total geral. Em Python, podemos usar a função *outer* da biblioteca numpy conforme segue:

```
# Calcule a tabela de valores esperados
esperados_table = pd.DataFrame(
    np.outer(
        cont_table.sum(axis=1),
        cont_table.sum(axis=0)
    ) / cont_table.sum().sum(),
    columns=cont_table.columns,
    index=cont_table.index
)
```

Valores esperados

```python
# Mostre a tabela de valores esperados
display(esperados_table)
```

✓ 0.0s

sex	female	male
survived		
0	193.474747	355.525253
1	120.525253	221.474747

```python
# Mostre a tabela de valores esperados normalizada pelo total geral
display(esperados_table / esperados_table.sum().sum() * 100)
```

✓ 0.0s

sex	female	male
survived		
0	21.714338	39.901824
1	13.526964	24.856874

Resíduos

Com a tabela de contingência (valores observados) e a tabela de valores esperados, podemos calcular uma tabela de resíduos, onde cada elemento da tabela é calculado pela equação:

$$r_{ij} = \frac{(\text{observado}_{ij} - \text{esperado}_{ij})^2}{esperado_{ij}}$$

Em Python, a tabela de resíduos pode ser calculada da seguinte forma:

```
# Calcule a tabela de resíduos
residuos_table = (cont_table - esperados_table)**2 / esperados_table
```

✓ 0.0s

A seguir é mostrada a tabela de resíduos, que será utilizada para calcular a medida de associação entre as variáveis analisadas.

Qui-quadrado

```
# Mostra a tabela de resíduos
display(residuos_table)
```

✓ 0.0s

sex	female	male
survived		
0	65.386150	35.582757
1	104.961977	57.119690

O qui-quadrado (χ^2) mede o afastamento global entre as variáveis categóricas analisadas, sendo calculado pela soma dos resíduos.

A seguir veremos o cálculo do qui-quadrado pela soma dos resíduos da nossa tabela e pela biblioteca statsmodels, que oferece classes e funções de vários modelos estatísticos.

qui-quadrado

```python
# Calcule o chi-quadrado
chi2 = residuos_table.sum().sum()
print("Qui-quadrado calculado:", chi2)

# Calcule o chi-quadrado pelo statsmodels
table = sm.stats.Table(cont_table)
chi2_sm = table.test_nominal_association().statistic
print("Qui-quadrado statsmodels:", chi2_sm)
```

✓ 0.0s

```
Qui-quadrado calculado: 263.05057407065567
Qui-quadrado statsmodels: 263.0505740706556
```

Como podemos ver, nosso qui-quadrado, calculado passo a passo, é basicamente o mesmo calculado através da biblioteca statsmodels. O mesmo é válido para variáveis com mais de duas categorias. Sugiro testar o passo a passo com as variáveis “survived” e “pclass” (categorias 1, 2 e 3), por exemplo.

Espero que tenham gostado!