

✓ <https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/UJ1HDR>

Comece a programar ou [gere código](#) com IA.

Análise Exploratória de Dados sobre "Learning Burnout" Vou criar um código em Python para abrir, tratar e explorar os dados sobre "learning burnout". Primeiro, precisamos carregar o arquivo .sav (que é um formato do SPSS) e depois realizar algumas análises exploratórias.

```
# Instalar o pacote necessário
!pip install pyreadstat
```

```
import pandas as pd
import pyreadstat
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
# Restante do código como fornecido anteriormente
```

```
Collecting pyreadstat
  Downloading pyreadstat-1.2.9-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.3 kB)
Requirement already satisfied: pandas>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from pyreadstat) (2.2.2)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2.0->pyreadstat) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2.0->pyreadstat) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2.0->pyreadstat) (2022.7)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2.0->pyreadstat) (2022.7)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.2.0->pyreadstat) (1.16.0)
Downloading pyreadstat-1.2.9-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (617 kB)
617.7/617.7 kB 9.6 MB/s eta 0:00:00
Installing collected packages: pyreadstat
Successfully installed pyreadstat-1.2.9
```

✓ Tratamento dos Dados criando Dicionarios

```
import pandas as pd
import numpy as np
```

```
# Carregar os dados (substitua pelo seu método de carregamento)
df = pd.read_spss('dados.sav')
```

```
# Ver colunas originais
print("Colunas originais:")
print(df.columns.tolist())
```

```
# Criar cópia para tratamento
df_tratado = df.copy()
```

```
# Mapeamento de valores para categorias
mapeamentos = {
    'Q1': {1.0: 'Masculino', 2.0: 'Feminino'},
    'Q2': {1.0: '7ª série', 2.0: '8ª série', 3.0: '9ª série'},
    'Q3': {1.0: 'Urbano', 2.0: 'Rural'},
    'Q4': {1.0: 'Sim', 2.0: 'Não'},
    'Q5.1': {
        1.0: 'Fundamental',
        2.0: 'Médio',
        3.0: 'Superior incompleto',
        4.0: 'Superior completo',
        5.0: 'Pós-graduação'
    },
    'Q5.2': {
        1.0: 'Fundamental',
        2.0: 'Médio',
        3.0: 'Superior incompleto',
        4.0: 'Superior completo',
        5.0: 'Pós-graduação'
    },
    'Q6': {
        1.0: '<3000',
        2.0: '3000-5000',
    },
}
```

```
3.0: '5000-10000',
4.0: '>10000'
}
}

# Aplicar mapeamentos
for col, mapping in mapeamentos.items():
    if col in df_tratado.columns:
        df_tratado[col] = df_tratado[col].map(mapping)

# Salvar dados tratados para uso nos próximos scripts
df_tratado.to_pickle('dados_tratados.pkl') # Formato que preserva os tipos de dados

print("\nDados tratados salvos em 'dados_tratados.pkl'")
print("Amostra dos dados tratados:")
display(df_tratado.head())
```



Colunas originais:
['生日', 'Q1', 'Q2', 'Q3', 'Q4', 'Q5.1', 'Q5.2', 'Q6', '自豪', '高兴', '希望', '满足', '平静', '放松', '焦虑', '羞愧', '恼火',

Dados tratados salvos em 'dados_tratados.pkl'
Amostra dos dados tratados:

	生日	Q1	Q2	Q3	Q4	Q5.1	Q5.2	Q6	自豪	高兴	...	消极低唤醒	心理弹性	积极情绪	消极情
0	2009-01-	Masculino	7ª série	Rural	Não	Fundamental	Fundamental	<3000	2.2	4.000000	...	1.945455	3.658333	3.791667	2.2393
1	2009-10-	Masculino	7ª série	Urbano	Não	Médio	Superior incompleto	3000-5000	3.2	4.857143	...	1.050000	4.047222	3.901190	1.3488
2	2008-04-	Feminino	7ª série	Rural	Não	Fundamental	Fundamental	3000-5000	3.2	4.428571	...	1.638636	3.644444	3.954762	1.7859
3	2008-04-	Masculino	7ª série	Urbano	Não	Médio	Médio	<3000	4.8	5.000000	...	1.072727	4.452778	4.791667	1.6554
4	2009-03-	Feminino	7ª série	Urbano	Não	Médio	Pós-graduação	3000-5000	4.2	5.000000	...	1.700000	4.483333	4.433333	2.0880

5 rows × 45 columns

```
# Carregar o arquivo .sav
df, meta = pyreadstat.read_sav('/content/123年级day.sav')

# Mostrar as primeiras linhas
print("Primeiras linhas do dataset:")
display(df.head())

# Instala a fonte SimHei
!apt-get -qq install -y fonts-noto-cjk

import matplotlib.font_manager as fm
font_dirs = ['/usr/share/fonts/truetype/noto']
font_files = fm.findSystemFonts(fontpaths=font_dirs)
for font_file in font_files:
    fm.fontManager.addfont(font_file)
```

✓ Vou criar uma análise completa do dataset de learning burnout, incluindo tratamento de dados, visualizações e análises estatísticas relevantes.

Comece a programar ou [gere código](#) com IA.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy import stats

# 1. Primeiro, vamos verificar as colunas reais do seu dataframe
print("Colunas disponíveis no dataframe:")
print(df.columns.tolist())

# 2. Vamos criar um mapeamento entre os nomes em português e os nomes originais em chinês
```

```

# Baseado no seu dicionário de dados, vamos mapear algumas colunas principais:
column_mapping = {
    'Orgulho': '自豪',
    'Felicidade': '高兴',
    'Ansiedade': '焦虑',
    'Cansaço': '厌倦',
    'Resiliência': '心理弹性',
    'Gênero': 'Q1 - 1.性别',
    'Série': 'Q2 - 2.年级',
    'Origem': 'Q3 - 3.生源地'
}

# 3. Vamos verificar quais colunas do mapeamento existem no dataframe
available_columns = {}
for pt_name, original_name in column_mapping.items():
    if original_name in df.columns:
        available_columns[pt_name] = original_name
    else:
        print(f"Aviso: Coluna '{original_name}' ({pt_name}) não encontrada no dataframe")

print("\nColunas disponíveis para análise:")
print(available_columns)

# 4. Configuração de estilo
plt.style.use('ggplot')
sns.set_style("whitegrid")
%matplotlib inline

# 5. Análise Descritiva Básica
print("\n## Estatísticas Descritivas ##")
display(df[list(available_columns.values())].describe().transpose())

# 6. Visualização de Distribuições
plt.figure(figsize=(15, 10))

for i, (pt_name, original_name) in enumerate(available_columns.items(), 1):
    if pt_name in ['Orgulho', 'Felicidade', 'Ansiedade', 'Cansaço', 'Resiliência']:
        plt.subplot(2, 3, i)
        sns.histplot(df[original_name], bins=15, kde=True)
        plt.title(f'Distribuição de {pt_name}')
        plt.xlabel('Pontuação')

plt.tight_layout()
plt.show()

# 7. Análise por Gênero (se disponível)
if 'Gênero' in available_columns:
    gender_mapping = {1.0: 'Masculino', 2.0: 'Feminino'}
    df['gender_label'] = df[available_columns['Gênero']].map(gender_mapping)

    plt.figure(figsize=(10, 6))
    sns.boxplot(data=df, x='gender_label', y=available_columns['Cansaço'])
    plt.title('Nível de Cansaço (Burnout) por Gênero')
    plt.xlabel('Gênero')
    plt.ylabel('Nível de Cansaço')
    plt.show()

# 8. Correlações entre variáveis emocionais
emotion_vars = []
for pt_name in ['Felicidade', 'Ansiedade', 'Cansaço', 'Resiliência']:
    if pt_name in available_columns:
        emotion_vars.append(available_columns[pt_name])

if len(emotion_vars) >= 2:
    plt.figure(figsize=(10, 8))
    corr_matrix = df[emotion_vars].corr()
    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0,
                xticklabels=[available_columns.get(pt_name, pt_name) for pt_name in ['Felicidade', 'Ansiedade', 'Cansaço', 'Resi
                yticklabels=[available_columns.get(pt_name, pt_name) for pt_name in ['Felicidade', 'Ansiedade', 'Cansaço', 'Resi
    plt.title('Matriz de Correlação entre Variáveis Emocionais')
    plt.show()

# 9. Relação entre Resiliência e Cansaço (Burnout)
if 'Resiliência' in available_columns and 'Cansaço' in available_columns:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=df, x=available_columns['Resiliência'], y=available_columns['Cansaço'], alpha=0.6)
    plt.title('Relação entre Resiliência e Cansaço (Burnout)')
    plt.xlabel('Nível de Resiliência')

```

```
plt.ylabel('Nível de Cansaço/Burnout')
plt.show()

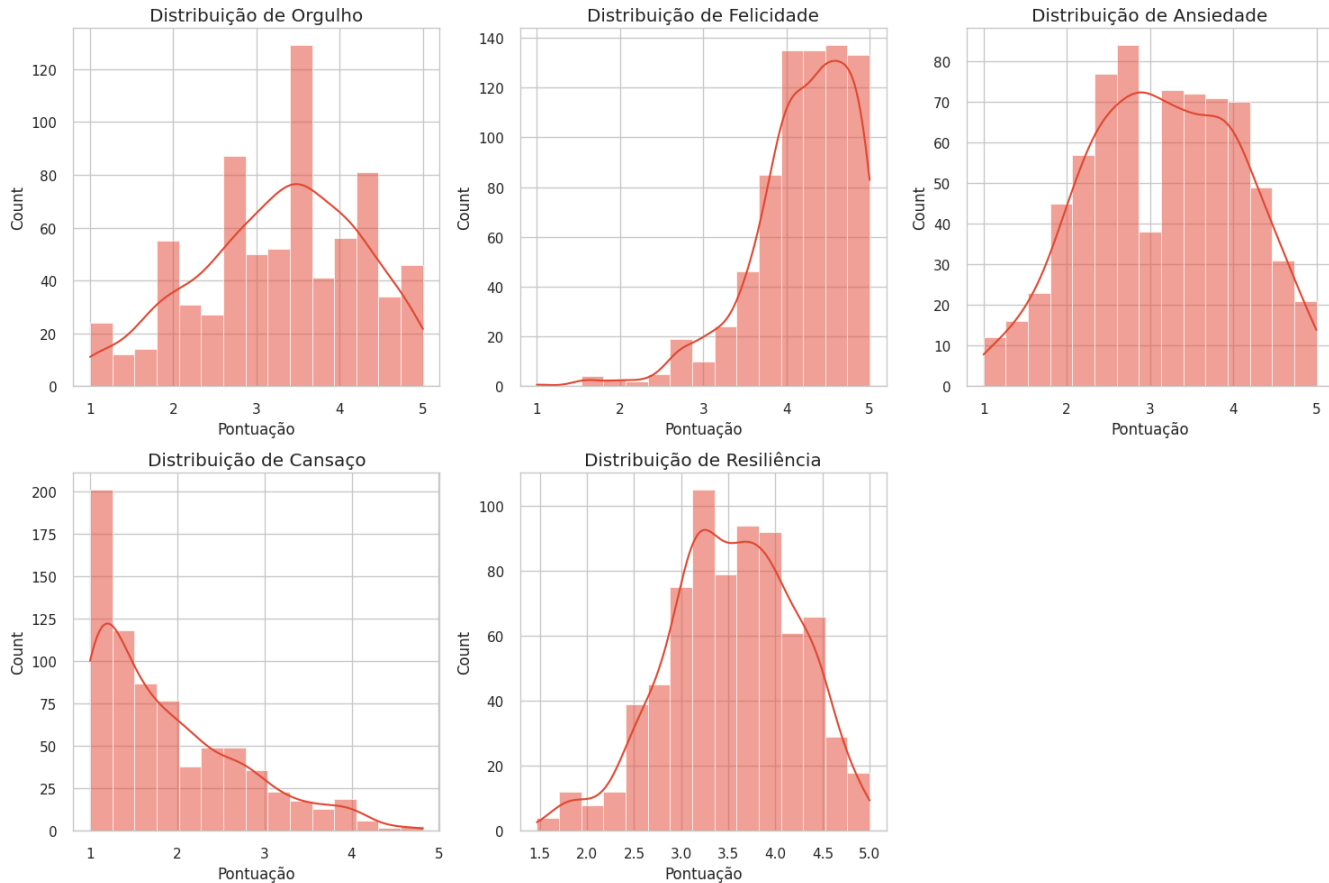
# Análise de regressão
print("\n## Análise de Regressão: Cansaço ~ Resiliência ##")
slope, intercept, r_value, p_value, std_err = stats.linregress(
    df[available_columns['Resiliência']].dropna(),
    df[available_columns['Cansaço']].dropna()
)
print(f"Coeficiente: {slope:.3f} (Para cada aumento unitário na resiliência, o cansaço muda em {slope:.3f})")
print(f"Intercepto: {intercept:.3f}")
print(f"R²: {r_value**2:.3f} ({r_value**2:.1%} da variação no cansaço é explicada pela resiliência)")
print(f"Valor p: {p_value:.4f}")
if p_value < 0.05:
    print("A relação é estatisticamente significativa (p < 0.05)")
else:
    print("A relação não é estatisticamente significativa")
```

Colunas disponíveis no dataframe:
['生日', 'Q1', 'Q2', 'Q3', 'Q4', 'Q5.1', 'Q5.2', 'Q6', '自豪', '高兴', '希望', '满足', '平静', '放松', '焦虑', '羞愧', '恼火',
Aviso: Coluna 'Q1 - 1.性别' (Gênero) não encontrada no dataframe
Aviso: Coluna 'Q2 - 2.年级' (Série) não encontrada no dataframe
Aviso: Coluna 'Q3 - 3.生源地' (Origem) não encontrada no dataframe

Colunas disponíveis para análise:
{'Orgulho': '自豪', 'Felicidade': '高兴', 'Ansiedade': '焦虑', 'Cansaço': '厌倦', 'Resiliência': '心理弹性'}

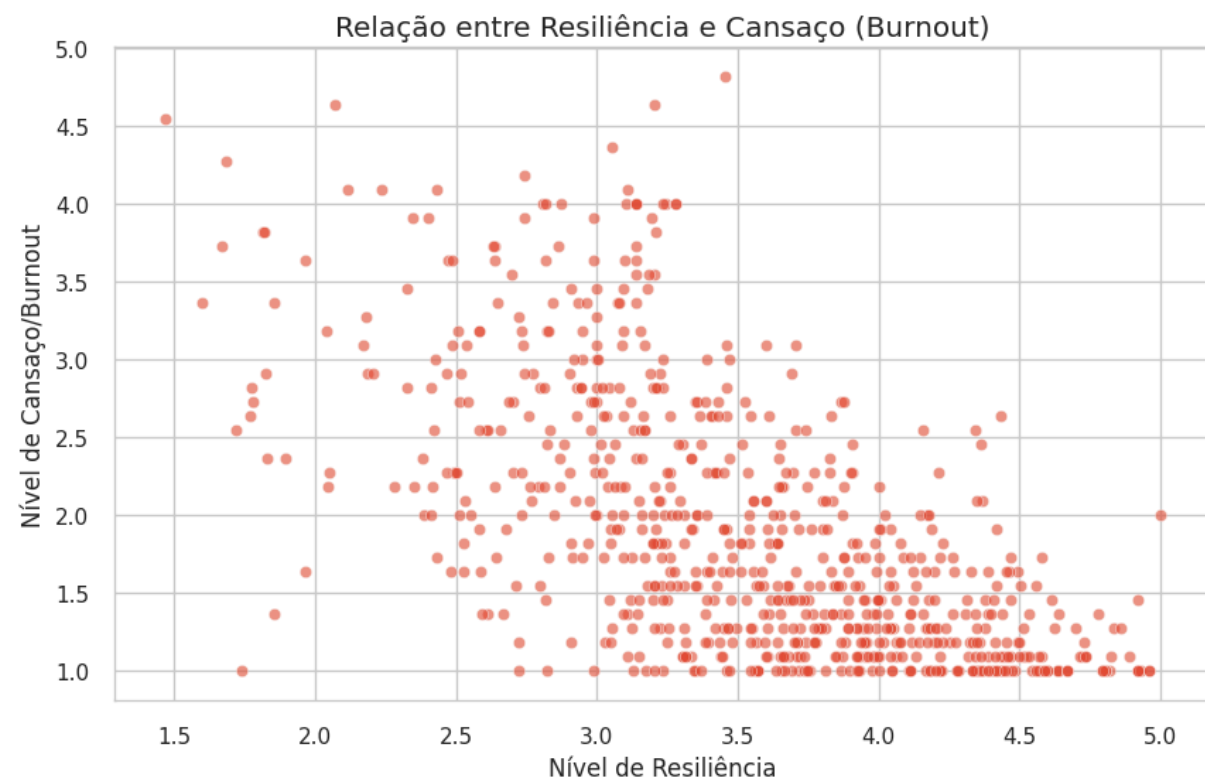
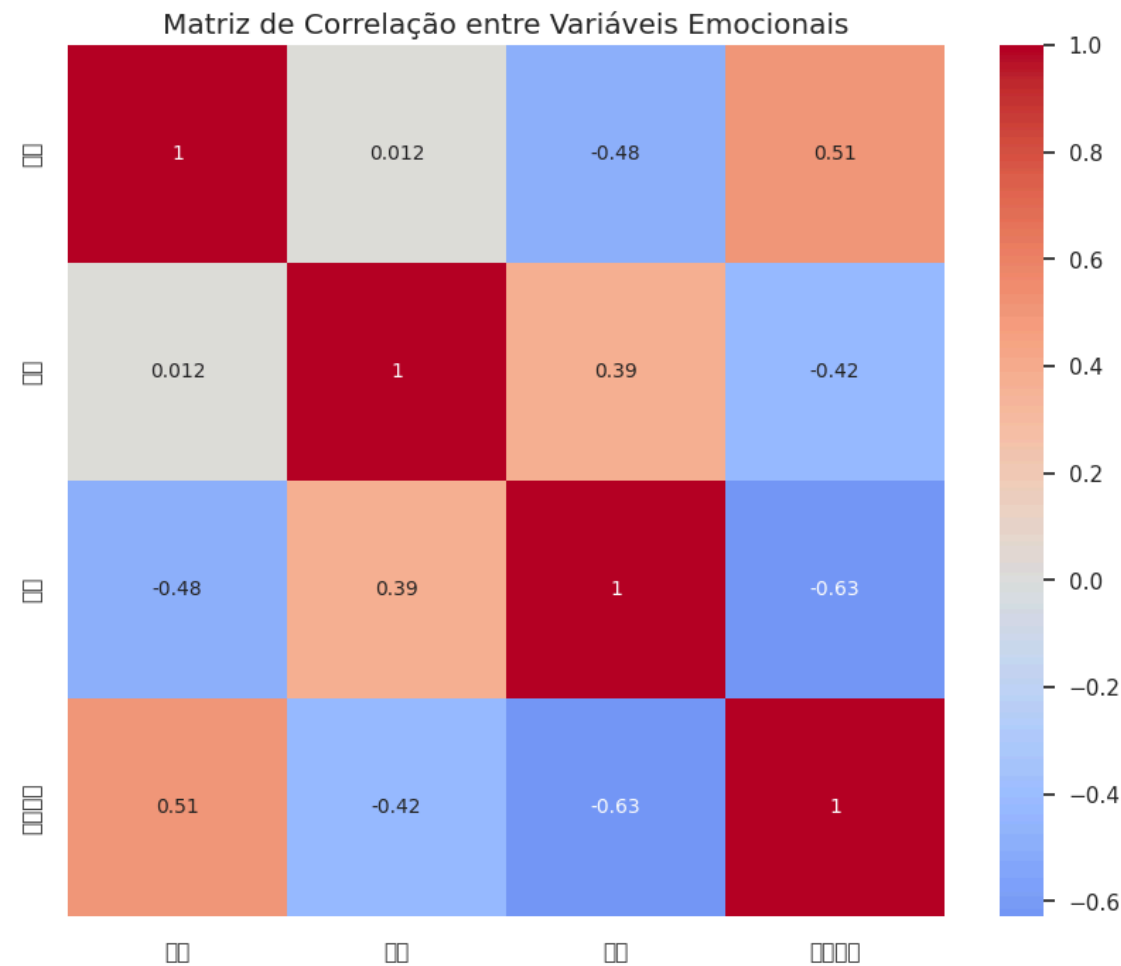
Estatísticas Descritivas

	count	mean	std	min	25%	50%	75%	max
自豪	739.0	3.264953	0.979853	1.000000	2.600000	3.400000	4.000000	5.000000
高兴	739.0	4.202590	0.629146	1.000000	3.857143	4.285714	4.714286	5.000000
焦虑	739.0	3.149236	0.919700	1.000000	2.428571	3.142857	3.857143	5.000000
厌倦	739.0	1.892238	0.840832	1.000000	1.181818	1.636364	2.363636	4.818182
心理弹性	739.0	3.532183	0.684268	1.466667	3.091667	3.552778	4.027778	5.000000



```
/usr/local/lib/python3.11/dist-packages/seaborn/utils.py:61: UserWarning: Glyph 39640 (\N{CJK UNIFIED IDEOGRAPH-9AD8}) missi
fig.canvas.draw()
/usr/local/lib/python3.11/dist-packages/seaborn/utils.py:61: UserWarning: Glyph 20852 (\N{CJK UNIFIED IDEOGRAPH-5174}) missi
fig.canvas.draw()
/usr/local/lib/python3.11/dist-packages/seaborn/utils.py:61: UserWarning: Glyph 28966 (\N{CJK UNIFIED IDEOGRAPH-7126}) missi
fig.canvas.draw()
/usr/local/lib/python3.11/dist-packages/seaborn/utils.py:61: UserWarning: Glyph 34385 (\N{CJK UNIFIED IDEOGRAPH-8651}) missi
fig.canvas.draw()
/usr/local/lib/python3.11/dist-packages/seaborn/utils.py:61: UserWarning: Glyph 21388 (\N{CJK UNIFIED IDEOGRAPH-538C}) missi
fig.canvas.draw()
/usr/local/lib/python3.11/dist-packages/seaborn/utils.py:61: UserWarning: Glyph 20518 (\N{CJK UNIFIED IDEOGRAPH-5026}) missi
fig.canvas.draw()
/usr/local/lib/python3.11/dist-packages/seaborn/utils.py:61: UserWarning: Glyph 24515 (\N{CJK UNIFIED IDEOGRAPH-5FC3}) missi
fig.canvas.draw()
/usr/local/lib/python3.11/dist-packages/seaborn/utils.py:61: UserWarning: Glyph 29702 (\N{CJK UNIFIED IDEOGRAPH-7406}) missi
fig.canvas.draw()
/usr/local/lib/python3.11/dist-packages/seaborn/utils.py:61: UserWarning: Glyph 24377 (\N{CJK UNIFIED IDEOGRAPH-5F39}) missi
fig.canvas.draw()
/usr/local/lib/python3.11/dist-packages/seaborn/utils.py:61: UserWarning: Glyph 24615 (\N{CJK UNIFIED IDEOGRAPH-6027}) missi
fig.canvas.draw()
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 39640 (\N{CJK UNIFIED IDEOGRAPH-9AD8}) missi
fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 20852 (\N{CJK UNIFIED IDEOGRAPH-5174}) missi
fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 28966 (\N{CJK UNIFIED IDEOGRAPH-7126}) missi
fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 34385 (\N{CJK UNIFIED IDEOGRAPH-8651}) missi
fig.canvas.print_figure(bytes_io, **kw)
```

```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 21388 (\N{CJK UNIFIED IDEOGRAPH-  
fig.canvas.print_figure(bytes_io, **kw)  
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 20518 (\N{CJK UNIFIED IDEOGRAPH-  
fig.canvas.print_figure(bytes_io, **kw)  
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 24515 (\N{CJK UNIFIED IDEOGRAPH-  
fig.canvas.print_figure(bytes_io, **kw)  
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 29702 (\N{CJK UNIFIED IDEOGRAPH-  
fig.canvas.print_figure(bytes_io, **kw)  
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 24377 (\N{CJK UNIFIED IDEOGRAPH-  
fig.canvas.print_figure(bytes_io, **kw)  
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 24615 (\N{CJK UNIFIED IDEOGRAPH-  
fig.canvas.print_figure(bytes_io, **kw)
```



Análise de Regressão: Cansaço ~ Resiliência

Coeficiente: -0.772 (Para cada aumento unitário na resiliência, o cansaço muda em -0.772)

Intercepto: 4.618

R²: 0.394 (39.4% da variação no cansaço é explicada pela resiliência)

Valor p: 0.0000

A relação é estatisticamente significativa ($p < 0.05$)

Clique duas vezes (ou pressione "Enter") para editar

✓ 1 Como estão distribuídos os alunos por características demográficas?

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Configurações
plt.style.use('ggplot')
sns.set_palette("pastel")
%matplotlib inline

# Carregar dados tratados
try:
    df = pd.read_pickle('dados_tratados.pkl')
    print("Dados carregados com sucesso!")
except:
    print("Erro ao carregar dados. Execute primeiro o Script 1 de tratamento.")
    raise

# 1. Função para plotar distribuições
def plot_distribuicao(coluna, titulo, ordem=None):
    plt.figure(figsize=(8, 5))
    sns.countplot(data=df, x=coluna, order=ordem)
    plt.title(titulo)
    plt.xlabel('')
    plt.ylabel('Contagem')
    plt.xticks(rotation=45 if len(df[coluna].unique()) > 3 else 0)
    plt.show()

# 2. Distribuições demográficas
plot_distribuicao('Q1', 'Distribuição por Gênero', ['Masculino', 'Feminino'])
plot_distribuicao('Q2', 'Distribuição por Série', ['7ª série', '8ª série', '9ª série'])
plot_distribuicao('Q3', 'Distribuição por Origem', ['Urbano', 'Rural'])
plot_distribuicao('Q4', 'Distribuição por Criança sob Tutela', ['Sim', 'Não'])

# 3. Educação dos pais
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.countplot(data=df, x='Q5.1', order=['Fundamental', 'Médio', 'Superior incompleto', 'Superior completo', 'Pós-graduação'])
plt.title('Educação das Mães')
plt.xticks(rotation=45)

plt.subplot(1, 2, 2)
sns.countplot(data=df, x='Q5.2', order=['Fundamental', 'Médio', 'Superior incompleto', 'Superior completo', 'Pós-graduação'])
plt.title('Educação dos Pais')
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()

# 4. Renda familiar
plot_distribuicao('Q6', 'Distribuição por Renda Familiar', ['<3000', '3000-5000', '5000-10000', '>10000'])

# 5. Tabelas resumo
print("\n=== Resumo Demográfico ===")
print("\nGênero:")
print(df['Q1'].value_counts(normalize=True))

print("\nSérie:")
print(df['Q2'].value_counts(normalize=True))

print("\nOrigem:")
print(df['Q3'].value_counts(normalize=True))

print("\nCriança sob tutela:")
print(df['Q4'].value_counts(normalize=True))

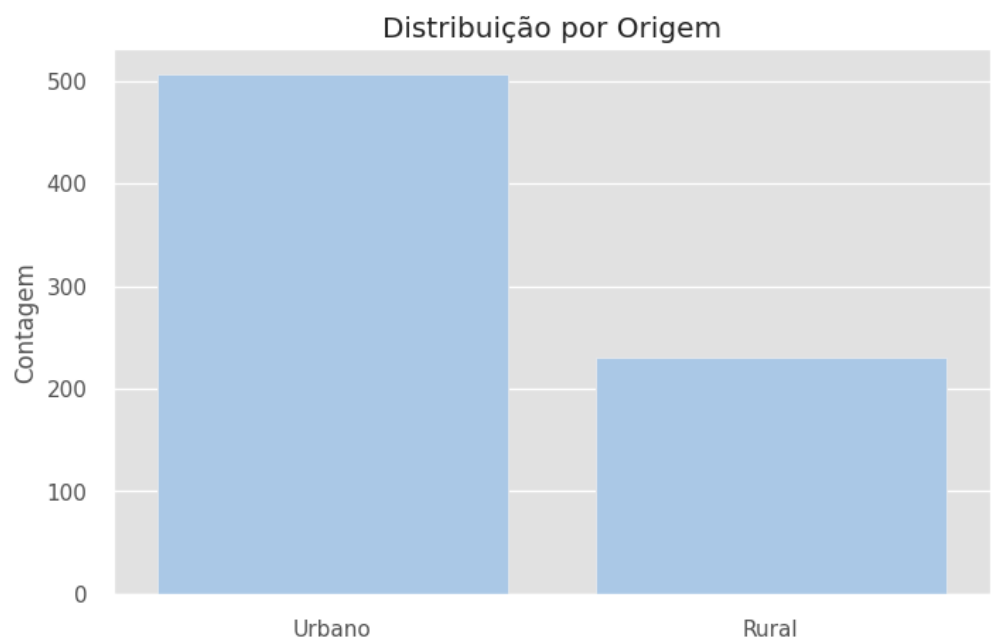
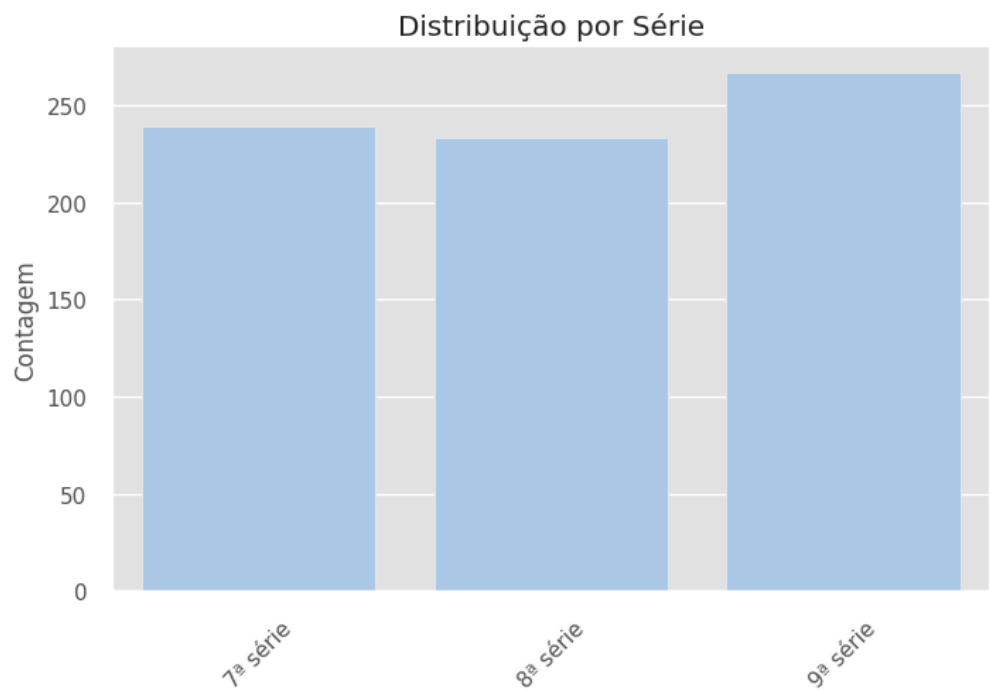
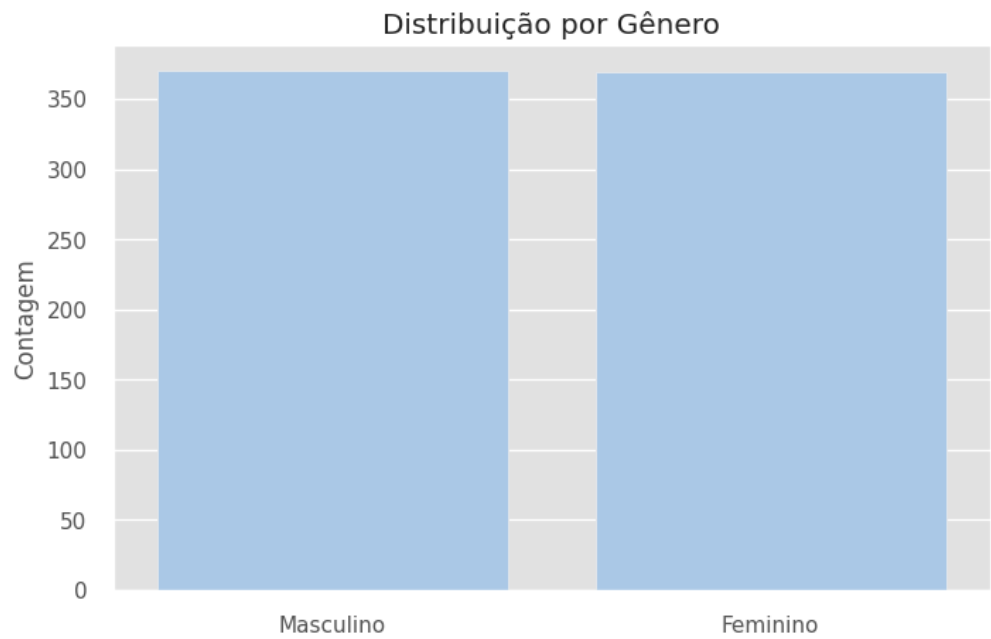
print("\nEducação das mães:")
print(df['Q5.1'].value_counts(normalize=True))

print("\nEducação dos pais:")
print(df['Q5.2'].value_counts(normalize=True))
```

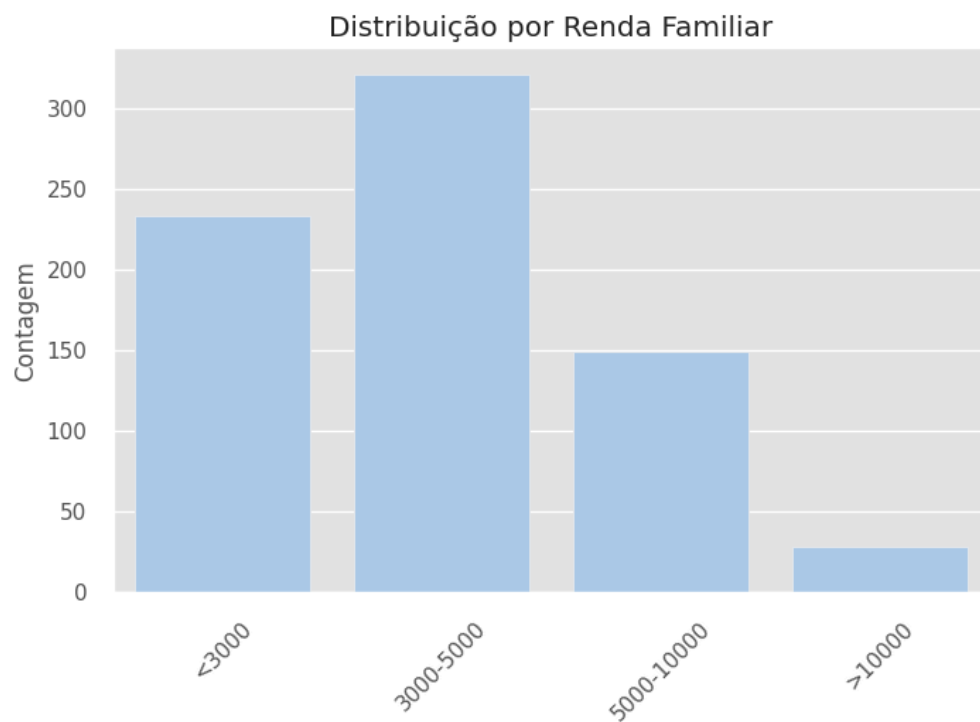
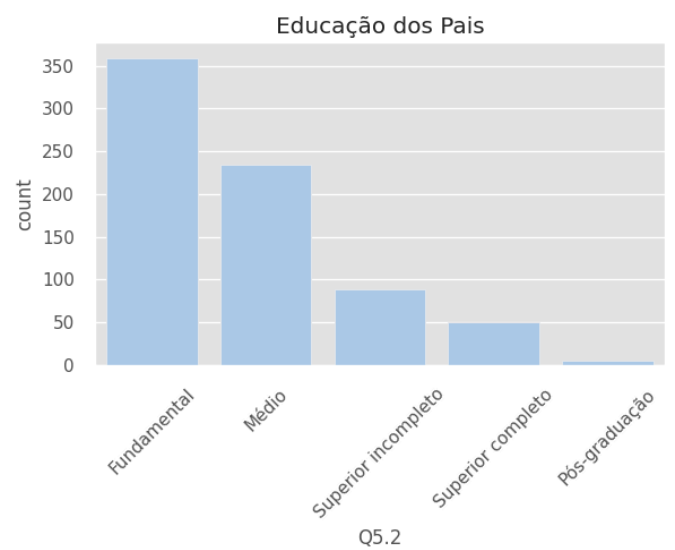
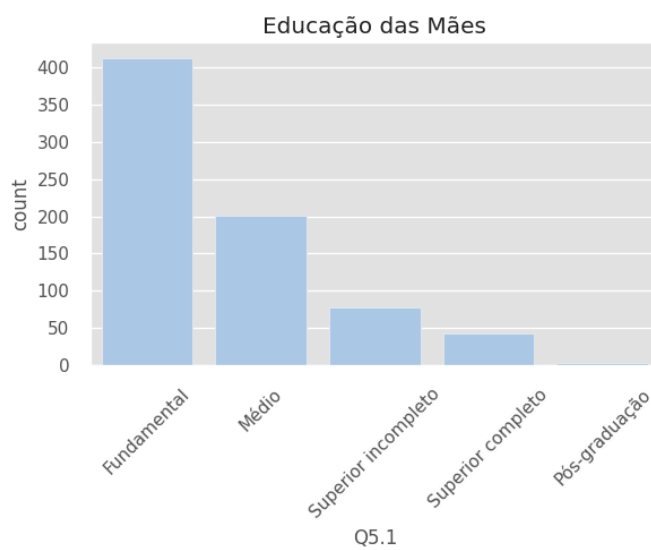
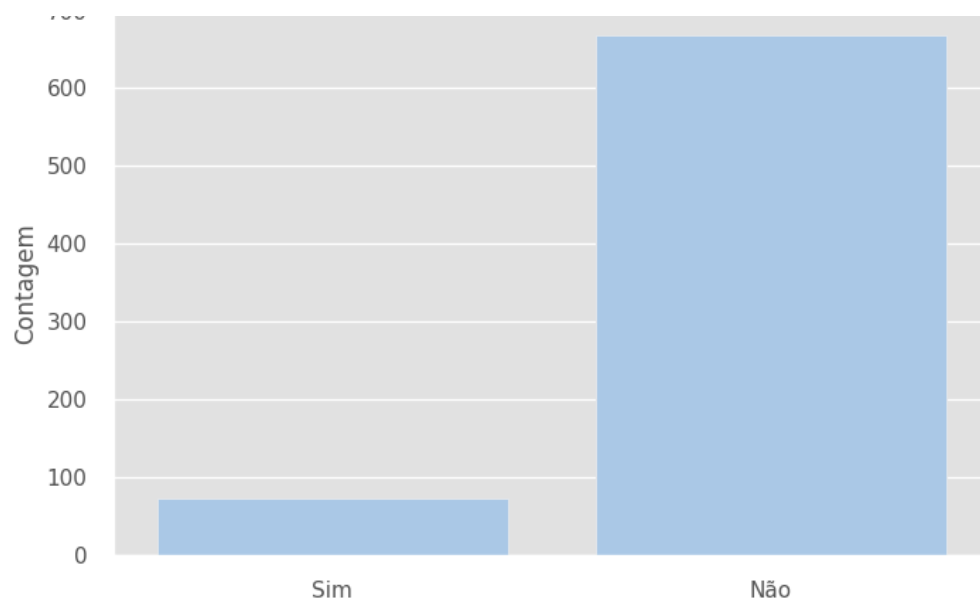


```
print("\nRenda familiar:")  
print(df['Q6'].value_counts(normalize=True))
```

 Dados carregados com sucesso!



Distribuição por Criança sob Tutela



=== Resumo Demográfico ===

Gênero:

Q1

Masculino 0.500677

Feminino 0.499323

Name: proportion, dtype: float64

Série:

Q2

9ª série 0.361299

7ª série 0.323410

8ª série 0.315291

Name: proportion, dtype: float64

Origem:

Q3

Urbano 0.6875

Rural 0.3125

Name: proportion, dtype: float64

Criança sob tutela:

Q4

Não 0.902571

Sim 0.097429

Name: proportion, dtype: float64

Educação das mães:

Q5.1

Fundamental 0.559621

Médio 0.272358

Superior incompleto 0.105691

Superior completo 0.058266

Pós-graduação 0.004065

Name: proportion, dtype: float64

Educação dos pais:

Q5.2

Fundamental 0.487110

Médio 0.317503

Superior incompleto 0.120760

Superior completo 0.067843

Pós-graduação 0.006784

Name: proportion, dtype: float64

Renda familiar:

Q6

3000-5000 0.439124

<3000 0.318741

5000-10000 0.203830

>10000 0.038304

Name: proportion, dtype: float64

```
import matplotlib
import matplotlib.pyplot as plt

# Instalar e configurar uma fonte compatível com chinês (ex: Noto Sans CJK)
!apt-get -qq install fonts-noto-cjk
matplotlib.rcParams['font.family'] = 'Noto Sans CJK JP'
matplotlib.rcParams['axes.unicode_minus'] = False # Corrige sinal de menos em eixos
```

✓ 2 Quais são os níveis médios das diferentes emoções relatadas?

```
# Instale a fonte Noto Sans CJK (executar apenas uma vez, pode ser ignorado se já tiver rodado)
!apt-get -qq install -y fonts-noto-cjk

import matplotlib.font_manager as fm
font_dirs = ['/usr/share/fonts/truetype/noto']
font_files = fm.findSystemFonts(fontpaths=font_dirs)
for font_file in font_files:
    fm.fontManager.addfont(font_file)

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Configurações de estilo
plt.style.use('ggplot')
sns.set_theme(style="whitegrid")
plt.rcParams['font.sans-serif'] = ['Noto Sans CJK SC', 'SimHei', 'Arial', 'DejaVu Sans']
plt.rcParams['axes.unicode_minus'] = False

# Simulação de DataFrame (substitua pelo seu df real)
# Exemplo de criação do DataFrame:
# df = pd.read_csv('seuarquivo.csv')
df = pd.DataFrame({
    '自豪': [3, 4, 2, 5, 3],
    '高兴': [4, 5, 3, 4, 4],
    '希望': [3, 3, 4, 4, 3],
    '满足': [4, 4, 4, 5, 3],
    '平静': [2, 3, 2, 4, 3],
    '放松': [3, 3, 2, 4, 2],
    '焦虑': [2, 2, 3, 1, 2],
    '羞愧': [1, 2, 1, 2, 1],
    '恼火': [2, 3, 2, 2, 2],
    '厌倦': [1, 2, 2, 1, 2],
    '无助': [1, 1, 2, 1, 1],
    '沮丧': [1, 1, 2, 2, 1],
    '心烦': [2, 2, 2, 1, 2]
})

# Dicionário das emoções básicas (chinês -> português)
emocoes = {
    '自豪': 'Orgulho',
    '高兴': 'Felicidade',
    '希望': 'Esperança',
    '满足': 'Satisfação',
    '平静': 'Calma',
    '放松': 'Relaxamento',
    '焦虑': 'Ansiedade',
    '羞愧': 'Vergonha',
    '恼火': 'Irritação',
    '厌倦': 'Cansaço',
    '无助': 'Desamparo',
    '沮丧': 'Desânimo',
    '心烦': 'Agitação'
}

# Filtrar apenas as colunas de emoções que existem no dataframe
cols_emocionais = [col for col in emocoes.keys() if col in df.columns]

# Calcular as médias
medias = df[cols_emocionais].mean()
medias.index = [emocoes[col] for col in medias.index]
medias = medias.sort_values(ascending=False)

# 1. Gráfico de todas as emoções
plt.figure(figsize=(12, 6))
```

```

cores = ['#3498db' if val >= 3 else '#e74c3c' for val in medias.values]
sns.barplot(x=medias.values, y=medias.index, hue=medias.index, palette=cores, legend=False)
plt.title('Intensidade Média das Emoções Relatadas', fontsize=16, pad=20)
plt.xlabel('Escala Média (1-5)', fontsize=12)
plt.ylabel('')
plt.xlim(0, 5)
plt.axvline(x=3, color='gray', linestyle='--', alpha=0.5)
plt.show()

# 2. Gráfico dividido por tipo de emoção
plt.figure(figsize=(14, 6))

# Emoções positivas
positivas = ['Felicidade', 'Esperança', 'Satisfação', 'Calma', 'Relaxamento', 'Orgulho']
plt.subplot(1, 2, 1)
sns.barplot(x=medias[positivas].values, y=medias[positivas].index, hue=medias[positivas].index, palette="Blues_d", legend=False)
plt.title('Emoções Positivas', fontsize=14)
plt.xlabel('Intensidade Média')
plt.xlim(0, 5)

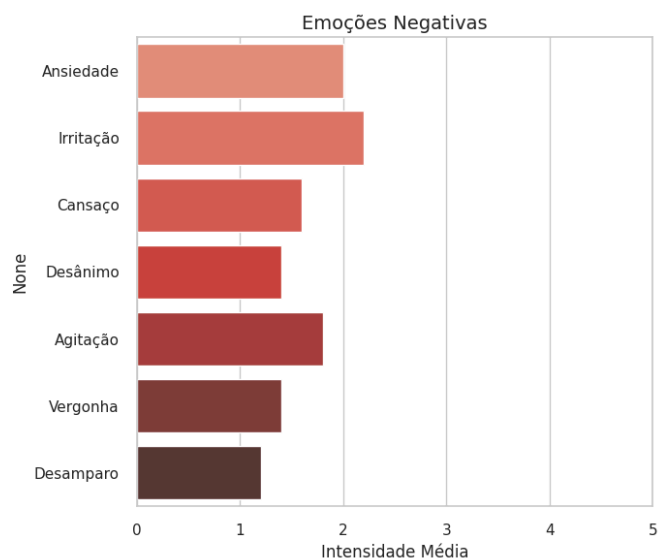
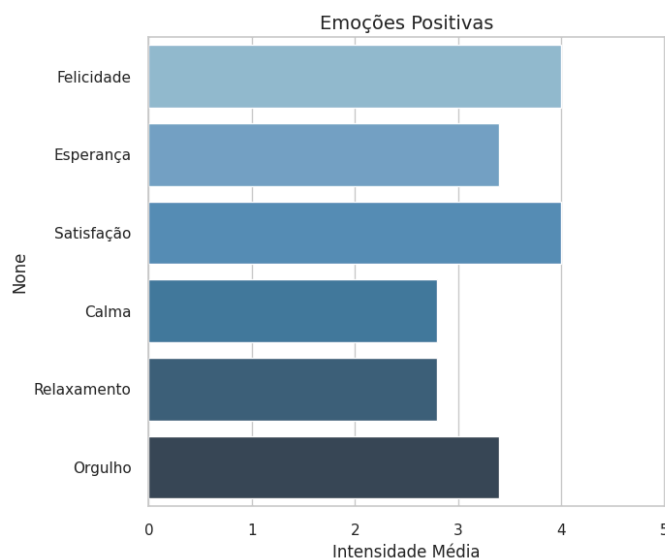
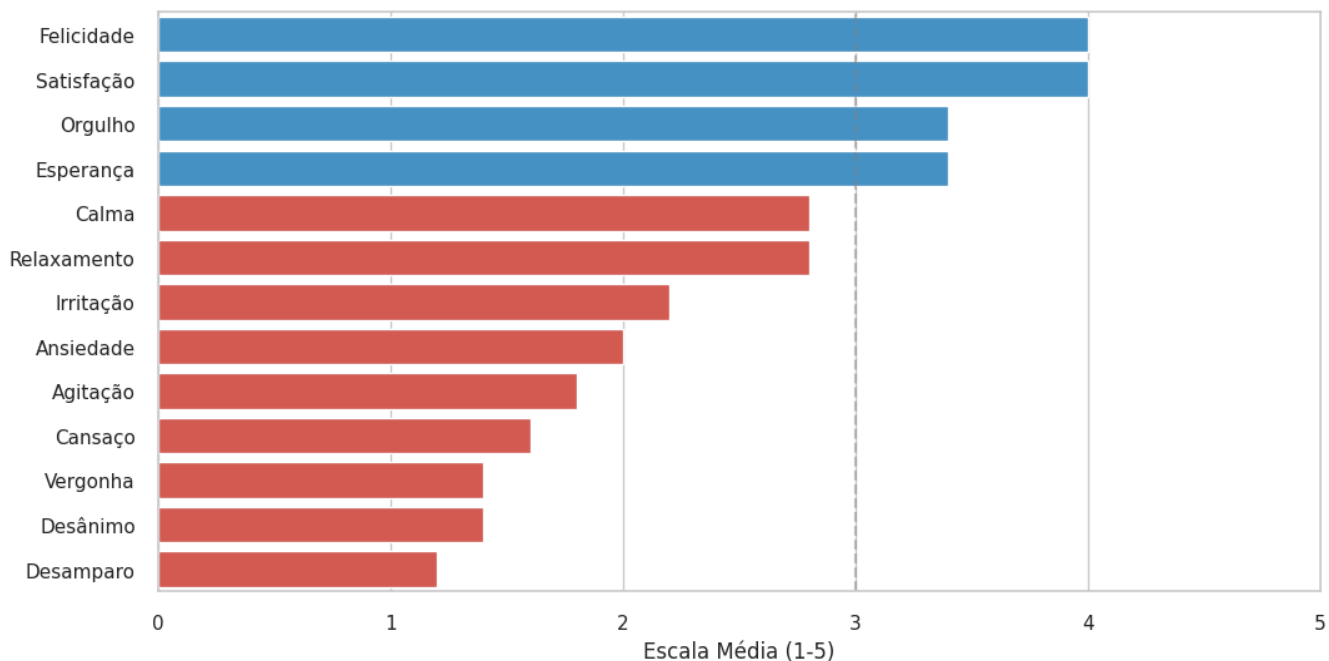
# Emoções negativas
negativas = ['Ansiedade', 'Irritação', 'Cansaço', 'Desânimo', 'Agitação', 'Vergonha', 'Desamparo']
plt.subplot(1, 2, 2)
sns.barplot(x=medias[negativas].values, y=medias[negativas].index, hue=medias[negativas].index, palette="Reds_d", legend=False)
plt.title('Emoções Negativas', fontsize=14)
plt.xlabel('Intensidade Média')
plt.xlim(0, 5)

plt.tight_layout()
plt.show()

```



Intensidade Média das Emoções Relatadas



3 Como as emoções positivas e negativas se correlacionam?

```
# 1. Instala a fonte para caracteres chineses (se necessário)
!apt-get -qq install -y fonts-noto-cjk

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.font_manager as fm
import os

# 2. Configuração de fontes para suportar caracteres chineses
fonte_cjk = "/usr/share/fonts/opentype/noto/NotoSansCJK-Regular.ttc"
if not os.path.exists(fonte_cjk):
    fonte_cjk = "/usr/share/fonts/truetype/noto/NotoSansCJK-Regular.ttc"
fm.fontManager.addfont(fonte_cjk)
prop_cjk = fm.FontProperties(fname=fonte_cjk)
plt.rcParams['font.sans-serif'] = [fonte_cjk, 'Noto Sans CJK SC', 'DejaVu Sans', 'Arial']
plt.rcParams['axes.unicode_minus'] = False

# 3. Dicionário das emoções (chinês -> português)
emocoes = {
    '自豪': 'Orgulho',
```

```

'高兴': 'Felicidade',
'希望': 'Esperança',
'满足': 'Satisfação',
'平静': 'Calma',
'放松': 'Relaxamento',
'焦虑': 'Ansiedade',
'羞愧': 'Vergonha',
'恼火': 'Irritação',
'厌倦': 'Cansaço',
'无助': 'Desamparo',
'沮丧': 'Desânimo',
'心烦': 'Agitação'
}

# 4. Emoções positivas e negativas
positivas = ['自豪', '高兴', '希望', '满足', '平静', '放松']
negativas = ['焦虑', '羞愧', '恼火', '厌倦', '无助', '沮丧', '心烦']

# 5. Carregue o DataFrame do arquivo SAV
df = pd.read_spss('/content/123年级day.sav')

# 6. Calcule o índice médio de emoções positivas e negativas para cada linha
df['Índice Positivo'] = df[positivas].mean(axis=1)
df['Índice Negativo'] = df[negativas].mean(axis=1)

# 7. Calcule a correlação de Pearson
correlacao = df[['Índice Positivo', 'Índice Negativo']].corr().iloc[0, 1]

# 8. Faça o gráfico de dispersão com linha de tendência
plt.figure(figsize=(8,6))
sns.regplot(
    data=df,
    x='Índice Positivo',
    y='Índice Negativo',
    scatter_kws={'s': 60, 'alpha': 0.7, 'color': '#3498db', 'edgecolor': 'black'},
    line_kws={"color": "#e74c3c", "lw": 2}
)
plt.title(f'Correlação entre Emoções Positivas e Negativas\nCoeficiente de Pearson: {correlacao:.2f}',
        fontproperties=prop_cjk, fontsize=15)
plt.xlabel('Índice Médio de Emoções Positivas', fontproperties=prop_cjk, fontsize=12)
plt.ylabel('Índice Médio de Emoções Negativas', fontproperties=prop_cjk, fontsize=12)
plt.grid(True, alpha=0.1)
plt.tight_layout()
plt.show()

```



Correlação entre Emoções Positivas e Negativas
Coeficiente de Pearson: -0.38




```
print(df.columns)
```

```
Index(['生日', 'Q1', 'Q2', 'Q3', 'Q4', 'Q5.1', 'Q5.2', 'Q6', '自豪', '高兴', '希望',  
      '满足', '平静', '放松', '焦虑', '羞愧', '恼火', '厌倦', '无助', '沮丧', '心烦', '目标专注',  
      '情绪控制', '积极认知', '家庭支持', '人际协助', '身心耗竭分量表', '学业疏离分量表', '低成就感分量表', '人际力',  
      '个人力', '学业倦怠', '积极高唤醒', '积极低唤醒', '消极高唤醒', '消极低唤醒', '心理弹性', '积极情绪',  
      '消极情绪', '积极低唤醒因子', '积极高唤醒因子', '消极高唤醒因子', '消极低唤醒因子', '消极情绪因子', '积极情绪因子'],  
      dtype='object')
```

✓ 4 Existem diferenças nos níveis de burnout entre gêneros, séries ou origens?

```
# Instale fonte chinesa se necessário  
!apt-get -qq install -y fonts-noto-cjk
```

```
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
import matplotlib.font_manager as fm  
import os
```

```
# Fonte chinesa  
fonte_cjk = "/usr/share/fonts/opentype/noto/NotoSansCJK-Regular.ttc"  
if not os.path.exists(fonte_cjk):  
    fonte_cjk = "/usr/share/fonts/truetype/noto/NotoSansCJK-Regular.ttc"  
fm.fontManager.addfont(fonte_cjk)  
prop_cjk = fm.FontProperties(fname=fonte_cjk)  
plt.rcParams['font.sans-serif'] = [fonte_cjk, 'Noto Sans CJK SC', 'DejaVu Sans', 'Arial']  
plt.rcParams['axes.unicode_minus'] = False
```

```
# Carrega dados  
df = pd.read_spss('/content/123年级day.sav')
```

```
# DEFINA AQUI quais são as colunas de agrupamento:  
col_burnout = '学业倦怠' # burnout acadêmico  
col_genero = 'Q1'        # substitua se souber o nome correto  
col_serie = 'Q2'         # substitua se souber o nome correto  
col_origem = 'Q3'        # substitua se souber o nome correto
```

```
# Gráfico de burnout por gênero  
plt.figure(figsize=(6,4))  
sns.boxplot(x=col_genero, y=col_burnout, data=df, palette='pastel')  
plt.title('Níveis de Burnout por Gênero', fontproperties=prop_cjk, fontsize=14)  
plt.xlabel('Gênero', fontproperties=prop_cjk)  
plt.ylabel('Índice de Burnout', fontproperties=prop_cjk)  
plt.tight_layout()  
plt.show()
```

```
# Gráfico de burnout por série  
plt.figure(figsize=(6,4))  
sns.boxplot(x=col_serie, y=col_burnout, data=df, palette='Set2')  
plt.title('Níveis de Burnout por Série', fontproperties=prop_cjk, fontsize=14)  
plt.xlabel('Série', fontproperties=prop_cjk)  
plt.ylabel('Índice de Burnout', fontproperties=prop_cjk)  
plt.tight_layout()  
plt.show()
```

```
# Gráfico de burnout por origem  
plt.figure(figsize=(6,4))  
sns.boxplot(x=col_origem, y=col_burnout, data=df, palette='Set3')  
plt.title('Níveis de Burnout por Origem', fontproperties=prop_cjk, fontsize=14)  
plt.xlabel('Origem', fontproperties=prop_cjk)  
plt.ylabel('Índice de Burnout', fontproperties=prop_cjk)  
plt.tight_layout()  
plt.show()
```


```
# Testes estatísticos (ANOVA)  
from scipy.stats import f_oneway
```

```
# Por gênero  
grupos_genero = [g[col_burnout].dropna() for n, g in df.groupby(col_genero)]  
stat_gen, p_gen = f_oneway(*grupos_genero)  
print(f'Teste ANOVA - Burnout por Gênero: p-valor = {p_gen:.4f}')
```

```
# Por série  
grupos_serie = [g[col_burnout].dropna() for n, g in df.groupby(col_serie)]  
stat_ser, p_ser = f_oneway(*grupos_serie)
```

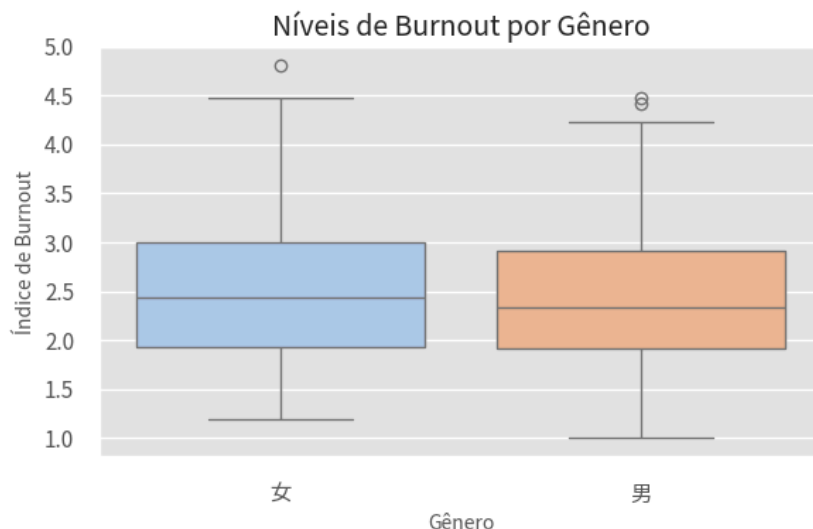
```
print(f'Teste ANOVA - Burnout por Série: p-valor = {p_ser:.4f}')

# Por origem
grupos_origem = [g[col_burnout].dropna() for n, g in df.groupby(col_origem)]
stat_ori, p_ori = f_oneway(*grupos_origem)
print(f'Teste ANOVA - Burnout por Origem: p-valor = {p_ori:.4f}')
```

 /tmp/ipython-input-59-2463192466.py:30: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

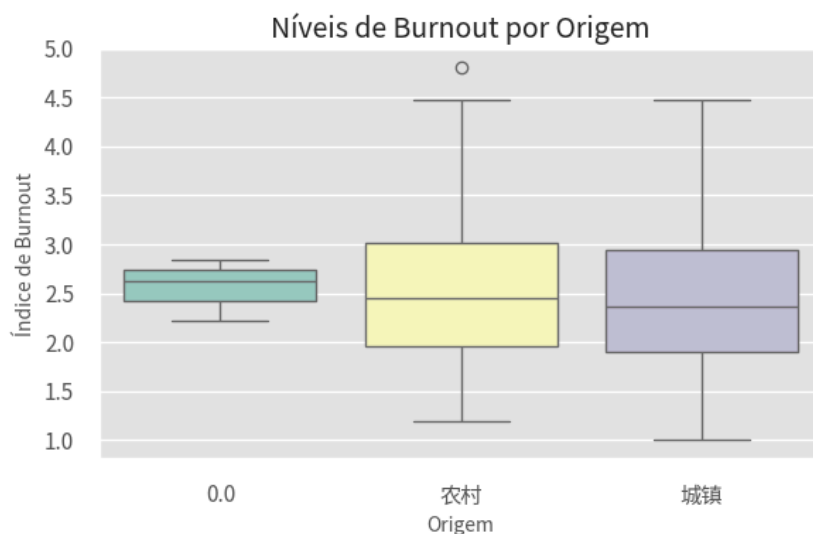
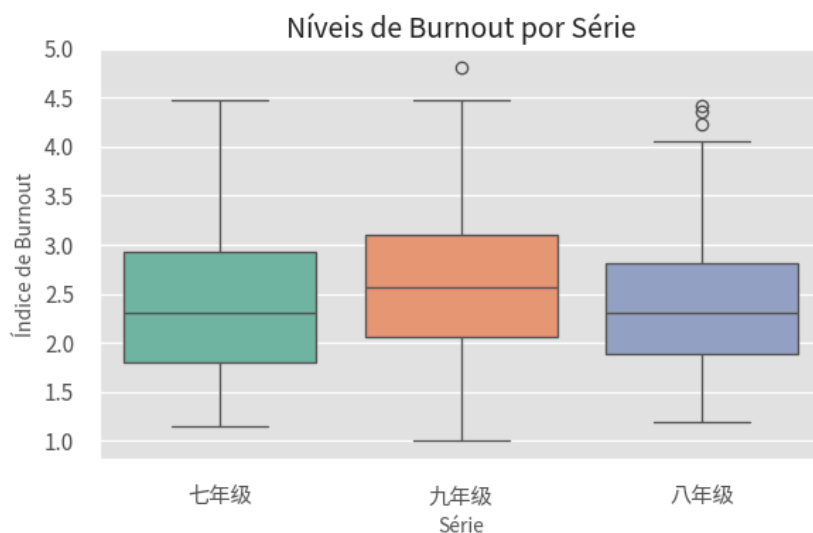
```
sns.boxplot(x=col_genero, y=col_burnout, data=df, palette='pastel')
```



/tmp/ipython-input-59-2463192466.py:39: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

```
sns.boxplot(x=col_serie, y=col_burnout, data=df, palette='Set2')
```



Teste ANOVA - Burnout por Gênero: p-valor = 0.3615

Teste ANOVA - Burnout por Série: p-valor = 0.0015

Teste ANOVA - Burnout por Origem: p-valor = 0.3311

/tmp/ipython-input-59-2463192466.py:59: FutureWarning: The default of observed=False is deprecated and will be changed to True
grupos_genero = [g[col_burnout].dropna() for n, g in df.groupby(col_genero)]

/tmp/ipython-input-59-2463192466.py:64: FutureWarning: The default of observed=False is deprecated and will be changed to True
grupos_serie = [g[col_burnout].dropna() for n, g in df.groupby(col_serie)]

/tmp/ipython-input-59-2463192466.py:69: FutureWarning: The default of observed=False is deprecated and will be changed to True

```
, emp_type) # Input of 20022000.py: not changing. The value of observed value is deprecated and will be changed to 0.  
grupos_origem = [g[col_burnout].dropna() for n, g in df.groupby(col_origem)]
```

✓ 5 Qual a relação entre resiliência psicológica e burnout?

```
# Instale a fonte para caracteres chineses (se necessário)  
!apt-get -qq install -y fonts-noto-cjk  
  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
import matplotlib.font_manager as fm  
import os  
  
# Fonte chinesa  
fonte_cjk = "/usr/share/fonts/opentype/noto/NotoSansCJK-Regular.ttc"  
if not os.path.exists(fonte_cjk):  
    fonte_cjk = "/usr/share/fonts/truetype/noto/NotoSansCJK-Regular.ttc"  
fm.fontManager.addfont(fonte_cjk)  
prop_cjk = fm.FontProperties(fname=fonte_cjk)  
plt.rcParams['font.sans-serif'] = [fonte_cjk, 'Noto Sans CJK SC', 'DejaVu Sans', 'Arial']  
plt.rcParams['axes.unicode_minus'] = False  
  
# Carregue os dados  
df = pd.read_spss('/content/123年级day.sav')  
  
# Nome das variáveis (ajuste se necessário dependendo dos nomes do seu arquivo)  
col_resiliencia = '心理弹性' # Resiliência psicológica  
col_burnout = '学业倦怠' # Burnout acadêmico  
  
# Gráfico de dispersão com linha de tendência  
plt.figure(figsize=(8,6))  
sns.regplot(  
    data=df,  
    x=col_resiliencia,  
    y=col_burnout,  
    scatter_kws={'s': 60, 'alpha': 0.7, 'color': '#2ecc71', 'edgecolor': 'black'},  
    line_kws={"color": "#e74c3c", "lw": 2}  
)  
plt.title('Relação entre Resiliência Psicológica e Burnout Acadêmico',  
        fontproperties=prop_cjk, fontsize=15)  
plt.xlabel('Resiliência Psicológica (心理弹性)', fontproperties=prop_cjk, fontsize=12)  
plt.ylabel('Burnout Acadêmico (学业倦怠)', fontproperties=prop_cjk, fontsize=12)  
plt.grid(True, alpha=0.1)  
plt.tight_layout()  
plt.show()
```



Relação entre Resiliência Psicológica e Burnout Acadêmico