**Métodos Quantitativos I**

**Camila Guedes de Farias**

1ª Lista de Exercícios, 31/3/2025

*Descrição:*

*Exercícios dos capítulos 2 e 3 do livro Estatística Aplicada a Administração e Economia.*

*Cap. 2: 10, 18, 37, 52.*

*Cap. 3: 12, 29, 34, 36, 44, 48, 56, 59.*

**Cap 2:10**

a)

Distribuição de Frequência:

>>> print(frequencia)

Rating

Terrible 41

Poor 62

Average 107

Very Good 252

Excellent 187

b)

Distribuição de Frequência Percentual:

>>> print(frequencia\_percentual)

Rating

Terrible 6.317411

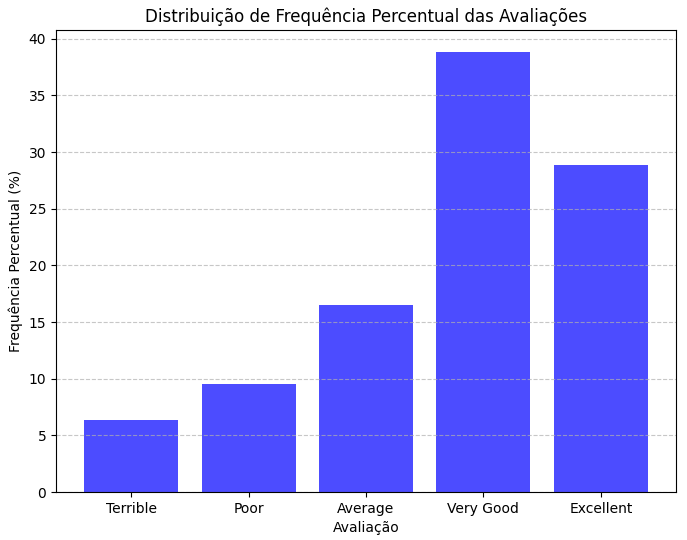
Poor 9.553159

Average 16.486903

Very Good 38.828968

Excellent 28.813559

c)



d)

>>> mapa\_avaliacoes = {"Terrible": 1, "Poor": 2, "Average": 3, "Very Good": 4, "Excellent": 5}

Comentários sobre as avaliações:

>>> print(f"- Média das avaliações: {media:.2f}")

- Média das avaliações: 3.74

>>> print(f"- Mediana das avaliações: {mediana}")

- Mediana das avaliações: 4.0

>>> print(f"- Moda das avaliações: {moda}")

- Moda das avaliações: Very Good

e)

Distribuição de Frequência - Disneys Grand Californian:

>>> print(dados\_disney)

Classificação Frequência Frequência Percentual

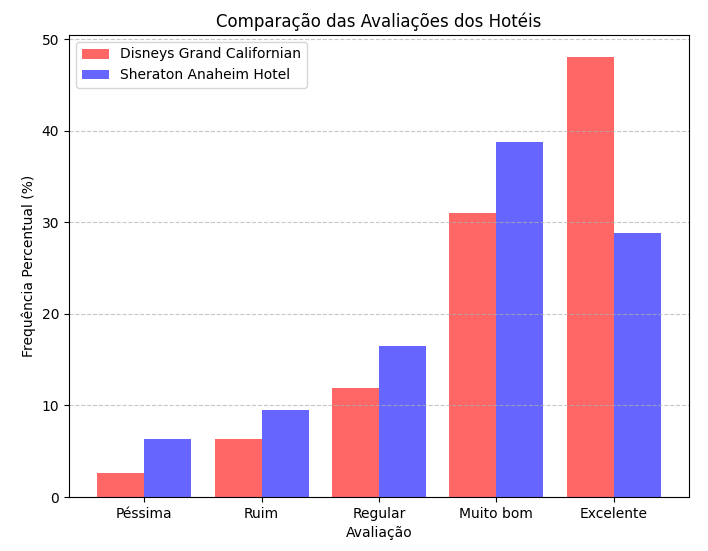
0 Excelente 807 48.064324

1 Muito bom 521 31.030375

2 Regular 200 11.911852

3 Ruim 107 6.372841

4 Péssima 44 2.620608



**Cap 2:18**

a)

Distribuição de Frequência:

>>> print(df\_frequencia)

Intervalo Frequência

0 10 - 12 0

1 12 - 14 0

2 14 - 16 6

3 16 - 18 19

4 18 - 20 6

5 20 - 22 4

6 22 - 24 2

7 24 - 26 0

8 26 - 28 3

9 28 - 30 1

b) e c)

>>> print("\nDistribuição de Frequência Completa:")

Distribuição de Frequência Completa:

>>> print(df\_frequencia)

Intervalo Frequência Frequência Relativa Frequência Percentual Acumulada

0 10 - 12 0 0.000000 0.000000

1 12 - 14 0 0.000000 0.000000

2 14 - 16 6 0.146341 14.634146

3 16 - 18 19 0.463415 60.975610

4 18 - 20 6 0.146341 75.609756

5 20 - 22 4 0.097561 85.365854

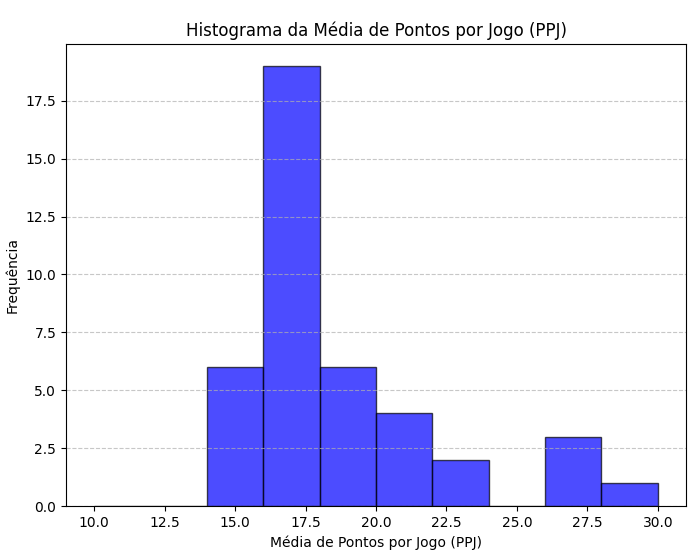
6 22 - 24 2 0.048780 90.243902

7 24 - 26 0 0.000000 90.243902

8 26 - 28 3 0.073171 97.560976

9 28 - 30 1 0.024390 100.000000

d)



e)

>>> print(f"\nMédia: {media:.2f}, Mediana: {mediana:.2f}")

Média: 18.65, Mediana: 17.40

>>> print("\nOs dados apresentam uma assimetria positiva (distorção à direita).")

Os dados apresentam uma assimetria positiva (distorção à direita).

f)

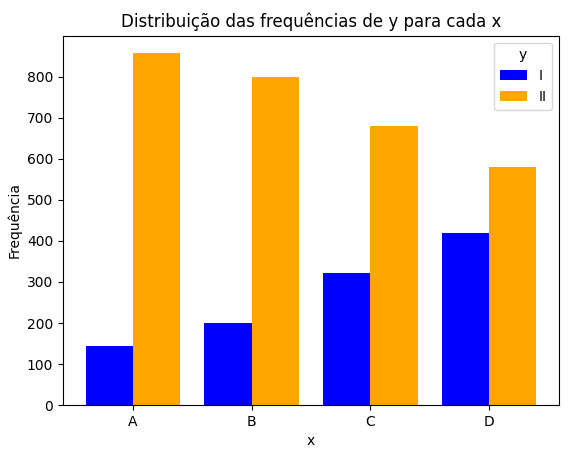
>>> porcentagem\_20\_mais = (sum(np.array(ppj) >= 20) / len(ppj)) \* 100

>>> print(f"\nPorcentagem de jogadores com pelo menos 20 pontos por jogo: {porcentagem\_20\_mais:.2f}%")

Porcentagem de jogadores com pelo menos 20 pontos por jogo: 24.39%

**Cap 2:37**

a)



b)

O comportamento dos dados sugere que a categoria y=II é mais frequente, mas essa diferença se reduz conforme x passa de A para D.

**Cap 2:52**

a)

>>> print(cross\_tab)

Size Large Midsized Small Total

Growth\_Class

-10 a -1 2 7 5 14

0 a 9 30 12 18 60

10 a 19 3 3 6 12

20 a 29 2 3 3 8

30 a 39 1 2 0 3

60 a 69 0 1 0 1

Total 38 28 32 98

b)

>>> print(growth\_freq)

Taxa de Crescimento (%) Frequência Frequência Relativa (%) Frequência Acumulada Frequência Relativa Acumulada (%)

1 -10 a -1 14 14.3 74 75.5

0 0 a 9 60 61.2 60 61.2

2 10 a 19 12 12.2 86 87.8

3 20 a 29 8 8.2 94 95.9

4 30 a 39 3 3.1 97 99.0

6 40 a 49 0 0.0 98 100.0

7 50 a 59 0 0.0 98 100.0

5 60 a 69 1 1.0 98 100.0

>>> print(size\_freq)

Tamanho Frequência Frequência Relativa (%) Frequência Acumulada Frequência Relativa Acumulada (%)

0 Large 38 38.8 38 38.8

1 Small 32 32.7 70 71.4

2 Midsized 28 28.6 98 100.0

c)

>>> print(cross\_tab\_col\_pct)

Size Large Midsized Small Total

Growth\_Class

-10 a -1 5.26 25.00 15.62 14.29

0 a 9 78.95 42.86 56.25 61.22

10 a 19 7.89 10.71 18.75 12.24

20 a 29 5.26 10.71 9.38 8.16

30 a 39 2.63 7.14 0.00 3.06

60 a 69 0.00 3.57 0.00 1.02

d)

>>> print(cross\_tab\_row\_pct)

Size Large Midsized Small

Growth\_Class

-10 a -1 14.29 50.00 35.71

0 a 9 50.00 20.00 30.00

10 a 19 25.00 25.00 50.00

20 a 29 25.00 37.50 37.50

30 a 39 33.33 66.67 0.00

60 a 69 0.00 100.00 0.00

Total 38.78 28.57 32.65

e)

>>> print(growth\_by\_size)

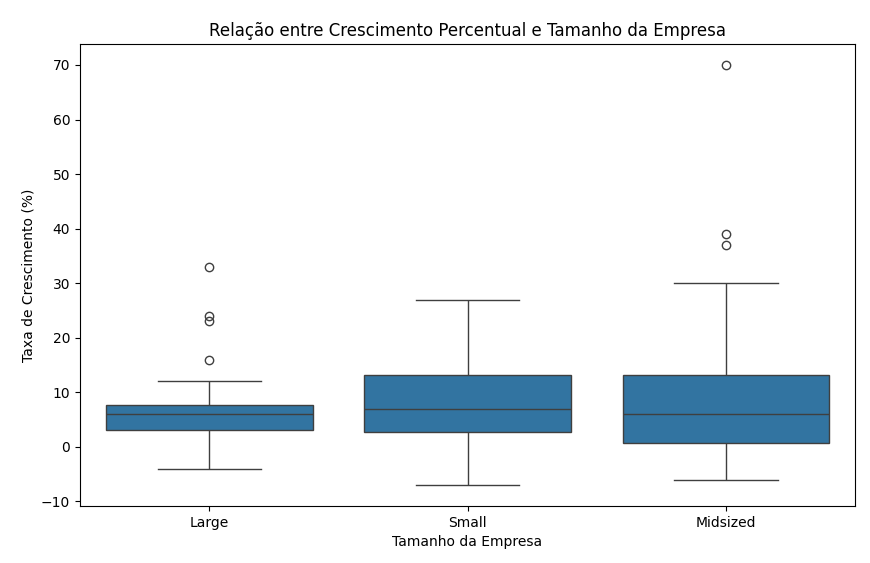
mean median min max std

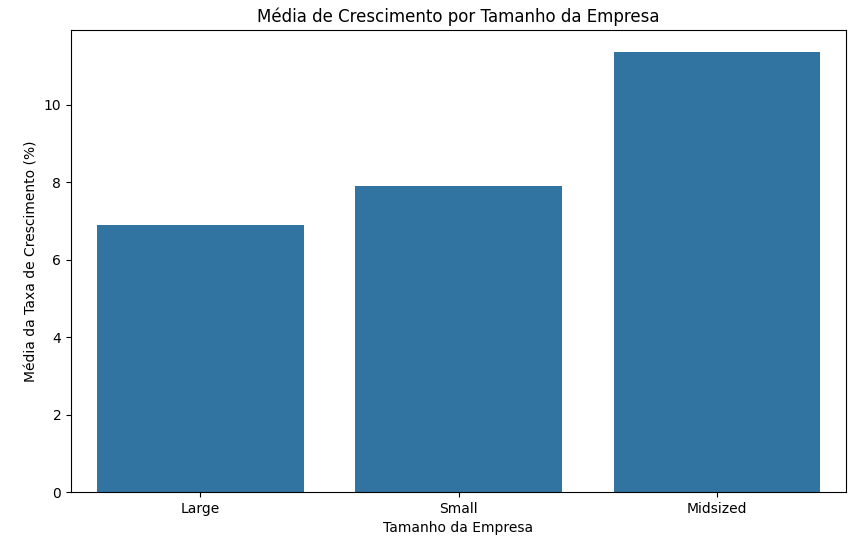
Size

Large 6.89 6.0 -4 33 7.12

Midsized 11.36 6.0 -6 70 16.77

Small 7.91 7.0 -7 27 8.16





**Cap 3:12**

a)

>>> print(f" Mínimo: {min\_viewers} milhões (em {min\_date.strftime('%d/%m/%Y')})")

Mínimo: 13.3 milhões (em 05/04/2012)

>>> print(f" Máximo: {max\_viewers} milhões (em {max\_date.strftime('%d/%m/%Y')})")

Máximo: 16.5 milhões (em 02/02/2012)

b)

>>> print(f" Média: {mean\_viewers:.2f} milhões")

Média: 15.04 milhões

>>> print(f" Mediana: {median\_viewers:.2f} milhões")

Mediana: 15.00 milhões

>>> print(f" Moda: {mode\_viewers:.1f} milhões")

Moda: 13.6 milhões

c)

>>> print(f" Primeiro quartil (Q1): {q1:.2f} milhões")

Primeiro quartil (Q1): 14.10 milhões

>>> print(f" Terceiro quartil (Q3): {q3:.2f} milhões")

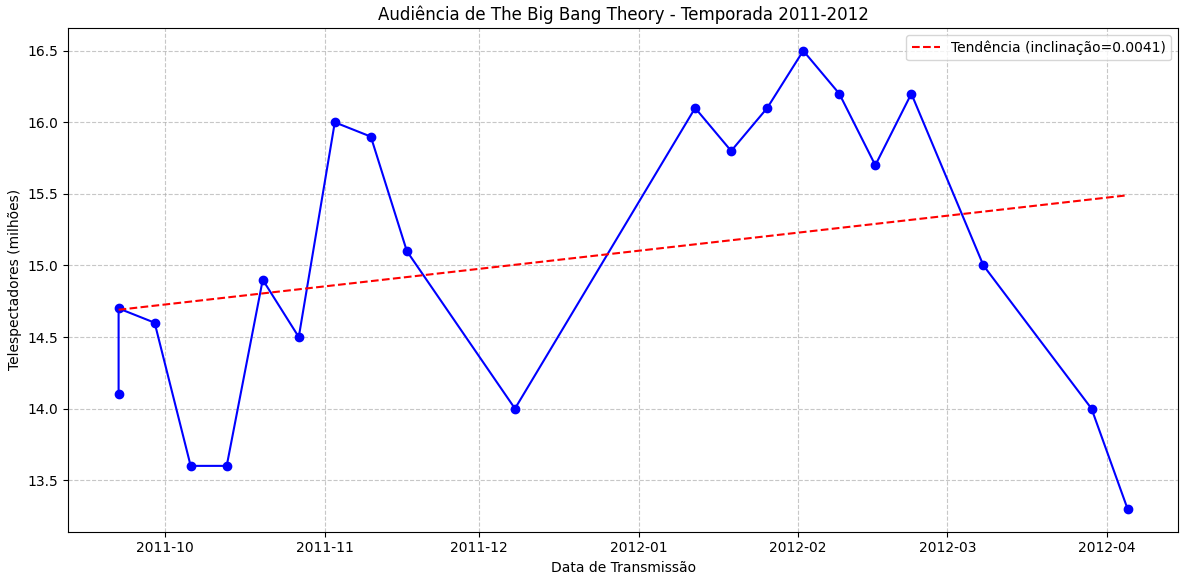
Terceiro quartil (Q3): 16.00 milhões

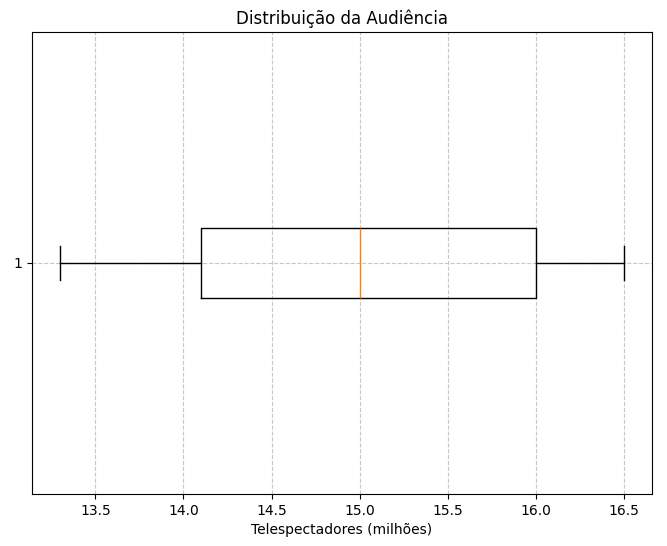
d)

>>> print(f" Inclinação da linha de tendência: {slope:.4f} milhões por dia")

Inclinação da linha de tendência: 0.0041 milhões por dia

A audiência teve uma tendência de AUMENTO ao longo da temporada e aumento de 0.65 milhões na segunda metade.





**Cap 3:29**

a)

>>> print(f"a) Amplitude: {pomona\_amplitude}")

Amplitude: 32

>>> print(f" Amplitude Interquartil (IQR): {pomona\_iqr:.2f}")

Amplitude Interquartil (IQR): 10.00

b)

>>> print(f"b) Variância Amostral: {pomona\_variancia:.2f}")

Variância Amostral: 92.75

>>> print(f" Desvio Padrão Amostral: {pomona\_desvio\_padrao:.2f}")

Desvio Padrão Amostral: 9.63

c)

Comparação entre Pomona e Anaheim:

>>> print(f"Média: Pomona = {pomona\_media:.2f}, Anaheim = {anaheim\_media}")

Média: Pomona = 48.33, Anaheim = 48.5

>>> print(f"Variância: Pomona = {pomona\_variancia:.2f}, Anaheim = {anaheim\_variancia}")

Variância: Pomona = 92.75, Anaheim = 136

>>> print(f"Desvio Padrão: Pomona = {pomona\_desvio\_padrao:.2f}, Anaheim = {anaheim\_desvio\_padrao}")

Desvio Padrão: Pomona = 9.63, Anaheim = 11.66

Interpretação dos resultados:

A qualidade do ar em Pomona tem um índice médio mais baixo (48.33) do que em Anaheim (48.5).

Os dados de Pomona apresentam menor variabilidade (variância = 92.75) do que os de Anaheim (variância = 136).

Os dados de Pomona apresentam menor dispersão (desvio padrão = 9.63) do que os de Anaheim (desvio padrão = 11.66).

**Cap 3:34**

Análise dos Tempos de Corrida - Equipe Universitária

Prova de Um Quarto de Milha:

>>> print(f"Tempos (minutos): {', '.join([str(t) for t in tempos\_quarto\_milha])}")

Tempos (minutos): 0.92, 0.98, 1.04, 0.9, 0.99

>>> print(f"Média: {media\_quarto:.3f} minutos")

Média: 0.966 minutos

>>> print(f"Desvio Padrão: {dp\_quarto:.3f} minutos")

Desvio Padrão: 0.056 minutos

>>> print(f"Coeficiente de Variação: {cv\_quarto:.2f}%")

Coeficiente de Variação: 5.84%

>>> print("\nProva de Uma Milha:")

Prova de Uma Milha:

>>> print(f"Tempos (minutos): {', '.join([str(t) for t in tempos\_milha])}")

Tempos (minutos): 4.52, 4.35, 4.6, 4.7, 4.5

>>> print(f"Média: {media\_milha:.3f} minutos")

Média: 4.534 minutos

>>> print(f"Desvio Padrão: {dp\_milha:.3f} minutos")

Desvio Padrão: 0.130 minutos

>>> print(f"Coeficiente de Variação: {cv\_milha:.2f}%")

Coeficiente de Variação: 2.86%

O coeficiente de variação da prova de um quarto de milha (5.84%) é maior que o da prova de uma milha (2.86%). Isso indica que os tempos na prova de uma milha são mais consistentes, o que confirma a declaração do treinador. A declaração do técnico é suportada pela análise estatística. O coeficiente de variação (CV) expressa a variabilidade dos dados em relação à média. Quanto menor o CV, maior a consistência dos dados (menor variabilidade relativa).

Um CV menor indica tempos mais homogêneos entre os corredores.

**Cap 3:36**

>>> print("\nCálculo de Escores-z com Média = 500 e Desvio Padrão = 100")

Cálculo de Escores-z com Média = 500 e Desvio Padrão = 100

>>> print(f"Fórmula do Escore-z: z = (x - μ) / σ, onde:")

Fórmula do Escore-z: z = (x - μ) / σ, onde:

>>> print(f" - x é o valor a ser padronizado")

- x é o valor a ser padronizado

>>> print(f" - μ é a média da distribuição (500)")

- μ é a média da distribuição (500)

>>> print(f" - σ é o desvio padrão da distribuição (100)")

- σ é o desvio padrão da distribuição (100)

Valor | Cálculo | Escore-z | Interpretação

520 | (520 - 500) / 100 | 0.20 | Próximo à média (menos de 1 DP de distância), acima da média

650 | (650 - 500) / 100 | 1.50 | Moderadamente distante da média (entre 1 e 2 DP), acima da média

500 | (500 - 500) / 100 | 0.00 | Próximo à média (menos de 1 DP de distância)

450 | (450 - 500) / 100 | -0.50 | Próximo à média (menos de 1 DP de distância), abaixo da média

280 | (280 - 500) / 100 | -2.20 | Consideravelmente distante da média (entre 2 e 3 DP), abaixo da média

- O escore-z representa quantos desvios padrão um valor está acima ou abaixo da média

- Valores positivos estão acima da média, valores negativos estão abaixo

- Na distribuição normal:

\* Aproximadamente 68% dos valores têm escores-z entre -1 e +1

\* Aproximadamente 95% dos valores têm escores-z entre -2 e +2

\* Aproximadamente 99.7% dos valores têm escores-z entre -3 e +3

**Cap 3:44**

a)

Pontos do time vencedor:

>>> print(f" Média: {media\_pontos\_vencedor:.2f}")

Média: 77.10

>>> print(f" Desvio padrão: {desvio\_padrao\_pontos\_vencedor:.2f}")

Desvio padrão: 7.22

b)

Porcentagem de jogos em que o time vencedor marca mais de 90 pontos:

>>> print(f" {porcentagem:.2f}%")

3.70%

c)

Margem de pontos do time vitorioso:

>>> print(f" Média: {media\_margem:.2f}")

Média: 12.80

>>> print(f" Desvio padrão: {desvio\_padrao\_margem:.2f}")

Desvio padrão: 8.73

Analisando os dados de pontuações dos times vencedores, perdedores e margens das partidas de basquete universitário da NCAA, podemos concluir que não existem outliers significativos em nenhum dos três conjuntos. Utilizando o método do intervalo interquartil (IQR), que define como outliers valores abaixo de Q1-1.5×IQR ou acima de Q3+1.5×IQR, todos os pontos se mantêm dentro dos limites aceitáveis. Os pontos dos times vencedores variam entre 65 e 90, os dos perdedores entre 56 e 70, e as margens entre 3 e 26 pontos, demonstrando uma variabilidade natural esperada em resultados esportivos, sem valores extremos que pudessem distorcer as análises estatísticas.

**Cap 3:48**

Dados ordenados: [5, 6, 8, 10, 10, 12, 15, 16, 18]

Resumo de cinco números (Five-number summary):

>>> print(f"Mínimo: {minimo}")

Mínimo: 5

>>> print(f"Q1 (Primeiro quartil): {q1}")

Q1 (Primeiro quartil): 8.0

>>> print(f"Mediana: {mediana}")

Mediana: 10.0

>>> print(f"Q3 (Terceiro quartil): {q3}")

Q3 (Terceiro quartil): 15.0

>>> print(f"Máximo: {maximo}")

Máximo: 18

>>> print(f"IQR (Intervalo interquartil): {iqr}")

IQR (Intervalo interquartil): 7.0

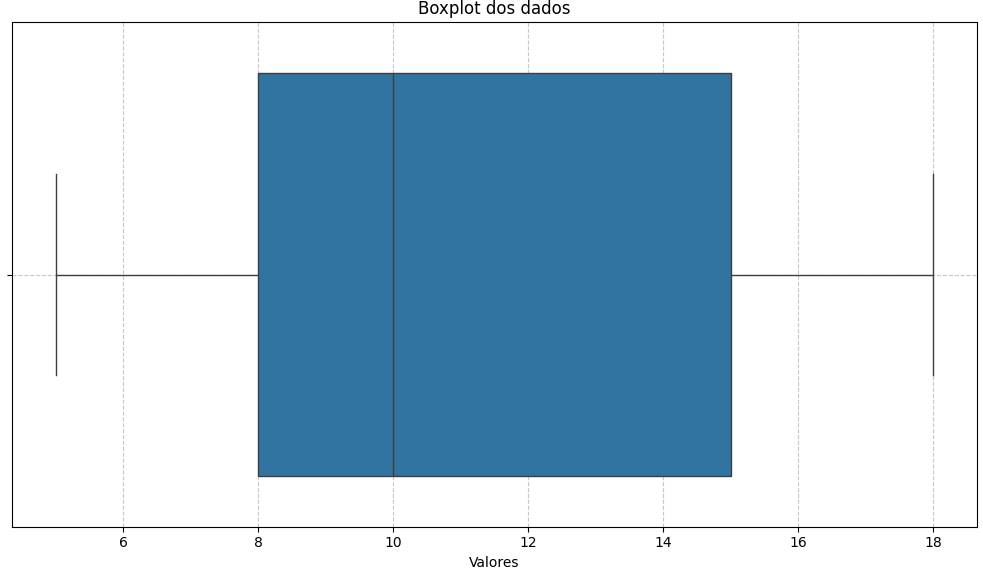
>>> print(f"\nLimite inferior para outliers: {limite\_inferior}")

Limite inferior para outliers: -2.5

>>> print(f"Limite superior para outliers: {limite\_superior}")

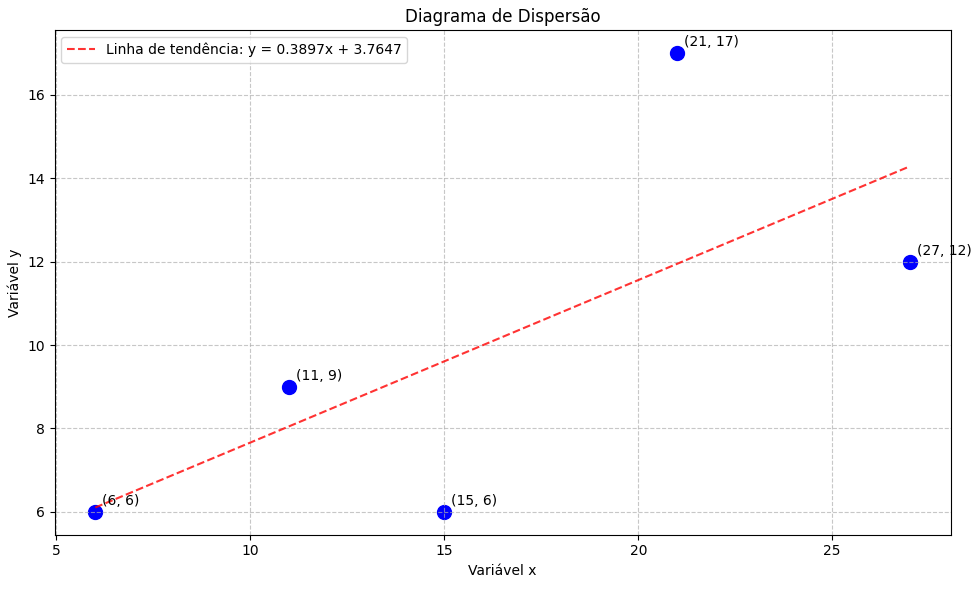
Limite superior para outliers: 25.5

Sem outliers



**Cap 3:56**

a)



b)

O diagrama demonstra potencial relação positiva entre as variáveis.

c)

Covariância amostral: 26.5000

Covariância positiva, indicando co-movimentação na mesma direção.

d)

Coeficiente de correlação amostral (r): 0.6931

>>> print(f" p-valor: {p\_valor:.4f}")

p-valor: 0.1945

Existe uma correlação moderada e positiva entre as variáveis x e y.

**Cap 3:59**

a)

Espera-se relação negativa já que a maior adoção de detectores de fumaça tem potencial para reduzir a taxa de mortalidade por incêndios residenciais.

b)

Relação entre uso de detectores de fumaça e mortes:

>>> print(f" Coeficiente de correlação: {correlacao:.4f}")

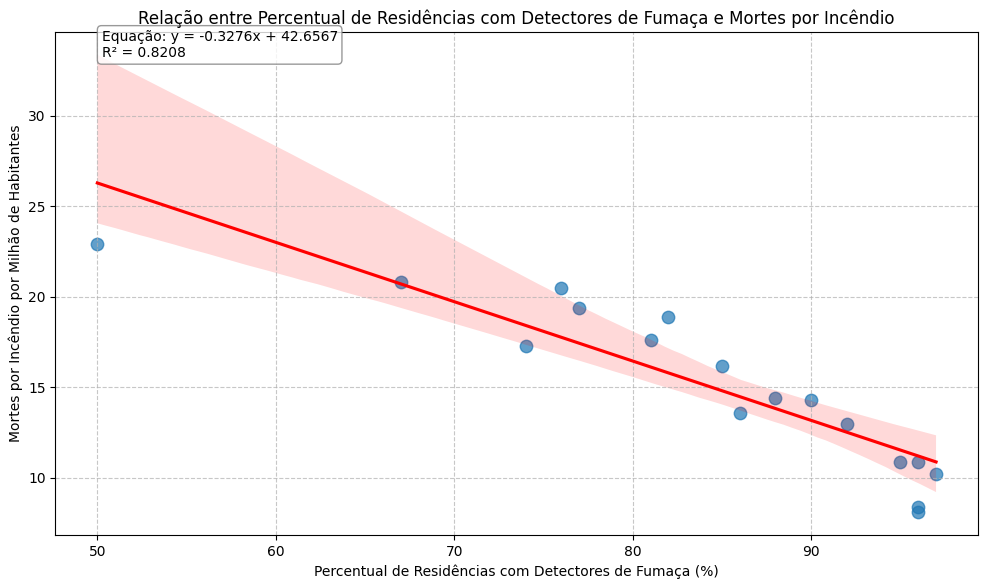
Coeficiente de correlação: -0.9060

>>> print(f" p-valor: {p\_valor:.8f}")

p-valor: 0.00000056

Existe uma relação NEGATIVA entre o uso de detectores de fumaça e mortes por incêndio. Isso significa que quanto maior a porcentagem de residências com detectores, menor é a taxa de mortalidade. O valor p de 0.00000056 indica que essa correlação é estatisticamente significativa.

c)



**Apêndice – Código em Python**

# 1a lista

'''

Nome da Tarefa:

1ª lista de exercícios: estatísticas descritivas (capítulos 2 e 3)

Descrição:

Exercícios dos capítulos 2 e 3 do livro Estatística Aplicada a Administração e Economia.

Cap. 2: 10, 18, 37, 52.

Cap. 3: 12, 29, 34, 36, 44, 48, 56, 59.

Descrição de gráfico

Período:

Inicia em 19/03/2025 às 00h00 e finaliza em 31/03/2025 às 23h59

'''

## Cap 2 - 10

import pandas as pd

import matplotlib.pyplot as plt

# Carregar os dados do Excel

arquivo = r"C:\Users\camil\Documents\repo\metodosquantitativosi\data\_files\SBE13e\_DATAfiles\_CH02\HotelRatings.xlsx"

df = pd.read\_excel(arquivo)

# Selecionar a primeira coluna (assumindo que contém as avaliações) e garantir que os valores sejam strings

avaliacoes = df.iloc[:, 0].astype(str).str.strip()

# Definir a ordem correta das categorias

ordem\_avaliacoes = ["Terrible", "Poor", "Average", "Very Good", "Excellent"]

# Contar as ocorrências corretamente sem reindexamento inicial

frequencia = avaliacoes.value\_counts()

# Reindexar garantindo que todas as categorias apareçam, preenchendo ausências com zero

frequencia = frequencia.reindex(ordem\_avaliacoes, fill\_value=0)

print("Distribuição de Frequência:")

print(frequencia)

# Distribuição de frequência percentual ordenada

frequencia\_percentual = (frequencia / frequencia.sum()) \* 100

print("\nDistribuição de Frequência Percentual:")

print(frequencia\_percentual)

# Gráfico de barras da distribuição de frequência percentual

plt.figure(figsize=(8, 6))

plt.bar(frequencia\_percentual.index, frequencia\_percentual.values, color='blue', alpha=0.7)

plt.xlabel("Avaliação")

plt.ylabel("Frequência Percentual (%)")

plt.title("Distribuição de Frequência Percentual das Avaliações")

plt.xticks(rotation=0)

plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.show()

# Comentários sobre as avaliações

mapa\_avaliacoes = {"Terrible": 1, "Poor": 2, "Average": 3, "Very Good": 4, "Excellent": 5}

avaliacoes\_numericas = avaliacoes.map(mapa\_avaliacoes)

media = avaliacoes\_numericas.mean()

mediana = avaliacoes\_numericas.median()

moda = avaliacoes.mode()[0]

print("\nComentários sobre as avaliações:")

print(f"- Média das avaliações: {media:.2f}")

print(f"- Mediana das avaliações: {mediana}")

print(f"- Moda das avaliações: {moda}")

# Criar um DataFrame com os dados do Disney’s Grand Californian

dados\_disney = pd.DataFrame({

    "Classificação": ["Excelente", "Muito bom", "Regular", "Ruim", "Péssima"],

    "Frequência": [807, 521, 200, 107, 44]

})

# Garantir a mesma ordem de categorias para comparação

dados\_disney["Classificação"] = pd.Categorical(dados\_disney["Classificação"],

                                               categories=["Péssima", "Ruim", "Regular", "Muito bom", "Excelente"],

                                               ordered=True)

# Calcular distribuição percentual

dados\_disney["Frequência Percentual"] = (dados\_disney["Frequência"] / dados\_disney["Frequência"].sum()) \* 100

# Exibir os resultados

print("\nDistribuição de Frequência - Disney’s Grand Californian:")

print(dados\_disney)

# Criar DataFrame para Disney’s Grand Californian

dados\_disney = pd.DataFrame({

    "Classificação": ["Péssima", "Ruim", "Regular", "Muito bom", "Excelente"],

    "Frequência": [44, 107, 200, 521, 807]

})

# Criar DataFrame para Sheraton Anaheim Hotel (frequencia já calculada anteriormente)

dados\_sheraton = pd.DataFrame({

    "Classificação": ["Péssima", "Ruim", "Regular", "Muito bom", "Excelente"],

    "Frequência": [41, 62, 107, 252, 187]

})

# Definir a mesma ordem categórica para os dois conjuntos de dados

ordem\_classes = ["Péssima", "Ruim", "Regular", "Muito bom", "Excelente"]

dados\_disney["Classificação"] = pd.Categorical(dados\_disney["Classificação"], categories=ordem\_classes, ordered=True)

dados\_sheraton["Classificação"] = pd.Categorical(dados\_sheraton["Classificação"], categories=ordem\_classes, ordered=True)

# Calcular distribuições percentuais

dados\_disney["Frequência Percentual"] = (dados\_disney["Frequência"] / dados\_disney["Frequência"].sum()) \* 100

dados\_sheraton["Frequência Percentual"] = (dados\_sheraton["Frequência"] / dados\_sheraton["Frequência"].sum()) \* 100

# Criar gráfico comparativo de barras lado a lado

largura\_barra = 0.4  # Definir largura para ajustar as barras lado a lado

x = range(len(ordem\_classes))  # Criar posições para as categorias

plt.figure(figsize=(8, 6))

plt.bar([pos - largura\_barra/2 for pos in x], dados\_disney["Frequência Percentual"], width=largura\_barra, color='red', alpha=0.6, label="Disney’s Grand Californian")

plt.bar([pos + largura\_barra/2 for pos in x], dados\_sheraton["Frequência Percentual"], width=largura\_barra, color='blue', alpha=0.6, label="Sheraton Anaheim Hotel")

plt.xlabel("Avaliação")

plt.ylabel("Frequência Percentual (%)")

plt.title("Comparação das Avaliações dos Hotéis")

plt.xticks(ticks=x, labels=ordem\_classes, rotation=0)

plt.legend()

plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.show()

## Cap 2 - 18

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Lista com os valores dos PPJ (média de pontos por jogo)

ppj = [27.0, 28.8, 26.4, 27.1, 22.9, 19.2, 21.0, 20.8, 17.6,

       21.1, 19.2, 21.2, 15.5, 17.2, 16.7, 18.5, 18.3, 18.2,

       23.3, 16.4, 18.9, 16.5, 17.7, 16.8, 17.0, 17.4, 14.6,

       15.7, 17.2, 17.0, 15.3, 17.8, 16.7, 17.4, 16.3, 16.7,

       17.0, 17.5, 14.0, 16.3, 14.6]

# Criar classes de 10 a 30 com intervalo de 2

bins = np.arange(10, 32, 2)

# Criar a distribuição de frequência

frequencia, bins\_edges = np.histogram(ppj, bins=bins)

# Criar DataFrame para visualizar melhor

df\_frequencia = pd.DataFrame({

    "Intervalo": [f"{bins\_edges[i]} - {bins\_edges[i+1]}" for i in range(len(bins\_edges)-1)],

    "Frequência": frequencia

})

# Exibir distribuição de frequência

print("\nDistribuição de Frequência:")

print(df\_frequencia)

# Calcular distribuição de frequência relativa

df\_frequencia["Frequência Relativa"] = df\_frequencia["Frequência"] / len(ppj)

# Calcular distribuição de frequência percentual acumulada

df\_frequencia["Frequência Percentual Acumulada"] = df\_frequencia["Frequência"].cumsum() / len(ppj) \* 100

# Exibir tabela completa

print("\nDistribuição de Frequência Completa:")

print(df\_frequencia)

# Criar histograma

plt.figure(figsize=(8,6))

plt.hist(ppj, bins=bins, edgecolor="black", alpha=0.7, color="blue")

plt.xlabel("Média de Pontos por Jogo (PPJ)")

plt.ylabel("Frequência")

plt.title("Histograma da Média de Pontos por Jogo (PPJ)")

plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.show()

# Calcular média e mediana

media = np.mean(ppj)

mediana = np.median(ppj)

# Exibir valores médios

print(f"\nMédia: {media:.2f}, Mediana: {mediana:.2f}")

# Responder sobre a distorção dos dados

print("\nOs dados apresentam uma assimetria positiva (distorção à direita).")

# Calcular a porcentagem de jogadores com PPJ >= 20

porcentagem\_20\_mais = (sum(np.array(ppj) >= 20) / len(ppj)) \* 100

print(f"\nPorcentagem de jogadores com pelo menos 20 pontos por jogo: {porcentagem\_20\_mais:.2f}%")

## Cap 2 - 37

import matplotlib.pyplot as plt

import numpy as np

# Dados extraídos da imagem

categorias\_x = ['A', 'B', 'C', 'D']

valores\_y\_I = [143, 200, 321, 420]

valores\_y\_II = [857, 800, 679, 580]

# Posição das barras

x = np.arange(len(categorias\_x))

largura = 0.4

# Criando o gráfico de barras

fig, ax = plt.subplots()

ax.bar(x - largura/2, valores\_y\_I, largura, label='I', color='blue')

ax.bar(x + largura/2, valores\_y\_II, largura, label='II', color='orange')

# Configurações do gráfico

ax.set\_xlabel('x')

ax.set\_ylabel('Frequência')

ax.set\_title('Distribuição das frequências de y para cada x')

ax.set\_xticks(x)

ax.set\_xticklabels(categorias\_x)

ax.legend(title="y")

# Exibir o gráfico

plt.show()

## Cap 2 - 52

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Criando o DataFrame a partir dos dados fornecidos

data = [

    [1, "Google", "Large", 33],

    [2, "Boston Consulting Group", "Small", 10],

    [3, "SAS Institute", "Midsized", 8],

    [4, "Wegmans Food Markets", "Large", 5],

    [5, "Edward Jones", "Large", 1],

    [6, "NetApp", "Midsized", 30],

    [7, "Camden Property Trust", "Small", -2],

    [8, "Recreational Equipment (REI)", "Large", 12],

    [9, "CHG Healthcare Services", "Small", 17],

    [10, "Quicken Loans", "Midsized", 20],

    [11, "Zappos.com", "Midsized", 70],

    [12, "Mercedes-Benz USA", "Small", 2],

    [13, "DPR Construction", "Small", 18],

    [14, "DreamWorks Animation", "Small", 8],

    [15, "NuStar Energy", "Small", 6],

    [16, "Kimpton Hotels & Restaurants", "Midsized", 4],

    [17, "JM Family Enterprises", "Midsized", -1],

    [18, "Chesapeake Energy", "Large", 23],

    [19, "Intuit", "Midsized", 9],

    [20, "USAA", "Large", 7],

    [21, "Robert W. Baird", "Midsized", 5],

    [22, "The Container Store", "Midsized", 11],

    [23, "Qualcomm", "Large", 6],

    [24, "Alston & Bird", "Small", 3],

    [25, "Ultimate Software", "Small", 15],

    [26, "Burns & McDonnell", "Midsized", 5],

    [27, "Salesforce.com", "Midsized", 39],

    [28, "Devon Energy", "Midsized", -6],

    [29, "PCL Construction", "Small", -5],

    [30, "Bingham McCutchen", "Small", -7],

    [31, "Scottrade", "Midsized", 9],

    [32, "Whole Foods Market", "Large", 6],

    [34, "Nugget Market", "Small", 8],

    [35, "Millennium: The Takeda Oncology Co.", "Small", 3],

    [36, "Southern Ohio Medical Center", "Small", 18],

    [37, "Plante Moran", "Small", 1],

    [38, "W. L. Gore & Associates", "Midsized", 2],

    [39, "St. Jude Children's Research Hospital", "Midsized", 1],

    [40, "SVB Financial Group", "Small", 9],

    [41, "Adobe", "Midsized", 11],

    [42, "Baptist Health South Florida", "Large", 10],

    [44, "Balfour Beatty Construction", "Small", -2],

    [45, "National Instruments", "Midsized", 7],

    [46, "Intel", "Large", 4],

    [47, "American Fidelity Assurance", "Small", 0],

    [48, "PricewaterhouseCoopers", "Large", 9],

    [49, "Children's Healthcare of Atlanta", "Midsized", -1],

    [50, "World Wide Technology", "Small", 23],

    [51, "Allianz Life Insurance", "Small", 2],

    [52, "Autodesk", "Midsized", 5],

    [53, "Methodist Hospital", "Large", 8],

    [54, "Baker Donelson", "Small", 3],

    [55, "Men's Wearhouse", "Large", 2],

    [56, "Scripps Health", "Large", 2],

    [57, "Marriott International", "Large", 3],

    [58, "Perkins Coie", "Small", 7],

    [59, "Ernst & Young", "Large", 6],

    [60, "American Express", "Large", 4],

    [61, "Nordstrom", "Large", 6],

    [62, "Build-A-Bear Workshop", "Midsized", 0],

    [63, "General Mills", "Large", 1],

    [64, "TDIndustries", "Small", 9],

    [65, "Atlantic Health", "Midsized", -2],

    [66, "QuikTrip", "Large", 3],

    [67, "Deloitte", "Large", 7],

    [68, "Genentech", "Large", 1],

    [69, "Umpqua Bank", "Small", 5],

    [70, "Teach For America", "Small", 14],

    [71, "Mayo Clinic", "Large", 3],

    [72, "EOG Resources", "Small", 13],

    [73, "Starbucks", "Large", 3],

    [74, "Rackspace Hosting", "Midsized", 37],

    [75, "FactSet Research Systems", "Small", 22],

    [76, "Microsoft", "Large", -4],

    [77, "Aflac", "Midsized", -4],

    [78, "Publix Super Markets", "Large", 1],

    [79, "Mattel", "Midsized", -4],

    [80, "Stryker", "Large", 24],

    [81, "SRC", "Small", 7],

    [82, "Hasbro", "Midsized", 3],

    [83, "Bright Horizons Family Solutions", "Large", 5],

    [84, "Booz Allen Hamilton", "Large", 7],

    [85, "Four Seasons Hotels & Resorts", "Large", 6],

    [86, "Hitachi Data Systems", "Small", 7],

    [87, "The Everett Clinic", "Small", 4],

    [88, "OhioHealth", "Large", 4],

    [89, "Morningstar", "Small", 8],

    [90, "Cisco", "Large", 7],

    [91, "CarMax", "Large", 16],

    [92, "Accenture", "Large", 9],

    [93, "GoDaddy.com", "Midsized", 25],

    [94, "KPMG", "Large", 5],

    [95, "Navy Federal Credit Union", "Midsized", 8],

    [96, "Meridian Health", "Midsized", 27],

    [97, "Schweitzer Engineering Labs", "Small", 27],

    [98, "Capital One", "Large", 7],

    [99, "Darden Restaurants", "Large", 12],

    [100, "Intercontinental Hotels Group", "Large", -2]

]

# Criando o DataFrame

df = pd.DataFrame(data, columns=['Rank', 'Company', 'Size', 'Job\_Growth'])

# a. Construa uma tabulação cruzada com Taxa de crescimento (%) como variável linha e Tamanho como variável coluna.

# Use classes começando em -10 e terminando em 70 em gradações de 10 para a Taxa de crescimento (%).

# Definindo os intervalos para a taxa de crescimento

bins = range(-10, 80, 10)

labels = [f"{i} a {i+9}" for i in range(-10, 70, 10)]

# Criando uma nova coluna com as classes de crescimento

df['Growth\_Class'] = pd.cut(df['Job\_Growth'], bins=bins, labels=labels, right=True)

# Criando a tabulação cruzada

cross\_tab = pd.crosstab(df['Growth\_Class'], df['Size'], margins=True, margins\_name='Total')

print("a. Tabulação cruzada com Taxa de crescimento (%) como variável linha e Tamanho como variável coluna:")

print(cross\_tab)

print("\n")

# b. Mostre a distribuição de frequência para a taxa de crescimento dos funcionários no trabalho (%)

# e a distribuição de frequência para Tamanho.

# Distribuição de frequência para taxa de crescimento

growth\_freq = pd.DataFrame(df['Growth\_Class'].value\_counts()).reset\_index()

growth\_freq.columns = ['Taxa de Crescimento (%)', 'Frequência']

growth\_freq['Frequência Relativa (%)'] = (growth\_freq['Frequência'] / len(df) \* 100).round(1)

growth\_freq['Frequência Acumulada'] = growth\_freq['Frequência'].cumsum()

growth\_freq['Frequência Relativa Acumulada (%)'] = (growth\_freq['Frequência Acumulada'] / len(df) \* 100).round(1)

growth\_freq = growth\_freq.sort\_values('Taxa de Crescimento (%)')

print("b.1 Distribuição de frequência para taxa de crescimento:")

print(growth\_freq)

print("\n")

# Distribuição de frequência para Tamanho

size\_freq = pd.DataFrame(df['Size'].value\_counts()).reset\_index()

size\_freq.columns = ['Tamanho', 'Frequência']

size\_freq['Frequência Relativa (%)'] = (size\_freq['Frequência'] / len(df) \* 100).round(1)

size\_freq['Frequência Acumulada'] = size\_freq['Frequência'].cumsum()

size\_freq['Frequência Relativa Acumulada (%)'] = (size\_freq['Frequência Acumulada'] / len(df) \* 100).round(1)

print("b.2 Distribuição de frequência para Tamanho:")

print(size\_freq)

print("\n")

# c. Usando a tabulação cruzada construída na parte (a), desenvolva uma tabulação cruzada mostrando as porcentagens em coluna.

# Tabulação cruzada com porcentagens em coluna

cross\_tab\_col\_pct = pd.crosstab(df['Growth\_Class'], df['Size'], normalize='columns', margins=True, margins\_name='Total')

cross\_tab\_col\_pct = cross\_tab\_col\_pct.round(4) \* 100

print("c. Tabulação cruzada mostrando as porcentagens em coluna:")

print(cross\_tab\_col\_pct)

print("\n")

# d. Usando a tabulação cruzada construída na parte (a), desenvolva uma tabulação cruzada mostrando as porcentagens em linhas.

# Tabulação cruzada com porcentagens em linha

cross\_tab\_row\_pct = pd.crosstab(df['Growth\_Class'], df['Size'], normalize='index', margins=True, margins\_name='Total')

cross\_tab\_row\_pct = cross\_tab\_row\_pct.round(4) \* 100

print("d. Tabulação cruzada mostrando as porcentagens em linhas:")

print(cross\_tab\_row\_pct)

print("\n")

# e. Comente a relação entre o crescimento percentual no emprego para os funcionários em tempo integral e o tamanho da empresa.

# Estatísticas descritivas por tamanho de empresa

growth\_by\_size = df.groupby('Size')['Job\_Growth'].agg(['mean', 'median', 'min', 'max', 'std']).round(2)

print("e. Estatísticas descritivas para crescimento percentual por tamanho de empresa:")

print(growth\_by\_size)

print("\n")

# Visualização da relação entre crescimento e tamanho da empresa

plt.figure(figsize=(10, 6))

boxplot = sns.boxplot(x='Size', y='Job\_Growth', data=df)

plt.title('Relação entre Crescimento Percentual e Tamanho da Empresa')

plt.xlabel('Tamanho da Empresa')

plt.ylabel('Taxa de Crescimento (%)')

plt.show()

# Gráfico de barras para média de crescimento por tamanho

plt.figure(figsize=(10, 6))

mean\_growth = df.groupby('Size')['Job\_Growth'].mean().sort\_values()

barplot = sns.barplot(x=mean\_growth.index, y=mean\_growth.values)

plt.title('Média de Crescimento por Tamanho da Empresa')

plt.xlabel('Tamanho da Empresa')

plt.ylabel('Média da Taxa de Crescimento (%)')

plt.show()

## Cap 3 - 12

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

import locale

from datetime import datetime

# Tentar configurar locale para português (opcional, pode não funcionar em todos os sistemas)

try:

    locale.setlocale(locale.LC\_TIME, 'pt\_BR.UTF-8')

except:

    pass  # Se não funcionar, usaremos método alternativo

# Criando o DataFrame com os dados da tabela

data = {

    'Data\_str': [

        '22 de setembro de 2011', '22 de setembro de 2011', '29 de setembro de 2011',

        '6 de outubro de 2011', '13 de outubro de 2011', '20 de outubro de 2011',

        '27 de outubro de 2011', '3 de novembro de 2011', '10 de novembro de 2011',

        '17 de novembro de 2011', '8 de dezembro de 2011', '12 de janeiro de 2012',

        '19 de janeiro de 2012', '26 de janeiro de 2012', '2 de fevereiro de 2012',

        '9 de fevereiro de 2012', '16 de fevereiro de 2012', '23 de fevereiro de 2012',

        '8 de março de 2012', '29 de março de 2012', '5 de abril de 2012'

    ],

    'Telespectadores': [

        14.1, 14.7, 14.6, 13.6, 13.6, 14.9, 14.5, 16.0, 15.9, 15.1, 14.0, 16.1,

        15.8, 16.1, 16.5, 16.2, 15.7, 16.2, 15.0, 14.0, 13.3

    ]

}

# Criando o DataFrame

df = pd.DataFrame(data)

# Função para converter as datas manualmente

def convert\_pt\_date(date\_str):

    day, month\_str, year = date\_str.split(' de ')

    month\_dict = {

        'janeiro': 1, 'fevereiro': 2, 'março': 3, 'abril': 4,

        'maio': 5, 'junho': 6, 'julho': 7, 'agosto': 8,

        'setembro': 9, 'outubro': 10, 'novembro': 11, 'dezembro': 12

    }

    month = month\_dict[month\_str]

    return pd.Timestamp(year=int(year), month=month, day=int(day))

# Aplicando a função de conversão

df['Data'] = df['Data\_str'].apply(convert\_pt\_date)

# Removendo a coluna com strings originais

df = df.drop('Data\_str', axis=1)

# Ordenando o DataFrame por data

df = df.sort\_values('Data')

# a. Calcule o número mínimo e máximo de telespectadores

min\_viewers = df['Telespectadores'].min()

max\_viewers = df['Telespectadores'].max()

min\_date = df.loc[df['Telespectadores'] == min\_viewers, 'Data'].iloc[0]

max\_date = df.loc[df['Telespectadores'] == max\_viewers, 'Data'].iloc[0]

print("a. Número mínimo e máximo de telespectadores:")

print(f"   Mínimo: {min\_viewers} milhões (em {min\_date.strftime('%d/%m/%Y')})")

print(f"   Máximo: {max\_viewers} milhões (em {max\_date.strftime('%d/%m/%Y')})")

print("\n")

# b. Calcule a média, a mediana e a moda

mean\_viewers = df['Telespectadores'].mean()

median\_viewers = df['Telespectadores'].median()

mode\_viewers = df['Telespectadores'].mode()[0]  # Pega a primeira moda (pode haver múltiplas)

print("b. Média, mediana e moda dos telespectadores:")

print(f"   Média: {mean\_viewers:.2f} milhões")

print(f"   Mediana: {median\_viewers:.2f} milhões")

print(f"   Moda: {mode\_viewers:.1f} milhões")

print("\n")

# c. Calcule o primeiro e o terceiro quartis

q1 = df['Telespectadores'].quantile(0.25)

q3 = df['Telespectadores'].quantile(0.75)

print("c. Primeiro e terceiro quartis:")

print(f"   Primeiro quartil (Q1): {q1:.2f} milhões")

print(f"   Terceiro quartil (Q3): {q3:.2f} milhões")

print("\n")

# d. Análise se a audiência aumentou ou diminuiu ao longo da temporada 2011-2012

# Criando uma coluna numérica para representar o tempo (dias desde o primeiro episódio)

df['Dias'] = (df['Data'] - df['Data'].min()).dt.days

# Calculando a correlação entre dias e número de telespectadores

correlation = df['Dias'].corr(df['Telespectadores'])

# Linha de tendência (regressão linear)

slope, intercept, r\_value, p\_value, std\_err = stats.linregress(

    df['Dias'], df['Telespectadores']

)

print("d. Análise da tendência de audiência ao longo da temporada 2011-2012:")

print(f"   Coeficiente de correlação: {correlation:.4f}")

print(f"   Inclinação da linha de tendência: {slope:.4f} milhões por dia")

if slope > 0:

    print("   A audiência teve uma tendência de AUMENTO ao longo da temporada.")

else:

    print("   A audiência teve uma tendência de DIMINUIÇÃO ao longo da temporada.")

if abs(correlation) < 0.3:

    print("   A correlação é fraca, indicando que a tendência não é muito significativa.")

elif abs(correlation) < 0.7:

    print("   A correlação é moderada.")

else:

    print("   A correlação é forte, indicando uma tendência clara.")

# Primeira metade vs. segunda metade da temporada

half\_point = len(df) // 2

first\_half\_avg = df.iloc[:half\_point]['Telespectadores'].mean()

second\_half\_avg = df.iloc[half\_point:]['Telespectadores'].mean()

print(f"   Média da primeira metade da temporada: {first\_half\_avg:.2f} milhões")

print(f"   Média da segunda metade da temporada: {second\_half\_avg:.2f} milhões")

if second\_half\_avg > first\_half\_avg:

    print(f"   Aumento de {second\_half\_avg - first\_half\_avg:.2f} milhões na segunda metade.")

else:

    print(f"   Diminuição de {first\_half\_avg - second\_half\_avg:.2f} milhões na segunda metade.")

# Visualização da tendência

plt.figure(figsize=(12, 6))

plt.plot(df['Data'], df['Telespectadores'], marker='o', linestyle='-', color='b')

plt.title('Audiência de The Big Bang Theory - Temporada 2011-2012')

plt.xlabel('Data de Transmissão')

plt.ylabel('Telespectadores (milhões)')

plt.grid(True, linestyle='--', alpha=0.7)

# Adicionando linha de tendência

plt.plot(df['Data'], intercept + slope \* df['Dias'], 'r--',

         label=f'Tendência (inclinação={slope:.4f})')

plt.legend()

plt.tight\_layout()

plt.show()

# Boxplot para análise estatística

plt.figure(figsize=(8, 6))

plt.boxplot(df['Telespectadores'], vert=False)

plt.title('Distribuição da Audiência')

plt.xlabel('Telespectadores (milhões)')

plt.grid(True, linestyle='--', alpha=0.7)

plt.show()

## Cap 3 - 29

import numpy as np

import matplotlib.pyplot as plt

# Dados de Pomona

pomona\_dados = np.array([28, 42, 58, 48, 45, 55, 60, 49, 50])

# a) Calcular a amplitude e a amplitude interquartil

pomona\_min = np.min(pomona\_dados)

pomona\_max = np.max(pomona\_dados)

pomona\_amplitude = pomona\_max - pomona\_min

# Amplitude interquartil (IQR)

pomona\_q1 = np.percentile(pomona\_dados, 25)

pomona\_q3 = np.percentile(pomona\_dados, 75)

pomona\_iqr = pomona\_q3 - pomona\_q1

# b) Calcular a variância amostral e o desvio padrão da amostra

pomona\_variancia = np.var(pomona\_dados, ddof=1)  # ddof=1 para variância amostral

pomona\_desvio\_padrao = np.std(pomona\_dados, ddof=1)

# c) Comparações entre Pomona e Anaheim

# Dados de Anaheim

anaheim\_media = 48.5

anaheim\_variancia = 136

anaheim\_desvio\_padrao = 11.66

# Calcular a média de Pomona para comparação

pomona\_media = np.mean(pomona\_dados)

# Função para visualizar as comparações

def visualizar\_comparacoes():

    # Criar um gráfico de barras para comparar as estatísticas

    labels = ['Média', 'Variância', 'Desvio Padrão']

    pomona\_stats = [pomona\_media, pomona\_variancia, pomona\_desvio\_padrao]

    anaheim\_stats = [anaheim\_media, anaheim\_variancia, anaheim\_desvio\_padrao]

    x = np.arange(len(labels))

    width = 0.35

    fig, ax = plt.subplots(figsize=(10, 6))

    rects1 = ax.bar(x - width/2, pomona\_stats, width, label='Pomona')

    rects2 = ax.bar(x + width/2, anaheim\_stats, width, label='Anaheim')

    ax.set\_ylabel('Valores')

    ax.set\_title('Comparação das Estatísticas de Qualidade do Ar')

    ax.set\_xticks(x)

    ax.set\_xticklabels(labels)

    ax.legend()

    # Adicionar os valores nas barras

    def autolabel(rects):

        for rect in rects:

            height = rect.get\_height()

            ax.annotate(f'{height:.2f}',

                        xy=(rect.get\_x() + rect.get\_width() / 2, height),

                        xytext=(0, 3),

                        textcoords="offset points",

                        ha='center', va='bottom')

    autolabel(rects1)

    autolabel(rects2)

    fig.tight\_layout()

    # Também podemos realizar um boxplot para comparação da distribuição

    fig2, ax2 = plt.subplots(figsize=(8, 6))

    ax2.boxplot([pomona\_dados], labels=['Pomona'])

    ax2.set\_title('Boxplot da Qualidade do Ar em Pomona')

    ax2.set\_ylabel('Índice de Qualidade do Ar')

    plt.show()

# Exibir os resultados

print("\nResultados para Pomona:")

print(f"a) Amplitude: {pomona\_amplitude}")

print(f"   Amplitude Interquartil (IQR): {pomona\_iqr:.2f}")

print(f"b) Variância Amostral: {pomona\_variancia:.2f}")

print(f"   Desvio Padrão Amostral: {pomona\_desvio\_padrao:.2f}")

print("\nComparação entre Pomona e Anaheim:")

print(f"Média: Pomona = {pomona\_media:.2f}, Anaheim = {anaheim\_media}")

print(f"Variância: Pomona = {pomona\_variancia:.2f}, Anaheim = {anaheim\_variancia}")

print(f"Desvio Padrão: Pomona = {pomona\_desvio\_padrao:.2f}, Anaheim = {anaheim\_desvio\_padrao}")

print("\nInterpretação dos resultados:")

if pomona\_media > anaheim\_media:

    print(f"- A qualidade do ar em Pomona tem um índice médio mais alto ({pomona\_media:.2f}) do que em Anaheim ({anaheim\_media}).")

elif pomona\_media < anaheim\_media:

    print(f"- A qualidade do ar em Pomona tem um índice médio mais baixo ({pomona\_media:.2f}) do que em Anaheim ({anaheim\_media}).")

else:

    print(f"- A qualidade do ar em Pomona e Anaheim têm índices médios semelhantes ({pomona\_media:.2f}).")

if pomona\_variancia > anaheim\_variancia:

    print(f"- Os dados de Pomona apresentam maior variabilidade (variância = {pomona\_variancia:.2f}) do que os de Anaheim (variância = {anaheim\_variancia}).")

elif pomona\_variancia < anaheim\_variancia:

    print(f"- Os dados de Pomona apresentam menor variabilidade (variância = {pomona\_variancia:.2f}) do que os de Anaheim (variância = {anaheim\_variancia}).")

else:

    print(f"- Os dados de Pomona e Anaheim apresentam variabilidade semelhante.")

if pomona\_desvio\_padrao > anaheim\_desvio\_padrao:

    print(f"- Os dados de Pomona apresentam maior dispersão (desvio padrão = {pomona\_desvio\_padrao:.2f}) do que os de Anaheim (desvio padrão = {anaheim\_desvio\_padrao}).")

elif pomona\_desvio\_padrao < anaheim\_desvio\_padrao:

    print(f"- Os dados de Pomona apresentam menor dispersão (desvio padrão = {pomona\_desvio\_padrao:.2f}) do que os de Anaheim (desvio padrão = {anaheim\_desvio\_padrao}).")

else:

    print(f"- Os dados de Pomona e Anaheim apresentam dispersão semelhante.")

## Cap 3 - 34

import numpy as np

import matplotlib.pyplot as plt

# Dados dos tempos dos corredores (em minutos)

tempos\_quarto\_milha = np.array([0.92, 0.98, 1.04, 0.90, 0.99])

tempos\_milha = np.array([4.52, 4.35, 4.60, 4.70, 4.50])

# Calcular estatísticas para as duas amostras

# Para os tempos de um quarto de milha

media\_quarto = np.mean(tempos\_quarto\_milha)

dp\_quarto = np.std(tempos\_quarto\_milha, ddof=1)  # desvio padrão amostral (ddof=1)

cv\_quarto = (dp\_quarto / media\_quarto) \* 100  # coeficiente de variação em percentual

# Para os tempos de uma milha

media\_milha = np.mean(tempos\_milha)

dp\_milha = np.std(tempos\_milha, ddof=1)  # desvio padrão amostral (ddof=1)

cv\_milha = (dp\_milha / media\_milha) \* 100  # coeficiente de variação em percentual

# Função para criar visualizações

def visualizar\_resultados():

    # Criar um gráfico para comparar os coeficientes de variação

    categorias = ['Quarto de Milha', 'Milha Completa']

    coeficientes = [cv\_quarto, cv\_milha]

    plt.figure(figsize=(10, 6))

    plt.bar(categorias, coeficientes, color=['blue', 'green'])

    plt.title('Coeficiente de Variação por Tipo de Prova')

    plt.ylabel('Coeficiente de Variação (%)')

    plt.grid(axis='y', linestyle='--', alpha=0.7)

    for i, v in enumerate(coeficientes):

        plt.text(i, v + 0.3, f'{v:.2f}%', ha='center')

    # Gráficos de dispersão para visualizar a variabilidade dos tempos

    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)

    plt.scatter(range(1, len(tempos\_quarto\_milha) + 1), tempos\_quarto\_milha, color='blue')

    plt.axhline(y=media\_quarto, color='red', linestyle='--', label=f'Média: {media\_quarto:.2f}')

    plt.fill\_between(range(1, len(tempos\_quarto\_milha) + 1),

                     media\_quarto - dp\_quarto,

                     media\_quarto + dp\_quarto,

                     alpha=0.2, color='red', label=f'DP: ±{dp\_quarto:.3f}')

    plt.title('Tempos da Prova de Quarto de Milha')

    plt.xlabel('Corredor')

    plt.ylabel('Tempo (minutos)')

    plt.grid(True, linestyle='--', alpha=0.7)

    plt.legend()

    plt.subplot(1, 2, 2)

    plt.scatter(range(1, len(tempos\_milha) + 1), tempos\_milha, color='green')

    plt.axhline(y=media\_milha, color='red', linestyle='--', label=f'Média: {media\_milha:.2f}')

    plt.fill\_between(range(1, len(tempos\_milha) + 1),

                     media\_milha - dp\_milha,

                     media\_milha + dp\_milha,

                     alpha=0.2, color='red', label=f'DP: ±{dp\_milha:.3f}')

    plt.title('Tempos da Prova de Uma Milha')

    plt.xlabel('Corredor')

    plt.ylabel('Tempo (minutos)')

    plt.grid(True, linestyle='--', alpha=0.7)

    plt.legend()

    plt.tight\_layout()

    plt.show()

# Exibir resultados

print("\nAnálise dos Tempos de Corrida - Equipe Universitária")

print("-" \* 50)

print("\nProva de Um Quarto de Milha:")

print(f"Tempos (minutos): {', '.join([str(t) for t in tempos\_quarto\_milha])}")

print(f"Média: {media\_quarto:.3f} minutos")

print(f"Desvio Padrão: {dp\_quarto:.3f} minutos")

print(f"Coeficiente de Variação: {cv\_quarto:.2f}%")

print("\nProva de Uma Milha:")

print(f"Tempos (minutos): {', '.join([str(t) for t in tempos\_milha])}")

print(f"Média: {media\_milha:.3f} minutos")

print(f"Desvio Padrão: {dp\_milha:.3f} minutos")

print(f"Coeficiente de Variação: {cv\_milha:.2f}%")

print("\nAnálise Comparativa:")

if cv\_quarto < cv\_milha:

    print(f"O coeficiente de variação da prova de um quarto de milha ({cv\_quarto:.2f}%) é menor que o da prova de uma milha ({cv\_milha:.2f}%).")

    print("Isso indica que os tempos na prova de um quarto de milha são mais consistentes, o que contradiz a declaração do treinador.")

    print("A declaração do técnico não se sustenta com base nessa análise estatística.")

elif cv\_quarto > cv\_milha:

    print(f"O coeficiente de variação da prova de um quarto de milha ({cv\_quarto:.2f}%) é maior que o da prova de uma milha ({cv\_milha:.2f}%).")

    print("Isso indica que os tempos na prova de uma milha são mais consistentes, o que confirma a declaração do treinador.")

    print("A declaração do técnico é suportada pela análise estatística.")

else:

    print(f"Os coeficientes de variação são iguais ({cv\_quarto:.2f}%).")

    print("Não há diferença de consistência entre as duas provas com base nessa análise.")

print("\nInterpretação do Coeficiente de Variação:")

print("O coeficiente de variação (CV) expressa a variabilidade dos dados em relação à média.")

print("Quanto menor o CV, maior a consistência dos dados (menor variabilidade relativa).")

print("Um CV menor indica tempos mais homogêneos entre os corredores.")

## Cap 3 - 36

import numpy as np

import matplotlib.pyplot as plt

from scipy import stats

# Parâmetros da amostra

media = 500

desvio\_padrao = 100

# Valores para calcular os escores-z

valores = np.array([520, 650, 500, 450, 280])

# Calcular os escores-z para cada valor

escores\_z = (valores - media) / desvio\_padrao

# Criar uma função para visualizar os resultados

def visualizar\_distribuicao():

    # Criar um gráfico da distribuição normal com os pontos marcados

    x = np.linspace(media - 4\*desvio\_padrao, media + 4\*desvio\_padrao, 1000)

    y = stats.norm.pdf(x, media, desvio\_padrao)

    plt.figure(figsize=(12, 6))

    plt.plot(x, y, 'b-', linewidth=2, label='Distribuição Normal')

    # Marcar a média

    plt.axvline(x=media, color='r', linestyle='--', label=f'Média = {media}')

    # Marcar cada valor e seu escore-z

    colors = ['green', 'purple', 'orange', 'brown', 'magenta']

    for i, (valor, z, color) in enumerate(zip(valores, escores\_z, colors)):

        plt.axvline(x=valor, color=color, linestyle='-',

                   label=f'Valor = {valor}, z = {z:.2f}')

    plt.title('Distribuição Normal com Escores-z')

    plt.xlabel('Valores')

    plt.ylabel('Densidade de Probabilidade')

    plt.legend(loc='best')

    plt.grid(True)

    # Gráfico de barras dos escores-z

    plt.figure(figsize=(10, 6))

    bars = plt.bar(range(len(valores)), escores\_z, color=colors)

    # Adicionar rótulos às barras

    for i, bar in enumerate(bars):

        height = bar.get\_height()

        if height < 0:

            va = 'top'

            offset = -0.3

        else:

            va = 'bottom'

            offset = 0.3

        plt.text(bar.get\_x() + bar.get\_width()/2., height + offset,

                f'z = {escores\_z[i]:.2f}\n({valores[i]})',

                ha='center', va=va)

    plt.axhline(y=0, color='black', linestyle='-')

    plt.axhline(y=1, color='green', linestyle='--', alpha=0.7, label='z = +1')

    plt.axhline(y=-1, color='green', linestyle='--', alpha=0.7, label='z = -1')

    plt.axhline(y=2, color='orange', linestyle='--', alpha=0.7, label='z = +2')

    plt.axhline(y=-2, color='orange', linestyle='--', alpha=0.7, label='z = -2')

    plt.title('Escores-z para os Valores Fornecidos')

    plt.xlabel('Valores da Amostra')

    plt.ylabel('Escore-z')

    plt.xticks(range(len(valores)), [str(v) for v in valores])

    plt.grid(True, axis='y')

    plt.legend()

    plt.tight\_layout()

    plt.show()

# Exibir os resultados

print("\nCálculo de Escores-z com Média = 500 e Desvio Padrão = 100")

print("-" \* 60)

print(f"Fórmula do Escore-z: z = (x - μ) / σ, onde:")

print(f"  - x é o valor a ser padronizado")

print(f"  - μ é a média da distribuição (500)")

print(f"  - σ é o desvio padrão da distribuição (100)")

print("-" \* 60)

# Criar uma tabela com os resultados

print("\nResultados:")

print(f"{'Valor':<10} | {'Cálculo':<20} | {'Escore-z':<10} | {'Interpretação':<40}")

print("-" \* 85)

for valor, z in zip(valores, escores\_z):

    calculo = f"({valor} - 500) / 100"

    # Interpretação do escore-z

    if abs(z) < 1:

        interpretacao = "Próximo à média (menos de 1 DP de distância)"

    elif abs(z) < 2:

        interpretacao = "Moderadamente distante da média (entre 1 e 2 DP)"

    elif abs(z) < 3:

        interpretacao = "Consideravelmente distante da média (entre 2 e 3 DP)"

    else:

        interpretacao = "Extremamente distante da média (mais de 3 DP)"

    if z > 0:

        interpretacao += ", acima da média"

    elif z < 0:

        interpretacao += ", abaixo da média"

    print(f"{valor:<10} | {calculo:<20} | {z:.2f}{' ':<6} | {interpretacao:<40}")

# Informações adicionais sobre escores-z

print("\nObservações:")

print("- O escore-z representa quantos desvios padrão um valor está acima ou abaixo da média")

print("- Valores positivos estão acima da média, valores negativos estão abaixo")

print("- Na distribuição normal:")

print("  \* Aproximadamente 68% dos valores têm escores-z entre -1 e +1")

print("  \* Aproximadamente 95% dos valores têm escores-z entre -2 e +2")

print("  \* Aproximadamente 99.7% dos valores têm escores-z entre -3 e +3")

## Cap 3 - 44

import numpy as np

import scipy.stats as stats

# Dados fornecidos na tabela

pontos\_vencedor = [90, 85, 75, 78, 71, 65, 72, 76, 77, 82]

pontos\_perdedor = [66, 66, 70, 57, 63, 62, 66, 70, 67, 56]

margem\_pontos = [24, 19, 5, 21, 8, 3, 6, 6, 10, 26]

# a. Calcule a média e o desvio padrão para os pontos marcados pelo time vencedor

media\_pontos\_vencedor = np.mean(pontos\_vencedor)

desvio\_padrao\_pontos\_vencedor = np.std(pontos\_vencedor, ddof=1)  # ddof=1 para desvio padrão amostral

print(f"a. Pontos do time vencedor:")

print(f"   Média: {media\_pontos\_vencedor:.2f}")

print(f"   Desvio padrão: {desvio\_padrao\_pontos\_vencedor:.2f}")

# b. Suponha distribuição normal, estimando a porcentagem de jogos onde o vencedor marca mais de 90 pontos

# Usando a distribuição normal com a média e desvio padrão calculados

z\_score = (90 - media\_pontos\_vencedor) / desvio\_padrao\_pontos\_vencedor

probabilidade = 1 - stats.norm.cdf(z\_score)

porcentagem = probabilidade \* 100

print(f"\nb. Porcentagem de jogos em que o time vencedor marca mais de 90 pontos:")

print(f"   {porcentagem:.2f}%")

# c. Calcule a média e o desvio padrão para a margem de pontos do time vitorioso

media\_margem = np.mean(margem\_pontos)

desvio\_padrao\_margem = np.std(margem\_pontos, ddof=1)

print(f"\nc. Margem de pontos do time vitorioso:")

print(f"   Média: {media\_margem:.2f}")

print(f"   Desvio padrão: {desvio\_padrao\_margem:.2f}")

## Cap 3 - 48

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Dados fornecidos

dados = [5, 15, 18, 10, 8, 12, 16, 10, 6]

# Ordenando os dados

dados\_ordenados = sorted(dados)

print("Dados ordenados:", dados\_ordenados)

# Cálculo do resumo de cinco números

minimo = min(dados)

maximo = max(dados)

mediana = np.median(dados)

q1 = np.percentile(dados, 25)

q3 = np.percentile(dados, 75)

iqr = q3 - q1

# Exibindo o resumo de cinco números

print("\nResumo de cinco números (Five-number summary):")

print(f"Mínimo: {minimo}")

print(f"Q1 (Primeiro quartil): {q1}")

print(f"Mediana: {mediana}")

print(f"Q3 (Terceiro quartil): {q3}")

print(f"Máximo: {maximo}")

print(f"IQR (Intervalo interquartil): {iqr}")

# Verificando outliers

limite\_inferior = q1 - 1.5 \* iqr

limite\_superior = q3 + 1.5 \* iqr

print(f"\nLimite inferior para outliers: {limite\_inferior}")

print(f"Limite superior para outliers: {limite\_superior}")

outliers = [x for x in dados if x < limite\_inferior or x > limite\_superior]

print(f"Outliers (se houver): {outliers}")

# Criando o boxplot

plt.figure(figsize=(10, 6))

sns.boxplot(x=dados)

plt.title('Boxplot dos dados')

plt.xlabel('Valores')

plt.grid(True, linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()

## Cap 3 - 56

import numpy as np

import matplotlib.pyplot as plt

from scipy import stats

import seaborn as sns

# Dados fornecidos

x = [6, 11, 15, 21, 27]

y = [6, 9, 6, 17, 12]

# a. Construir um diagrama de dispersão

plt.figure(figsize=(10, 6))

plt.scatter(x, y, color='blue', s=100)

plt.title('Diagrama de Dispersão')

plt.xlabel('Variável x')

plt.ylabel('Variável y')

plt.grid(True, linestyle='--', alpha=0.7)

# Adicionar labels aos pontos

for i, (xi, yi) in enumerate(zip(x, y)):

    plt.annotate(f'({xi}, {yi})', (xi, yi), xytext=(5, 5), textcoords='offset points')

# Adicionar linha de tendência

z = np.polyfit(x, y, 1)

p = np.poly1d(z)

plt.plot(x, p(x), "r--", alpha=0.8, label=f'Linha de tendência: y = {z[0]:.4f}x + {z[1]:.4f}')

plt.legend()

plt.tight\_layout()

plt.show()

# c. Calcular a covariância amostral

covariancia = np.cov(x, y, ddof=1)[0, 1]  # ddof=1 para covariância amostral

print(f"c. Covariância amostral: {covariancia:.4f}")

# d. Calcular e interpretar o coeficiente de correlação amostral

correlacao, p\_valor = stats.pearsonr(x, y)

print(f"d. Coeficiente de correlação amostral (r): {correlacao:.4f}")

print(f"   p-valor: {p\_valor:.4f}")

# Interpretação do coeficiente de correlação

if abs(correlacao) < 0.3:

    interpretacao = "fraca"

elif abs(correlacao) < 0.7:

    interpretacao = "moderada"

else:

    interpretacao = "forte"

if correlacao > 0:

    direcao = "positiva"

else:

    direcao = "negativa"

print(f"   Interpretação: Existe uma correlação {interpretacao} e {direcao} entre as variáveis x e y.")

# Estatísticas descritivas adicionais

print("\nEstatísticas descritivas:")

print(f"Média de x: {np.mean(x):.4f}")

print(f"Média de y: {np.mean(y):.4f}")

print(f"Desvio padrão de x: {np.std(x, ddof=1):.4f}")

print(f"Desvio padrão de y: {np.std(y, ddof=1):.4f}")

# b. Análise do diagrama de dispersão

print("\nb. O diagrama de dispersão indica:")

if correlacao > 0.5:

    print("- Uma tendência de crescimento de y conforme x aumenta")

elif correlacao < -0.5:

    print("- Uma tendência de decrescimento de y conforme x aumenta")

else:

    print("- Uma relação não muito clara entre as variáveis")

if abs(correlacao) > 0.7:

    print("- Uma forte associação linear entre as variáveis")

elif abs(correlacao) > 0.3:

    print("- Uma associação linear moderada entre as variáveis")

else:

    print("- Uma associação linear fraca entre as variáveis")

## Cap 3 - 59

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

# Dados fornecidos

percentagem\_detectores = [0.50, 0.67, 0.74, 0.76, 0.77, 0.82, 0.81, 0.85, 0.86, 0.88, 0.90, 0.92, 0.95, 0.96, 0.97, 0.96, 0.96]

mortes\_por\_milhao = [22.9, 20.8, 17.3, 20.5, 19.4, 18.9, 17.6, 16.2, 13.6, 14.4, 14.3, 13.0, 10.9, 10.9, 10.2, 8.4, 8.1]

# Convertendo para percentuais (0-100) para melhor visualização

percentagem\_detectores\_viz = [p \* 100 for p in percentagem\_detectores]

# Criar DataFrame

df = pd.DataFrame({

    'Percentual\_detectores': percentagem\_detectores,

    'Percentual\_detectores\_viz': percentagem\_detectores\_viz,

    'Mortes\_por\_milhao': mortes\_por\_milhao

})

# a. Relação entre uso de detectores e mortes

correlacao, p\_valor = stats.pearsonr(percentagem\_detectores, mortes\_por\_milhao)

print(f"a. Relação entre uso de detectores de fumaça e mortes:")

print(f"   Coeficiente de correlação: {correlacao:.4f}")

print(f"   p-valor: {p\_valor:.8f}")

if correlacao < 0:

    print("   Existe uma relação NEGATIVA entre o uso de detectores de fumaça e mortes por incêndio.")

    print("   Isso significa que quanto maior a porcentagem de residências com detectores, menor é a taxa de mortalidade.")

else:

    print("   Existe uma relação POSITIVA entre o uso de detectores de fumaça e mortes por incêndio.")

    print("   Isso significa que quanto maior a porcentagem de residências com detectores, maior é a taxa de mortalidade.")

# b. Cálculo do coeficiente de correlação

print(f"\nb. Coeficiente de correlação: {correlacao:.4f}")

if abs(correlacao) < 0.3:

    forca = "fraca"

elif abs(correlacao) < 0.7:

    forca = "moderada"

else:

    forca = "forte"

print(f"   Existe uma correlação {forca} e negativa entre o uso de detectores de fumaça e mortes por incêndios.")

print(f"   Isso sugere que o aumento na adoção de detectores de fumaça está associado a uma diminuição nas mortes por incêndio.")

print(f"   O valor p de {p\_valor:.8f} indica que essa correlação é estatisticamente significativa.")

# c. Gráfico de dispersão

plt.figure(figsize=(10, 6))

sns.regplot(x='Percentual\_detectores\_viz', y='Mortes\_por\_milhao', data=df,

            scatter\_kws={'s': 80, 'alpha': 0.7}, line\_kws={'color': 'red'})

plt.title('Relação entre Percentual de Residências com Detectores de Fumaça e Mortes por Incêndio')

plt.xlabel('Percentual de Residências com Detectores de Fumaça (%)')

plt.ylabel('Mortes por Incêndio por Milhão de Habitantes')

plt.grid(True, linestyle='--', alpha=0.7)

# Adicionar equação da linha de tendência

slope, intercept, r\_value, p\_value, std\_err = stats.linregress(percentagem\_detectores\_viz, mortes\_por\_milhao)

equation = f'y = {slope:.4f}x + {intercept:.4f}'

plt.annotate(f'Equação: {equation}\nR² = {r\_value\*\*2:.4f}',

             xy=(0.05, 0.95), xycoords='axes fraction',

             bbox=dict(boxstyle="round,pad=0.3", fc="white", ec="gray", alpha=0.8))

plt.tight\_layout()

plt.show()