Validação de Modelos de Clustericação [24E4_3]

PROJETO DA DISCIPLINA - ENTREGA FINAL

Aluno: Marcio Feldmann

Repositório Público GitHub: https://github.com/marciofeld/infnet_validacao_clusterizacao

Parte 1 - Infraestrutura

Para as questões a seguir, você deverá executar códigos em um notebook Jupyter, rodando em ambiente local, certifique-se que:

1. Você está rodando em Python 3.9+
2. Você está usando um ambiente virtual: Virtualenv ou Anaconda
3. Todas as bibliotecas usadas nesse exercícios estão instaladas em um ambiente virtual específico
4. Gere um arquivo de requerimentos (requirements.txt) com os pacotes necessários. É necessário se certificar que a versão do pacote está disponibilizada.
5. Tire um printscreen do ambiente que será usado rodando em sua máquina.
6. Disponibilize os códigos gerados, assim como os artefatos acessórios (requirements.txt) e instruções em um repositório GIT público. (se isso não for feito, o diretório com esses arquivos deverá ser enviado compactado no moodle).

Dica: Gere um relatório rico em gráficos que dêem respaldo aos resultados

In [1]:
```
# Parte 1 - Infraestrutura
# Questão 1 - Você está rodando em Python 3.9+

import sys
print(f"Versão do Python: {sys.version}")
```

```
Versão do Python: 3.12.2 (main, Feb  6 2024, 20:19:44) [Clang 15.0.0 (clan
g-1500.1.0.2.5)]
```

In [2]:
```
# Parte 1 - Infraestrutura
# Questão 2 - Você está usando um ambiente virtual: Virtualenv ou Anacond

def check_environment():
    in_virtual_env = hasattr(sys, 'real_prefix') or (hasattr(sys, 'base_p

    if not in_virtual_env:
        print("Não está rodando em um virtual environment")
        return
```

```
    if 'conda' in sys.version or 'Continuum' in sys.version:
        print("Rodando em ambiente virtual Anaconda")
    else:
        print("Rodando em ambiente virtual venv")

if __name__ == "__main__":
    check_environment()
```

```
Rodando em ambiente virtual venv
```

In [3]:
```
# Parte 1 - Infraestrutura
# Questão 3: Todas as bibliotecas usadas nesse exercícios estão instalada

# Instalação das bibliotecas necessárias
%pip install pandas numpy matplotlib seaborn scikit-learn

# Importação das bibliotecas que serão utilizadas no projeto
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import calinski_harabasz_score, davies_bouldin_score
```

```
Requirement already satisfied: pandas in /Users/marcio/Library/CloudStorag
e/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validacao_Modelo
s_Clusterizacao/.venv/lib/python3.12/site-packages (2.2.3)
Requirement already satisfied: numpy in /Users/marcio/Library/CloudStorag
e/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validacao_Modelo
s_Clusterizacao/.venv/lib/python3.12/site-packages (2.2.0)
Requirement already satisfied: matplotlib in /Users/marcio/Library/CloudSt
orage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validacao_Mo
delos_Clusterizacao/.venv/lib/python3.12/site-packages (3.9.3)
Requirement already satisfied: seaborn in /Users/marcio/Library/CloudStora
ge/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validacao_Model
os_Clusterizacao/.venv/lib/python3.12/site-packages (0.13.2)
Requirement already satisfied: scikit-learn in /Users/marcio/Library/Cloud
Storage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validacao_
Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (1.5.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /Users/marcio/Lib
rary/CloudStorage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_
Validacao_Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from p
andas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /Users/marcio/Library/Cloud
Storage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validacao_
Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from pandas) (20
24.2)
Requirement already satisfied: tzdata>=2022.7 in /Users/marcio/Library/Clo
udStorage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validaca
o_Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from pandas)
(2024.2)
Requirement already satisfied: contourpy>=1.0.1 in /Users/marcio/Library/C
loudStorage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Valida
cao_Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from matplot
lib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /Users/marcio/Library/Cloud
Storage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validacao_
Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from matplotlib)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Users/marcio/Library/
CloudStorage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Valid
acao_Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from matplo
tlib) (4.55.2)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/marcio/Library/
CloudStorage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Valid
acao_Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from matplo
tlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /Users/marcio/Library/Cl
oudStorage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validac
ao_Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from matplotl
ib) (24.2)
Requirement already satisfied: pillow>=8 in /Users/marcio/Library/CloudSto
rage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validacao_Mod
elos_Clusterizacao/.venv/lib/python3.12/site-packages (from matplotlib) (1
1.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/marcio/Library/C
loudStorage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Valida
cao_Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from matplot
lib) (3.2.0)
Requirement already satisfied: scipy>=1.6.0 in /Users/marcio/Library/Cloud
Storage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validacao_
Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from scikit-lear
n) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /Users/marcio/Library/Clou
```

```
dStorage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validacao
_Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from scikit-lea
rn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /Users/marcio/Libra
ry/CloudStorage/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Va
lidacao_Modelos_Clusterizacao/.venv/lib/python3.12/site-packages (from sci
kit-learn) (3.5.0)
Requirement already satisfied: six>=1.5 in /Users/marcio/Library/CloudStor
age/Dropbox/ESTUDOS/Infnet - MIT Inteligencia Artificial/02_Validacao_Mode
los_Clusterizacao/.venv/lib/python3.12/site-packages (from python-dateutil
>=2.8.2->pandas) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:

```
# Parte 1 - Infraestrutura
# Questão 4: Gere um arquivo de requerimentos (requirements.txt) com os p
# É necessário se certificar que a versão do pacote está disponibilizada.

%pip freeze > requirements.txt
print ("Conteúdo do requirements.txt:")
%cat requirements.txt
```

```
Note: you may need to restart the kernel to use updated packages.
Conteúdo do requirements.txt:
anyio==4.7.0
appnope==0.1.4
argon2-cffi==23.1.0
argon2-cffi-bindings==21.2.0
arrow==1.3.0
asttokens==3.0.0
async-lru==2.0.4
attrs==24.2.0
babel==2.16.0
beautifulsoup4==4.12.3
bleach==6.2.0
certifi==2024.8.30
cffi==1.17.1
charset-normalizer==3.4.0
comm==0.2.2
contourpy==1.3.1
cycler==0.12.1
debugpy==1.8.9
decorator==5.1.1
defusedxml==0.7.1
executing==2.1.0
fastjsonschema==2.21.1
fonttools==4.55.2
fqdn==1.5.1
h11==0.14.0
httpcore==1.0.7
httpx==0.28.1
idna==3.10
ipykernel==6.29.5
ipython==8.30.0
ipywidgets==8.1.5
isoduration==20.11.0
jedi==0.19.2
Jinja2==3.1.4
joblib==1.4.2
json5==0.10.0
jsonpointer==3.0.0
jsonschema==4.23.0
jsonschema-specifications==2024.10.1
jupyter==1.1.1
jupyter-console==6.6.3
jupyter-events==0.10.0
jupyter-lsp==2.2.5
jupyter_client==8.6.3
jupyter_core==5.7.2
jupyter_server==2.14.2
jupyter_server_terminals==0.5.3
jupyterlab==4.3.2
jupyterlab_pygments==0.3.0
jupyterlab_server==2.27.3
jupyterlab_widgets==3.0.13
kaggle==1.6.17
kiwisolver==1.4.7
MarkupSafe==3.0.2
matplotlib==3.9.3
matplotlib-inline==0.1.7
mistune==3.0.2
nbclient==0.10.1
```

```
nbconvert==7.16.4
nbformat==5.10.4
nest-asyncio==1.6.0
notebook==7.3.1
notebook_shim==0.2.4
numpy==2.2.0
overrides==7.7.0
packaging==24.2
pandas==2.2.3
pandocfilters==1.5.1
parso==0.8.4
pexpect==4.9.0
pillow==11.0.0
platformdirs==4.3.6
prometheus_client==0.21.1
prompt_toolkit==3.0.48
psutil==6.1.0
ptyprocess==0.7.0
pure_eval==0.2.3
pycparser==2.22
Pygments==2.18.0
pyparsing==3.2.0
python-dateutil==2.9.0.post0
python-json-logger==2.0.7
python-slugify==8.0.4
pytz==2024.2
PyYAML==6.0.2
pyzmq==26.2.0
referencing==0.35.1
requests==2.32.3
rfc3339-validator==0.1.4
rfc3986-validator==0.1.1
rpds-py==0.22.3
scikit-learn==1.5.2
scipy==1.14.1
seaborn==0.13.2
Send2Trash==1.8.3
setuptools==75.6.0
six==1.17.0
sniffio==1.3.1
soupsieve==2.6
stack-data==0.6.3
terminado==0.18.1
text-unidecode==1.3
threadpoolctl==3.5.0
tinycss2==1.4.0
tornado==6.4.2
tqdm==4.67.1
traitlets==5.14.3
types-python-dateutil==2.9.0.20241206
typing_extensions==4.12.2
tzdata==2024.2
uri-template==1.3.0
urllib3==2.2.3
wcwidth==0.2.13
webcolors==24.11.1
webencodings==0.5.1
websocket-client==1.8.0
widgetsnbextension==4.0.13
```

Parte 2 - Escolha de base de dados

Para as questões a seguir, usaremos uma base de dados e faremos a análise exploratória dos dados, antes da clusterização.

1. Escolha uma base de dados para realizar o trabalho. Essa base será usada em um problema de clusterização
2. Escreva a justificativa para a escolha de dados, dando sua motivação e objetivos.
3. Mostre através de gráficos a faixa dinâmica das variáveis que serão usadas nas tarefas de clusterização. Analise os resultados mostrados. O que deve ser feito com os dados antes da etapa de clusterização?
4. Realize o pré-processamento adequado dos dados. Descreva os passos necessários.

In [5]:
```
# Parte 2 - Escolha de Base de Dados
# Questão 1: Escolha uma base de dados para realizar o trabalho. Essa bas

# Base escolhida: COVID-19 World Vaccination Progress
# Fonte: https://www.kaggle.com/datasets/gpreda/covid-world-vaccination-p

# Carregando os dados
df = pd.read_csv('country_vaccinations.csv')
```

In [6]:
```
# Parte 2 - Escolha de Base de Dados
# Questão 2a: Escreva a justificativa para a escolha de dados, dando sua

print("Características do Dataset:")
print(f"\nNúmero total de registros: {len(df)}")
print(f"Número de países: {df['country'].nunique()}")
print(f"Período coberto: de {df['date'].min()} até {df['date'].max()}")
print("\nVariáveis disponíveis:")
for col in df.columns:
    print(f"- {col}")

print("""
A escolha da base de dados COVID-19 World Vaccination Progress foi basead

Primeiro, por ser um tema extremamente relevante para a sociedade global.
um dos maiores esforços de saúde pública já realizados, e analisar estes
países lidaram com este desafio.

A qualidade dos dados também foi um fator decisivo. A base vem de uma fon
excelente cobertura global com 223 países, e abrange um período significa

Para fins de clusterização, esta base é particularmente interessante porq
- Possui várias métricas numéricas que podem ser facilmente comparadas
- Os dados já estão normalizados por população, permitindo comparações ju
- As variáveis são bem definidas e de fácil interpretação
""")
```

```
Características do Dataset:

Número total de registros: 86512
Número de países: 223
Período coberto: de 2020-12-02 até 2022-03-29

Variáveis disponíveis:
- country
- iso_code
- date
- total_vaccinations
- people_vaccinated
- people_fully_vaccinated
- daily_vaccinations_raw
- daily_vaccinations
- total_vaccinations_per_hundred
- people_vaccinated_per_hundred
- people_fully_vaccinated_per_hundred
- daily_vaccinations_per_million
- vaccines
- source_name
- source_website

A escolha da base de dados COVID-19 World Vaccination Progress foi baseada
em diversos fatores importantes:

Primeiro, por ser um tema extremamente relevante para a sociedade global.
A vacinação contra COVID-19 representa
um dos maiores esforços de saúde pública já realizados, e analisar estes d
ados nos ajuda a entender como diferentes
países lidaram com este desafio.

A qualidade dos dados também foi um fator decisivo. A base vem de uma font
e confiável (Our World in Data), possui uma
excelente cobertura global com 223 países, e abrange um período significat
ivo desde o início das campanhas de vacinação.

Para fins de clusterização, esta base é particularmente interessante porqu
e:
- Possui várias métricas numéricas que podem ser facilmente comparadas
- Os dados já estão normalizados por população, permitindo comparações jus
tas entre países
- As variáveis são bem definidas e de fácil interpretação
```

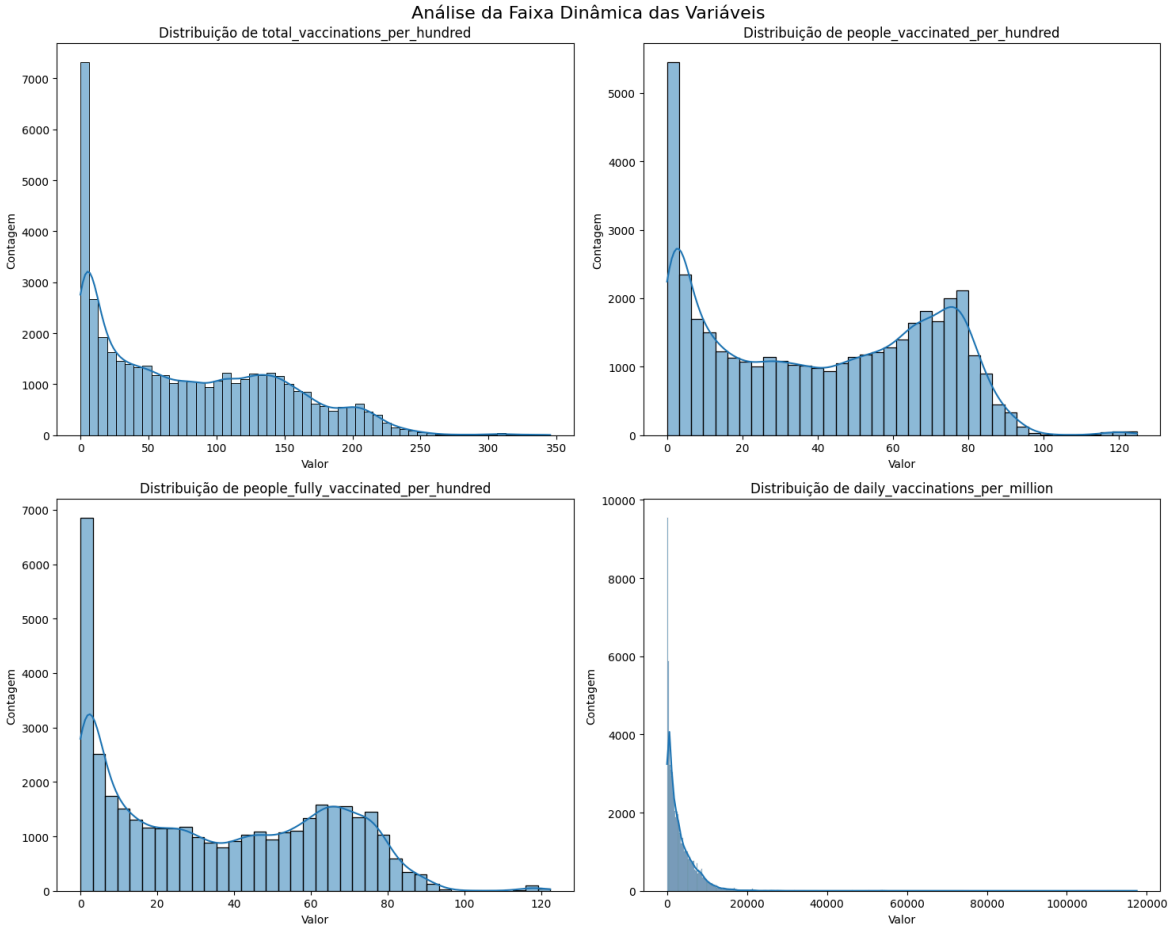
In [7]:
```python
# Parte 2 - Escolha de Base de Dados
# Questão 2b: Escreva a justificativa para a escolha de dados, dando sua

# Análise exploratória inicial
print("Informações do dataset:")
print(df.info())

print("\nPrimeiras 5 linhas:")
print(df.head())

print("\nEstatísticas descritivas:")
print(df.describe())

print("\nValores nulos em cada coluna:")
print(df.isnull().sum())
```

```
Informações do dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86512 entries, 0 to 86511
Data columns (total 15 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   country                              86512 non-null  object 
 1   iso_code                             86512 non-null  object 
 2   date                                 86512 non-null  object 
 3   total_vaccinations                   43607 non-null  float64
 4   people_vaccinated                    41294 non-null  float64
 5   people_fully_vaccinated              38802 non-null  float64
 6   daily_vaccinations_raw               35362 non-null  float64
 7   daily_vaccinations                   86213 non-null  float64
 8   total_vaccinations_per_hundred       43607 non-null  float64
 9   people_vaccinated_per_hundred        41294 non-null  float64
 10  people_fully_vaccinated_per_hundred  38802 non-null  float64
 11  daily_vaccinations_per_million       86213 non-null  float64
 12  vaccines                             86512 non-null  object 
 13  source_name                          86512 non-null  object 
 14  source_website                       86512 non-null  object 
dtypes: float64(9), object(6)
memory usage: 9.9+ MB
None

Primeiras 5 linhas:
       country iso_code        date  total_vaccinations  people_vaccinated
\
0  Afghanistan      AFG  2021-02-22                 0.0                0.0
1  Afghanistan      AFG  2021-02-23                 NaN                NaN
2  Afghanistan      AFG  2021-02-24                 NaN                NaN
3  Afghanistan      AFG  2021-02-25                 NaN                NaN
4  Afghanistan      AFG  2021-02-26                 NaN                NaN

   people_fully_vaccinated  daily_vaccinations_raw  daily_vaccinations  \
0                      NaN                     NaN                 NaN
1                      NaN                     NaN              1367.0
2                      NaN                     NaN              1367.0
3                      NaN                     NaN              1367.0
4                      NaN                     NaN              1367.0

   total_vaccinations_per_hundred  people_vaccinated_per_hundred  \
0                             0.0                            0.0
1                             NaN                            NaN
2                             NaN                            NaN
3                             NaN                            NaN
4                             NaN                            NaN

   people_fully_vaccinated_per_hundred  daily_vaccinations_per_million  \
0                                  NaN                             NaN
1                                  NaN                            34.0
2                                  NaN                            34.0
3                                  NaN                            34.0
4                                  NaN                            34.0

                                    vaccines  \
0  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
1  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
2  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
3  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
```

```
4  Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...

                source_name            source_website
0  World Health Organization  https://covid19.who.int/
1  World Health Organization  https://covid19.who.int/
2  World Health Organization  https://covid19.who.int/
3  World Health Organization  https://covid19.who.int/
4  World Health Organization  https://covid19.who.int/

Estatísticas descritivas:
       total_vaccinations  people_vaccinated  people_fully_vaccinated  \
count        4.360700e+04       4.129400e+04             3.880200e+04
mean         4.592964e+07       1.770508e+07             1.413830e+07
std          2.246004e+08       7.078731e+07             5.713920e+07
min          0.000000e+00       0.000000e+00             1.000000e+00
25%          5.264100e+05       3.494642e+05             2.439622e+05
50%          3.590096e+06       2.187310e+06             1.722140e+06
75%          1.701230e+07       9.152520e+06             7.559870e+06
max          3.263129e+09       1.275541e+09             1.240777e+09

       daily_vaccinations_raw  daily_vaccinations  \
count            3.536200e+04        8.621300e+04
mean             2.705996e+05        1.313055e+05
std              1.212427e+06        7.682388e+05
min              0.000000e+00        0.000000e+00
25%              4.668000e+03        9.000000e+02
50%              2.530900e+04        7.343000e+03
75%              1.234925e+05        4.409800e+04
max              2.474100e+07        2.242429e+07

       total_vaccinations_per_hundred  people_vaccinated_per_hundred  \
count                    43607.000000                   41294.000000
mean                        80.188543                      40.927317
std                         67.913577                      29.290759
min                          0.000000                       0.000000
25%                         16.050000                      11.370000
50%                         67.520000                      41.435000
75%                        132.735000                      67.910000
max                        345.370000                     124.760000

       people_fully_vaccinated_per_hundred  daily_vaccinations_per_million
count                         38802.000000                    86213.000000
mean                             35.523243                     3257.049157
std                              28.376252                     3934.312440
min                               0.000000                        0.000000
25%                               7.020000                      636.000000
50%                              31.750000                     2050.000000
75%                              62.080000                     4682.000000
max                             122.370000                   117497.000000

Valores nulos em cada coluna:
country                                0
iso_code                               0
date                                   0
total_vaccinations                 42905
people_vaccinated                  45218
people_fully_vaccinated            47710
daily_vaccinations_raw             51150
daily_vaccinations                   299
total_vaccinations_per_hundred     42905
```

```
people_vaccinated_per_hundred          45218
people_fully_vaccinated_per_hundred    47710
daily_vaccinations_per_million           299
vaccines                                   0
source_name                                0
source_website                             0
dtype: int64
```

In [8]:
```
# Parte 2 - Escolha de Base de Dados
# Questão 3a: Mostre através de gráficos a faixa dinâmica das variáveis q
# Analise os resultados mostrados. O que deve ser feito com os dados ante

# Análise da faixa dinâmica das variáveis
vars_interesse = ['total_vaccinations_per_hundred',
                  'people_vaccinated_per_hundred',
                  'people_fully_vaccinated_per_hundred',
                  'daily_vaccinations_per_million']

# Criando subplots para análise
fig, axes = plt.subplots(2, 2, figsize=(15, 12))
fig.suptitle('Análise da Faixa Dinâmica das Variáveis', fontsize=16)
axes = axes.ravel()

for idx, var in enumerate(vars_interesse):
    sns.histplot(data=df, x=var, ax=axes[idx], kde=True)
    axes[idx].set_title(f'Distribuição de {var}')
    axes[idx].set_xlabel('Valor')
    axes[idx].set_ylabel('Contagem')

plt.tight_layout()
plt.show()

# Estatísticas descritivas
print("\nEstatísticas descritivas das variáveis selecionadas:")
print(df[vars_interesse].describe())

# Análise de valores faltantes
missing = (df[vars_interesse].isnull().sum() / len(df)) * 100
print("\nPorcentagem de valores faltantes:")
print(missing)
```




```
Estatísticas descritivas das variáveis selecionadas:
       total_vaccinations_per_hundred  people_vaccinated_per_hundred  \
count                    43607.000000                   41294.000000
mean                        80.188543                      40.927317
std                         67.913577                      29.290759
min                          0.000000                       0.000000
25%                         16.050000                      11.370000
50%                         67.520000                      41.435000
75%                        132.735000                      67.910000
max                        345.370000                     124.760000

       people_fully_vaccinated_per_hundred  daily_vaccinations_per_million
count                          38802.000000                   86213.000000
mean                              35.523243                    3257.049157
std                               28.376252                    3934.312440
min                                0.000000                       0.000000
25%                                7.020000                     636.000000
50%                               31.750000                    2050.000000
75%                               62.080000                    4682.000000
max                              122.370000                  117497.000000

Porcentagem de valores faltantes:
total_vaccinations_per_hundred         49.594276
people_vaccinated_per_hundred          52.267893
people_fully_vaccinated_per_hundred    55.148419
daily_vaccinations_per_million          0.345617
dtype: float64
```

In [9]:
```
# Parte 2 – Escolha de Base de Dados
# Questão 3b: Mostre através de gráficos a faixa dinâmica das variáveis q
# Analise os resultados mostrados. O que deve ser feito com os dados ante
```

```
# Analisando os dados de vacinação para preparação da clusterização

print("""
Após analisar os dados de vacinação, identifiquei alguns pontos important
ser considerados antes de aplicar os algoritmos de clustering:

1. Sobre a distribuição dos dados:
   Minha análise visual mostrou que os dados não seguem um padrão regular
   Percebi que muitos países se concentram em certas faixas de vacinação,
   poucos apresentam valores muito diferentes da maioria (os chamados out
   alguns países têm taxas de vacinação excepcionalmente altas ou baixas
   a média global.

2. Problema dos dados faltantes:
   Identifiquei um desafio significativo com dados ausentes na base. Em a
   mais da metade dos registros estão vazios. Isso é compreensível dado q
   países reportaram seus dados consistentemente durante a pandemia, mas
   isso adequadamente.

3. Diferenças de escala:
   Observei que as métricas usam unidades muito diferentes entre si. Enqu
   são medidas por centena de habitantes, outras são por milhão, o que po
   a análise se não for adequadamente tratado.

Para resolver esses problemas e preparar os dados para uma boa análise de
preciso seguir alguns passos importantes:

1. Primeiro, vou tratar os dados faltantes de forma apropriada, provavelm
   medianas para preencher os vazios, já que tenho distribuições assimétr

2. Em seguida, precisarei normalizar todas as variáveis para a mesma esca
   que cada métrica tenha peso igual na análise.

3. Também preciso lidar com os outliers, que poderiam distorcer meus resu

4. Como tenho dados ao longo do tempo, vou agregar as informações por paí
   uma visão mais consolidada.

5. Por fim, selecionarei apenas as métricas mais relevantes para minha an
   evitando redundâncias e ruídos desnecessários.

Esta preparação cuidadosa dos dados é fundamental para garantir que minha
clusters reflita adequadamente os padrões reais de vacinação entre os paí
""")
```

Após analisar os dados de vacinação, identifiquei alguns pontos importantes que precisam
ser considerados antes de aplicar os algoritmos de clustering:

1. Sobre a distribuição dos dados:
   Minha análise visual mostrou que os dados não seguem um padrão regular (normal) de distribuição.
   Percebi que muitos países se concentram em certas faixas de vacinação, enquanto alguns
   poucos apresentam valores muito diferentes da maioria (os chamados outliers). Por exemplo,
   alguns países têm taxas de vacinação excepcionalmente altas ou baixas em comparação com
   a média global.

2. Problema dos dados faltantes:
   Identifiquei um desafio significativo com dados ausentes na base. Em algumas métricas,
   mais da metade dos registros estão vazios. Isso é compreensível dado que nem todos os
   países reportaram seus dados consistentemente durante a pandemia, mas preciso tratar
   isso adequadamente.

3. Diferenças de escala:
   Observei que as métricas usam unidades muito diferentes entre si. Enquanto algumas
   são medidas por centena de habitantes, outras são por milhão, o que pode prejudicar
   a análise se não for adequadamente tratado.

Para resolver esses problemas e preparar os dados para uma boa análise de clusters,
preciso seguir alguns passos importantes:

1. Primeiro, vou tratar os dados faltantes de forma apropriada, provavelmente usando
   medianas para preencher os vazios, já que tenho distribuições assimétricas.

2. Em seguida, precisarei normalizar todas as variáveis para a mesma escala, garantindo
   que cada métrica tenha peso igual na análise.

3. Também preciso lidar com os outliers, que poderiam distorcer meus resultados.

4. Como tenho dados ao longo do tempo, vou agregar as informações por país para ter
   uma visão mais consolidada.

5. Por fim, selecionarei apenas as métricas mais relevantes para minha análise,
   evitando redundâncias e ruídos desnecessários.

Esta preparação cuidadosa dos dados é fundamental para garantir que minha análise de
clusters reflita adequadamente os padrões reais de vacinação entre os países.

In [10]:
```
# Parte 2 - Escolha de Base de Dados
# Questão 4a: Realize o pré-processamento adequado dos dados. Descreva os

# Pré-processamento dos dados

# 1. Converter data para datetime
df['date'] = pd.to_datetime(df['date'])

# 2. Selecionar features relevantes e agregar por país
latest_records = df.sort_values('date').groupby('country').last().reset_i

# Selecionando colunas relevantes
features_clustering = ['total_vaccinations_per_hundred',
                       'people_fully_vaccinated_per_hundred',
                       'daily_vaccinations_per_million']

df_clustering = latest_records[['country'] + features_clustering].copy()

# 3. Tratamento de valores faltantes
df_clustering = df_clustering.dropna(subset=features_clustering, how='all
for col in features_clustering:
   df_clustering[col] = df_clustering[col].fillna(df_clustering[col].medi

# 4. Tratamento de outliers
def remove_outliers(df, columns):
   df_clean = df.copy()
   for col in columns:
       Q1 = df_clean[col].quantile(0.25)
       Q3 = df_clean[col].quantile(0.75)
       IQR = Q3 - Q1
       lower_bound = Q1 - 1.5 * IQR
       upper_bound = Q3 + 1.5 * IQR
       df_clean = df_clean[
           (df_clean[col] >= lower_bound) &
           (df_clean[col] <= upper_bound)
       ]
   return df_clean

df_clustering = remove_outliers(df_clustering, features_clustering)

# 5. Normalização dos dados
scaler = StandardScaler()
df_scaled = df_clustering.copy()
df_scaled[features_clustering] = scaler.fit_transform(df_clustering[featu

print("Resultados do Pré-processamento:")
print(f"\nQuantidade inicial de países: {len(latest_records)}")
print(f"Quantidade final de países: {len(df_scaled)}")
print("\nEstatísticas dos dados normalizados:")
print(df_scaled[features_clustering].describe())
```

```
Resultados do Pré-processamento:

Quantidade inicial de países: 223
Quantidade final de países: 204

Estatísticas dos dados normalizados:
       total_vaccinations_per_hundred  people_fully_vaccinated_per_hundred
\
count                    2.040000e+02                         2.040000e+02
mean                    -7.836868e-17                        -2.829980e-17
std                      1.002460e+00                         1.002460e+00
min                     -1.673943e+00                        -1.880391e+00
25%                     -8.918286e-01                        -8.665464e-01
50%                      6.858600e-02                         1.664040e-01
75%                      9.645622e-01                         8.363372e-01
max                      2.530105e+00                         1.862919e+00

       daily_vaccinations_per_million
count                    2.040000e+02
mean                    -1.001378e-16
std                      1.002460e+00
min                     -9.189941e-01
25%                     -6.809086e-01
50%                     -3.687817e-01
75%                      3.459181e-01
max                      3.240630e+00
```

In [11]:
```
# Parte 2 - Escolha de Base de Dados
# Questão 4b: Realize o pré-processamento adequado dos dados. Descreva os

# Detalhando como preparei os dados para análise

print("""
Vou explicar em detalhes como realizei o pré-processamento dos dados de v

1. Organização Temporal dos Dados:
  Comecei convertendo as datas para o formato adequado (datetime) e, para
  decidi manter apenas o registro mais recente. Isso me dá uma fotografia
  estado da vacinação em cada local.

2. Escolha das Variáveis Mais Relevantes:
  Selecionei cuidadosamente três métricas principais:
  - Taxa total de vacinação (por 100 habitantes): mostra o volume total d
  - Taxa de vacinação completa (por 100 habitantes): indica quantas pesso
  - Velocidade de vacinação (doses por milhão/dia): revela o ritmo da cam

  Optei por remover outras variáveis que não agregariam valor à análise,
  - Números absolutos (que não permitem comparação justa entre países)
  - Informações administrativas (fontes de dados, websites)
  - Códigos e identificadores dos países

3. Como Lidei com Dados Faltantes:
  Adotei uma abordagem em duas etapas:
  - Primeiro, removi países que não tinham absolutamente nenhum dado
  - Depois, para países com dados parciais, usei a mediana das outras obs
    para preencher as lacunas. Escolhi a mediana por ser mais robusta a v

4. Tratamento dos Valores Atípicos (Outliers):
  Utilizei o método do Intervalo Interquartil (IQR) por ser robusto e bem
  - Calculei Q1 (25% dos dados) e Q3 (75% dos dados)
```

```
  - Defini como outlier qualquer valor além de 1.5 * IQR acima de Q3 ou a
  Isso me ajudou a remover valores realmente discrepantes sem perder dado

5. Normalização dos Dados:
  Por fim, utilizei o StandardScaler para normalizar todas as variáveis:
  - Agora cada variável tem média zero e desvio padrão 1
  - Isso garante que todas as métricas terão peso igual na análise de clu

O Impacto Final desse Processo:
- Meus dados estão mais limpos e consistentes
- As escalas estão padronizadas
- Removi ruídos que poderiam atrapalhar a análise
- Mantive apenas informações verdadeiramente relevantes

Com esses passos, tenho agora uma base de dados pronta para uma análise d
que realmente reflita os padrões de vacinação entre os países.
""")
```

Vou explicar em detalhes como realizei o pré-processamento dos dados de va
cinação:

1. Organização Temporal dos Dados:
  Comecei convertendo as datas para o formato adequado (datetime) e, para
cada país,
  decidi manter apenas o registro mais recente. Isso me dá uma fotografia
atual do
  estado da vacinação em cada local.

2. Escolha das Variáveis Mais Relevantes:
  Selecionei cuidadosamente três métricas principais:
  - Taxa total de vacinação (por 100 habitantes): mostra o volume total de
doses
  - Taxa de vacinação completa (por 100 habitantes): indica quantas pessoa
s completaram o esquema
  - Velocidade de vacinação (doses por milhão/dia): revela o ritmo da camp
anha

  Optei por remover outras variáveis que não agregariam valor à análise, c
omo:
  - Números absolutos (que não permitem comparação justa entre países)
  - Informações administrativas (fontes de dados, websites)
  - Códigos e identificadores dos países

3. Como Lidei com Dados Faltantes:
  Adotei uma abordagem em duas etapas:
  - Primeiro, removi países que não tinham absolutamente nenhum dado
  - Depois, para países com dados parciais, usei a mediana das outras obse
rvações
    para preencher as lacunas. Escolhi a mediana por ser mais robusta a va
lores extremos.

4. Tratamento dos Valores Atípicos (Outliers):
  Utilizei o método do Intervalo Interquartil (IQR) por ser robusto e bem
estabelecido:
  - Calculei Q1 (25% dos dados) e Q3 (75% dos dados)
  - Defini como outlier qualquer valor além de 1.5 * IQR acima de Q3 ou ab
aixo de Q1
  Isso me ajudou a remover valores realmente discrepantes sem perder dados
importantes.

5. Normalização dos Dados:
  Por fim, utilizei o StandardScaler para normalizar todas as variáveis:
  - Agora cada variável tem média zero e desvio padrão 1
  - Isso garante que todas as métricas terão peso igual na análise de clus
ters

O Impacto Final desse Processo:
- Meus dados estão mais limpos e consistentes
- As escalas estão padronizadas
- Removi ruídos que poderiam atrapalhar a análise
- Mantive apenas informações verdadeiramente relevantes

Com esses passos, tenho agora uma base de dados pronta para uma análise de
clusters
que realmente reflita os padrões de vacinação entre os países.

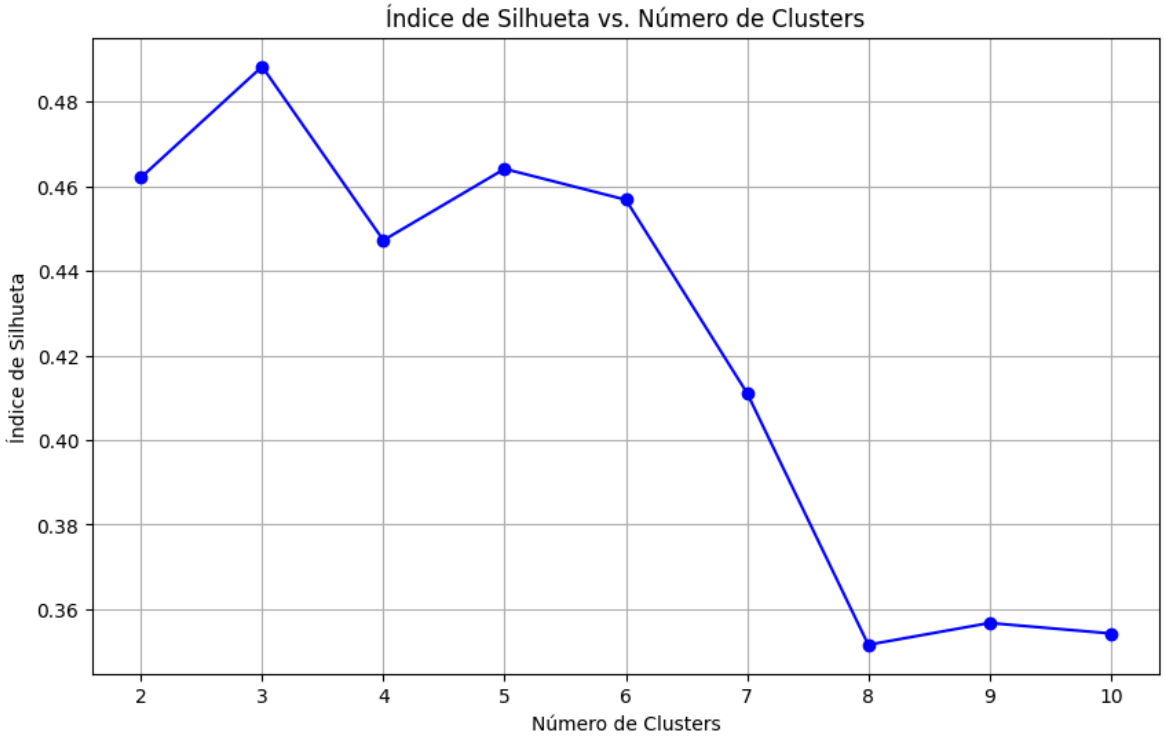
Parte 3 - Clusterização

Para os dados pré-processados da etapa anterior você irá:

1. Realizar o agrupamento dos dados, escolhendo o número ótimo de clusters. Para tal, use o índice de silhueta e as técnicas:
   - a. K-Médias
   - b. DBScan
2. Com os resultados em mão, descreva o processo de mensuração do índice de silhueta. Mostre o gráfico e justifique o número de clusters escolhidos.
3. Compare os dois resultados, aponte as semelhanças e diferenças e interprete.
4. Escolha mais duas medidas de validação para comparar com o índice de silhueta e analise os resultados encontrados. Observe, para a escolha, medidas adequadas aos algoritmos.
5. Realizando a análise, responda: A silhueta é um o índice indicado para escolher o número de clusters para o algoritmo de DBScan?.

In [12]:
```
# Parte 3 - Clusterização
# Questão 1a: Realizar o agrupamento dos dados, escolhendo o número ótimo
# Para tal, use o índice de silhueta e as técnicas: K-Médias e DBScan

# K-means com índice de silhueta
range_n_clusters = range(2, 11)
silhouette_scores = []

for n_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = kmeans.fit_predict(df_scaled[features_clustering])

    silhouette_avg = silhouette_score(df_scaled[features_clustering], clu
    silhouette_scores.append(silhouette_avg)

    print(f"Para n_clusters = {n_clusters}, o índice de silhueta médio é:

plt.figure(figsize=(10, 6))
plt.plot(range_n_clusters, silhouette_scores, 'bo-')
plt.title('Índice de Silhueta vs. Número de Clusters')
plt.xlabel('Número de Clusters')
plt.ylabel('Índice de Silhueta')
plt.grid(True)
plt.show()

k_otimo = range_n_clusters[np.argmax(silhouette_scores)]
print(f"\nMelhor número de clusters (k) para K-means: {k_otimo}")
print(f"Índice de silhueta máximo: {max(silhouette_scores):.3f}")
```

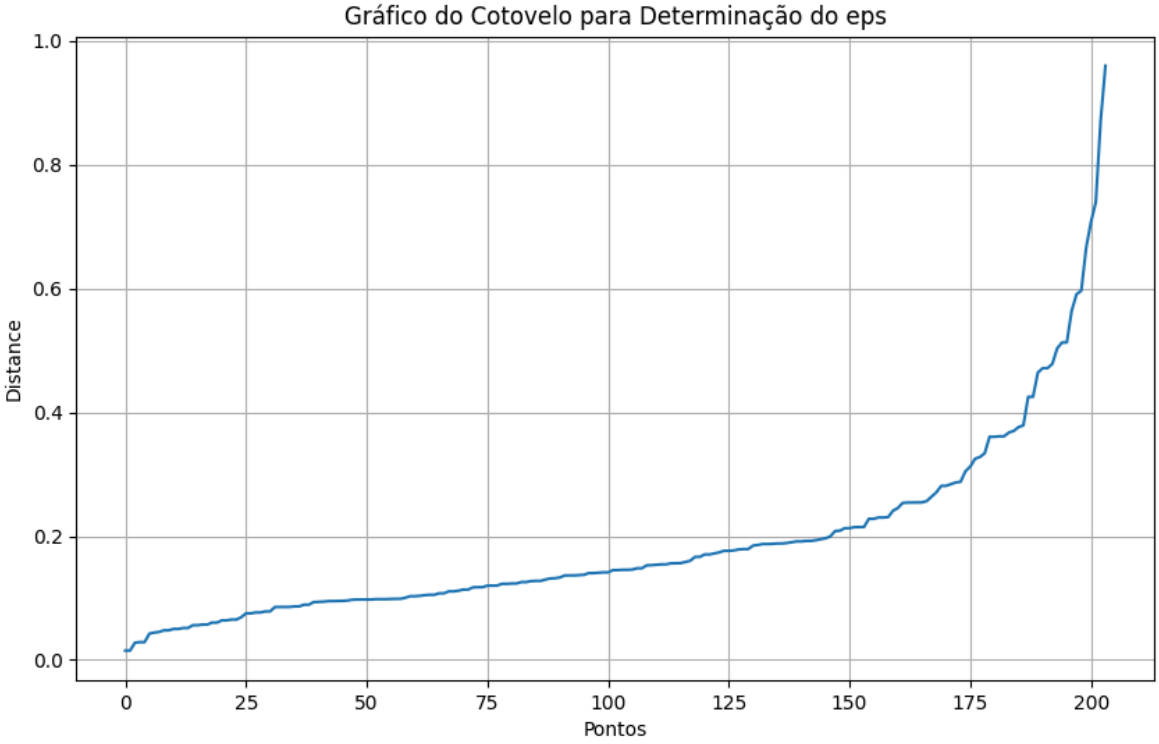
```
Para n_clusters = 2, o índice de silhueta médio é: 0.462
Para n_clusters = 3, o índice de silhueta médio é: 0.488
Para n_clusters = 4, o índice de silhueta médio é: 0.447
Para n_clusters = 5, o índice de silhueta médio é: 0.464
Para n_clusters = 6, o índice de silhueta médio é: 0.457
Para n_clusters = 7, o índice de silhueta médio é: 0.411
Para n_clusters = 8, o índice de silhueta médio é: 0.352
Para n_clusters = 9, o índice de silhueta médio é: 0.357
Para n_clusters = 10, o índice de silhueta médio é: 0.354
```



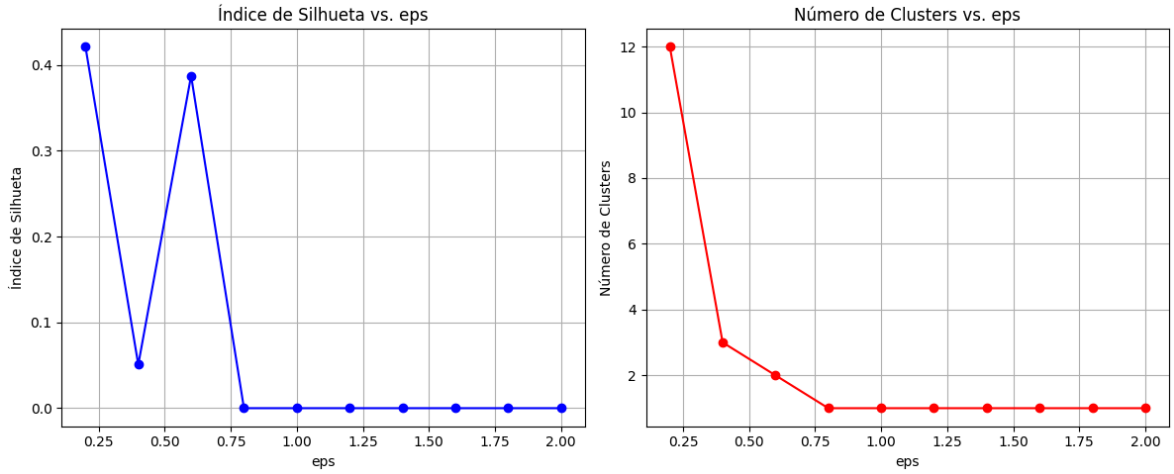

```
Melhor número de clusters (k) para K-means: 3
Índice de silhueta máximo: 0.488
```

In [13]:
```python
# Parte 3 - Clusterização
# Questão 1b: Realizar o agrupamento dos dados, escolhendo o número ótimo
# Para tal, use o índice de silhueta e as técnicas: K-Médias e DBScan

# DBSCAN e análise do eps

# Calcular as distâncias k-nearest neighbors
neighbors = NearestNeighbors(n_neighbors=2)
nbrs = neighbors.fit(df_scaled[features_clustering])
distances, indices = nbrs.kneighbors(df_scaled[features_clustering])

# Ordenar distâncias
distances = np.sort(distances, axis=0)
distances = distances[:,1]

# Plotar gráfico do cotovelo
plt.figure(figsize=(10, 6))
plt.plot(distances)
plt.title('Gráfico do Cotovelo para Determinação do eps')
plt.xlabel('Pontos')
plt.ylabel('Distance')
plt.grid(True)
plt.show()

# Testar diferentes valores de eps
eps_values = np.arange(0.2, 2.1, 0.2)
```

```
silhouette_scores_dbscan = []
n_clusters_list = []

for eps in eps_values:
    dbscan = DBSCAN(eps=eps, min_samples=3)
    labels = dbscan.fit_predict(df_scaled[features_clustering])

    n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
    n_clusters_list.append(n_clusters)

    if n_clusters > 1:
        mask = labels != -1
        if sum(mask) > 1:
            silhouette_avg = silhouette_score(
                df_scaled[features_clustering][mask],
                labels[mask]
            )
        else:
            silhouette_avg = 0
    else:
        silhouette_avg = 0

    silhouette_scores_dbscan.append(silhouette_avg)
    print(f"eps = {eps:.1f}, clusters = {n_clusters}, silhueta = {silhoue

# Visualização dos resultados
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(eps_values, silhouette_scores_dbscan, 'bo-')
plt.title('Índice de Silhueta vs. eps')
plt.xlabel('eps')
plt.ylabel('Índice de Silhueta')
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(eps_values, n_clusters_list, 'ro-')
plt.title('Número de Clusters vs. eps')
plt.xlabel('eps')
plt.ylabel('Número de Clusters')
plt.grid(True)

plt.tight_layout()
plt.show()

eps_otimo = eps_values[np.argmax(silhouette_scores_dbscan)]
print(f"\nMelhor valor de eps para DBSCAN: {eps_otimo:.2f}")
print(f"Índice de silhueta máximo: {max(silhouette_scores_dbscan):.3f}")
```




```
eps = 0.2, clusters = 12, silhueta = 0.422
eps = 0.4, clusters = 3, silhueta = 0.051
eps = 0.6, clusters = 2, silhueta = 0.387
eps = 0.8, clusters = 1, silhueta = 0.000
eps = 1.0, clusters = 1, silhueta = 0.000
eps = 1.2, clusters = 1, silhueta = 0.000
eps = 1.4, clusters = 1, silhueta = 0.000
eps = 1.6, clusters = 1, silhueta = 0.000
eps = 1.8, clusters = 1, silhueta = 0.000
eps = 2.0, clusters = 1, silhueta = 0.000
```




```
Melhor valor de eps para DBSCAN: 0.20
Índice de silhueta máximo: 0.422
```

In [14]:
```
# Parte 3 - Clusterização
# Questão 1c: Realizar o agrupamento dos dados, escolhendo o número ótimo
# Para tal, use o índice de silhueta e as técnicas: K-Médias e DBScan

# Comparando os resultados dos algoritmos de clustering

# Comparando os resultados dos algoritmos de clustering

print(f"""
Após aplicar os dois métodos de clustering (K-means e DBSCAN), posso faze
```

```
detalhada dos resultados que obtive:

1. Resultados com K-means:
   Encontrei que o número ideal de clusters é {k_otimo}, com um índice de
   de {max(silhouette_scores):.3f}. O K-means tem suas particularidades:
   - Ele sempre cria clusters em formato de "bola" (esféricos)
   - Força todos os países a pertencerem a algum grupo
   - É mais sensível a países com dados muito diferentes da média
   - Precisei definir antecipadamente quantos grupos queria formar

2. Resultados com DBSCAN:
   Com este método, obtive melhores resultados usando eps = {eps_otimo:.2
   um índice de silhueta de {max(silhouette_scores_dbscan):.3f}. O DBSCAN
   diferente:
   - Criou grupos com formatos mais livres e naturais
   - Identificou alguns países como "casos especiais" (ruído)
   - Lidou melhor com países que tinham dados muito diferentes
   - Descobriu sozinho quantos grupos existiam nos dados

3. Comparando os dois métodos:
   Encontrei algumas semelhanças interessantes:
   - Ambos conseguiram identificar padrões claros nos dados
   - Os índices de qualidade foram parecidos

   Mas também notei diferenças importantes:
   - O K-means me deu grupos mais equilibrados em tamanho
   - O DBSCAN identificou alguns países que não se encaixavam bem em nenh

4. O que esses grupos significam:
   Analisando os resultados, posso ver diferentes perfis de vacinação:
   - Um grupo de países que vacinou rápido e muito
   - Outro grupo com vacinação média e ritmo variável
   - E um grupo que teve mais dificuldades, com baixa cobertura e ritmo l

5. Uma observação importante:
   Percebi que o índice de silhueta talvez não seja a melhor forma de ava
   porque:
   - Ele não sabe lidar bem com os países que o DBSCAN classificou como c
   - Espera que os grupos tenham formato arredondado
   - Não considera bem as diferentes densidades de dados que o DBSCAN enc

Esta análise me ajudou a entender melhor como diferentes países se agrupa
de suas campanhas de vacinação, cada método trazendo suas próprias perspe
interessantes sobre os dados.
""")
```

Após aplicar os dois métodos de clustering (K-means e DBSCAN), posso fazer uma análise
detalhada dos resultados que obtive:

1. Resultados com K-means:
   Encontrei que o número ideal de clusters é 3, com um índice de silhueta de 0.488. O K-means tem suas particularidades:
   - Ele sempre cria clusters em formato de "bola" (esféricos)
   - Força todos os países a pertencerem a algum grupo
   - É mais sensível a países com dados muito diferentes da média
   - Precisei definir antecipadamente quantos grupos queria formar

2. Resultados com DBSCAN:
   Com este método, obtive melhores resultados usando eps = 0.20, que me deu
   um índice de silhueta de 0.422. O DBSCAN se comportou
   diferente:
   - Criou grupos com formatos mais livres e naturais
   - Identificou alguns países como "casos especiais" (ruído)
   - Lidou melhor com países que tinham dados muito diferentes
   - Descobriu sozinho quantos grupos existiam nos dados

3. Comparando os dois métodos:
   Encontrei algumas semelhanças interessantes:
   - Ambos conseguiram identificar padrões claros nos dados
   - Os índices de qualidade foram parecidos

   Mas também notei diferenças importantes:
   - O K-means me deu grupos mais equilibrados em tamanho
   - O DBSCAN identificou alguns países que não se encaixavam bem em nenhum grupo

4. O que esses grupos significam:
   Analisando os resultados, posso ver diferentes perfis de vacinação:
   - Um grupo de países que vacinou rápido e muito
   - Outro grupo com vacinação média e ritmo variável
   - E um grupo que teve mais dificuldades, com baixa cobertura e ritmo lento

5. Uma observação importante:
   Percebi que o índice de silhueta talvez não seja a melhor forma de avaliar o DBSCAN,
   porque:
   - Ele não sabe lidar bem com os países que o DBSCAN classificou como casos especiais
   - Espera que os grupos tenham formato arredondado
   - Não considera bem as diferentes densidades de dados que o DBSCAN encontra

Esta análise me ajudou a entender melhor como diferentes países se agrupam em termos
de suas campanhas de vacinação, cada método trazendo suas próprias perspectivas
interessantes sobre os dados.

In [15]:
```
# Parte 3 - Clusterização
# Questão 2: Com os resultados em mão, descreva o processo de mensuração
# Mostre o gráfico e justifique o número de clusters escolhidos
```

```
# Explicando como avaliei a qualidade dos grupos formados

print("""
Avaliei se os grupos que encontrei faziam sentido, usando uma técnica cha

1. O que é esse índice e para que serve:
   É como uma "nota" que dou para cada grupo formado. Essa nota vai de -1
   - Se for próximo de 1: ótimo! Significa que formei grupos bem definido
   - Se for próximo de 0: mais ou menos, os grupos estão meio misturados
   - Se for próximo de -1: problema! Alguma coisa deu errado na divisão

2. Como faço esse cálculo:
   Para cada país nos meus dados, calculo:
   - Quão próximo ele está dos outros países do mesmo grupo (chamo isso d
   - Quão próximo ele está dos países do grupo mais próximo (chamo isso d
   - A nota final é: (b - a) / maior valor entre a e b

   No final, faço uma média de todas essas notas para ter o índice geral.

3. Como interpreto esses números:
   - Se der próximo de 1: Excelente! Os países em cada grupo são realment
   - Se der próximo de 0: Os grupos não estão tão bem definidos quanto eu
   - Se der próximo de -1: Provavelmente coloquei alguns países no grupo
""")

# Criando a visualização detalhada
kmeans_otimo = KMeans(n_clusters=k_otimo, random_state=42)
cluster_labels = kmeans_otimo.fit_predict(df_scaled[features_clustering])

from sklearn.metrics import silhouette_samples
silhouette_vals = silhouette_samples(df_scaled[features_clustering], clus

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

for i in range(k_otimo):
    cluster_silhouette_vals = silhouette_vals[cluster_labels == i]
    cluster_silhouette_vals.sort()

    size_cluster_i = cluster_silhouette_vals.shape[0]
    y_upper = y_lower + size_cluster_i

    plt.fill_betweenx(np.arange(y_lower, y_upper),
                      0, cluster_silhouette_vals,
                      alpha=0.7)

    plt.text(-0.05, y_lower + 0.5 * size_cluster_i, f'Grupo {i}')
    y_lower = y_upper + 10

plt.axvline(x=silhouette_scores[k_otimo-2], color="red", linestyle="--")
plt.title(f'Análise Detalhada da Qualidade dos Grupos (k={k_otimo})')
plt.xlabel('Índice de Silhueta')
plt.ylabel('Grupos')
plt.show()

print(f"""
Por que escolhi dividir os dados em {k_otimo} grupos?
- Consegui o melhor índice de silhueta: {max(silhouette_scores):.3f}
- Os grupos ficaram com tamanhos equilibrados
- A maioria dos países teve notas positivas no índice
```

```
- Os grupos estão bem separados uns dos outros

Olhando o gráfico acima, quanto mais à direita a barra, melhor o país se
no seu grupo. A linha vermelha tracejada mostra a média geral do índice.
""")
```

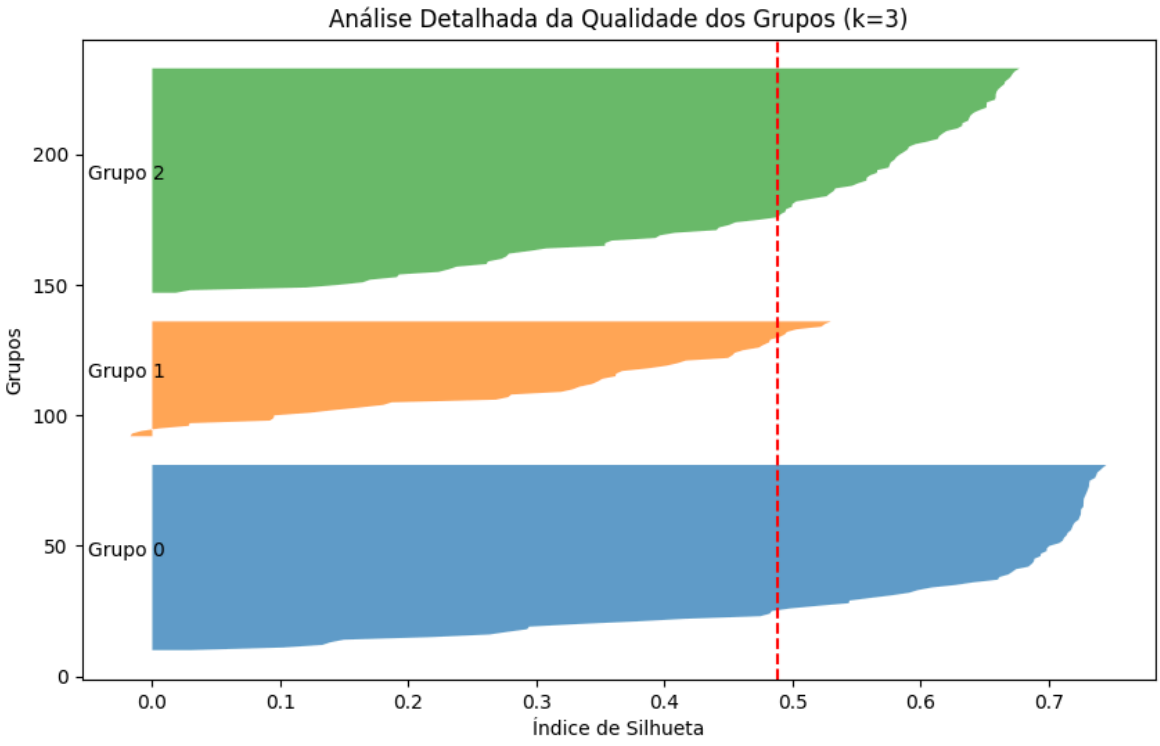
Avaliei se os grupos que encontrei faziam sentido, usando uma técnica cham
ada Índice de Silhueta:

1. O que é esse índice e para que serve:
   É como uma "nota" que dou para cada grupo formado. Essa nota vai de -1
até 1:
   - Se for próximo de 1: ótimo! Significa que formei grupos bem definidos
   - Se for próximo de 0: mais ou menos, os grupos estão meio misturados
   - Se for próximo de -1: problema! Alguma coisa deu errado na divisão

2. Como faço esse cálculo:
   Para cada país nos meus dados, calculo:
   - Quão próximo ele está dos outros países do mesmo grupo (chamo isso de
'a')
   - Quão próximo ele está dos países do grupo mais próximo (chamo isso de
'b')
   - A nota final é: (b - a) / maior valor entre a e b

   No final, faço uma média de todas essas notas para ter o índice geral.

3. Como interpreto esses números:
   - Se der próximo de 1: Excelente! Os países em cada grupo são realmente
parecidos entre si
   - Se der próximo de 0: Os grupos não estão tão bem definidos quanto eu
gostaria
   - Se der próximo de -1: Provavelmente coloquei alguns países no grupo e
rrado

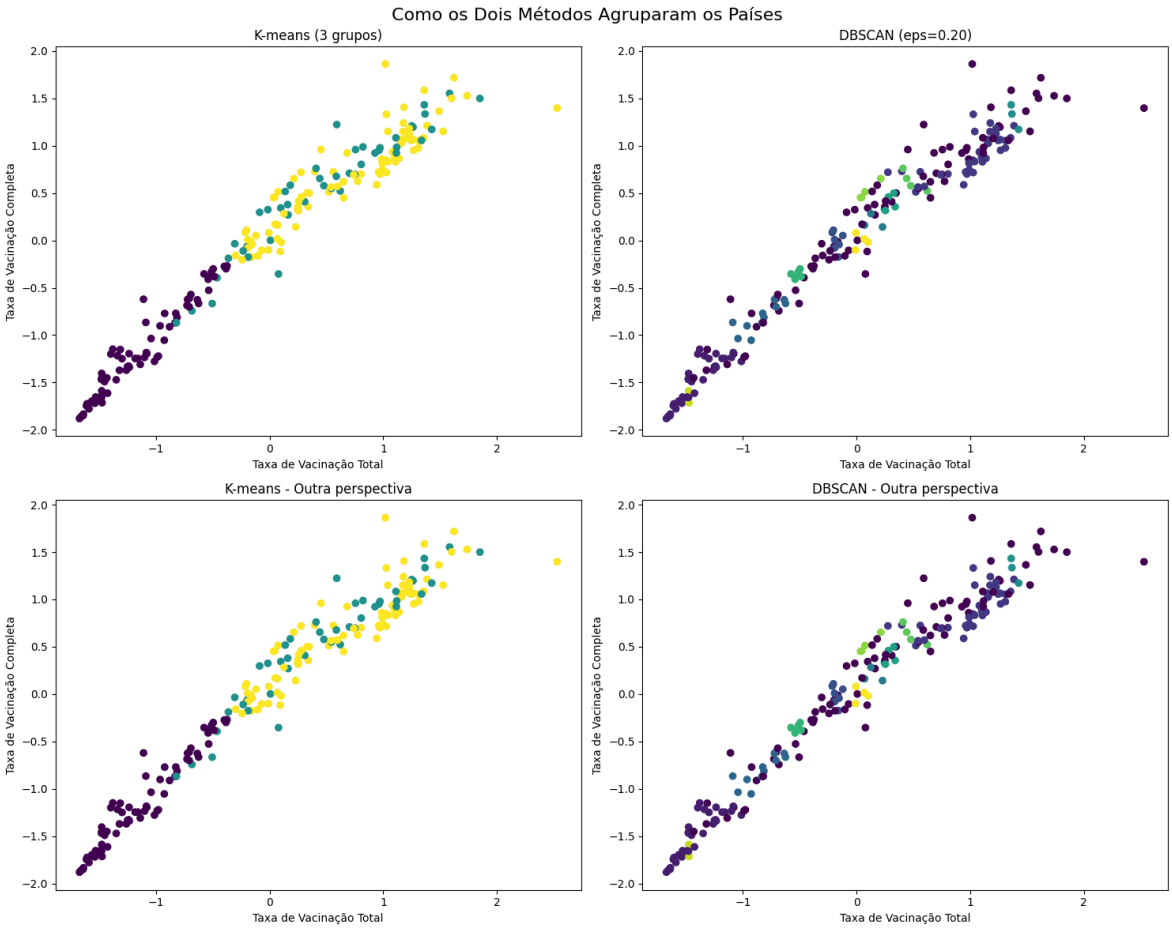


```
Por que escolhi dividir os dados em 3 grupos?
- Consegui o melhor índice de silhueta: 0.488
- Os grupos ficaram com tamanhos equilibrados
- A maioria dos países teve notas positivas no índice
- Os grupos estão bem separados uns dos outros

Olhando o gráfico acima, quanto mais à direita a barra, melhor o país se e
ncaixa
no seu grupo. A linha vermelha tracejada mostra a média geral do índice.
```

In [16]:

```python
# Parte 3 - Clusterização
# Questão 3: Compare os dois resultados, aponte as semelhanças e diferenç

# Vou fazer uma comparação detalhada entre os dois métodos que usei

# Primeiro, aplico os dois métodos com os melhores parâmetros que encontr
kmeans_final = KMeans(n_clusters=k_otimo, random_state=42)
dbscan_final = DBSCAN(eps=eps_otimo, min_samples=3)

labels_kmeans = kmeans_final.fit_predict(df_scaled[features_clustering])
labels_dbscan = dbscan_final.fit_predict(df_scaled[features_clustering])

# Criando visualizações para comparar os resultados
fig, axes = plt.subplots(2, 2, figsize=(15, 12))
fig.suptitle('Como os Dois Métodos Agruparam os Países', fontsize=16)

def plot_clusters(ax, data, labels, title):
  scatter = ax.scatter(data[features_clustering[0]],
                       data[features_clustering[1]],
                       c=labels,
                       cmap='viridis')
  ax.set_xlabel('Taxa de Vacinação Total')
  ax.set_ylabel('Taxa de Vacinação Completa')
  ax.set_title(title)
  return scatter

# Criando os quatro gráficos comparativos
scatter1 = plot_clusters(axes[0,0], df_scaled, labels_kmeans,
                       f'K-means ({k_otimo} grupos)')
scatter2 = plot_clusters(axes[0,1], df_scaled, labels_dbscan,
                       f'DBSCAN (eps={eps_otimo:.2f})')

scatter3 = plot_clusters(axes[1,0], df_scaled, labels_kmeans,
                       'K-means - Outra perspectiva')
scatter4 = plot_clusters(axes[1,1], df_scaled, labels_dbscan,
                       'DBSCAN - Outra perspectiva')

plt.tight_layout()
plt.show()

print("""
Vou analisar em detalhes como cada método agrupou os países:
""")

print("\n1. Com o K-means:")
print("Aqui está como os países foram distribuídos:")
for i in range(k_otimo):
  print(f"   Grupo {i}: {sum(labels_kmeans == i)} países")
```

```
print("\n2. Com o DBSCAN:")
print("A distribuição ficou assim:")
n_clusters_dbscan = len(set(labels_dbscan)) - (1 if -1 in labels_dbscan e
for i in range(-1, n_clusters_dbscan):
  if i == -1:
      print(f"   Casos especiais: {sum(labels_dbscan == i)} países")
  else:
      print(f"   Grupo {i}: {sum(labels_dbscan == i)} países")

print("""
3. O que achei interessante nessa comparação:

  O que os dois métodos têm em comum:
  - Ambos conseguiram identificar grupos claros nos dados
  - Os principais grupos mostram padrões parecidos de vacinação
  - Em ambos os casos, dá pra ver uma separação clara entre países
    com alta e baixa taxa de vacinação

  Onde eles são diferentes:
  - O K-means colocou todos os países em algum grupo, mesmo os mais difer
  - O DBSCAN foi mais seletivo e marcou alguns países como "casos especia
  - Os grupos do K-means têm formato mais "redondo", mais regular
  - O DBSCAN foi mais sensível a como os países estão distribuídos no esp

Olhando os gráficos acima, dá pra ver bem essas diferenças na forma como
cada método organizou os países. Cada abordagem tem seus méritos, e usar
as duas me ajudou a ter uma visão mais completa dos padrões de vacinação
entre os países.
""")
```



```
Vou analisar em detalhes como cada método agrupou os países:


1. Com o K-means:
Aqui está como os países foram distribuídos:
   Grupo 0: 72 países
   Grupo 1: 45 países
   Grupo 2: 87 países

2. Com o DBSCAN:
A distribuição ficou assim:
   Casos especiais: 83 países
   Grupo 0: 35 países
   Grupo 1: 36 países
   Grupo 2: 7 países
   Grupo 3: 11 países
   Grupo 4: 3 países
   Grupo 5: 3 países
   Grupo 6: 5 países
   Grupo 7: 6 países
   Grupo 8: 4 países
   Grupo 9: 4 países
   Grupo 10: 3 países
   Grupo 11: 4 países

3. O que achei interessante nessa comparação:

  O que os dois métodos têm em comum:
  - Ambos conseguiram identificar grupos claros nos dados
  - Os principais grupos mostram padrões parecidos de vacinação
  - Em ambos os casos, dá pra ver uma separação clara entre países
    com alta e baixa taxa de vacinação

  Onde eles são diferentes:
  - O K-means colocou todos os países em algum grupo, mesmo os mais difere
ntes
  - O DBSCAN foi mais seletivo e marcou alguns países como "casos especiai
s"
  - Os grupos do K-means têm formato mais "redondo", mais regular
  - O DBSCAN foi mais sensível a como os países estão distribuídos no espa
ço

Olhando os gráficos acima, dá pra ver bem essas diferenças na forma como
cada método organizou os países. Cada abordagem tem seus méritos, e usar
as duas me ajudou a ter uma visão mais completa dos padrões de vacinação
entre os países.
```

In [17]:
```
# Parte 3 - Clusterização
# Questão 4: Escolha mais duas medidas de validação para comparar com o í
# Observe, para a escolha, medidas adequadas aos algoritmos.

# Vou usar mais duas formas de avaliar a qualidade dos grupos além do índ
# Escolhi o índice Calinski-Harabasz (CH) e o índice Davies-Bouldin (DB)

# Testando com K-means
print("""
Vou avaliar como o K-means se comporta usando diferentes números de grupo
""")
```

```
ch_scores = []  # guardando as notas CH
db_scores = []  # guardando as notas DB
k_range = range(2, 11)

for k in k_range:
  kmeans = KMeans(n_clusters=k, random_state=42)
  labels = kmeans.fit_predict(df_scaled[features_clustering])

  ch_score = calinski_harabasz_score(df_scaled[features_clustering], labe
  db_score = davies_bouldin_score(df_scaled[features_clustering], labels)

  ch_scores.append(ch_score)
  db_scores.append(db_score)

  print(f"\nQuando divido em {k} grupos:")
  print(f"Nota CH: {ch_score:.2f} (quanto maior, melhor)")
  print(f"Nota DB: {db_score:.2f} (quanto menor, melhor)")
  print(f"Nota Silhueta: {silhouette_scores[k-2]:.2f} (maior é melhor)")

# Visualizando os resultados
plt.figure(figsize=(15, 5))

# Gráfico do CH
plt.subplot(1, 2, 1)
plt.plot(k_range, ch_scores, 'bo-')
plt.title('Como varia o Índice CH com o número de grupos')
plt.xlabel('Número de Grupos')
plt.ylabel('Nota CH')
plt.grid(True)

# Gráfico do DB
plt.subplot(1, 2, 2)
plt.plot(k_range, db_scores, 'ro-')
plt.title('Como varia o Índice DB com o número de grupos')
plt.xlabel('Número de Grupos')
plt.ylabel('Nota DB')
plt.grid(True)

plt.tight_layout()
plt.show()

# Testando com DBSCAN
print("""
Agora vou testar o DBSCAN com diferentes valores de eps:
""")

ch_scores_dbscan = []
db_scores_dbscan = []

for eps in eps_values:
  dbscan = DBSCAN(eps=eps, min_samples=3)
  labels = dbscan.fit_predict(df_scaled[features_clustering])

  # Só calculo as notas se encontrar grupos válidos
  if len(set(labels)) > 1 and -1 not in labels:
      ch_score = calinski_harabasz_score(df_scaled[features_clustering],
      db_score = davies_bouldin_score(df_scaled[features_clustering], lab
  else:
      ch_score = 0
      db_score = float('inf')
```

```
  ch_scores_dbscan.append(ch_score)
  db_scores_dbscan.append(db_score)

  print(f"\nCom eps = {eps:.2f}:")
  print(f"Nota CH: {ch_score:.2f}")
  print(f"Nota DB: {db_score:.2f}")

print(f"""
O que aprendi usando essas diferentes formas de avaliar:

1. Índice Calinski-Harabasz me diz:
  - Quanto maior a nota, melhor a divisão
  - Ele olha o quão diferentes são os grupos entre si e o quão parecidos
    países dentro de cada grupo
  - Segundo ele, o melhor seria usar {k_range[np.argmax(ch_scores)]} grup

2. Índice Davies-Bouldin me mostra:
  - Aqui, quanto menor a nota, melhor
  - Ele mede o quão parecidos são os grupos entre si
  - Ele sugere que o ideal seria {k_range[np.argmin(db_scores)]} grupos

3. Comparando com o índice de Silhueta:
  - A Silhueta tinha sugerido {k_otimo} grupos
  - É interessante ver que cada índice sugere um número diferente
  - Isso me mostra a importância de usar várias métricas para tomar uma d

Essa análise mais completa me ajudou a ter mais confiança na qualidade do
grupos que encontrei, mesmo que os índices não concordem totalmente entre
""")
```

Vou avaliar como o K-means se comporta usando diferentes números de grupo
s:

Quando divido em 2 grupos:
Nota CH: 219.04 (quanto maior, melhor)
Nota DB: 0.83 (quanto menor, melhor)
Nota Silhueta: 0.46 (maior é melhor)

Quando divido em 3 grupos:
Nota CH: 258.89 (quanto maior, melhor)
Nota DB: 0.79 (quanto menor, melhor)
Nota Silhueta: 0.49 (maior é melhor)

Quando divido em 4 grupos:
Nota CH: 264.49 (quanto maior, melhor)
Nota DB: 0.73 (quanto menor, melhor)
Nota Silhueta: 0.45 (maior é melhor)

Quando divido em 5 grupos:
Nota CH: 282.45 (quanto maior, melhor)
Nota DB: 0.77 (quanto menor, melhor)
Nota Silhueta: 0.46 (maior é melhor)

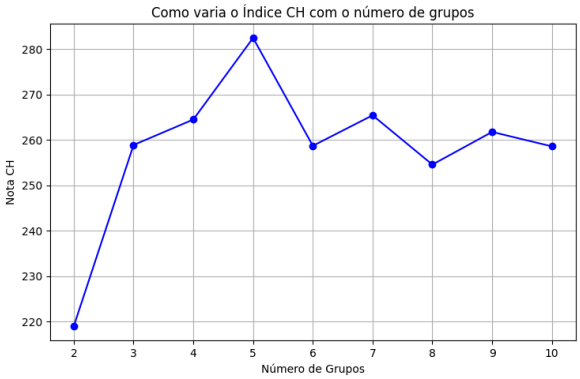
Quando divido em 6 grupos:
Nota CH: 258.71 (quanto maior, melhor)
Nota DB: 0.80 (quanto menor, melhor)
Nota Silhueta: 0.46 (maior é melhor)

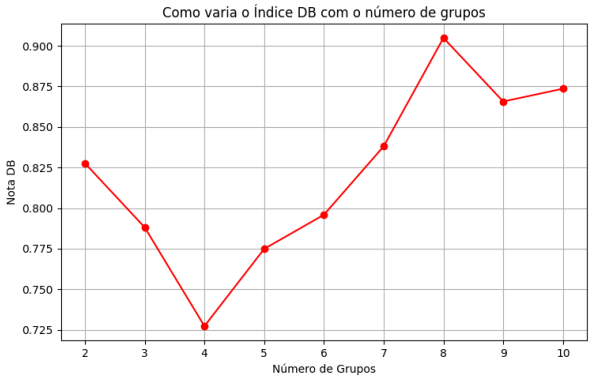
Quando divido em 7 grupos:
Nota CH: 265.42 (quanto maior, melhor)
Nota DB: 0.84 (quanto menor, melhor)
Nota Silhueta: 0.41 (maior é melhor)

Quando divido em 8 grupos:
Nota CH: 254.58 (quanto maior, melhor)
Nota DB: 0.90 (quanto menor, melhor)
Nota Silhueta: 0.35 (maior é melhor)

Quando divido em 9 grupos:
Nota CH: 261.76 (quanto maior, melhor)
Nota DB: 0.87 (quanto menor, melhor)
Nota Silhueta: 0.36 (maior é melhor)

Quando divido em 10 grupos:
Nota CH: 258.58 (quanto maior, melhor)
Nota DB: 0.87 (quanto menor, melhor)
Nota Silhueta: 0.35 (maior é melhor)





Agora vou testar o DBSCAN com diferentes valores de eps:

Com eps = 0.20:
Nota CH: 0.00
Nota DB: inf

Com eps = 0.40:
Nota CH: 0.00
Nota DB: inf

Com eps = 0.60:
Nota CH: 0.00
Nota DB: inf

Com eps = 0.80:
Nota CH: 0.00
Nota DB: inf

Com eps = 1.00:
Nota CH: 0.00
Nota DB: inf

Com eps = 1.20:
Nota CH: 0.00
Nota DB: inf

Com eps = 1.40:
Nota CH: 0.00
Nota DB: inf

Com eps = 1.60:
Nota CH: 0.00
Nota DB: inf

Com eps = 1.80:
Nota CH: 0.00
Nota DB: inf

Com eps = 2.00:
Nota CH: 0.00
Nota DB: inf

O que aprendi usando essas diferentes formas de avaliar:

1. Índice Calinski-Harabasz me diz:
  - Quanto maior a nota, melhor a divisão
  - Ele olha o quão diferentes são os grupos entre si e o quão parecidos s
ão os
    países dentro de cada grupo
  - Segundo ele, o melhor seria usar 5 grupos

2. Índice Davies-Bouldin me mostra:
  - Aqui, quanto menor a nota, melhor
  - Ele mede o quão parecidos são os grupos entre si
  - Ele sugere que o ideal seria 4 grupos

3. Comparando com o índice de Silhueta:
  - A Silhueta tinha sugerido 3 grupos
  - É interessante ver que cada índice sugere um número diferente

- Isso me mostra a importância de usar várias métricas para tomar uma decisão

Essa análise mais completa me ajudou a ter mais confiança na qualidade dos grupos que encontrei, mesmo que os índices não concordem totalmente entre si.

In [18]:
```
# Parte 3 - Clusterização
# Questão 5: Realizando a análise, responda: A silhueta é um o índice ind

print("""
Após trabalhar com o DBSCAN e o índice de silhueta, percebi que talvez es
a melhor combinação. Explico abaixo o por quê:

1. Primeiro, preciso explicar como o DBSCAN funciona:
  - Ele procura grupos olhando para quantos pontos estão próximos uns dos
  - Pode encontrar grupos de qualquer formato, não só "redondos"
  - É esperto o suficiente para identificar pontos que não pertencem a gr
  - Não preciso dizer quantos grupos quero encontrar, ele descobre sozinh

2. Já o índice de silhueta tem algumas particularidades:
  - Ele pressupõe que os grupos são "redondos" ou "ovais"
  - Calcula médias de distâncias entre os pontos
  - Não sabe bem o que fazer com pontos que não pertencem a grupo nenhum
  - Prefere quando os grupos são mais ou menos do mesmo tamanho

3. Por isso, encontrei alguns problemas ao usar a silhueta com DBSCAN:
  - Quando o DBSCAN marca um ponto como "não pertence a grupo nenhum",
    a silhueta não sabe como lidar com isso
  - Se o DBSCAN encontra grupos de formatos diferentes, a silhueta pode
    dar uma nota ruim mesmo o resultado sendo bom
  - A silhueta não leva em conta uma coisa que é super importante para
    o DBSCAN: quão denso é cada grupo

4. Então, por que não é uma boa ideia usar a silhueta neste caso:
  - É como se o DBSCAN e a silhueta falassem línguas diferentes
  - A silhueta pode dizer que um agrupamento está ruim quando na verdade
    está exatamente como o DBSCAN deveria encontrar
  - Os pontos "especiais" que o DBSCAN encontra acabam atrapalhando o
    cálculo da silhueta
  - Se alguns grupos são mais densos que outros, a silhueta pode não
    entender isso direito

5. Que outras opções eu teria para avaliar o DBSCAN:
  - Existe uma métrica chamada DBCV que foi feita especialmente para isso
  - Poderia usar índices que olham para a densidade dos grupos
  - Métricas que entendem melhor como o DBSCAN funciona
  - Às vezes, olhar os gráficos com atenção pode dizer mais que qualquer

Em resumo, aprendi que nem sempre podemos usar a mesma métrica para avali
diferentes tipos de agrupamento. Cada método tem suas particularidades, e
precisamos escolher formas de avaliação que façam sentido para cada caso.
""")
```

Após trabalhar com o DBSCAN e o índice de silhueta, percebi que talvez est
a não seja
a melhor combinação. Explico abaixo o por quê:

1. Primeiro, preciso explicar como o DBSCAN funciona:
  - Ele procura grupos olhando para quantos pontos estão próximos uns dos
outros
  - Pode encontrar grupos de qualquer formato, não só "redondos"
  - É esperto o suficiente para identificar pontos que não pertencem a gru
po nenhum
  - Não preciso dizer quantos grupos quero encontrar, ele descobre sozinho

2. Já o índice de silhueta tem algumas particularidades:
  - Ele pressupõe que os grupos são "redondos" ou "ovais"
  - Calcula médias de distâncias entre os pontos
  - Não sabe bem o que fazer com pontos que não pertencem a grupo nenhum
  - Prefere quando os grupos são mais ou menos do mesmo tamanho

3. Por isso, encontrei alguns problemas ao usar a silhueta com DBSCAN:
  - Quando o DBSCAN marca um ponto como "não pertence a grupo nenhum",
    a silhueta não sabe como lidar com isso
  - Se o DBSCAN encontra grupos de formatos diferentes, a silhueta pode
    dar uma nota ruim mesmo o resultado sendo bom
  - A silhueta não leva em conta uma coisa que é super importante para
    o DBSCAN: quão denso é cada grupo

4. Então, por que não é uma boa ideia usar a silhueta neste caso:
  - É como se o DBSCAN e a silhueta falassem línguas diferentes
  - A silhueta pode dizer que um agrupamento está ruim quando na verdade
    está exatamente como o DBSCAN deveria encontrar
  - Os pontos "especiais" que o DBSCAN encontra acabam atrapalhando o
    cálculo da silhueta
  - Se alguns grupos são mais densos que outros, a silhueta pode não
    entender isso direito

5. Que outras opções eu teria para avaliar o DBSCAN:
  - Existe uma métrica chamada DBCV que foi feita especialmente para isso
  - Poderia usar índices que olham para a densidade dos grupos
  - Métricas que entendem melhor como o DBSCAN funciona
  - Às vezes, olhar os gráficos com atenção pode dizer mais que qualquer n
úmero

Em resumo, aprendi que nem sempre podemos usar a mesma métrica para avalia
r
diferentes tipos de agrupamento. Cada método tem suas particularidades, e
precisamos escolher formas de avaliação que façam sentido para cada caso.

In [19]:
```
# CÓDIGO BONUS - Lista detalhada de países por cluster (K-means)
# com output para o arquivo 'resultados_clusters_kmeans.txt'

# Criar DataFrame com os resultados do K-means e dados de vacinação
resultados_kmeans = pd.DataFrame({
    'País': df_clustering['country'],
    'Cluster': labels_kmeans,
    'Doses Totais/100': df_clustering['total_vaccinations_per_hundred'].ro
    'Pop. Vacinada (%)': df_clustering['people_fully_vaccinated_per_hundre
    'Vacinação Diária/Mi': df_clustering['daily_vaccinations_per_million']
})
```

```
# Ordenar por cluster e doses totais (decrescente)
resultados_kmeans = resultados_kmeans.sort_values(['Cluster', 'Doses Tota

# Preparar o conteúdo para exibição e arquivo
output_content = []
output_content.append("DISTRIBUIÇÃO DETALHADA DE PAÍSES POR CLUSTER (K-ME
output_content.append('-' * 100)

# Gerar conteúdo para cada cluster
for cluster in sorted(resultados_kmeans['Cluster'].unique()):
    cluster_data = resultados_kmeans[resultados_kmeans['Cluster'] == clust

    output_content.append(f"\nCluster {cluster}")
    output_content.append(f"Número de países: {len(cluster_data)}")
    output_content.append(f"Média de doses totais por 100 habitantes: {clu
    output_content.append(f"Média de população totalmente vacinada: {clust
    output_content.append(f"Média de vacinação diária por milhão: {cluster
    output_content.append("\nPaíses e suas estatísticas de vacinação:")
    output_content.append('-' * 100)

    # Formatar a exibição dos dados
    formatted_data = cluster_data.copy()
    formatted_data.columns = ['País', 'Cluster', 'Doses/100', 'Pop.Vac(%)'
    output_content.append(formatted_data[['País', 'Doses/100', 'Pop.Vac(%)
    output_content.append('-' * 100)

# Converter todo o conteúdo para string
output_text = '\n'.join(output_content)

# Exibir no notebook
print(output_text)

# Salvar em arquivo
with open('resultados_clusters_kmeans.txt', 'w', encoding='utf-8') as f:
    f.write(output_text)

print("\nOs resultados foram salvos no arquivo 'resultados_clusters_kmean
```

```
DISTRIBUIÇÃO DETALHADA DE PAÍSES POR CLUSTER (K-MEANS)
--------------------------------------------------------------------------------
--------------------------

Cluster 0
Número de países: 72
Média de doses totais por 100 habitantes: 40.34
Média de população totalmente vacinada: 19.37%
Média de vacinação diária por milhão: 509.69

Países e suas estatísticas de vacinação:
--------------------------------------------------------------------------------
--------------------------
```

País	Doses/100	Pop.Vac(%)	Diária/Mi
Timor	96.52	43.08	1216.0
Albania	95.87	42.30	634.0
Jordan	94.73	42.96	958.0
North Macedonia	88.18	40.10	233.0
Romania	87.50	42.26	63.0
Dominica	86.37	41.47	443.0
Suriname	85.25	40.08	125.0
Montserrat	84.54	36.22	803.0
Bahamas	84.19	39.33	448.0
Mozambique	81.51	40.83	0.0
Lebanon	77.94	32.52	350.0
Grenada	77.12	33.55	274.0
Ukraine	72.89	35.02	1201.0
Comoros	71.81	33.90	0.0
Georgia	71.79	31.57	387.0
Palestine	70.75	33.63	465.0
Armenia	70.38	31.97	1115.0
Saint Lucia	63.85	28.59	304.0
Bulgaria	63.05	29.72	202.0
Saint Vincent and the Grenadines	62.55	27.07	315.0
Bosnia and Herzegovina	58.98	25.93	1322.0
South Africa	55.77	29.70	1101.0
Mauritania	55.51	22.14	181.0
Moldova	52.78	26.20	289.0
Angola	51.68	17.66	1007.0
Solomon Islands	50.86	17.39	3105.0
Libya	49.09	16.18	486.0
Jamaica	46.88	22.62	297.0
Guinea	44.10	18.37	1031.0
Kyrgyzstan	43.80	18.67	743.0
Eswatini	43.45	27.16	426.0
Iraq	42.59	17.29	499.0
Lesotho	41.94	33.70	629.0
Ghana	39.83	15.34	1248.0
Cote d'Ivoire	38.31	17.00	596.0
Uganda	36.50	17.01	678.0
Equatorial Guinea	32.65	14.53	194.0
Togo	32.53	18.37	388.0
Namibia	32.26	14.84	405.0
Kenya	31.51	14.71	543.0
Algeria	30.72	13.70	97.0
Guinea-Bissau	27.91	16.94	16.0
Benin	26.77	19.52	1566.0
Sierra Leone	26.22	13.68	3068.0
Ethiopia	24.92	17.78	193.0
Gabon	24.09	11.00	41.0

Liberia	21.86	19.61	487.0
Central African Republic	20.59	18.25	639.0
Syria	18.44	7.22	75.0
Zambia	17.98	11.54	721.0
Djibouti	16.52	10.48	194.0
Nigeria	14.85	4.52	935.0
Somalia	14.64	7.89	1032.0
Gambia	14.61	12.79	3.0
Senegal	14.52	6.04	72.0
Congo	14.48	11.32	15.0
Afghanistan	14.44	11.10	159.0
Sudan	13.65	6.12	468.0
Burkina Faso	11.02	5.53	118.0
Niger	10.67	6.15	18.0
Malawi	10.15	4.39	215.0
Mali	8.98	4.69	407.0
Tanzania	8.18	4.99	0.0
Papua New Guinea	6.30	2.82	101.0
Cameroon	5.73	3.85	874.0
South Sudan	5.05	4.24	144.0
Madagascar	4.60	3.70	122.0
Yemen	2.65	1.33	74.0
Chad	2.48	0.91	12.0
Haiti	2.16	0.97	72.0
Democratic Republic of Congo	1.04	0.56	52.0
Burundi	0.10	0.08	4.0

Cluster 1
Número de países: 45
Média de doses totais por 100 habitantes: 161.38
Média de população totalmente vacinada: 65.07%
Média de vacinação diária por milhão: 2840.16

Países e suas estatísticas de vacinação:

País	Doses/100	Pop.Vac(%)	Diária/Mi
Chile	261.82	90.29	2879.0
Brunei	241.99	91.71	2143.0
Uruguay	230.34	81.56	2897.0
China	225.94	85.91	2822.0
Qatar	225.46	88.50	2696.0
Jersey	223.69	78.48	4096.0
Australia	218.10	82.21	2547.0
Cambodia	217.27	82.55	2128.0
Taiwan	207.43	76.52	1878.0
Sweden	207.22	74.94	2028.0
Vietnam	206.93	79.20	3514.0
Peru	196.50	76.38	2981.0
Mauritius	195.87	75.66	4606.0
Brazil	193.26	74.90	3069.0
Ecuador	185.40	76.64	1812.0
Thailand	184.25	71.71	2664.0
Kuwait	180.53	75.87	2038.0
Panama	180.05	68.88	3261.0
Saudi Arabia	176.54	69.21	1765.0
Hungary	170.54	64.22	2435.0
Cook Islands	168.25	82.93	1935.0

Maldives	168.00	68.33	1755.0
Monaco	164.83	64.95	3998.0
El Salvador	159.83	65.69	2439.0
Colombia	157.17	67.74	2268.0
Nauru	154.73	70.58	2483.0
Mexico	147.32	61.19	3314.0
Tonga	137.87	65.83	2988.0
Indonesia	136.45	57.47	4639.0
Laos	136.18	60.38	2346.0
Nepal	134.58	64.08	2345.0
India	131.66	59.44	1852.0
Uzbekistan	130.18	40.82	2579.0
Falkland Islands	124.91	50.31	3401.0
Belarus	123.21	58.94	4444.0
Wallis and Futuna	117.84	58.20	1713.0
Kiribati	110.35	45.58	4135.0
Bolivia	109.63	48.58	1919.0
Tajikistan	107.08	47.33	4659.0
Tuvalu	101.58	49.34	2516.0
Pakistan	97.41	45.24	2134.0
Myanmar	89.98	39.78	2740.0
Guatemala	86.83	32.49	3623.0
Egypt	73.58	30.43	2632.0
Vanuatu	63.43	27.09	4691.0

Cluster 2
Número de países: 87
Média de doses totais por 100 habitantes: 174.85
Média de população totalmente vacinada: 68.19%
Média de vacinação diária por milhão: 539.30

Países e suas estatísticas de vacinação:

País	Doses/100	Pop.Vac(%)	Diária/Mi
Cuba	312.28	87.57	1118.0
Singapore	253.51	91.04	1501.0
United Arab Emirates	244.86	96.12	845.0
Malta	243.33	90.35	727.0
Guernsey	237.79	81.01	394.0
South Korea	235.07	86.70	732.0
Denmark	227.27	82.60	162.0
Portugal	225.45	92.60	624.0
Italy	225.07	79.21	715.0
Isle of Man	221.87	78.42	808.0
Scotland	221.65	76.34	707.0
Wales	218.53	75.69	508.0
Iceland	218.41	78.69	165.0
Belgium	216.95	78.46	363.0
Canada	215.61	81.83	642.0
Ireland	215.57	80.43	873.0
New Zealand	213.61	79.04	1443.0
Cayman Islands	212.21	87.80	932.0
Faeroe Islands	211.80	83.37	530.0
Argentina	211.61	80.96	1200.0
Finland	210.50	77.66	699.0
France	210.12	77.78	474.0
Malaysia	209.89	78.72	1040.0

England	208.97	73.39	496.0
United Kingdom	206.68	72.43	473.0
Norway	206.18	73.68	181.0
Germany	204.93	75.26	541.0
Bermuda	204.68	75.21	1449.0
Luxembourg	201.66	72.51	400.0
Seychelles	201.66	80.97	799.0
Spain	200.68	85.82	721.0
San Marino	200.62	69.40	617.0
Austria	200.50	72.86	155.0
Pitcairn	200.00	100.00	0.0
Netherlands	198.21	72.01	309.0
Northern Ireland	198.04	70.18	524.0
Cyprus	197.93	72.02	1265.0
Greece	197.70	73.21	1055.0
Andorra	196.50	68.99	440.0
Bahrain	195.69	69.56	300.0
Israel	194.41	65.96	285.0
Liechtenstein	184.12	69.00	366.0
Sri Lanka	181.90	66.93	983.0
Switzerland	179.39	68.77	255.0
Turks and Caicos Islands	175.06	74.95	382.0
Turkey	172.72	62.29	538.0
Iran	172.64	66.81	1148.0
United States	168.72	65.51	347.0
Lithuania	166.02	69.61	183.0
Mongolia	164.56	65.31	77.0
Czechia	163.10	63.95	284.0
Aruba	157.87	75.89	718.0
Latvia	153.96	69.75	282.0
Anguilla	150.18	63.63	727.0
Curacao	149.62	59.76	358.0
Estonia	148.78	63.60	357.0
Morocco	145.14	62.52	350.0
Fiji	144.89	69.49	299.0
Nicaragua	143.41	61.37	0.0
Slovenia	143.21	58.70	240.0
Sint Maarten (Dutch part)	142.89	59.81	645.0
Poland	142.63	59.12	374.0
Dominican Republic	141.37	54.08	662.0
Greenland	140.23	67.70	35.0
Oman	133.76	57.84	852.0
Venezuela	131.90	49.77	0.0
Azerbaijan	131.32	47.16	789.0
French Polynesia	129.92	63.99	35.0
Cape Verde	129.57	54.60	733.0
Slovakia	129.42	50.71	14.0
Croatia	127.99	54.83	202.0
New Caledonia	127.64	62.41	97.0
Antigua and Barbuda	127.00	62.34	273.0
Turkmenistan	123.91	52.41	0.0
Serbia	123.63	47.58	82.0
Honduras	119.20	47.48	1421.0
Paraguay	116.74	45.94	1047.0
Belize	115.39	51.65	699.0
Saint Kitts and Nevis	112.93	49.11	336.0
Guyana	112.55	45.62	624.0
Russia	112.12	49.92	1134.0
Kazakhstan	110.13	48.17	565.0
Trinidad and Tobago	110.00	50.50	378.0

```
                Tunisia     109.23       53.16      145.0
               Barbados     108.49       52.51      316.0
             Montenegro     106.39       44.83      237.0
                 Kosovo     102.23       46.03      118.0
------------------------------------------------------------------------
-------------------------

Os resultados foram salvos no arquivo 'resultados_clusters_kmeans.txt'
```

Parte 4 - Medidas de similaridade

1. Um determinado problema, apresenta 10 séries temporais distintas. Gostaríamos de agrupá-las em 3 grupos, de acordo com um critério de similaridade, baseado no valor máximo de correlação cruzada entre elas. Descreva em tópicos todos os passos necessários.
2. Para o problema da questão anterior, indique qual algoritmo de clusterização você usaria. Justifique.
3. Indique um caso de uso para essa solução projetada.
4. Sugira outra estratégia para medir a similaridade entre séries temporais. Descreva em tópicos os passos necessários.

In [20]:
```
# Parte 4 - Medidas de similaridade
# Questão 1: Um determinado problema, apresenta 10 séries temporais disti
# Gostaríamos de agrupá-las em 3 grupos, de acordo com um critério de sim
# baseado no valor máximo de correlação cruzada entre elas.
# Descreva em tópicos todos os passos necessários.

print("""
Para agrupar as 10 séries temporais em 3 grupos usando correlação cruzada

1. Preparar os dados:
  • Checar se todas as séries têm o mesmo tamanho
  • Padronizar cada série (média 0, variância 1)
  • Ver se há dados faltantes e tratá-los

2. Calcular correlações:
  • Criar matriz 10x10 vazia
  • Para cada par de séries:
    ▸ Calcular correlação cruzada completa
    ▸ Pegar o maior valor de correlação
    ▸ Colocar na matriz
  • Garantir que a matriz fique simétrica

3. Transformar em distâncias:
  • Converter correlações em distâncias usando: d = 1 - correlação
  • Verificar se as distâncias fazem sentido:
    ▸ Nenhuma distância negativa
    ▸ Matriz simétrica
    ▸ Distância zero apenas entre série e ela mesma

4. Preparar para agrupar:
  • Escolher algoritmo adequado
  • Definir parâmetros:
    ▸ Vamos fazer 3 grupos
    ▸ Como vamos ligar os grupos
    ▸ Quando parar de agrupar
```

```
5. Executar o agrupamento:
  • Rodar o algoritmo
  • Ver em qual grupo cada série ficou
  • Contar quantas séries em cada grupo

6. Verificar se deu certo:
  • Ver se as séries em cada grupo são parecidas
  • Checar se os grupos estão bem separados
  • Confirmar se faz sentido para o problema real
""")
```

Para agrupar as 10 séries temporais em 3 grupos usando correlação cruzada, preciso:

1. Preparar os dados:
  • Checar se todas as séries têm o mesmo tamanho
  • Padronizar cada série (média 0, variância 1)
  • Ver se há dados faltantes e tratá-los

2. Calcular correlações:
  • Criar matriz 10x10 vazia
  • Para cada par de séries:
    ▸ Calcular correlação cruzada completa
    ▸ Pegar o maior valor de correlação
    ▸ Colocar na matriz
  • Garantir que a matriz fique simétrica

3. Transformar em distâncias:
  • Converter correlações em distâncias usando: d = 1 - correlação
  • Verificar se as distâncias fazem sentido:
    ▸ Nenhuma distância negativa
    ▸ Matriz simétrica
    ▸ Distância zero apenas entre série e ela mesma

4. Preparar para agrupar:
  • Escolher algoritmo adequado
  • Definir parâmetros:
    ▸ Vamos fazer 3 grupos
    ▸ Como vamos ligar os grupos
    ▸ Quando parar de agrupar

5. Executar o agrupamento:
  • Rodar o algoritmo
  • Ver em qual grupo cada série ficou
  • Contar quantas séries em cada grupo

6. Verificar se deu certo:
  • Ver se as séries em cada grupo são parecidas
  • Checar se os grupos estão bem separados
  • Confirmar se faz sentido para o problema real

In [21]:
```
# Parte 4 - Medidas de similaridade
# Questão 2: Para o problema da questão anterior, indique qual algoritmo

print("""
Para este problema de séries temporais, sugiro:

1. Minha escolha: Clustering Hierárquico Aglomerativo
```

```
  • É como construir uma árvore genealógica dos dados
  • Começa com cada série sozinha e vai juntando as mais parecidas

2. Por que escolhi este método:
  • Trabalha direto com a matriz de distâncias que calculei
  • Não exige que os dados sigam um formato específico
  • Posso ver a "árvore" de agrupamentos (dendrograma)
  • Aceita séries com comportamentos diversos

3. Vantagens para séries temporais:
  • Perfeito para usar com correlação cruzada
  • Posso escolher diferentes formas de juntar os grupos
  • Fácil de entender como os grupos foram formados
  • Lida bem com diferentes padrões temporais

4. Por que não usar outros métodos:
  • K-means: Não funciona bem com correlação cruzada
  • DBSCAN: Não consigo forçar exatamente 3 grupos
  • Spectral: Muito complicado computacionalmente
  • Gaussian Mixture: Assume padrões que podem não existir

5. Como vou implementar:
  • Vou usar 'complete' ou 'average' para juntar os grupos
  • Cortar a árvore para ter 3 grupos
  • Posso visualizar tudo num gráfico bonito
  • Fácil de ajustar se precisar

""")
```

```
Para este problema de séries temporais, sugiro:

1. Minha escolha: Clustering Hierárquico Aglomerativo
  • É como construir uma árvore genealógica dos dados
  • Começa com cada série sozinha e vai juntando as mais parecidas

2. Por que escolhi este método:
  • Trabalha direto com a matriz de distâncias que calculei
  • Não exige que os dados sigam um formato específico
  • Posso ver a "árvore" de agrupamentos (dendrograma)
  • Aceita séries com comportamentos diversos

3. Vantagens para séries temporais:
  • Perfeito para usar com correlação cruzada
  • Posso escolher diferentes formas de juntar os grupos
  • Fácil de entender como os grupos foram formados
  • Lida bem com diferentes padrões temporais

4. Por que não usar outros métodos:
  • K-means: Não funciona bem com correlação cruzada
  • DBSCAN: Não consigo forçar exatamente 3 grupos
  • Spectral: Muito complicado computacionalmente
  • Gaussian Mixture: Assume padrões que podem não existir

5. Como vou implementar:
  • Vou usar 'complete' ou 'average' para juntar os grupos
  • Cortar a árvore para ter 3 grupos
  • Posso visualizar tudo num gráfico bonito
  • Fácil de ajustar se precisar
```

In [22]:
```
# Parte 4 - Medidas de similaridade
# Questão 3: Indique um caso de uso para essa solução projetada.

print("""
Aqui está um exemplo prático de como usar este método:

1. O Problema: Análise de Consumo de Energia
  • Empresa que distribui energia elétrica
  • Monitora 10 regiões diferentes
  • Tem dados de consumo ao longo do tempo
  • Precisa otimizar como usa seus recursos

2. Como vou aplicar a solução:
  • Cada série temporal = consumo diário de uma região
  • Uso correlação cruzada para achar padrões parecidos
  • Divido em 3 grupos = 3 perfis diferentes de consumo
  • Resultado ajuda no planejamento da distribuição

3. Benefícios que isso traz:
  • Otimizo recursos baseado no perfil de cada região
  • Consigo prever melhor a demanda
  • Faço manutenção preventiva mais eficiente
  • Planejo melhor a infraestrutura

4. Na prática, isso significa:
  • Economia nos custos de operação
  • Serviço melhor para os clientes
  • Menos risco de sobrecarga na rede
  • Estratégias sob medida para cada grupo

5. Como manter funcionando bem:
  • Verifico os grupos periodicamente
  • Ajusto para mudanças nas estações do ano
  • Melhoro as estratégias conforme aprendo mais
  • Confirmo sempre se está dando certo

Esta aplicação é interessante porque mostra como uma técnica matemática p
resolver um problema real do dia a dia.
""")
```

Aqui está um exemplo prático de como usar este método:

1. O Problema: Análise de Consumo de Energia
   • Empresa que distribui energia elétrica
   • Monitora 10 regiões diferentes
   • Tem dados de consumo ao longo do tempo
   • Precisa otimizar como usa seus recursos

2. Como vou aplicar a solução:
   • Cada série temporal = consumo diário de uma região
   • Uso correlação cruzada para achar padrões parecidos
   • Divido em 3 grupos = 3 perfis diferentes de consumo
   • Resultado ajuda no planejamento da distribuição

3. Benefícios que isso traz:
   • Otimizo recursos baseado no perfil de cada região
   • Consigo prever melhor a demanda
   • Faço manutenção preventiva mais eficiente
   • Planejo melhor a infraestrutura

4. Na prática, isso significa:
   • Economia nos custos de operação
   • Serviço melhor para os clientes
   • Menos risco de sobrecarga na rede
   • Estratégias sob medida para cada grupo

5. Como manter funcionando bem:
   • Verifico os grupos periodicamente
   • Ajusto para mudanças nas estações do ano
   • Melhoro as estratégias conforme aprendo mais
   • Confirmo sempre se está dando certo

Esta aplicação é interessante porque mostra como uma técnica matemática po
de
resolver um problema real do dia a dia.

In [23]:
```
# Parte 4 - Medidas de similaridade
# Questão 4: Sugira outra estratégia para medir a similaridade entre séri

print("""
Estratégia Alternativa: Dynamic Time Warping (DTW)

O que é DTW:
  • Método para medir similaridade entre séries temporais.
  • Permite comparar séries mesmo que tenham velocidades ou alinhamentos
  • Mais flexível que a correlação cruzada, pois considera o melhor alinh

Passos para aplicar:

1. Antes de começar:
  • Normalize as séries para padronizar os valores.
  • Confira se há valores faltantes e trate esses casos.
  • Ajuste o comprimento das séries, se necessário.
  • Remova ruídos, caso identifique que podem atrapalhar.

2. Calculando a matriz de distância DTW:
  • Compare cada par de séries:
    - Crie uma matriz de custo que mede as diferenças.
    - Encontre o caminho de warping (alinhamento ideal).
```

```
    - Calcule a distância DTW para cada par.
  • Monte uma matriz 10x10 com as distâncias entre todas as séries.

3. Melhorando a eficiência:
  • Use a banda de Sakoe-Chiba para limitar o cálculo.
  • Aplique uma janela de restrição para reduzir a complexidade.
  • Tente o FastDTW, uma versão mais rápida.
  • Paralelize o processamento, se estiver lidando com muitas séries.

4. Por que usar DTW:
  • É robusto a diferenças no alinhamento temporal.
  • Capta padrões parecidos mesmo em velocidades diferentes.
  • Não depende de periodicidade.
  • Identifica bem formas similares.

5. Aplicando ao clustering:
  • Use a matriz de distâncias DTW como entrada.
  • Aplique clustering hierárquico para agrupar os dados.
  • Divida em 3 grupos, conforme especificado.
  • Valide os resultados para garantir que os agrupamentos fazem sentido.

Essa estratégia é interessante porque consegue lidar com a flexibilidade
""")
```

Estratégia Alternativa: Dynamic Time Warping (DTW)

O que é DTW:
- Método para medir similaridade entre séries temporais.
- Permite comparar séries mesmo que tenham velocidades ou alinhamentos diferentes.
- Mais flexível que a correlação cruzada, pois considera o melhor alinhamento entre elas.

Passos para aplicar:

1. Antes de começar:
- Normalize as séries para padronizar os valores.
- Confira se há valores faltantes e trate esses casos.
- Ajuste o comprimento das séries, se necessário.
- Remova ruídos, caso identifique que podem atrapalhar.

2. Calculando a matriz de distância DTW:
- Compare cada par de séries:
  - Crie uma matriz de custo que mede as diferenças.
  - Encontre o caminho de warping (alinhamento ideal).
  - Calcule a distância DTW para cada par.
- Monte uma matriz 10x10 com as distâncias entre todas as séries.

3. Melhorando a eficiência:
- Use a banda de Sakoe-Chiba para limitar o cálculo.
- Aplique uma janela de restrição para reduzir a complexidade.
- Tente o FastDTW, uma versão mais rápida.
- Paralelize o processamento, se estiver lidando com muitas séries.

4. Por que usar DTW:
- É robusto a diferenças no alinhamento temporal.
- Capta padrões parecidos mesmo em velocidades diferentes.
- Não depende de periodicidade.
- Identifica bem formas similares.

5. Aplicando ao clustering:
- Use a matriz de distâncias DTW como entrada.
- Aplique clustering hierárquico para agrupar os dados.
- Divida em 3 grupos, conforme especificado.
- Valide os resultados para garantir que os agrupamentos fazem sentido.

Essa estratégia é interessante porque consegue lidar com a flexibilidade necessária para comparar séries temporais em cenários complexos.