Universidade Estadual de Santa Cruz - UESC
Departamento de Ciências Exatas - DCEX
CET699 - Estatística Aplicada às Ciências Sociais ICurso de Ciências Sociais
Prof. José Cláudio Faria
Prova final
Pontuação total: 10
Prazo: 14/07/2024 - 16/07/2024
NÚMERO DO GRUPO: [redacted]
Nome: [redacted]
Matrícula: [redacted]

1. *Diagrama de caixa para Y1 e Y2 antes da eliminação de outliers*

python
```
import matplotlib.pyplot as plt
import seaborn as sns

# Boxplot para Y1 e Y2 antes da eliminação de outliers
fig, axes = plt.subplots(1, 2, figsize=(12, 6))
sns.boxplot(data=dados_df, y='Y1', ax=axes[0]).set_title('Boxplot de Y1 (antes)')
sns.boxplot(data=dados_df, y='Y2', ax=axes[1]).set_title('Boxplot de Y2 (antes)')
plt.show()
```

2. *Diagrama de caixa para Y1 e Y2 após a eliminação de outliers*

python
```
# Função para remover outliers
def remove_outliers(df, column):
 Q1 = df[column].quantile(0.25)
 Q3 = df[column].quantile(0.75)
  IQR = Q3 - Q1
  lower_bound = Q1 - 1.5 * IQR
  upper_bound = Q3 + 1.5 * IQR   return df[(df[column] >= lower_bound) &
(df[column] <= upper_bound)]

# Remover outliers
dados_no_outliers_y1 = remove_outliers(dados_df, 'Y1')
dados_no_outliers_y2 = remove_outliers(dados_df, 'Y2')

# Boxplot para Y1 e Y2 após a eliminação de outliers
fig, axes = plt.subplots(1, 2, figsize=(12, 6))
sns.boxplot(data=dados_no_outliers_y1, y='Y1', ax=axes[0]).set_title('Boxplot de Y1
(após)')
```

```
sns.boxplot(data=dados_no_outliers_y2, y='Y2', ax=axes[1]).set_title('Boxplot de Y2 (após)')
plt.show()
```

1. *Tabela de frequências: absoluta (Fi), relativa (Fr, %) e acumulada (Fac, %)*

python
```
# Calcular frequências
freq_absoluta = dados_df['Y1'].value_counts().sort_index()
freq_relativa = dados_df['Y1'].value_counts(normalize=True).sort_index() * 100
freq_acumulada = freq_relativa.cumsum()

# Criar a tabela
tabela_freq = pd.DataFrame({
    'Frequência Absoluta (Fi)': freq_absoluta,
    'Frequência Relativa (Fr, %)': freq_relativa,
    'Frequência Acumulada (Fac, %)': freq_acumulada
})

tabela_freq
```

2. *Histograma e polígono de frequência acumulada dos dados*

python
```
# Histograma
plt.figure(figsize=(10, 6))
sns.histplot(dados_df['Y1'], bins=10, kde=False)
plt.title('Histograma de Y1')
plt.xlabel('Y1')
plt.ylabel('Frequência')
plt.show()

# Polígono de frequência acumulada
plt.figure(figsize=(10, 6))
sns.ecdfplot(dados_df['Y1'])
plt.title('Polígono de Frequência Acumulada de Y1')
```

```
plt.xlabel('Y1')
plt.ylabel('Frequência Acumulada')
plt.show()
```

Questão 2: Medidas Estatísticas Básicas (5.0)

2.1 Medidas determinadas a partir dos dados (3.0)

Para as variáveis Y1 e Y2:

1. *Tendência central: média, mediana e moda (1.0)*

python
```
# Tendência central para Y1 e Y2
tendencia_central = dados_df[['Y1', 'Y2']].agg(['mean', 'median', pd.Series.mode])
tendencia_central
```

2. *Posição: quartis e decis (1.0)*

python
```
# Posição para Y1 e Y2
posicao = dados_df[['Y1', 'Y2']].describe(percentiles=[.25, .5, .75, .1, .2, .3, .4, .6, .7, .8, .9])
posicao
```

3. *Dispersão: amplitude total, variância, desvio padrão e coeficiente de variação (1.0)*

python
```
# Dispersão para Y1 e Y2
dispersao = dados_df[['Y1', 'Y2']].agg(['max', 'min', 'var', 'std'])
dispersao.loc['range'] = dispersao.loc['max'] - dispersao.loc['min']
dispersao.loc['coef_var'] = dispersao.loc['std'] / dados_df[['Y1', 'Y2']].mean()
dispersao
```

2.2 Medidas determinadas a partir de apresentações tabulares (2.0)

1. *Tendência central: média, mediana e moda (0.8)*

python
```
# Usando a tabela de frequências para calcular medidas de tendência central
frequencia = tabelas_df['Freq']
valores = tabelas_df.index + 1  # assumindo que os valores de Y1 são as classes 1, 2, 3, ...

# Média
media = sum(valores * frequencia) / sum(frequencia)

# Mediana (aproximada, considerando classes de frequências)
mediana_classe = valores[frequencia.cumsum() >= frequencia.sum() / 2].iloc[0]

# Moda
moda = valores[frequencia.idxmax()]

media, mediana_classe, moda
```

2. *Posição: quartis (0.4)*

python
```
# Quartis (aproximados, considerando classes de frequências)
q1_classe = valores[frequencia.cumsum() >= frequencia.sum() / 4].iloc[0]
q3_classe = valores[frequencia.cumsum() >= 3 * frequencia.sum() / 4].iloc[0]

q1_classe, q3_classe
```

3. *Dispersão: amplitude total, variância, desvio padrão e coeficiente de variação (0.8)*

python
```
# Dispersão (aproximada, considerando classes de frequências)
amplitude_total = valores.max() - valores.min()
variancia = sum(frequencia * (valores - media)**2) / (sum(frequencia) - 1)
desvio_padrao = variancia**0.5
coef_var = desvio_padrao / media

amplitude_total, variancia, desvio_padrao, coef_var
```

Questão 3: Medidas Estatísticas de Associação (2.0)

1. *Estimativas: covariância e correlação linear simples (0.8)*

python
```
# Covariância e correlação
covariancia = dados_df[['Y1', 'Y2']].cov().iloc[0, 1]
correlacao = dados_df[['Y1', 'Y2']].corr().iloc[0, 1]
covariancia, correlacao
```

2. *Diagramas de dispersão dos dados (0.8)*

python
```
# Diagrama de dispersão
plt.figure(figsize=(10, 6))
sns.scatterplot(data=dados_df, x='Y1', y='Y2')
plt.title('Diagrama de Dispersão de Y1 e Y2')
plt.xlabel('Y1')
plt.ylabel('Y2')
plt.show()
```