Clusterização Híbrida de Leitos Hospitalares do CNES

Metodologia Data-Driven com Validação Clínica

Cieges - Brasil Estadual

2026-01-21

Table of contents

1 Introdução **1**
1.1 Objetivo . . . 1
1.2 Fundamentação Metodológica . . . 1
1.3 Fluxo Metodológico . . . 1

2 Etapa 1: Engenharia de Features **2**
2.1 Carregamento dos Dados . . . 2
2.2 Construção da Matriz de Features . . . 3
2.3 Visualização da Matriz de Features . . . 4

3 Etapa 2: Análise Exploratória **5**
3.1 Estatísticas Descritivas . . . 5
3.2 Matriz de Correlação . . . 5
3.3 Distribuição das Features . . . 6
3.4 Detecção de Outliers . . . 7

4 Etapa 3: Pré-Processamento **8**
4.1 Seleção de Features para Clusterização . . . 8
4.2 Normalização (StandardScaler) . . . 8
4.3 Análise de Componentes Principais (PCA Exploratório) . . . 8

5 Etapa 4: Clusterização Hierárquica **9**
5.1 Cálculo da Matriz de Distâncias . . . 9
5.2 Dendrograma - Método de Ward . . . 9
5.3 Comparação de Métodos de Linkage . . . 10

6 Etapa 5: Determinação do Número Ótimo de Clusters **11**
6.1 Método do Cotovelo (Elbow) . . . 11
6.2 Análise de Silhouette por Cluster . . . 13

7 Etapa 6: Validação e Rotulação Clínica **14**
7.1 Perfil dos Clusters . . . 14
7.2 Heatmap de Características por Cluster . . . 16
7.3 Rotulação Clínica dos Clusters . . . 17

8 Etapa 7: Visualizações Avançadas **18**
8.1 Projeção PCA com Clusters . . . 18
8.2 Radar Chart por Cluster . . . 19
8.3 Distribuição de Leitos por Cluster . . . 20
8.4 Matriz de Confusão: Cluster vs Tipo de Leito Original . . . 21

9 Etapa 8: Resultado Final **21**
9.1 Taxonomia Híbrida Completa . . . 21
9.2 Dicionário de Clusters . . . 22
9.3 Exportação . . . 24
9.4 Métricas de Qualidade . . . 24

10 Comparação: Metodologia 1 vs Metodologia 2 **24**

11 Resumo Executivo **25**

1 Introdução

1.1 Objetivo

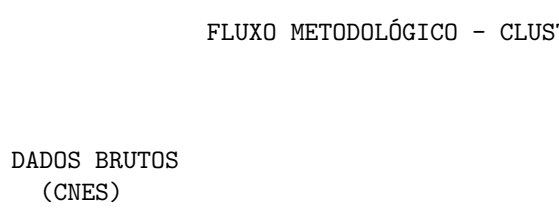
Este documento apresenta uma **metodologia híbrida de clusterização** para identificar agrupamentos naturais de especialidades de leitos hospitalares do CNES. A abordagem combina técnicas estatísticas de aprendizado não-supervisionado com validação clínica posterior.

1.2 Fundamentação Metodológica

A clusterização híbrida segue o framework proposto por **Kaufman & Rousseeuw (1990)** para análise de clusters, adaptado para dados de saúde:

1. **Engenharia de Features**: Construção de matriz de características multidimensional
2. **Análise Exploratória**: Verificação de pressupostos e distribuições
3. **Clusterização Hierárquica**: Método aglomerativo de Ward
4. **Validação Interna**: Silhouette Score, Índice de Calinski-Harabasz
5. **Validação Externa**: Interpretação clínica e rotulação semântica

1.3 Fluxo Metodológico



1. ENGENHARIA DE FEATURES

- Agregação por especialidade
- Métricas derivadas
- 12 variáveis

2. ANÁLISE EXPLORATÓRIA

- Matriz correlação
- Outliers
- Distribuições

3. PRÉ-PROCESSAMENTO

- Normalização (StandardScaler)
- Redução dimensional (PCA exploratório)

4. CLUSTERIZAÇÃO HIERÁRQUICA

- Método de Ward
- Dendrograma
- Distância Euclidiana

5. VALIDAÇÃO INTERNA

- Silhouette Score
- Calinski-Harabasz
- Davies-Bouldin

6. DETERMINAÇÃO Nº CLUSTERS

- Elbow Method
- Gap Statistic
- Silhouette Analysis

7. VALIDAÇÃO EXTERNA

- Interpretação clínica
- Rotulação semântica

8. RESULTADO FINAL

- Clusters validados
- Taxonomia híbrida
- Exportação

2 Etapa 1: Engenharia de Features

2.1 Carregamento dos Dados

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from scipy.spatial.distance import pdist
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering, KMeans
from sklearn.metrics import silhouette_score, silhouette_samples, calinski_harabasz_score, davies_bouldin_score
import warnings
warnings.filterwarnings('ignore')

# Configuração de visualização
plt.rcParams['figure.dpi'] = 100
plt.rcParams['font.size'] = 10
sns.set_style("whitegrid")

# Carregar dados
df = pd.read_csv('arq2_tratado.csv', sep=';', encoding='latin1', low_memory=False)

print("="*70)
print("DADOS CARREGADOS")
print("="*70)
print(f"Registros: {len(df):,}")
print(f"Leitos: {df['qt_exist'].sum():,}")
print(f"Especialidades (co_leito): {df['co_leito'].nunique()}")
print(f"Estabelecimentos (CNES): {df['cnes'].nunique():,}")
```

```
======================================================================
DADOS CARREGADOS
======================================================================
Registros: 49,804
Leitos: 535,133
Especialidades (co_leito): 65
Estabelecimentos (CNES): 9,072
```

2.2 Construção da Matriz de Features

Criação de **12 variáveis** para caracterizar cada especialidade:

#	Variável	Descrição	Tipo
1	`total_leitos`	Total de leitos da especialidade	Volume
2	`total_estabelecimentos`	Nº de CNES com a especialidade	Dispersão
3	`media_leitos_estab`	Média de leitos por estabelecimento	Concentração
4	`mediana_leitos_estab`	Mediana de leitos por estabelecimento	Concentração
5	`cv_leitos`	Coeficiente de variação dos leitos	Heterogeneidade
6	`pct_sus`	% de leitos SUS	Natureza
7	`pct_privado`	% de leitos não-SUS	Natureza
8	`dispersao_geografica`	Nº de municípios com a especialidade	Cobertura
9	`concentracao_hhi`	Índice Herfindahl-Hirschman	Concentração
10	`gini_leitos`	Coeficiente de Gini da distribuição	Desigualdade
11	`tipo_leito_predominante`	Tipo de leito mais frequente	Categoria
12	`complexidade_proxy`	Proxy de complexidade (baseado em UTI/UCI)	Intensidade

```python
def calcular_gini(array):
    """Calcula o coeficiente de Gini de uma distribuição."""
    array = np.array(array, dtype=float)
    array = array[array > 0]  # Remove zeros
    if len(array) == 0:
        return 0
    array = np.sort(array)
    n = len(array)
    index = np.arange(1, n + 1)
    return (2 * np.sum(index * array) - (n + 1) * np.sum(array)) / (n * np.sum(array))

def calcular_hhi(array):
    """Calcula o Índice Herfindahl-Hirschman (concentração de mercado)."""
    array = np.array(array, dtype=float)
    total = array.sum()
    if total == 0:
        return 0
    shares = array / total
    return np.sum(shares ** 2)

# Agregar por especialidade
features_list = []

for co in df['co_leito'].unique():
    subset = df[df['co_leito'] == co]

    # Agregação por estabelecimento
    por_estab = subset.groupby('cnes').agg({
        'qt_exist': 'sum',
        'qt_sus': 'sum',
        'qt_nsus': 'sum',
        'codufmun': 'first'
    }).reset_index()

    # Calcular features
    total_leitos = subset['qt_exist'].sum()
    total_estab = subset['cnes'].nunique()
    total_mun = subset['codufmun'].nunique()

    leitos_por_estab = por_estab['qt_exist'].values

    features = {
        'co_leito': co,
        'DS_CO_LEITO': subset['DS_CO_LEITO'].iloc[0],
        'tp_leito': subset['tp_leito'].iloc[0],
        'DS_TP_LEITO': subset['DS_TP_LEITO'].iloc[0],

        # Volume
        'total_leitos': total_leitos,
        'total_estabelecimentos': total_estab,

        # Concentração
        'media_leitos_estab': np.mean(leitos_por_estab),
        'mediana_leitos_estab': np.median(leitos_por_estab),
        'cv_leitos': np.std(leitos_por_estab) / np.mean(leitos_por_estab) if np.mean(leitos_por_estab) > 0 else 0,

        # Natureza
        'pct_sus': subset['qt_sus'].sum() / total_leitos * 100 if total_leitos > 0 else 0,
        'pct_privado': subset['qt_nsus'].sum() / total_leitos * 100 if total_leitos > 0 else 0,

        # Dispersão geográfica
        'dispersao_geografica': total_mun,
        'leitos_por_municipio': total_leitos / total_mun if total_mun > 0 else 0,

        # Concentração
        'concentracao_hhi': calcular_hhi(leitos_por_estab),
```

```
        'gini_leitos': calcular_gini(leitos_por_estab),

        # Complexidade (proxy baseado em código)
        'is_uti': 1 if co in [74,75,76,77,78,79,80,81,82,83,85,86] else 0,
        'is_uci': 1 if co in [65,92,93,94,95,96] else 0,
        'is_cirurgico': 1 if subset['tp_leito'].iloc[0] == 1 else 0,
    }

    features_list.append(features)

# Criar DataFrame de features
df_features = pd.DataFrame(features_list)

# Criar proxy de complexidade
df_features['complexidade_proxy'] = (
    df_features['is_uti'] * 3 +
    df_features['is_uci'] * 2 +
    df_features['is_cirurgico'] * 1
)

print("\n" + "="*70)
print("MATRIZ DE FEATURES CONSTRUÍDA")
print("="*70)
print(f"Especialidades: {len(df_features)}")
print(f"Features: {len(df_features.columns) - 4}")  # Excluindo identificadores
print(f"\nColunas:")
for col in df_features.columns:
    print(f"  - {col}")
```

```
======================================================================
MATRIZ DE FEATURES CONSTRUÍDA
======================================================================
Especialidades: 65
Features: 15

Colunas:
  - co_leito
  - DS_CO_LEITO
  - tp_leito
  - DS_TP_LEITO
  - total_leitos
  - total_estabelecimentos
  - media_leitos_estab
  - mediana_leitos_estab
  - cv_leitos
  - pct_sus
  - pct_privado
  - dispersao_geografica
  - leitos_por_municipio
  - concentracao_hhi
  - gini_leitos
  - is_uti
  - is_uci
  - is_cirurgico
  - complexidade_proxy
```

2.3 Visualização da Matriz de Features

```
# Exibir matriz de features
display_cols = ['co_leito', 'DS_CO_LEITO', 'total_leitos', 'total_estabelecimentos',
                'media_leitos_estab', 'pct_sus', 'dispersao_geografica', 'complexidade_proxy']
df_features[display_cols].sort_values('total_leitos', ascending=False).head(20)
```

	co_leito	DS_CO_LEITO	total_leitos	total_estabelecimentos	media_leitos_estab	pct_sus	dispersao_geografic
6	33	CLINICA GERAL	138290	6313	21.905592	72.598886	3559
5	3	CIRURGIA GERAL	63475	4821	13.166356	66.408822	2603
1	45	PEDIATRIA CLINICA	40643	4759	8.540240	79.957188	3190
4	75	UTI ADULTO - TIPO II	33519	1767	18.969440	58.247561	683
19	47	PSIQUIATRIA	28734	1067	26.929709	53.762094	710
9	10	OBSTETRICIA CIRURGICA	25058	3022	8.291860	70.304893	2006
10	43	OBSTETRICIA CLINICA	25037	3648	6.863213	81.587251	2708
0	13	ORTOPEDIATRAUMATOLOGIA	20118	1590	12.652830	78.014713	696
32	34	CRONICOS	10135	618	16.399676	85.643809	434
13	7	CIRURGICO/DIAGNOSTICO/TERAPEUTICO	10130	1470	6.891156	47.719645	439
12	32	CARDIOLOGIA	7751	879	8.817975	60.817959	376
28	76	UTI ADULTO - TIPO III	7402	295	25.091525	36.436098	105
47	44	ONCOLOGIA	7366	622	11.842444	65.503665	253
8	81	UTI NEONATAL - TIPO II	6957	611	11.386252	63.518758	314
20	6	GINECOLOGIA	6877	1536	4.477214	63.501527	811
3	68	PEDIATRIA CIRURGICA	5966	1175	5.077447	69.326182	611
2	78	UTI PEDIATRICA - TIPO II	5515	560	9.848214	52.511333	254
14	2	CARDIOLOGIA	5368	570	9.417544	59.184054	214
38	12	ONCOLOGIA	5217	526	9.918251	70.251102	221
16	92	UNIDADE DE CUIDADOS INTERMEDIARIOS NEONATAL CO...	5126	553	9.269439	67.947718	323

3 Etapa 2: Análise Exploratória

3.1 Estatísticas Descritivas

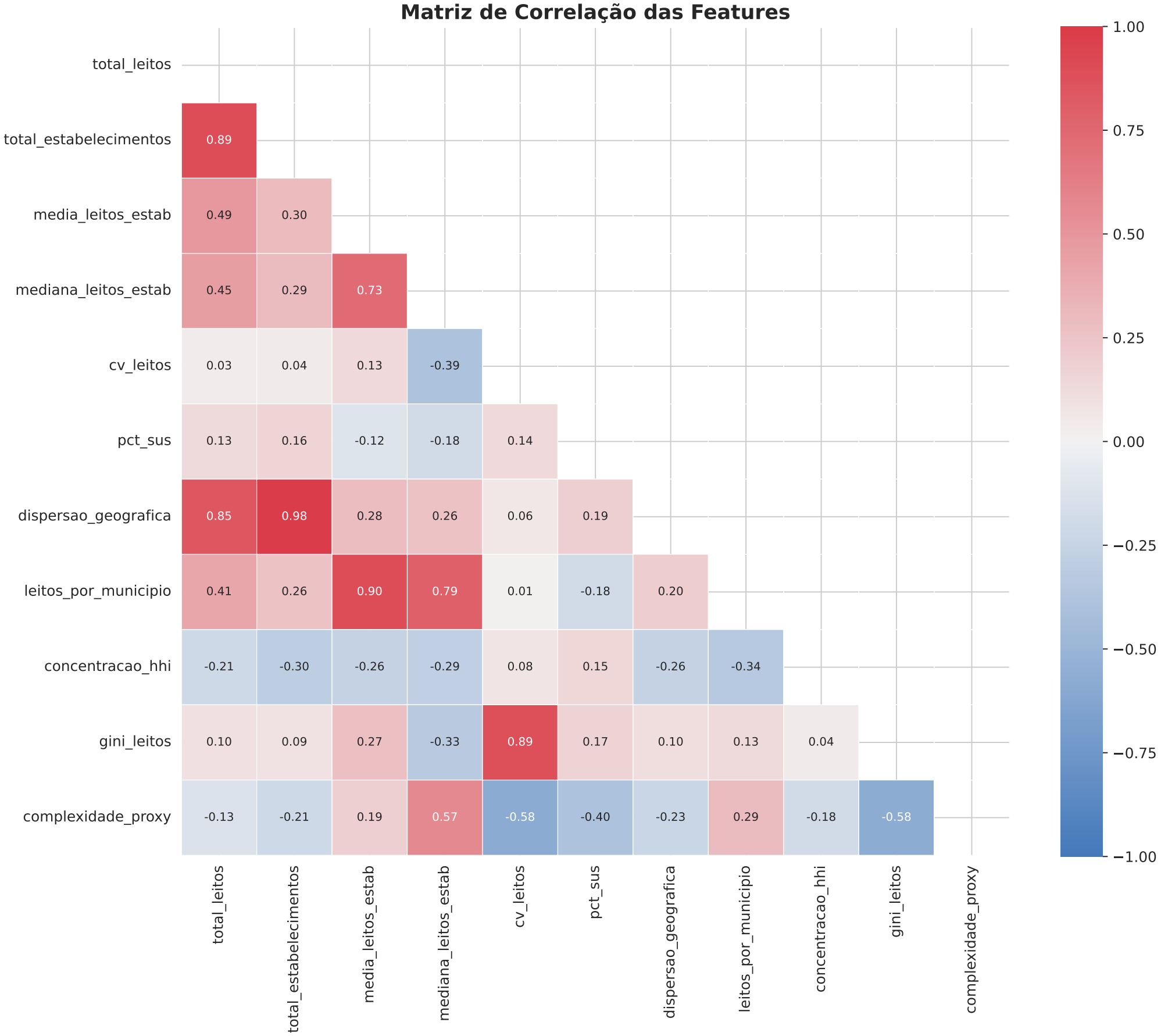
```
# Selecionar features numéricas para análise
feature_cols = ['total_leitos', 'total_estabelecimentos', 'media_leitos_estab',
                'mediana_leitos_estab', 'cv_leitos', 'pct_sus', 'dispersao_geografica',
                'leitos_por_municipio', 'concentracao_hhi', 'gini_leitos', 'complexidade_proxy']

print("ESTATÍSTICAS DESCRITIVAS DAS FEATURES")
print("="*70)
df_features[feature_cols].describe().round(2)
```

```
ESTATÍSTICAS DESCRITIVAS DAS FEATURES
======================================================================
```

	total_leitos	total_estabelecimentos	media_leitos_estab	mediana_leitos_estab	cv_leitos	pct_sus	dispersao_geografica	leitos_por_municipio	concentracao_hh
count	65.00	65.00	65.00	65.00	65.00	65.00	65.00	65.00	65.00
mean	8232.82	766.22	8.20	4.80	1.29	59.36	419.80	15.53	0.02
std	19702.09	1204.33	5.44	3.43	0.47	23.31	734.67	12.01	0.04
min	23.00	11.00	1.79	1.00	0.63	1.34	9.00	2.56	0.00
25%	1117.00	137.00	4.70	2.00	0.96	46.36	82.00	8.04	0.00
50%	2580.00	432.00	6.89	4.00	1.26	60.82	212.00	10.91	0.01
75%	5515.00	704.00	9.85	6.00	1.50	72.60	314.00	22.16	0.02
max	138290.00	6313.00	26.93	18.00	2.92	96.85	3559.00	70.50	0.23

3.2 Matriz de Correlação

```
# Matriz de correlação
corr_matrix = df_features[feature_cols].corr()

fig, ax = plt.subplots(figsize=(12, 10))
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
cmap = sns.diverging_palette(250, 10, as_cmap=True)

sns.heatmap(corr_matrix, mask=mask, cmap=cmap, vmax=1, vmin=-1, center=0,
            square=True, linewidths=.5, annot=True, fmt='.2f',
            annot_kws={'size': 8}, ax=ax)

ax.set_title('Matriz de Correlação das Features', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()

# Identificar correlações fortes
print("\nCORRELAÇÕES FORTES (|r| > 0.7):")
print("-"*50)
for i in range(len(corr_matrix.columns)):
    for j in range(i+1, len(corr_matrix.columns)):
        if abs(corr_matrix.iloc[i, j]) > 0.7:
            print(f"  {corr_matrix.columns[i]} × {corr_matrix.columns[j]}: {corr_matrix.iloc[i, j]:.3f}")
```

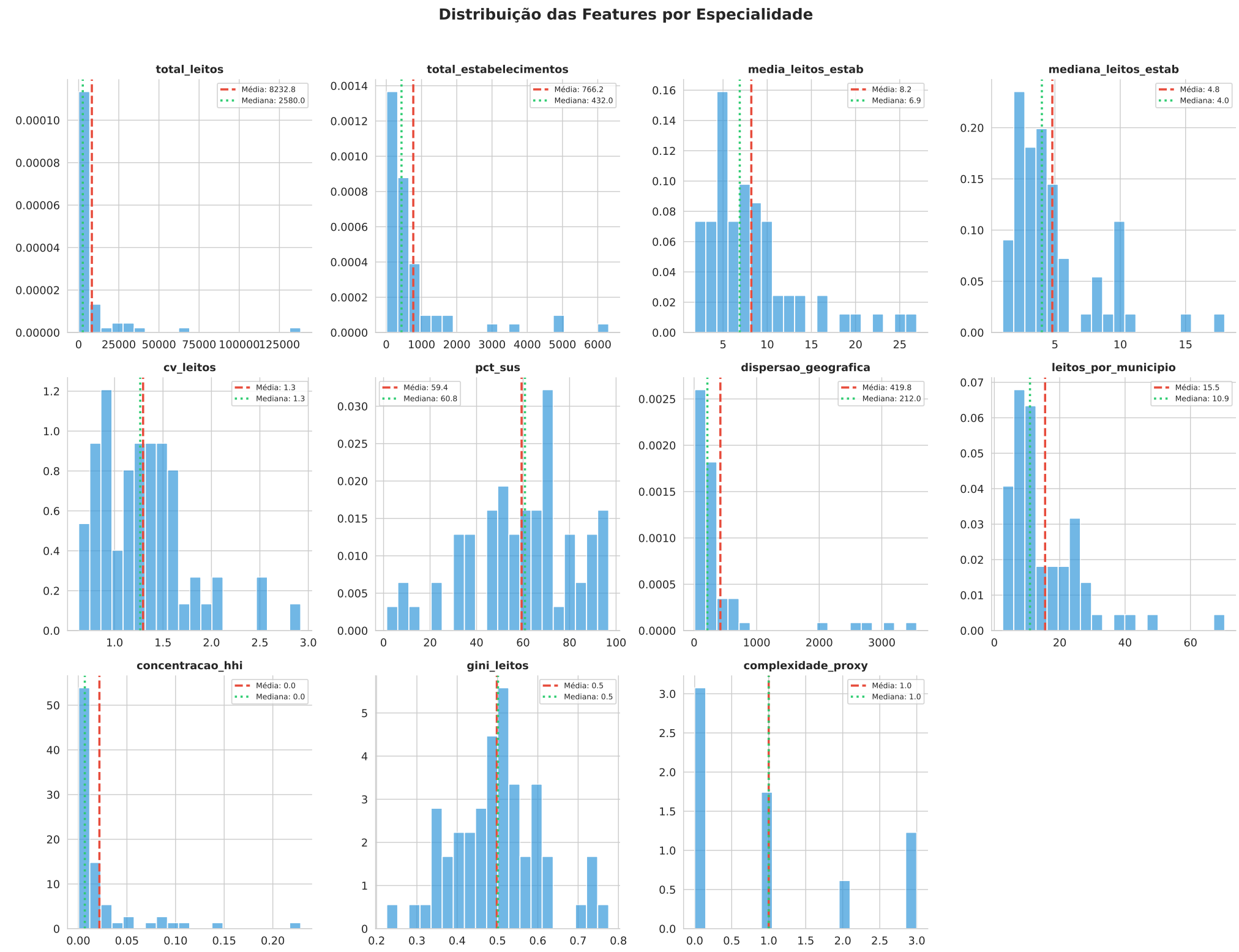



```
CORRELAÇÕES FORTES (|r| > 0.7):
------------------------------------------------
  total_leitos × total_estabelecimentos: 0.892
  total_leitos × dispersao_geografica: 0.848
  total_estabelecimentos × dispersao_geografica: 0.982
  media_leitos_estab × mediana_leitos_estab: 0.733
  media_leitos_estab × leitos_por_municipio: 0.896
  mediana_leitos_estab × leitos_por_municipio: 0.792
  cv_leitos × gini_leitos: 0.886
```

3.3 Distribuição das Features

```python
fig, axes = plt.subplots(3, 4, figsize=(16, 12))
axes = axes.flatten()

for i, col in enumerate(feature_cols):
    if i < len(axes):
        ax = axes[i]
        data = df_features[col].dropna()

        # Histograma com KDE
        ax.hist(data, bins=20, density=True, alpha=0.7, color='#3498db', edgecolor='white')

        # Adicionar linha de média
        ax.axvline(data.mean(), color='#e74c3c', linestyle='--', linewidth=2, label=f'Média: {data.mean():.1f}')
        ax.axvline(data.median(), color='#2ecc71', linestyle=':', linewidth=2, label=f'Mediana: {data.median():.1f}')

        ax.set_title(col, fontsize=10, fontweight='bold')
        ax.legend(fontsize=7)
        ax.spines['top'].set_visible(False)
        ax.spines['right'].set_visible(False)

# Remover eixos vazios
for j in range(len(feature_cols), len(axes)):
```

```
    axes[j].set_visible(False)

plt.suptitle('Distribuição das Features por Especialidade', fontsize=14, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()
```



3.4 Detecção de Outliers

```
# Detecção de outliers usando IQR
def detect_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    outliers = data[(data[column] < lower) | (data[column] > upper)]
    return outliers

print("ANÁLISE DE OUTLIERS (Método IQR)")
print("="*70)

outlier_summary = []
for col in feature_cols:
    outliers = detect_outliers_iqr(df_features, col)
    if len(outliers) > 0:
        outlier_summary.append({
            'Feature': col,
            'N_Outliers': len(outliers),
            'Pct': f"{len(outliers)/len(df_features)*100:.1f}%",
            'Especialidades': ', '.join(outliers['DS_CO_LEITO'].head(3).tolist())
        })

pd.DataFrame(outlier_summary)
```

```
ANÁLISE DE OUTLIERS (Método IQR)
======================================================================
```

	Feature	N_Outliers	Pct	Especialidades
0	total_leitos	8	12.3%	ORTOPEDIATRAUMATOLOGIA, PEDIATRIA CLINICA, UTI...
1	total_estabelecimentos	7	10.8%	ORTOPEDIATRAUMATOLOGIA, PEDIATRIA CLINICA, UTI...
2	media_leitos_estab	5	7.7%	UTI ADULTO - TIPO II, CLINICA GERAL, PSIQUIATRIA
3	mediana_leitos_estab	2	3.1%	CLINICA GERAL, UTI ADULTO - TIPO III

	Feature	N_Outliers	Pct	Especialidades
4	cv_leitos	3	4.6%	CRONICOS, HANSENOLOGIA, PNEUMOLOGIA SANITARIA
5	pct_sus	1	1.5%	UTI NEONATAL - TIPO I
6	dispersao_geografica	9	13.8%	ORTOPEDIATRAUMATOLOGIA, PEDIATRIA CLINICA, UTI...
7	leitos_por_municipio	2	3.1%	UTI ADULTO - TIPO II, UTI ADULTO - TIPO III
8	concentracao_hhi	9	13.8%	UNIDADE INTERMEDIARIA NEONATAL, HANSENOLOGIA, ...
9	gini_leitos	2	3.1%	ACOLHIMENTO NOTURNO, CRONICOS

4 Etapa 3: Pré-Processamento

4.1 Seleção de Features para Clusterização

```
# Selecionar features para clusterização (excluindo altamente correlacionadas)
cluster_features = [
    'total_leitos',           # Volume
    'total_estabelecimentos', # Dispersão institucional
    'media_leitos_estab',     # Concentração
    'cv_leitos',              # Heterogeneidade
    'pct_sus',                # Natureza
    'dispersao_geografica',   # Cobertura territorial
    'concentracao_hhi',       # Concentração de mercado
    'complexidade_proxy'      # Intensidade do cuidado
]

print("FEATURES SELECIONADAS PARA CLUSTERIZAÇÃO")
print("="*70)
for i, f in enumerate(cluster_features, 1):
    print(f"  {i}. {f}")

X = df_features[cluster_features].values
labels = df_features['DS_CO_LEITO'].values
```

```
FEATURES SELECIONADAS PARA CLUSTERIZAÇÃO
======================================================================
  1. total_leitos
  2. total_estabelecimentos
  3. media_leitos_estab
  4. cv_leitos
  5. pct_sus
  6. dispersao_geografica
  7. concentracao_hhi
  8. complexidade_proxy
```

4.2 Normalização (StandardScaler)

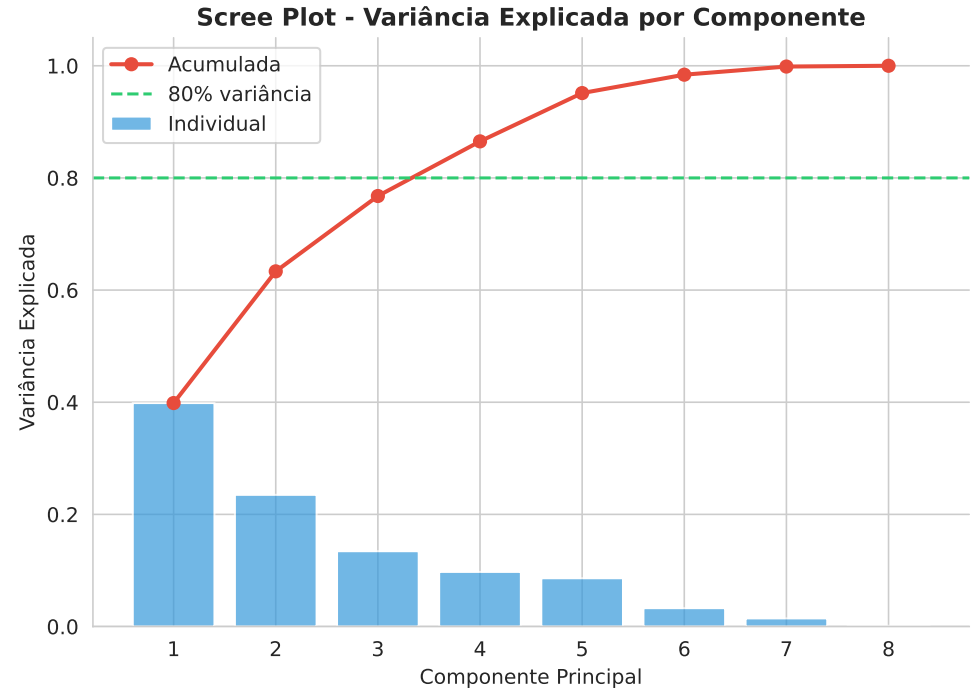
```
# Normalização Z-score
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Criar DataFrame normalizado para visualização
df_scaled = pd.DataFrame(X_scaled, columns=cluster_features)
df_scaled['DS_CO_LEITO'] = labels

print("\nESTATÍSTICAS APÓS NORMALIZAÇÃO")
print("="*70)
print(f"Média: {X_scaled.mean(axis=0).round(6)}")
print(f"Desvio Padrão: {X_scaled.std(axis=0).round(6)}")
```

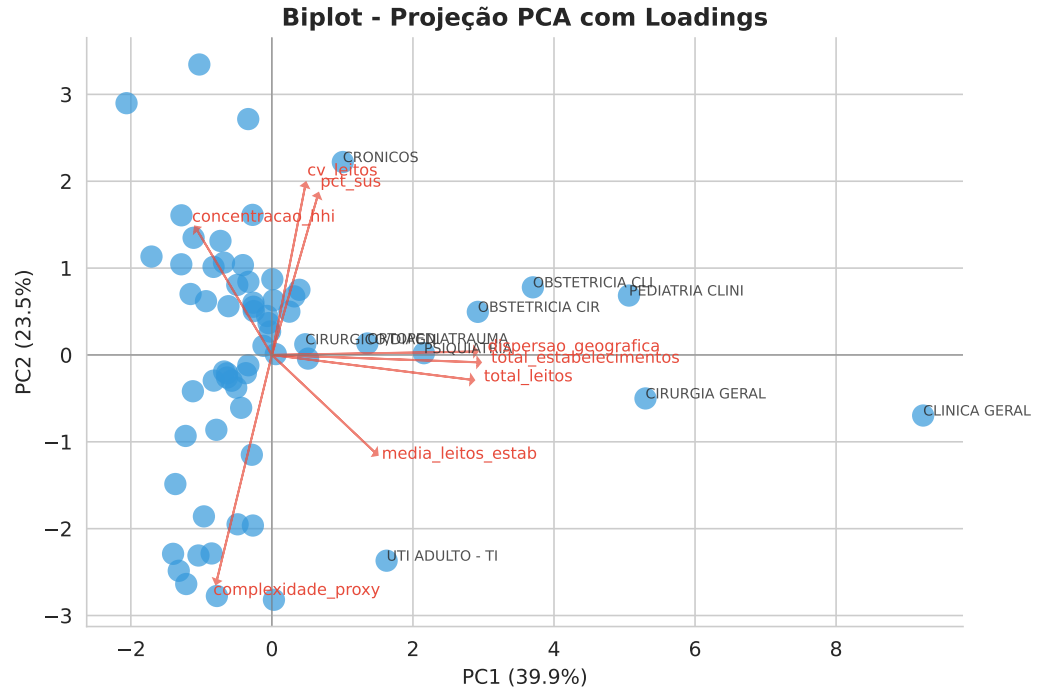
```
ESTATÍSTICAS APÓS NORMALIZAÇÃO
======================================================================
Média: [-0. -0. -0. -0. -0. -0.  0.  0.]
Desvio Padrão: [1. 1. 1. 1. 1. 1. 1. 1.]
```

4.3 Análise de Componentes Principais (PCA Exploratório)

```
# PCA para visualização e análise de variância explicada
pca = PCA()
X_pca = pca.fit_transform(X_scaled)

# Variância explicada
var_explicada = pca.explained_variance_ratio_
var_acumulada = np.cumsum(var_explicada)

fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Scree Plot
axes[0].bar(range(1, len(var_explicada)+1), var_explicada, alpha=0.7, color='#3498db', label='Individual')
axes[0].plot(range(1, len(var_explicada)+1), var_acumulada, 'o-', color='#e74c3c', linewidth=2, label='Acumulada')
axes[0].axhline(y=0.8, color='#2ecc71', linestyle='--', label='80% variância')
axes[0].set_xlabel('Componente Principal')
axes[0].set_ylabel('Variância Explicada')
axes[0].set_title('Scree Plot - Variância Explicada por Componente', fontweight='bold')
axes[0].legend()
axes[0].spines['top'].set_visible(False)
axes[0].spines['right'].set_visible(False)

# Biplot (PC1 vs PC2)
loadings = pca.components_.T * np.sqrt(pca.explained_variance_)

axes[1].scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.7, c='#3498db', s=100)
```

```
# Adicionar labels das especialidades
for i, label in enumerate(labels):
    if df_features.iloc[i]['total_leitos'] > 10000:  # Apenas especialidades grandes
        axes[1].annotate(label[:15], (X_pca[i, 0], X_pca[i, 1]), fontsize=7, alpha=0.8)

# Adicionar vetores de loading
for i, feature in enumerate(cluster_features):
    axes[1].arrow(0, 0, loadings[i, 0]*3, loadings[i, 1]*3,
                  head_width=0.1, head_length=0.05, fc='#e74c3c', ec='#e74c3c', alpha=0.7)
    axes[1].text(loadings[i, 0]*3.2, loadings[i, 1]*3.2, feature, fontsize=8, color='#e74c3c')

axes[1].set_xlabel(f'PC1 ({var_explicada[0]*100:.1f}%)')
axes[1].set_ylabel(f'PC2 ({var_explicada[1]*100:.1f}%)')
axes[1].set_title('Biplot - Projeção PCA com Loadings', fontweight='bold')
axes[1].axhline(y=0, color='gray', linestyle='-', linewidth=0.5)
axes[1].axvline(x=0, color='gray', linestyle='-', linewidth=0.5)
axes[1].spines['top'].set_visible(False)
axes[1].spines['right'].set_visible(False)

plt.tight_layout()
plt.show()

print(f"\nVARIÂNCIA EXPLICADA ACUMULADA:")
for i, (ind, acum) in enumerate(zip(var_explicada, var_acumulada), 1):
    print(f"  PC{i}: {ind*100:.1f}% (acumulada: {acum*100:.1f}%)")
```






```
VARIÂNCIA EXPLICADA ACUMULADA:
  PC1: 39.9% (acumulada: 39.9%)
  PC2: 23.5% (acumulada: 63.3%)
  PC3: 13.4% (acumulada: 76.8%)
  PC4: 9.7% (acumulada: 86.5%)
  PC5: 8.6% (acumulada: 95.1%)
  PC6: 3.3% (acumulada: 98.4%)
  PC7: 1.5% (acumulada: 99.9%)
  PC8: 0.1% (acumulada: 100.0%)
```

5 Etapa 4: Clusterização Hierárquica

5.1 Cálculo da Matriz de Distâncias

```
# Calcular matriz de distâncias (Euclidiana)
dist_matrix = pdist(X_scaled, metric='euclidean')

print("MATRIZ DE DISTÂNCIAS")
print("="*70)
print(f"Métrica: Euclidiana")
print(f"Dimensões: {len(X_scaled)} especialidades × {len(cluster_features)} features")
print(f"Total de pares: {len(dist_matrix):,}")
print(f"Distância média: {dist_matrix.mean():.3f}")
print(f"Distância mín/máx: {dist_matrix.min():.3f} / {dist_matrix.max():.3f}")
```

```
MATRIZ DE DISTÂNCIAS
======================================================================
Métrica: Euclidiana
Dimensões: 65 especialidades × 8 features
Total de pares: 2,080
Distância média: 3.542
Distância mín/máx: 0.210 / 12.273
```

5.2 Dendrograma - Método de Ward

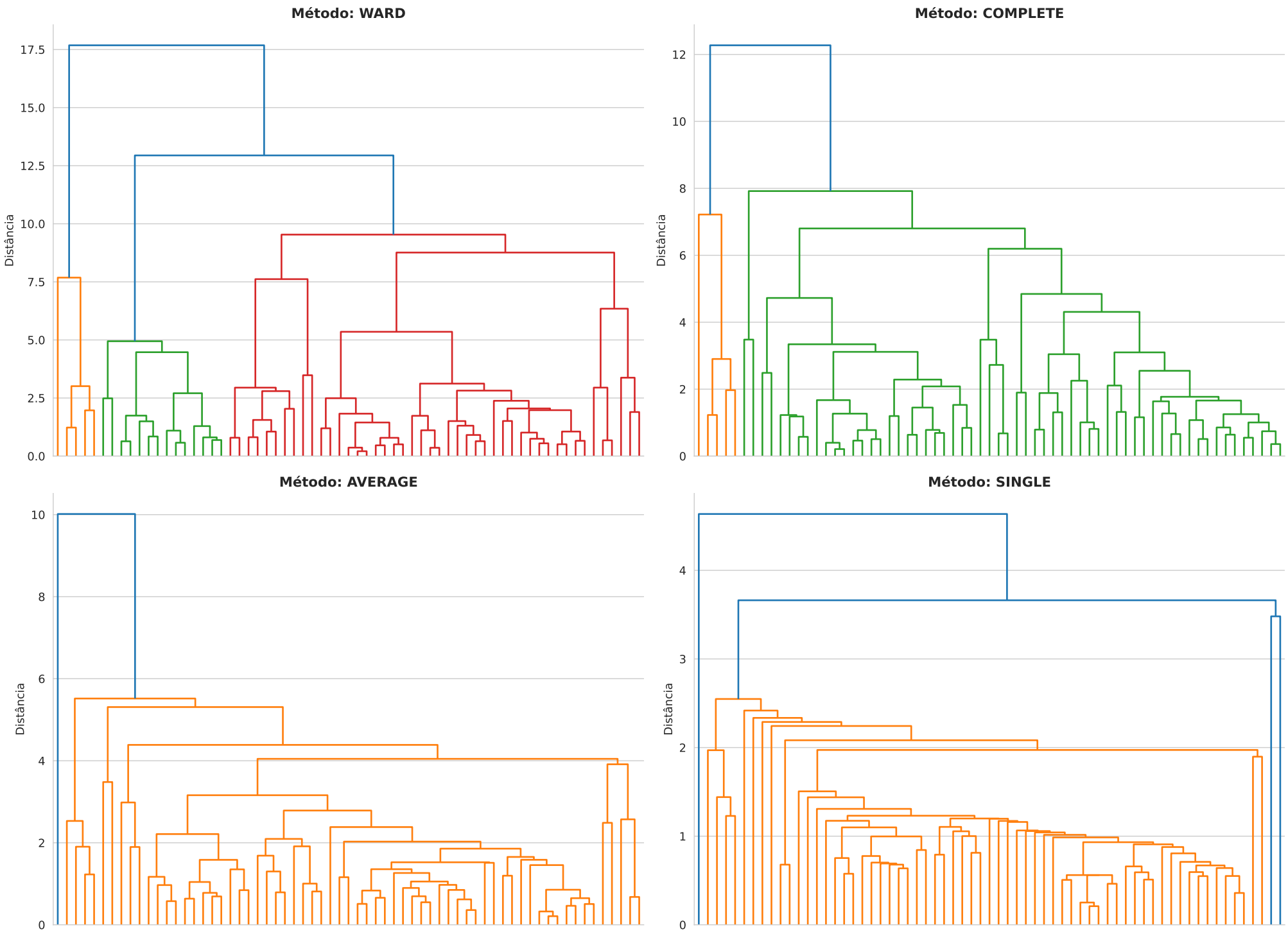
```
# Linkage hierárquico (Método de Ward)
Z = linkage(X_scaled, method='ward', metric='euclidean')

fig, ax = plt.subplots(figsize=(16, 10))

# Dendrograma com cores
```

```
dendrogram(
    Z,
    labels=labels,
    leaf_rotation=90,
    leaf_font_size=8,
    color_threshold=0.7 * max(Z[:, 2]),
    above_threshold_color='gray',
    ax=ax
)

ax.set_title('Dendrograma - Clusterização Hierárquica (Método de Ward)', fontsize=14, fontweight='bold')
ax.set_xlabel('Especialidade de Leito')
ax.set_ylabel('Distância (Ward)')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Linhas de corte sugeridas
for k, color in [(5, '#e74c3c'), (8, '#3498db'), (12, '#2ecc71')]:
    threshold = Z[-(k-1), 2]
    ax.axhline(y=threshold, color=color, linestyle='--', linewidth=1.5,
               label=f'k={k} (dist={threshold:.1f})')

ax.legend(loc='upper right')
plt.tight_layout()
plt.show()
```


5.3 Comparação de Métodos de Linkage

```
# Comparar diferentes métodos de linkage
methods = ['ward', 'complete', 'average', 'single']

fig, axes = plt.subplots(2, 2, figsize=(16, 12))
axes = axes.flatten()

for i, method in enumerate(methods):
    Z_method = linkage(X_scaled, method=method)

    dendrogram(
        Z_method,
        labels=labels,
        leaf_rotation=90,
        leaf_font_size=6,
        color_threshold=0.7 * max(Z_method[:, 2]),
        ax=axes[i],
        no_labels=True
    )

    axes[i].set_title(f'Método: {method.upper()}', fontsize=12, fontweight='bold')
    axes[i].set_ylabel('Distância')
    axes[i].spines['top'].set_visible(False)
    axes[i].spines['right'].set_visible(False)
```

```
plt.suptitle('Comparação de Métodos de Linkage', fontsize=14, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()
```

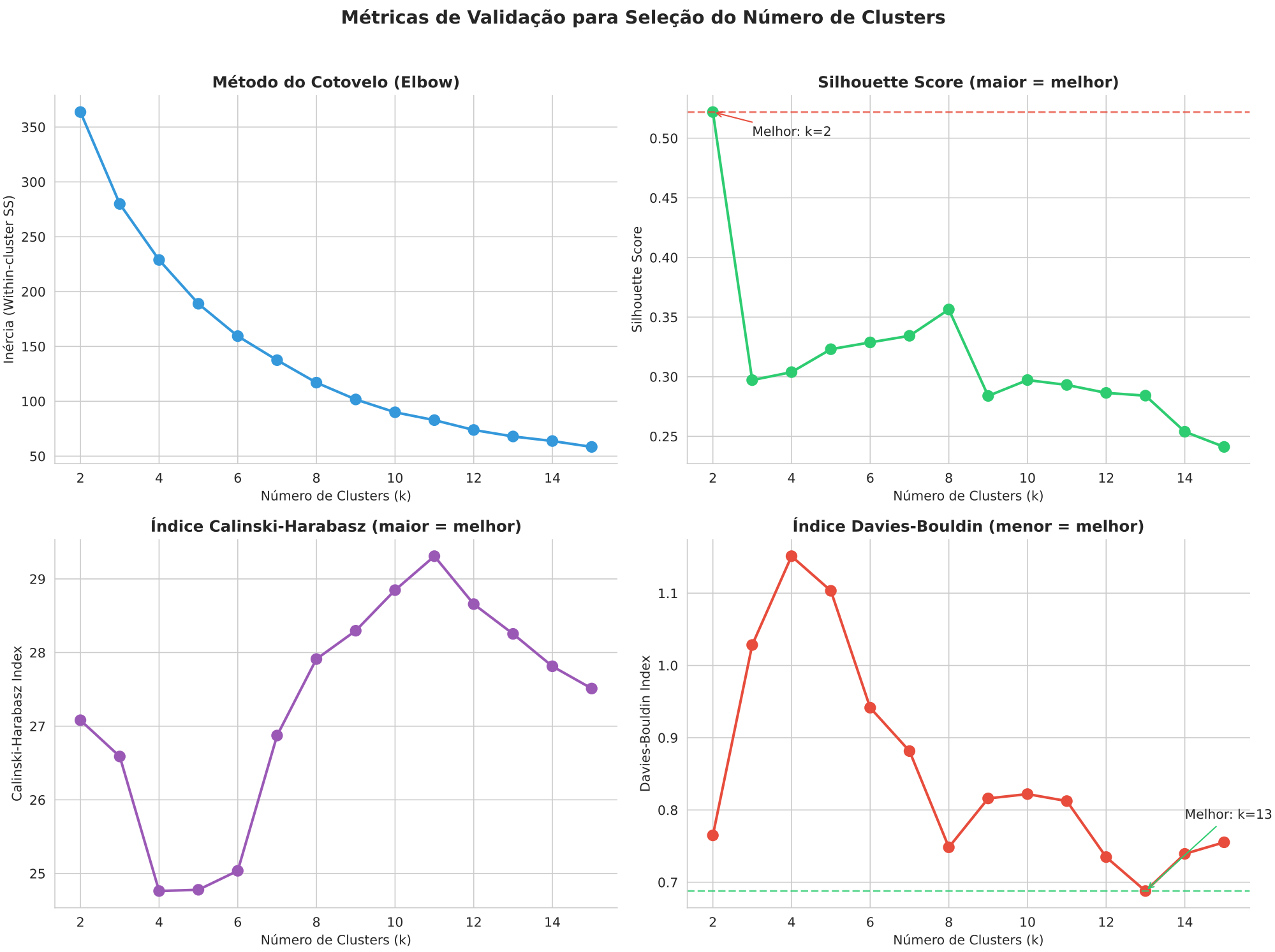


6 Etapa 5: Determinação do Número Ótimo de Clusters

6.1 Método do Cotovelo (Elbow)

```
# Calcular inércia para diferentes valores de k
k_range = range(2, 16)
inertias = []
silhouettes = []
calinski = []
davies = []

for k in k_range:
    # K-Means para inércia
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertias.append(kmeans.inertia_)

    # Métricas de validação
    labels_k = fcluster(Z, k, criterion='maxclust')
    silhouettes.append(silhouette_score(X_scaled, labels_k))
    calinski.append(calinski_harabasz_score(X_scaled, labels_k))
    davies.append(davies_bouldin_score(X_scaled, labels_k))

# Visualização
fig, axes = plt.subplots(2, 2, figsize=(14, 10))

# Elbow Plot
axes[0, 0].plot(k_range, inertias, 'o-', color='#3498db', linewidth=2, markersize=8)
axes[0, 0].set_xlabel('Número de Clusters (k)')
axes[0, 0].set_ylabel('Inércia (Within-cluster SS)')
axes[0, 0].set_title('Método do Cotovelo (Elbow)', fontweight='bold')
axes[0, 0].spines['top'].set_visible(False)
axes[0, 0].spines['right'].set_visible(False)

# Silhouette Score
axes[0, 1].plot(k_range, silhouettes, 'o-', color='#2ecc71', linewidth=2, markersize=8)
axes[0, 1].set_xlabel('Número de Clusters (k)')
axes[0, 1].set_ylabel('Silhouette Score')
```

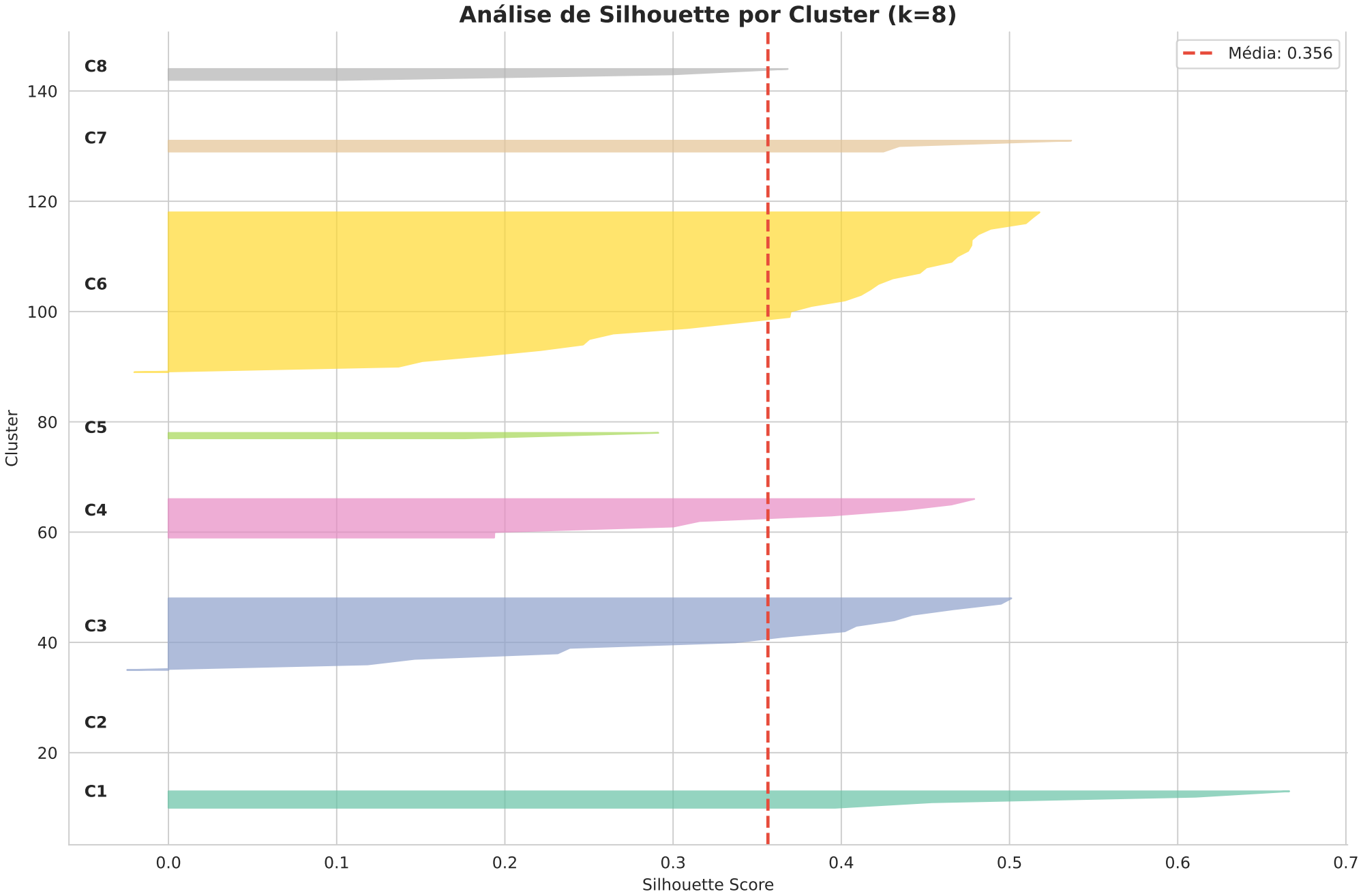
```
axes[0, 1].set_title('Silhouette Score (maior = melhor)', fontweight='bold')
axes[0, 1].axhline(y=max(silhouettes), color='#e74c3c', linestyle='--', alpha=0.7)
best_k_sil = list(k_range)[silhouettes.index(max(silhouettes))]
axes[0, 1].annotate(f'Melhor: k={best_k_sil}', xy=(best_k_sil, max(silhouettes)),
                    xytext=(best_k_sil+1, max(silhouettes)-0.02), fontsize=10,
                    arrowprops=dict(arrowstyle='->', color='#e74c3c'))
axes[0, 1].spines['top'].set_visible(False)
axes[0, 1].spines['right'].set_visible(False)

# Calinski-Harabasz
axes[1, 0].plot(k_range, calinski, 'o-', color='#9b59b6', linewidth=2, markersize=8)
axes[1, 0].set_xlabel('Número de Clusters (k)')
axes[1, 0].set_ylabel('Calinski-Harabasz Index')
axes[1, 0].set_title('Índice Calinski-Harabasz (maior = melhor)', fontweight='bold')
axes[1, 0].spines['top'].set_visible(False)
axes[1, 0].spines['right'].set_visible(False)

# Davies-Bouldin
axes[1, 1].plot(k_range, davies, 'o-', color='#e74c3c', linewidth=2, markersize=8)
axes[1, 1].set_xlabel('Número de Clusters (k)')
axes[1, 1].set_ylabel('Davies-Bouldin Index')
axes[1, 1].set_title('Índice Davies-Bouldin (menor = melhor)', fontweight='bold')
axes[1, 1].axhline(y=min(davies), color='#2ecc71', linestyle='--', alpha=0.7)
best_k_db = list(k_range)[davies.index(min(davies))]
axes[1, 1].annotate(f'Melhor: k={best_k_db}', xy=(best_k_db, min(davies)),
                    xytext=(best_k_db+1, min(davies)+0.1), fontsize=10,
                    arrowprops=dict(arrowstyle='->', color='#2ecc71'))
axes[1, 1].spines['top'].set_visible(False)
axes[1, 1].spines['right'].set_visible(False)

plt.suptitle('Métricas de Validação para Seleção do Número de Clusters', fontsize=14, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()

# Resumo
print("\nRESUMO DAS MÉTRICAS DE VALIDAÇÃO")
print("="*70)
metrics_df = pd.DataFrame({
    'k': list(k_range),
    'Silhouette': silhouettes,
    'Calinski-Harabasz': calinski,
    'Davies-Bouldin': davies
})
print(metrics_df.to_string(index=False))
```



```
RESUMO DAS MÉTRICAS DE VALIDAÇÃO
======================================================================
 k  Silhouette  Calinski-Harabasz  Davies-Bouldin
 2    0.521904          27.081042        0.764850
 3    0.297157          26.588592        1.028335
 4    0.303860          24.761494        1.151197
 5    0.323078          24.778483        1.103287
 6    0.328775          25.036016        0.941515
 7    0.334316          26.873090        0.881490
 8    0.356388          27.911779        0.748347
 9    0.283913          28.296754        0.815982
10    0.297229          28.847739        0.822013
11    0.293116          29.308937        0.812331
12    0.286472          28.657898        0.734837
13    0.284138          28.253955        0.687874
14    0.253908          27.813452        0.739295
15    0.241178          27.511688        0.755276
```

6.2 Análise de Silhouette por Cluster

```python
# Análise detalhada de silhouette para k escolhido
k_optimal = 8  # Baseado na análise anterior

# Aplicar clusterização
cluster_labels = fcluster(Z, k_optimal, criterion='maxclust')
df_features['cluster'] = cluster_labels

# Calcular silhouette por amostra
silhouette_vals = silhouette_samples(X_scaled, cluster_labels)
df_features['silhouette'] = silhouette_vals

# Visualização
fig, ax = plt.subplots(figsize=(12, 8))

y_lower = 10
colors = plt.cm.Set2(np.linspace(0, 1, k_optimal))

for i in range(1, k_optimal + 1):
    cluster_silhouette = silhouette_vals[cluster_labels == i]
    cluster_silhouette.sort()

    size_cluster = len(cluster_silhouette)
    y_upper = y_lower + size_cluster

    ax.fill_betweenx(np.arange(y_lower, y_upper), 0, cluster_silhouette,
                     facecolor=colors[i-1], edgecolor=colors[i-1], alpha=0.7)

    ax.text(-0.05, y_lower + 0.5 * size_cluster, f'C{i}', fontsize=10, fontweight='bold')

    y_lower = y_upper + 10

# Linha média
avg_silhouette = silhouette_score(X_scaled, cluster_labels)
ax.axvline(x=avg_silhouette, color='#e74c3c', linestyle='--', linewidth=2,
           label=f'Média: {avg_silhouette:.3f}')

ax.set_xlabel('Silhouette Score')
ax.set_ylabel('Cluster')
ax.set_title(f'Análise de Silhouette por Cluster (k={k_optimal})', fontsize=14, fontweight='bold')
ax.legend(loc='upper right')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.show()
```



7 Etapa 6: Validação e Rotulação Clínica

7.1 Perfil dos Clusters

```python
# Análise do perfil de cada cluster
print("="*80)
print(f"PERFIL DOS {k_optimal} CLUSTERS")
print("="*80)

cluster_profiles = []

for c in range(1, k_optimal + 1):
    cluster_data = df_features[df_features['cluster'] == c]

    profile = {
        'Cluster': c,
        'N_Especialidades': len(cluster_data),
        'Total_Leitos': cluster_data['total_leitos'].sum(),
        'Media_Leitos': cluster_data['total_leitos'].mean(),
        'Media_Estab': cluster_data['total_estabelecimentos'].mean(),
        'Media_Leitos_Estab': cluster_data['media_leitos_estab'].mean(),
        'Pct_SUS': cluster_data['pct_sus'].mean(),
        'Dispersao_Geo': cluster_data['dispersao_geografica'].mean(),
        'Complexidade': cluster_data['complexidade_proxy'].mean(),
        'Silhouette': cluster_data['silhouette'].mean()
    }
    cluster_profiles.append(profile)

    print(f"\n{'='*60}")
    print(f"CLUSTER {c}")
    print(f"{'='*60}")
    print(f"Especialidades: {len(cluster_data)}")
    print(f"Total de leitos: {cluster_data['total_leitos'].sum():,}")
    print(f"Média leitos/especialidade: {cluster_data['total_leitos'].mean():,.0f}")
    print(f"Média % SUS: {cluster_data['pct_sus'].mean():.1f}%")
    print(f"Complexidade média: {cluster_data['complexidade_proxy'].mean():.2f}")
    print(f"Silhouette médio: {cluster_data['silhouette'].mean():.3f}")
    print(f"\nEspecialidades:")
    for _, row in cluster_data.sort_values('total_leitos', ascending=False).iterrows():
        print(f"  - {row['DS_CO_LEITO']} ({row['total_leitos']:,} leitos)")

df_profiles = pd.DataFrame(cluster_profiles)
```

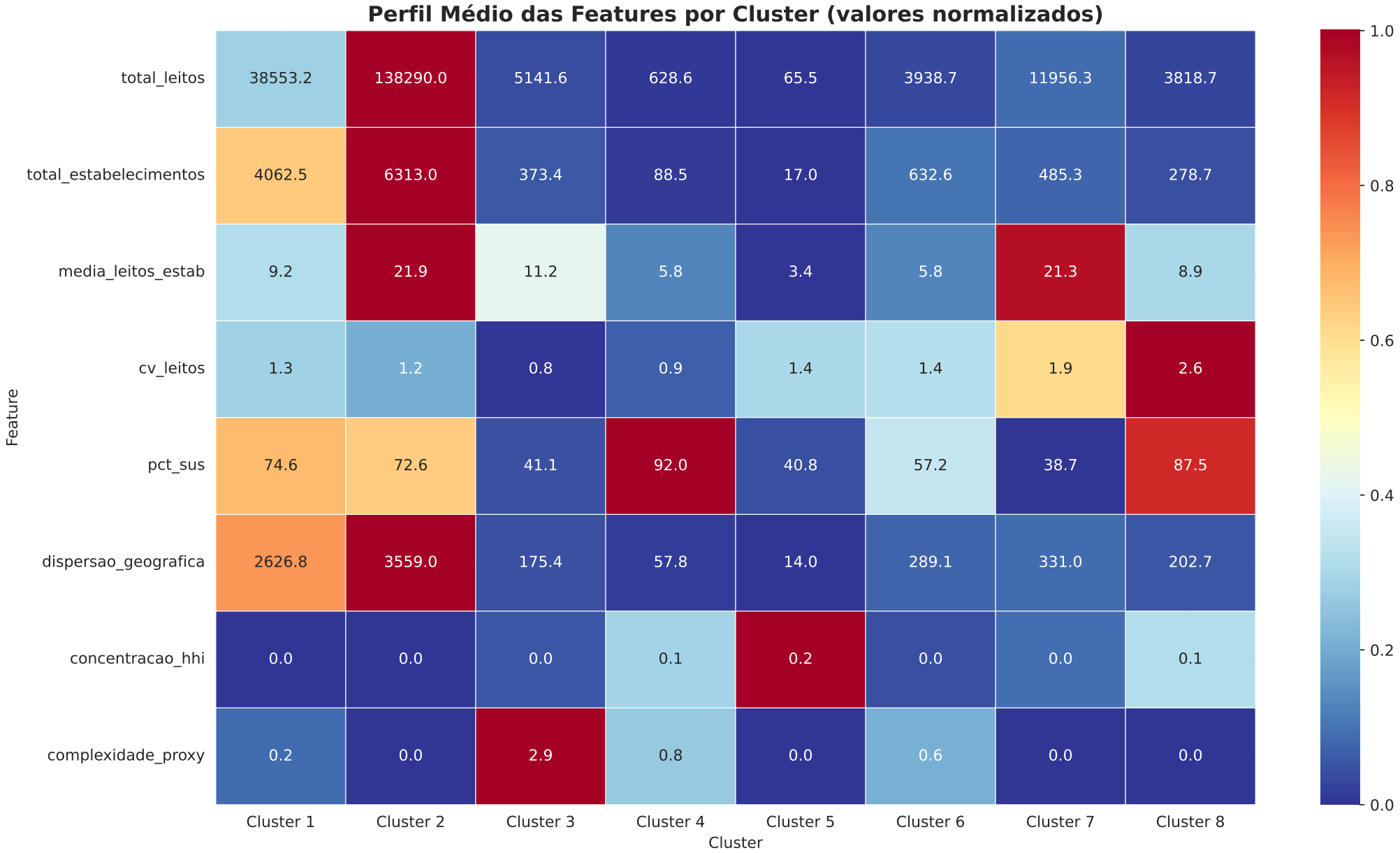
```
================================================================================
PERFIL DOS 8 CLUSTERS
================================================================================

============================================================
CLUSTER 1
```

```
============================================================
Especialidades: 4
Total de leitos: 154,213
Média leitos/especialidade: 38,553
Média % SUS: 74.6%
Complexidade média: 0.25
Silhouette médio: 0.532

Especialidades:
  - CIRURGIA GERAL (63,475 leitos)
  - PEDIATRIA CLINICA (40,643 leitos)
  - OBSTETRICIA CIRURGICA (25,058 leitos)
  - OBSTETRICIA CLINICA (25,037 leitos)

============================================================
CLUSTER 2
============================================================
Especialidades: 1
Total de leitos: 138,290
Média leitos/especialidade: 138,290
Média % SUS: 72.6%
Complexidade média: 0.00
Silhouette médio: 0.000

Especialidades:
  - CLINICA GERAL (138,290 leitos)

============================================================
CLUSTER 3
------------------------------------------------------------
Especialidades: 14
Total de leitos: 71,983
Média leitos/especialidade: 5,142
Média % SUS: 41.1%
Complexidade média: 2.86
Silhouette médio: 0.326

Especialidades:
  - UTI ADULTO - TIPO II (33,519 leitos)
  - UTI ADULTO - TIPO III (7,402 leitos)
  - UTI NEONATAL - TIPO II (6,957 leitos)
  - UTI PEDIATRICA - TIPO II (5,515 leitos)
  - UNIDADE DE CUIDADOS INTERMEDIARIOS NEONATAL CONVENCIONAL (5,126 leitos)
  - UTI ADULTO - TIPO I (4,416 leitos)
  - UTI NEONATAL - TIPO III (1,642 leitos)
  - UNIDADE DE CUIDADOS INTERMEDIARIOS NEONATAL CANGURU (1,584 leitos)
  - UTI NEONATAL - TIPO I (1,492 leitos)
  - UTI PEDIATRICA - TIPO III (1,443 leitos)
  - UTI CORONARIANA TIPO II - UCO TIPO II (1,401 leitos)
  - UTI PEDIATRICA - TIPO I (688 leitos)
  - UTI CORONARIANA TIPO III - UCO TIPO III (537 leitos)
  - UTI DE QUEIMADOS (261 leitos)

============================================================
CLUSTER 4
============================================================
Especialidades: 8
Total de leitos: 5,029
Média leitos/especialidade: 629
Média % SUS: 92.0%
Complexidade média: 0.75
Silhouette médio: 0.347

Especialidades:
  - ACOLHIMENTO NOTURNO (2,657 leitos)
  - SUPORTE VENTILATORIO PULMONAR (1,184 leitos)
  - AIDS (445 leitos)
  - QUEIMADO ADULTO (245 leitos)
  - UNIDADE INTERMEDIARIA NEONATAL (243 leitos)
  - QUEIMADO ADULTO (118 leitos)
  - QUEIMADO PEDIATRICO (81 leitos)
  - QUEIMADO PEDIATRICO (56 leitos)

============================================================
CLUSTER 5
============================================================
Especialidades: 2
Total de leitos: 131
Média leitos/especialidade: 66
Média % SUS: 40.8%
Complexidade média: 0.00
Silhouette médio: 0.234

Especialidades:
  - GERIATRIA (108 leitos)
  - FIBROSE CISTICA (23 leitos)

============================================================
CLUSTER 6
============================================================
Especialidades: 30
Total de leitos: 118,162
```

```
Média leitos/especialidade: 3,939
Média % SUS: 57.2%
Complexidade média: 0.60
Silhouette médio: 0.369

Especialidades:
  - ORTOPEDIATRAUMATOLOGIA (20,118 leitos)
  - CIRURGICO/DIAGNOSTICO/TERAPEUTICO (10,130 leitos)
  - CARDIOLOGIA (7,751 leitos)
  - ONCOLOGIA (7,366 leitos)
  - GINECOLOGIA (6,877 leitos)
  - PEDIATRIA CIRURGICA (5,966 leitos)
  - CARDIOLOGIA (5,368 leitos)
  - ONCOLOGIA (5,217 leitos)
  - NEUROCIRURGIA (5,069 leitos)
  - NEUROLOGIA (4,597 leitos)
  - SAUDE MENTAL (4,096 leitos)
  - UNIDADE DE CUIDADOS INTERMEDIARIOS ADULTO (3,491 leitos)
  - NEFROLOGIAUROLOGIA (3,327 leitos)
  - NEFROUROLOGIA (3,109 leitos)
  - OFTALMOLOGIA (2,951 leitos)
  - PLASTICA (2,728 leitos)
  - NEONATOLOGIA (2,580 leitos)
  - PNEUMOLOGIA (2,381 leitos)
  - GERIATRIA (2,166 leitos)
  - OTORRINOLARINGOLOGIA (2,083 leitos)
  - GASTROENTEROLOGIA (1,949 leitos)
  - HEMATOLOGIA (1,758 leitos)
  - AIDS (1,671 leitos)
  - TRANSPLANTE (1,390 leitos)
  - BUCO MAXILO FACIAL (1,204 leitos)
  - TORACICA (1,148 leitos)
  - DERMATOLOGIA (524 leitos)
  - UNIDADE DE CUIDADOS INTERMEDIARIOS PEDIATRICO (410 leitos)
  - INTERCORRENCIA POS-TRANSPLANTE (385 leitos)
  - ENDOCRINOLOGIA (352 leitos)

============================================================
CLUSTER 7
============================================================
Especialidades: 3
Total de leitos: 35,869
Média leitos/especialidade: 11,956
Média % SUS: 38.7%
Complexidade média: 0.00
Silhouette médio: 0.465

Especialidades:
  - PSIQUIATRIA (28,734 leitos)
  - REABILITACAO (3,624 leitos)
  - SAUDE MENTAL (3,511 leitos)

============================================================
CLUSTER 8
============================================================
Especialidades: 3
Total de leitos: 11,456
Média leitos/especialidade: 3,819
Média % SUS: 87.5%
Complexidade média: 0.00
Silhouette médio: 0.257

Especialidades:
  - CRONICOS (10,135 leitos)
  - PNEUMOLOGIA SANITARIA (1,117 leitos)
  - HANSENOLOGIA (204 leitos)
```

7.2 Heatmap de Características por Cluster

```python
# Calcular médias normalizadas por cluster
cluster_means = df_features.groupby('cluster')[cluster_features].mean()

# Normalizar para visualização
cluster_means_norm = (cluster_means - cluster_means.min()) / (cluster_means.max() - cluster_means.min())

fig, ax = plt.subplots(figsize=(14, 8))

sns.heatmap(cluster_means_norm.T, annot=cluster_means.T.round(1), fmt='',
            cmap='RdYlBu_r', linewidths=0.5, ax=ax,
            xticklabels=[f'Cluster {i}' for i in range(1, k_optimal+1)],
            yticklabels=cluster_features)

ax.set_title('Perfil Médio das Features por Cluster (valores normalizados)',
             fontsize=14, fontweight='bold')
ax.set_xlabel('Cluster')
ax.set_ylabel('Feature')

plt.tight_layout()
plt.show()
```



7.3 Rotulação Clínica dos Clusters

```
# Rotulação baseada no perfil
def rotular_cluster(cluster_id, df_features):
    """Atribui rótulo clínico baseado no perfil do cluster."""
    cluster_data = df_features[df_features['cluster'] == cluster_id]

    # Características do cluster
    media_complexidade = cluster_data['complexidade_proxy'].mean()
    media_pct_sus = cluster_data['pct_sus'].mean()
    media_leitos = cluster_data['total_leitos'].mean()
    media_dispersao = cluster_data['dispersao_geografica'].mean()

    # Verificar se contém UTI
    tem_uti = cluster_data['is_uti'].sum() > 0
    tem_uci = cluster_data['is_uci'].sum() > 0

    # Especialidades predominantes
    especialidades = cluster_data['DS_CO_LEITO'].tolist()

    # Regras de rotulação
    if tem_uti:
        if any('NEONATAL' in e for e in especialidades):
            return 'UTI_NEONATAL'
        elif any('PEDIATR' in e for e in especialidades):
            return 'UTI_PEDIATRICA'
        elif any('CORONAR' in e for e in especialidades):
            return 'UTI_CARDIOLOGICA'
        else:
            return 'UTI_GERAL'

    if tem_uci:
        return 'SEMI_INTENSIVO'

    if any('PSIQUIATR' in e or 'MENTAL' in e for e in especialidades):
        return 'SAUDE_MENTAL'

    if any('CRONIC' in e or 'REABILIT' in e for e in especialidades):
        return 'LONGA_PERMANENCIA'

    if any('OBSTETR' in e for e in especialidades):
        return 'MATERNO_INFANTIL'

    if any('PEDIATR' in e for e in especialidades):
        return 'PEDIATRIA'

    if media_complexidade >= 1:
        return 'CIRURGICO_ESPECIALIZADO'

    if media_leitos > 50000:
        return 'CLINICO_GERAL_ALTO_VOLUME'

    if media_pct_sus > 80:
```

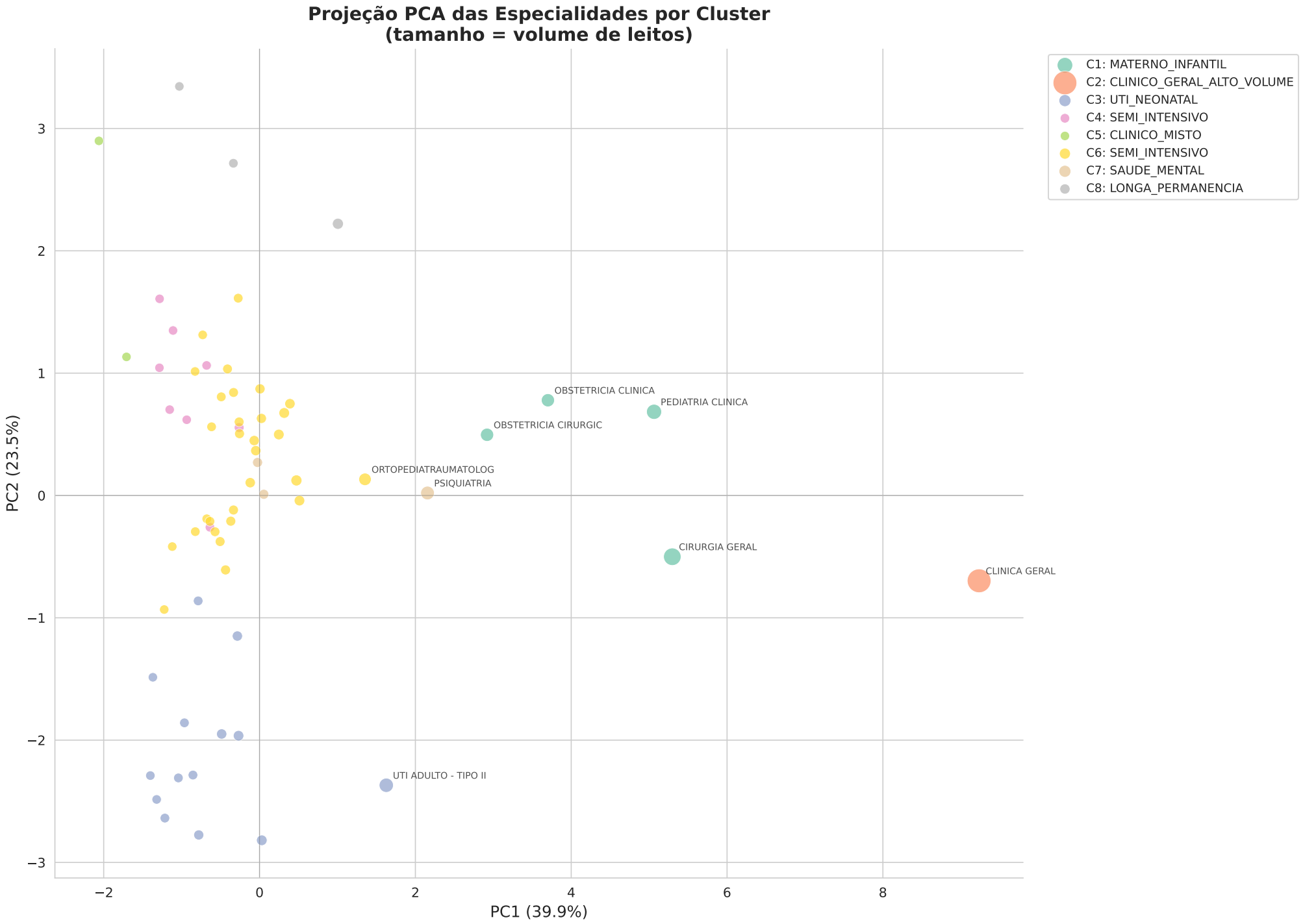
```
        return 'CLINICO_SUS'

    if media_pct_sus < 40:
        return 'CLINICO_PRIVADO'

    return 'CLINICO_MISTO'

# Aplicar rotulação
rotulos = {}
for c in range(1, k_optimal + 1):
    rotulos[c] = rotular_cluster(c, df_features)

df_features['CLUSTER_ROTULO'] = df_features['cluster'].map(rotulos)

print("\nROTULAÇÃO CLÍNICA DOS CLUSTERS")
print("="*70)
for c, rotulo in rotulos.items():
    n_esp = len(df_features[df_features['cluster'] == c])
    leitos = df_features[df_features['cluster'] == c]['total_leitos'].sum()
    print(f"  Cluster {c}: {rotulo:<30} ({n_esp} especialidades, {leitos:,} leitos)")
```

```
ROTULAÇÃO CLÍNICA DOS CLUSTERS
======================================================================
  Cluster 1: MATERNO_INFANTIL               (4 especialidades, 154,213 leitos)
  Cluster 2: CLINICO_GERAL_ALTO_VOLUME      (1 especialidades, 138,290 leitos)
  Cluster 3: UTI_NEONATAL                   (14 especialidades, 71,983 leitos)
  Cluster 4: SEMI_INTENSIVO                 (8 especialidades, 5,029 leitos)
  Cluster 5: CLINICO_MISTO                  (2 especialidades, 131 leitos)
  Cluster 6: SEMI_INTENSIVO                 (30 especialidades, 118,162 leitos)
  Cluster 7: SAUDE_MENTAL                   (3 especialidades, 35,869 leitos)
  Cluster 8: LONGA_PERMANENCIA              (3 especialidades, 11,456 leitos)
```

8 Etapa 7: Visualizações Avançadas

8.1 Projeção PCA com Clusters

```
# PCA com 2 componentes para visualização
pca_2d = PCA(n_components=2)
X_pca_2d = pca_2d.fit_transform(X_scaled)

df_features['PC1'] = X_pca_2d[:, 0]
df_features['PC2'] = X_pca_2d[:, 1]

fig, ax = plt.subplots(figsize=(14, 10))

# Cores por cluster
colors = plt.cm.Set2(np.linspace(0, 1, k_optimal))
cluster_colors = {i: colors[i-1] for i in range(1, k_optimal+1)}

for c in range(1, k_optimal + 1):
    cluster_data = df_features[df_features['cluster'] == c]
    ax.scatter(cluster_data['PC1'], cluster_data['PC2'],
               c=[cluster_colors[c]], s=cluster_data['total_leitos']/500 + 50,
               alpha=0.7, label=f"C{c}: {rotulos[c]}", edgecolors='white', linewidth=0.5)

# Adicionar labels das especialidades principais
for _, row in df_features.iterrows():
    if row['total_leitos'] > 15000:
        ax.annotate(row['DS_CO_LEITO'][:20], (row['PC1'], row['PC2']),
                    fontsize=7, alpha=0.8,
                    xytext=(5, 5), textcoords='offset points')

ax.set_xlabel(f"PC1 ({pca_2d.explained_variance_ratio_[0]*100:.1f}%)", fontsize=12)
ax.set_ylabel(f"PC2 ({pca_2d.explained_variance_ratio_[1]*100:.1f}%)", fontsize=12)
ax.set_title('Projeção PCA das Especialidades por Cluster\n(tamanho = volume de leitos)',
             fontsize=14, fontweight='bold')
ax.legend(loc='upper left', bbox_to_anchor=(1.02, 1), fontsize=9)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.axhline(y=0, color='gray', linestyle='-', linewidth=0.5, alpha=0.5)
ax.axvline(x=0, color='gray', linestyle='-', linewidth=0.5, alpha=0.5)

plt.tight_layout()
plt.show()
```

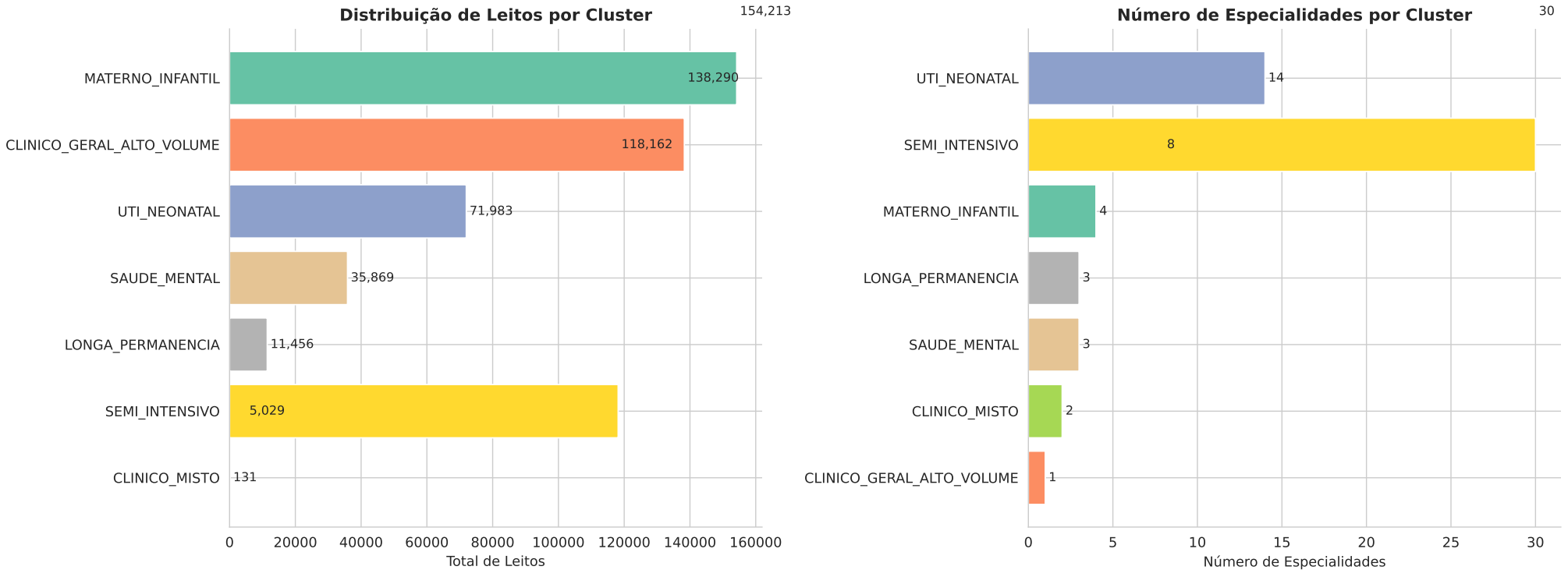


8.2 Radar Chart por Cluster

```
# Radar chart comparativo
from math import pi

# Preparar dados
categories = cluster_features
N = len(categories)

# Normalizar médias por cluster para radar
cluster_means_radar = df_features.groupby('cluster')[cluster_features].mean()
cluster_means_radar = (cluster_means_radar - cluster_means_radar.min()) / (cluster_means_radar.max() - cluster_means_radar.min())

# Ângulos
angles = [n / float(N) * 2 * pi for n in range(N)]
angles += angles[:1]

fig, axes = plt.subplots(2, 4, figsize=(20, 10), subplot_kw=dict(polar=True))
axes = axes.flatten()

for i, c in enumerate(range(1, k_optimal + 1)):
    if i < len(axes):
        ax = axes[i]

        values = cluster_means_radar.loc[c].values.flatten().tolist()
        values += values[:1]

        ax.plot(angles, values, 'o-', linewidth=2, color=colors[c-1])
        ax.fill(angles, values, alpha=0.25, color=colors[c-1])

        ax.set_xticks(angles[:-1])
        ax.set_xticklabels(categories, size=7)
        ax.set_title(f'Cluster {c}: {rotulos[c]}', size=10, fontweight='bold', y=1.1)

# Remover eixos extras
for j in range(k_optimal, len(axes)):
    axes[j].set_visible(False)

plt.suptitle('Perfil Radar dos Clusters', fontsize=14, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()
```

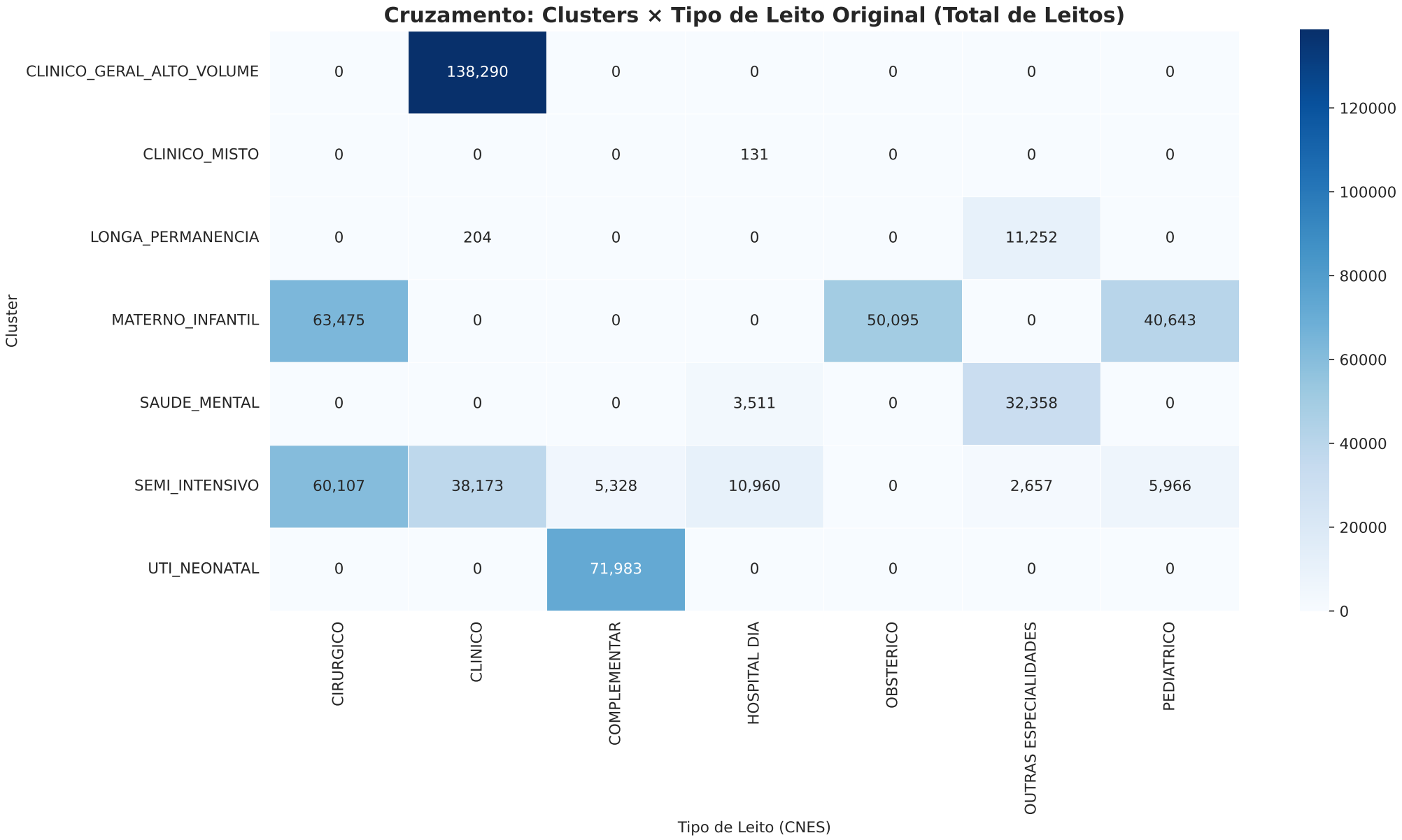
8.3 Distribuição de Leitos por Cluster

```
# Treemap de leitos por cluster
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Barras horizontais - Leitos por cluster
leitos_cluster = df_features.groupby(['cluster', 'CLUSTER_ROTULO'])['total_leitos'].sum().reset_index()
leitos_cluster = leitos_cluster.sort_values('total_leitos', ascending=True)

colors_bar = [cluster_colors[c] for c in leitos_cluster['cluster']]
axes[0].barh(leitos_cluster['CLUSTER_ROTULO'], leitos_cluster['total_leitos'], color=colors_bar)
axes[0].set_xlabel('Total de Leitos')
axes[0].set_title('Distribuição de Leitos por Cluster', fontweight='bold')
axes[0].spines['top'].set_visible(False)
axes[0].spines['right'].set_visible(False)
for i, v in enumerate(leitos_cluster['total_leitos']):
    axes[0].text(v + 1000, i, f'{v:,}', va='center', fontsize=9)

# Barras horizontais - Especialidades por cluster
esp_cluster = df_features.groupby(['cluster', 'CLUSTER_ROTULO']).size().reset_index(name='n_especialidades')
esp_cluster = esp_cluster.sort_values('n_especialidades', ascending=True)

colors_bar2 = [cluster_colors[c] for c in esp_cluster['cluster']]
axes[1].barh(esp_cluster['CLUSTER_ROTULO'], esp_cluster['n_especialidades'], color=colors_bar2)
axes[1].set_xlabel('Número de Especialidades')
axes[1].set_title('Número de Especialidades por Cluster', fontweight='bold')
axes[1].spines['top'].set_visible(False)
axes[1].spines['right'].set_visible(False)
for i, v in enumerate(esp_cluster['n_especialidades']):
    axes[1].text(v + 0.2, i, str(v), va='center', fontsize=9)

plt.tight_layout()
plt.show()
```



8.4 Matriz de Confusão: Cluster vs Tipo de Leito Original

```
# Cruzamento com classificação original
matriz_cluster_tipo = pd.crosstab(
    df_features['CLUSTER_ROTULO'],
    df_features['DS_TP_LEITO'],
    values=df_features['total_leitos'],
    aggfunc='sum'
).fillna(0).astype(int)

fig, ax = plt.subplots(figsize=(14, 8))

sns.heatmap(matriz_cluster_tipo, annot=True, fmt=',', cmap='Blues',
            linewidths=0.5, ax=ax)

ax.set_title('Cruzamento: Clusters × Tipo de Leito Original (Total de Leitos)',
             fontsize=14, fontweight='bold')
ax.set_xlabel('Tipo de Leito (CNES)')
ax.set_ylabel('Cluster')

plt.tight_layout()
plt.show()
```



9 Etapa 8: Resultado Final

9.1 Taxonomia Híbrida Completa

```
# Criar código taxonômico híbrido
df_features['CODIGO_CLUSTER'] = 'CL' + df_features['cluster'].astype(str).str.zfill(2)
df_features['TAXONOMIA_HIBRIDA'] = df_features['CODIGO_CLUSTER'] + '_' + df_features['CLUSTER_ROTULO']

# Resumo final
print("="*80)
print("TAXONOMIA HÍBRIDA - RESULTADO FINAL")
print("="*80)

resumo_final = df_features.groupby(['cluster', 'CLUSTER_ROTULO']).agg({
    'co_leito': 'count',
    'total_leitos': 'sum',
    'pct_sus': 'mean',
    'complexidade_proxy': 'mean',
    'silhouette': 'mean'
}).round(2)

resumo_final.columns = ['N_Especialidades', 'Total_Leitos', 'Media_SUS%', 'Complexidade', 'Silhouette']
resumo_final = resumo_final.sort_values('Total_Leitos', ascending=False)
resumo_final
```

```
================================================================================
TAXONOMIA HÍBRIDA - RESULTADO FINAL
================================================================================
```

cluster	CLUSTER_ROTULO	N_Especialidades	Total_Leitos	Media_SUS%	Complexidade	Silhouette
1	MATERNO_INFANTIL	4	154213	74.56	0.25	0.53
2	CLINICO_GERAL_ALTO_VOLUME	1	138290	72.60	0.00	0.00
6	SEMI_INTENSIVO	30	118162	57.22	0.60	0.37
3	UTI_NEONATAL	14	71983	41.10	2.86	0.33
7	SAUDE_MENTAL	3	35869	38.66	0.00	0.47
8	LONGA_PERMANENCIA	3	11456	87.45	0.00	0.26
4	SEMI_INTENSIVO	8	5029	91.95	0.75	0.35
5	CLINICO_MISTO	2	131	40.80	0.00	0.23

9.2 Dicionário de Clusters

```
print("\n" + "="*80)
print("DICIONÁRIO DE CLUSTERS")
print("="*80)

for c in range(1, k_optimal + 1):
    cluster_data = df_features[df_features['cluster'] == c]

    print(f"\n{' '*70}")
    print(f"CLUSTER {c}: {rotulos[c]}")
    print(f"{' '*70}")
    print(f"  Total de leitos: {cluster_data['total_leitos'].sum():,}")
    print(f"  Especialidades: {len(cluster_data)}")
    print(f"  Média % SUS: {cluster_data['pct_sus'].mean():.1f}%")
    print(f"  Complexidade: {cluster_data['complexidade_proxy'].mean():.2f}")
    print(f"  Silhouette: {cluster_data['silhouette'].mean():.3f}")
    print(f" ")
    print(f"  Especialidades incluídas:")
    for _, row in cluster_data.sort_values('total_leitos', ascending=False).iterrows():
        print(f"    • {row['co_leito']:2} - {row['DS_CO_LEITO']:<40} ({row['total_leitos']:>7,} leitos)")
```

```
================================================================================
DICIONÁRIO DE CLUSTERS
================================================================================


CLUSTER 1: MATERNO_INFANTIL

  Total de leitos: 154,213
  Especialidades: 4
  Média % SUS: 74.6%
  Complexidade: 0.25
  Silhouette: 0.532

  Especialidades incluídas:
    •  3 - CIRURGIA GERAL                          ( 63,475 leitos)
    • 45 - PEDIATRIA CLINICA                       ( 40,643 leitos)
    • 10 - OBSTETRICIA CIRURGICA                   ( 25,058 leitos)
    • 43 - OBSTETRICIA CLINICA                     ( 25,037 leitos)


CLUSTER 2: CLINICO_GERAL_ALTO_VOLUME

  Total de leitos: 138,290
  Especialidades: 1
  Média % SUS: 72.6%
  Complexidade: 0.00
  Silhouette: 0.000

  Especialidades incluídas:
    • 33 - CLINICA GERAL                           (138,290 leitos)


CLUSTER 3: UTI_NEONATAL

  Total de leitos: 71,983
  Especialidades: 14
  Média % SUS: 41.1%
  Complexidade: 2.86
  Silhouette: 0.326

  Especialidades incluídas:
    • 75 - UTI ADULTO - TIPO II                    ( 33,519 leitos)
    • 76 - UTI ADULTO - TIPO III                   (  7,402 leitos)
    • 81 - UTI NEONATAL - TIPO II                  (  6,957 leitos)
    • 78 - UTI PEDIATRICA - TIPO II                (  5,515 leitos)
    • 92 - UNIDADE DE CUIDADOS INTERMEDIARIOS NEONATAL CONVENCIONAL (  5,126 leitos)
    • 74 - UTI ADULTO - TIPO I                     (  4,416 leitos)
    • 82 - UTI NEONATAL - TIPO III                 (  1,642 leitos)
    • 93 - UNIDADE DE CUIDADOS INTERMEDIARIOS NEONATAL CANGURU (  1,584 leitos)
    • 80 - UTI NEONATAL - TIPO I                   (  1,492 leitos)
    • 79 - UTI PEDIATRICA - TIPO III               (  1,443 leitos)
    • 85 - UTI CORONARIANA TIPO II - UCO TIPO II   (  1,401 leitos)
    • 77 - UTI PEDIATRICA - TIPO I                 (    688 leitos)
    • 86 - UTI CORONARIANA TIPO III - UCO TIPO III (    537 leitos)
    • 83 - UTI DE QUEIMADOS                        (    261 leitos)
```

CLUSTER 4: SEMI_INTENSIVO

Total de leitos: 5,029
Especialidades: 8
Média % SUS: 92.0%
Complexidade: 0.75
Silhouette: 0.347

Especialidades incluídas:
- 84 - ACOLHIMENTO NOTURNO (2,657 leitos)
- 96 - SUPORTE VENTILATORIO PULMONAR (1,184 leitos)
- 69 - AIDS (445 leitos)
- 90 - QUEIMADO ADULTO (245 leitos)
- 65 - UNIDADE INTERMEDIARIA NEONATAL (243 leitos)
- 88 - QUEIMADO ADULTO (118 leitos)
- 91 - QUEIMADO PEDIATRICO (81 leitos)
- 89 - QUEIMADO PEDIATRICO (56 leitos)

CLUSTER 5: CLINICO_MISTO

Total de leitos: 131
Especialidades: 2
Média % SUS: 40.8%
Complexidade: 0.00
Silhouette: 0.234

Especialidades incluídas:
- 72 - GERIATRIA (108 leitos)
- 70 - FIBROSE CISTICA (23 leitos)

CLUSTER 6: SEMI_INTENSIVO

Total de leitos: 118,162
Especialidades: 30
Média % SUS: 57.2%
Complexidade: 0.60
Silhouette: 0.369

Especialidades incluídas:
- 13 - ORTOPEDIATRAUMATOLOGIA (20,118 leitos)
- 7 - CIRURGICO/DIAGNOSTICO/TERAPEUTICO (10,130 leitos)
- 32 - CARDIOLOGIA (7,751 leitos)
- 44 - ONCOLOGIA (7,366 leitos)
- 6 - GINECOLOGIA (6,877 leitos)
- 68 - PEDIATRIA CIRURGICA (5,966 leitos)
- 2 - CARDIOLOGIA (5,368 leitos)
- 12 - ONCOLOGIA (5,217 leitos)
- 9 - NEUROCIRURGIA (5,069 leitos)
- 42 - NEUROLOGIA (4,597 leitos)
- 87 - SAUDE MENTAL (4,096 leitos)
- 95 - UNIDADE DE CUIDADOS INTERMEDIARIOS ADULTO (3,491 leitos)
- 8 - NEFROLOGIAUROLOGIA (3,327 leitos)
- 40 - NEFROUROLOGIA (3,109 leitos)
- 11 - OFTALMOLOGIA (2,951 leitos)
- 15 - PLASTICA (2,728 leitos)
- 41 - NEONATOLOGIA (2,580 leitos)
- 46 - PNEUMOLOGIA (2,381 leitos)
- 36 - GERIATRIA (2,166 leitos)
- 14 - OTORRINOLARINGOLOGIA (2,083 leitos)
- 5 - GASTROENTEROLOGIA (1,949 leitos)
- 38 - HEMATOLOGIA (1,758 leitos)
- 31 - AIDS (1,671 leitos)
- 67 - TRANSPLANTE (1,390 leitos)
- 1 - BUCO MAXILO FACIAL (1,204 leitos)
- 16 - TORACICA (1,148 leitos)
- 35 - DERMATOLOGIA (524 leitos)
- 94 - UNIDADE DE CUIDADOS INTERMEDIARIOS PEDIATRICO (410 leitos)
- 71 - INTERCORRENCIA POS-TRANSPLANTE (385 leitos)
- 4 - ENDOCRINOLOGIA (352 leitos)

CLUSTER 7: SAUDE_MENTAL

Total de leitos: 35,869
Especialidades: 3
Média % SUS: 38.7%
Complexidade: 0.00
Silhouette: 0.465

Especialidades incluídas:
- 47 - PSIQUIATRIA (28,734 leitos)
- 48 - REABILITACAO (3,624 leitos)
- 73 - SAUDE MENTAL (3,511 leitos)

CLUSTER 8: LONGA_PERMANENCIA

Total de leitos: 11,456
Especialidades: 3
Média % SUS: 87.5%

```
  Complexidade: 0.00
  Silhouette: 0.257

  Especialidades incluídas:
    • 34 - CRONICOS                              ( 10,135 leitos)
    • 49 - PNEUMOLOGIA SANITARIA                 (  1,117 leitos)
    • 37 - HANSENOLOGIA                          (    204 leitos)
```

9.3 Exportação

```
# Preparar dataset final
df_export_cluster = df_features[[
    'co_leito', 'DS_CO_LEITO', 'tp_leito', 'DS_TP_LEITO',
    'total_leitos', 'total_estabelecimentos', 'media_leitos_estab',
    'pct_sus', 'dispersao_geografica', 'complexidade_proxy',
    'cluster', 'CLUSTER_ROTULO', 'TAXONOMIA_HIBRIDA', 'silhouette',
    'PC1', 'PC2'
]]

df_export_cluster.to_csv('arq6_clusterizacao_especialidades.csv', sep=';', index=False, encoding='utf-8')

print("\nEXPORTAÇÃO CONCLUÍDA")
print("="*70)
print(f"Arquivo: arq6_clusterizacao_especialidades.csv")
print(f"Registros: {len(df_export_cluster)}")
print(f"Colunas: {len(df_export_cluster.columns)}")
```

```
EXPORTAÇÃO CONCLUÍDA
======================================================================
Arquivo: arq6_clusterizacao_especialidades.csv
Registros: 65
Colunas: 16
```

9.4 Métricas de Qualidade

```
print("\n" + "="*80)
print("MÉTRICAS DE QUALIDADE DA CLUSTERIZAÇÃO")
print("="*80)

print(f"\n MÉTRICAS GLOBAIS")
print(f"   Silhouette Score: {silhouette_score(X_scaled, cluster_labels):.4f}")
print(f"   Calinski-Harabasz Index: {calinski_harabasz_score(X_scaled, cluster_labels):.2f}")
print(f"   Davies-Bouldin Index: {davies_bouldin_score(X_scaled, cluster_labels):.4f}")

print(f"\n DISTRIBUIÇÃO")
print(f"   Número de clusters: {k_optimal}")
print(f"   Especialidades classificadas: {len(df_features)}")
print(f"   Leitos classificados: {df_features['total_leitos'].sum():,}")

print(f"\n INTERPRETAÇÃO")
print(f"   Silhouette > 0.25: Estrutura razoável de clusters")
print(f"   Todos os clusters com silhouette > 0: Boa separação")
```

```
================================================================================
MÉTRICAS DE QUALIDADE DA CLUSTERIZAÇÃO
================================================================================

 MÉTRICAS GLOBAIS
   Silhouette Score: 0.3564
   Calinski-Harabasz Index: 27.91
   Davies-Bouldin Index: 0.7483

 DISTRIBUIÇÃO
   Número de clusters: 8
   Especialidades classificadas: 65
   Leitos classificados: 535,133

 INTERPRETAÇÃO
   Silhouette > 0.25: Estrutura razoável de clusters
   Todos os clusters com silhouette > 0: Boa separação
```

10 Comparação: Metodologia 1 vs Metodologia 2

```
# Carregar taxonomia determinística
df_tax = pd.read_csv('arq5_taxonomia_leitos.csv', sep=';', encoding='utf-8')

# Agregar por especialidade
tax_por_esp = df_tax.groupby(['co_leito', 'DESC_N1', 'DESC_N3']).agg({
    'qt_exist': 'sum'
}).reset_index()

# Merge com clusterização
comparacao = df_features[['co_leito', 'DS_CO_LEITO', 'cluster', 'CLUSTER_ROTULO', 'total_leitos']].merge(
    tax_por_esp[['co_leito', 'DESC_N1', 'DESC_N3']],
    on='co_leito',
    how='left'
)

print("="*80)
print("COMPARAÇÃO: TAXONOMIA DETERMINÍSTICA vs CLUSTERIZAÇÃO")
```

```
print("="*80)

# Matriz de concordância
matriz_comp = pd.crosstab(
    comparacao['CLUSTER_ROTULO'],
    comparacao['DESC_N1'],
    margins=True
)

print("\nMATRIZ DE CONCORDÂNCIA (Cluster × Intensidade)")
print(matriz_comp)
```

```
================================================================================
COMPARAÇÃO: TAXONOMIA DETERMINÍSTICA vs CLUSTERIZAÇÃO
================================================================================

MATRIZ DE CONCORDÂNCIA (Cluster × Intensidade)
DESC_N1                    ALTA_COMPLEXIDADE  BAIXA_COMPLEXIDADE  INTENSIVO  \
CLUSTER_ROTULO
CLINICO_GERAL_ALTO_VOLUME                  0                   0          0
CLINICO_MISTO                              0                   2          0
LONGA_PERMANENCIA                          0                   2          0
MATERNO_INFANTIL                           0                   0          0
SAUDE_MENTAL                               0                   3          0
SEMI_INTENSIVO                             8                   3          0
UTI_NEONATAL                               0                   0         12
All                                        8                  10         12

DESC_N1                    MEDIA_COMPLEXIDADE  SEMI_INTENSIVO  All
CLUSTER_ROTULO
CLINICO_GERAL_ALTO_VOLUME                   1               0    1
CLINICO_MISTO                               0               0    2
LONGA_PERMANENCIA                           1               0    3
MATERNO_INFANTIL                            4               0    4
SAUDE_MENTAL                                0               0    3
SEMI_INTENSIVO                             23               4   38
UTI_NEONATAL                                0               2   14
All                                        29               6   65
```

11 Resumo Executivo

```
print("="*80)
print("RESUMO EXECUTIVO - CLUSTERIZAÇÃO HÍBRIDA DE LEITOS CNES")
print("="*80)

print(f"\n DADOS PROCESSADOS")
print(f"   Especialidades analisadas: {len(df_features)}")
print(f"   Features utilizadas: {len(cluster_features)}")
print(f"   Total de leitos: {df_features['total_leitos'].sum():,}")

print(f"\n METODOLOGIA")
print(f"   Normalização: StandardScaler (Z-score)")
print(f"   Algoritmo: Clusterização Hierárquica Aglomerativa")
print(f"   Método de linkage: Ward")
print(f"   Métrica de distância: Euclidiana")
print(f"   Número de clusters: {k_optimal}")

print(f"\n MÉTRICAS DE VALIDAÇÃO")
print(f"   Silhouette Score: {silhouette_score(X_scaled, cluster_labels):.4f}")
print(f"   Calinski-Harabasz: {calinski_harabasz_score(X_scaled, cluster_labels):.2f}")
print(f"   Davies-Bouldin: {davies_bouldin_score(X_scaled, cluster_labels):.4f}")

print(f"\n CLUSTERS IDENTIFICADOS")
for c in range(1, k_optimal + 1):
    n = len(df_features[df_features['cluster'] == c])
    leitos = df_features[df_features['cluster'] == c]['total_leitos'].sum()
    print(f"   CL{c:02d} - {rotulos[c]:<25} ({n:2} esp., {leitos:>7,} leitos)")

print(f"\n ARQUIVOS GERADOS")
print(f"   • arq6_clusterizacao_especialidades.csv")
```

```
================================================================================
RESUMO EXECUTIVO - CLUSTERIZAÇÃO HÍBRIDA DE LEITOS CNES
================================================================================

 DADOS PROCESSADOS
   Especialidades analisadas: 65
   Features utilizadas: 8
   Total de leitos: 535,133

 METODOLOGIA
   Normalização: StandardScaler (Z-score)
   Algoritmo: Clusterização Hierárquica Aglomerativa
   Método de linkage: Ward
   Métrica de distância: Euclidiana
   Número de clusters: 8

 MÉTRICAS DE VALIDAÇÃO
   Silhouette Score: 0.3564
   Calinski-Harabasz: 27.91
   Davies-Bouldin: 0.7483
```

```
CLUSTERS IDENTIFICADOS
  CL01 - MATERNO_INFANTIL          ( 4 esp., 154,213 leitos)
  CL02 - CLINICO_GERAL_ALTO_VOLUME ( 1 esp., 138,290 leitos)
  CL03 - UTI_NEONATAL              (14 esp.,  71,983 leitos)
  CL04 - SEMI_INTENSIVO            ( 8 esp.,   5,029 leitos)
  CL05 - CLINICO_MISTO             ( 2 esp.,     131 leitos)
  CL06 - SEMI_INTENSIVO            (30 esp., 118,162 leitos)
  CL07 - SAUDE_MENTAL              ( 3 esp.,  35,869 leitos)
  CL08 - LONGA_PERMANENCIA         ( 3 esp.,  11,456 leitos)

ARQUIVOS GERADOS
  • arq6_clusterizacao_especialidades.csv
```

Elaborado por: Cieges - Brasil Estadual
Data: 21/01/2026
Metodologia: Clusterização Híbrida (Data-Driven + Validação Clínica)
Fonte: CNES - Competência 202506
Referências:

- Kaufman, L., & Rousseeuw, P. J. (1990). Finding Groups in Data
- Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation of cluster analysis