Analise de cluster K-Means

Code

O conteúdo original pode ser encontrado em: Univesity of Cincinnati (http://uc-r.github.io/kmeans_clustering), tradução minha.

Cluster é um amplo conjunto de técnicas para encontrar subgrupos de observações dentro de um conjunto de dados. Quando agrupamos observações, queremos que as observações no mesmo grupo sejam semelhantes e as observações em diferentes grupos sejam diferentes. Como não há uma variável de resposta, este é um método não supervisionado, o que implica em procura relações entre as n observações e sem treinar utilizando uma variável resposta. As técnicas de clustering nos permite identificar quais observações são semelhantes e potencialmente categorizá-las. O agrupamento de K-means é o método de cluster mais simples e mais usado para dividir um conjunto de dados em um conjunto de grupos k.

Para realizar essa técnica, vamos utilzar os seguintes pacotes:

Hide

```
library(tidyverse)  # Manipulação de dados
library(cluster)    # Algoritmos de cluster
library(factoextra) # Algoritmos de cluster e visualização
library(dendextend) # Comparar Dendrogramas
```

Para rodar uma análise de cluster, geralmente, os dados devem ser tratados levando em consideração alguns pontos.

1. Linhas são observações (individuais) e colunas são variáveis
2. Qualquer valor faltante deve ser removido ou estimado (é necessário conhecer bem a base para optar por alguma dessas decisões)
3. Os dados devem ser normalizados para transformar as variáveis comparáveis. Lembre-se, padronização consiste em transformar as variáveis de tal forma que elas possuam média zero e desvio padrão 1.

Vamos utilzar uma base de dados disponível no R chamada **USArrests**. Essa base contém estatísticas de prisões a cada 100.000 habitantes por assalto, assassinato e estupro para cada um dos 50 estados dos EUA em 1973. Inclui também a porcentagem da população que vive em áreas urbanas:

Hide

```
df <- USArrests
```

Para remover qualquer valor faltante (missings) em nossos dados, podemos digitar:

Hide

```
df <- na.omit(df)
```

Como não queremos que nosso algoritmo dependa de uma variável de valor arbitrário, vamos padronizar os dados utilizando a função scale

Hide

```
df <- scale(df)
head(df)
```

```
               Murder   Assault   UrbanPop         Rape
Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
Arizona    0.07163341 1.4788032  0.9989801  1.042878388
Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
California 0.27826823 1.2628144  1.7589234  2.067820292
Colorado   0.02571456 0.3988593  0.8608085  1.864967207
```

A classificação das observações em grupos requer alguns métodos para calcular a distância ou a (dis)similaridade entre cada par de observações. O resultado dessa computação é conhecido como uma matriz de dissimilaridade ou distância. Existem muitos métodos para calcular essa informação de distância; A escolha das medidas à distância é um passo crítico no agrupamento. Ele define como a similaridade de dois elementos (x, y) é calculada e influenciará a forma dos clusters.

Os métodos clássicos para meidir essas distâncias são $Euclidiana$ e $Manhattan$, que são definidos da seguinte forma:

Distância Euclidiana

$$d_{euc}(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

Distância Manhttan

$$d_{man}(x,y) = \sum_{i=1}^{n}|(x_i - y_i) \tag{2}$$

Onde, x e y são dois vetores de tamanho n.

Existem outras medidas de dissimilaridade, como distâncias baseadas em correlação, onde a a distância é definida substituindo o coeficiente de correlação de 1. Podemos utilizar outros métodos (que não vamos nos aprofundar por hora) como:

Distância de Correlação de Pearson

$$d_{cor}(x,y) = 1 - \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3}$$

Distância de Correlação de Spearman

O método de correlação de Spearman calcula a correlação entre o grau de x eo grau de variáveis y.

$$d_{spear}(x,y) = 1 - \frac{\sum_{i=1}^{n}(x'_i - \bar{x}')(y'_i - \bar{y}')}{\sqrt{\sum_{i=1}^{n}(x'_i - \bar{x}')^2 \sum_{i=1}^{n}(y'_i - \bar{y}')^2}} \tag{4}$$

Onde $x\prime_i = rank(x_i)$ e $y\prime_i = rank(y_i)$.

Cluster K-means (K-médias)

O método de K-means é o mais utilizado para modelos de machine learning não-supervisionados para particionar uma base de dados em um conjunto de k grupos, onde o k representa o número de grupos já predefinido pelo analista. O algoritmo classifica os objetos em vários grupos, de modo que os objetos

pertencentes ao mesmo grupo sejam o mais parecidos possível entre si e o mais diferente dos outros grupos. Para o cluster K-means, cada grupo é representado pelo seu centro (esse é o centróide), que corresponde a média de pontos atribuidas ao cluster.

Ideia básica

A ideia básica por trás do cluster k-means consiste em definir os grupos de tal forma que a variação dentro do cluster é minimizado. Existem vários algoritmos k-means disponíveis. O algoritmo *default* é o algoritmo Hartigan-Wong (1979), que define a variação total dentro do cluster como a soma das distâncias quadradas das distâncias euclidianas entre os itens e o centroide correspondente:

$$W(C_k) = \sum_{x_i \in C_k} (x_i - \mu_k)^2 \qquad (5)$$

Onde:

• x_i é o dados que pertence ao grupo C_k
• μ_k é a média dos valores atribuidos ao grupo C_k

Cada observação de (x_i) é atribuido a um determinado cluster de tal forma que a soma dos quadrados (SS) distância da observação para os centros de agrupamento atribuídos μ_k é minimizado.

Definimos a variação total dentro do cluster como:

$$tot.withiness = \sum_{k=1}^{k} W(C_k) = \sum_{k=1}^{k} \sum_{x_i \in C_k} (x_i - \mu_k)^2 \qquad (6)$$

A soma total de quadrado dentro do cluster mede a compacidade (ou seja, a qualidade) do agrupamento e queremos que seja tão pequeno quanto possível.

O primeiro passo ao usar o agrupamento k-means é indicar o número de clusters (k) que serão gerados na solução final. O algoritmo começa selecionando aleatoriamente k objetos do conjunto de dados para servir como sementes iniciais para os clusters. Os objetos selecionados também são conhecidos como centróides.

Em seguida, cada um dos objetos restantes é atribuído ao centroide mais próximo, onde o mais próximo é definido usando a distância euclidiana entre o objeto e a média do cluster. Este passo é chamado de “atribuição do cluster”. Após o passo de atribuição, o algoritmo calcula o novo valor médio de cada cluster. Então o centróide é recalculado. Agora que os centros foram recalculados, cada observação é verificada novamente para ver se ela pode estar mais próxima de um cluster diferente. Todos os objetos são reatribuídos de novo usando o cluster atualizado. As etapas de atualização do centróide são repetidas iterativamente até que as atribuições de cluster parem de mudar (ou seja, até a convergência). Ou seja, os clusters formados na iteração atual são os mesmos que os obtidos na iteração anterior.

Podemos resumir o algoritmos K-means da seguinte forma:

1. Especificar o número de grupos (k) para ser criados.
2. Selecionar aleatoriamente as sementes aleatórias iniciais
3. Atribui cada observação ao centroide mais próximo, com base na distância euclidiana entre o objeto e o centróide
4. Para cada um dos k clusters, atualizamos o centróide do cluster calculando os novos valores médios de todos os valores do cluster. O centróide de um cluster K^{th} é um vetor de comprimento p contendo os meios de todas as variáveis para as observações no cluster k^{th}. p é o número de variáveis.

Rodando o cluster K-means

Podemos criar um cluster K-means utilizando a função kmeans. Aqui vamos agrupar os dados em dois grupos centers. A função kmeans também tem uma opção nstart que tenta múltiplas configurações iniciais e reporta o melhor. Por exemplo, adicionando nstart = 25 irá gerar 25 configurações iniciais.

Hide

```
k2 <- kmeans(df, centers = 2, nstart = 25)
str(k2)
```

```
List of 9
 $ cluster     : Named int [1:50] 1 1 1 2 1 1 2 2 1 1 ...
  ..- attr(*, "names")= chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...
 $ centers     : num [1:2, 1:4] 1.005 -0.67 1.014 -0.676 0.198 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "1" "2"
  .. ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
 $ totss       : num 196
 $ withinss    : num [1:2] 46.7 56.1
 $ tot.withinss: num 103
 $ betweenss   : num 93.1
 $ size        : int [1:2] 20 30
 $ iter        : int 1
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
```

A saída do kmeans é uma lista com muitas informações, e os mais importantes são:

• cluster: Um vetor integers (de 1:k) indicando o cluster na qual cada ponto está alocado;
• centers: Uma matriz com os centróides;
• totss: O total da soma dos quadrados;
• withinss: Vector de soma de quadrados dentro do cluster, um componente por cluster. • tot.withinss: Soma total de quadrados dentro do cluster • betweenss: A soma dos quadrados entre os clusters • size: O número de observações em cada cluster

Vamos printar os resultados:

Hide

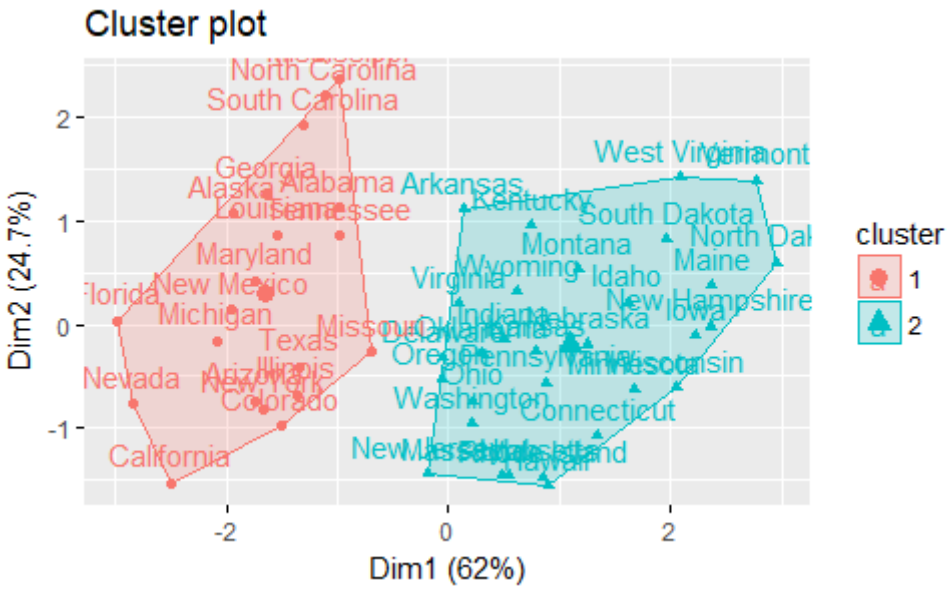
```
k2
```

```
K-means clustering with 2 clusters of sizes 20, 30

Cluster means:
     Murder    Assault   UrbanPop       Rape
1  1.004934  1.0138274  0.1975853  0.8469650
2 -0.669956 -0.6758849 -0.1317235 -0.5646433

Clustering vector:
       Alabama         Alaska        Arizona       Arkansas     California       Colorado
 Connecticut       Delaware
             1              1              1              2              1              1
         2              2
       Florida        Georgia         Hawaii          Idaho       Illinois        Indiana
       Iowa         Kansas
             1              1              2              2              1              2
         2              2
      Kentucky      Louisiana          Maine       Maryland  Massachusetts       Michigan
  Minnesota    Mississippi
             2              1              2              1              2              1
         2              1
      Missouri        Montana       Nebraska         Nevada  New Hampshire     New Jersey
New Mexico       New York
             1              2              2              1              2              2
          1              1
North Carolina   North Dakota           Ohio       Oklahoma         Oregon   Pennsylvania   R
hode Island South Carolina
             1              2              2              2              2              2
          2              1
  South Dakota      Tennessee          Texas           Utah        Vermont       Virginia
Washington  West Virginia
             2              1              1              2              2              2
          2              2
     Wisconsin        Wyoming
             2              2

Within cluster sum of squares by cluster:
[1] 46.74796 56.11445
 (between_SS / total_SS =  47.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
"size"
[8] "iter"         "ifault"
```

Podemos visualizar o resultado utilizando fviz_cluster Veja como fica bonito. Se houver mais de duas dimensões (variáveis), a função fviz_cluster irá rodar uma análise de componente principal (PCA) e plotar os dados de acordo com as primeiras duas dimensões principais dos componentes que explicam a maior parte da variância.

Hide

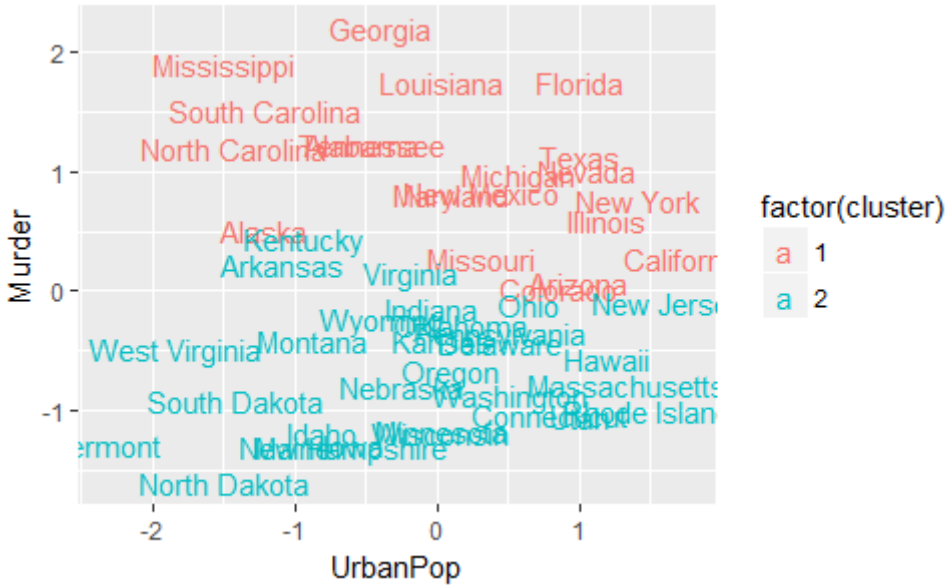
```
fviz_cluster(k2, data = df)
```



Alternativamente, podemos utilizar o gráfico de dispersão para ilustrar os grupos comparados com as variáveis originais.

Hide

```
df %>%
  as_tibble() %>%
  mutate(cluster = k2$cluster,
         state = row.names(USArrests)) %>%
  ggplot(aes(UrbanPop, Murder, color = factor(cluster), label = state)) +
  geom_text()
```



Como o número de clusters (k) deve ser definido antes de iniciar o algoritmo, muitas vezes é vantajoso usar vários valores diferentes de k e examinar as diferenças nos resultados (ciência). Podemos executar o mesmo processo para 3, 4 e 5 clusters, e os resultados analisar os resultados:

Hide

```
k3 <- kmeans(df, centers = 3, nstart = 25)
k4 <- kmeans(df, centers = 4, nstart = 25)
k5 <- kmeans(df, centers = 5, nstart = 25)
# Gráficos para comparação
p1 <- fviz_cluster(k2, geom = "point", data = df) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data = df) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data = df) + ggtitle("k = 4")
p4 <- fviz_cluster(k5, geom = "point",  data = df) + ggtitle("k = 5")
library(gridExtra)
```

```
Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

    combine
```

Hide

```
grid.arrange(p1, p2, p3, p4, nrow = 2)
```

Determinando o número ideal de Cluster

Método Elbow

Lembre-se de que, a idéia básica por trás dos métodos de particionamento de cluster, é definir clusters de modo que a variação dentro do cluster total seja minimizada:

$$minimize(\sum_{i=1}^{n} W(C_k))$$

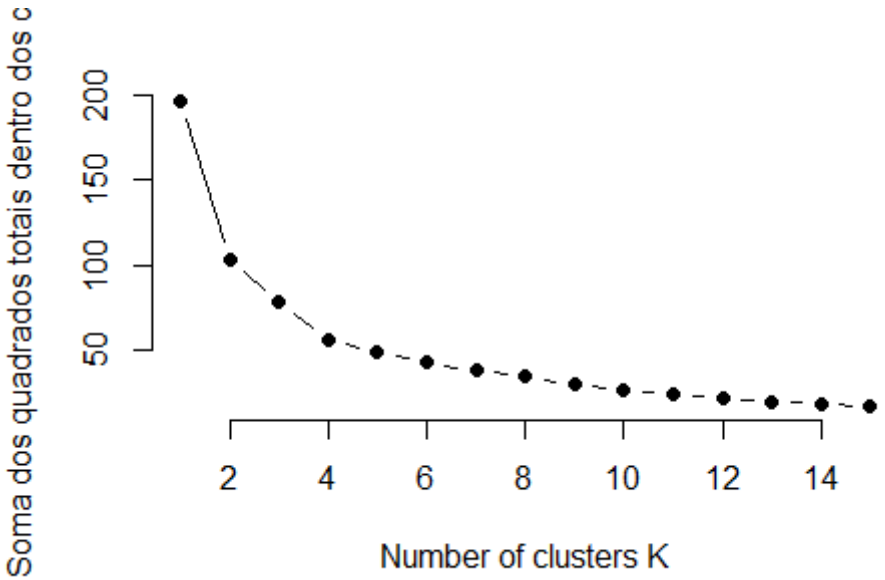
Onde C_k é o K cluster e $W(C_k)$ é a variação dentro do cluster. Então, o total da soma dos quadrados dentro do cluster (wss) mede a compacidade do cluster e queremos que seja tão pequeno quanto possível. Assim, podemos usar o seguinte algoritmo para definir os clusters como sendo ótimos:

1. Rodar o algoritmo de agrupamento para diferentes valores de k. Por exemplo, variando k de 1 a 10 clusters

2. Para cada k, calcular a soma dos quadrados total dentro do cluster (wss)
3. Traçar a curva de wss de acordo com o número de clusters k.
4. A localização de uma curva (joelho) na trama é geralmente considerada como um indicador do número apropriado de clusters.

Hide

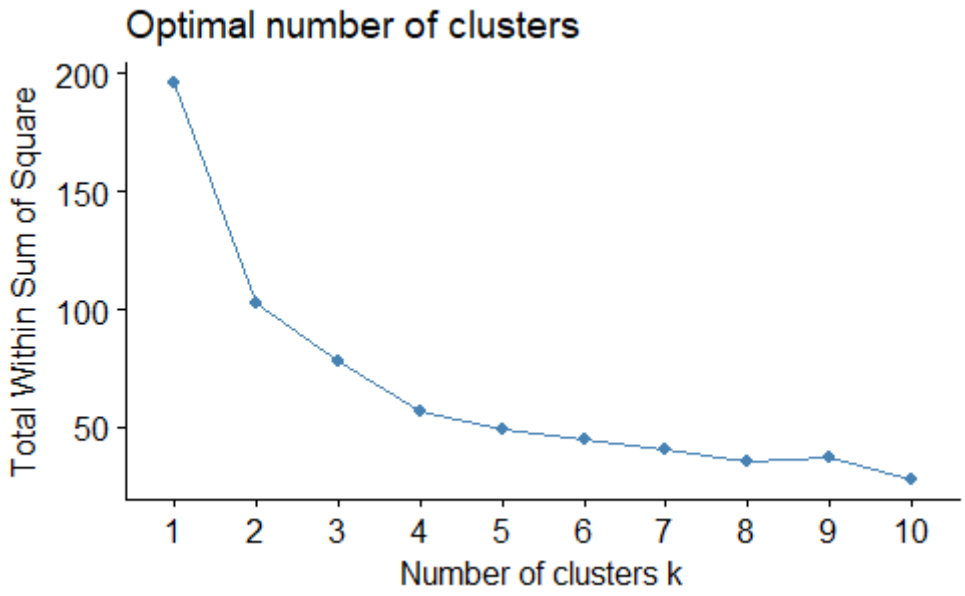
```
set.seed(123)
#  Função para calcular o total das somas dos quadrados dentro do cluster
wss <- function(k) {
  kmeans(df, k, nstart = 10 )$tot.withinss
}
# Calcular e plotar wss para k = 1 até k = 15
k.values <- 1:15
# Extrair o wss para 2-15 clusters
wss_values <- map_dbl(k.values, wss)
plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Soma dos quadrados totais dentro dos cluster")
```



Felizmente, este processo para calcular o “método Elbow” foi empacotado em uma única função (fviz_nbclust)

Hide

```
set.seed(123)
fviz_nbclust(df, kmeans, method = "wss")
```



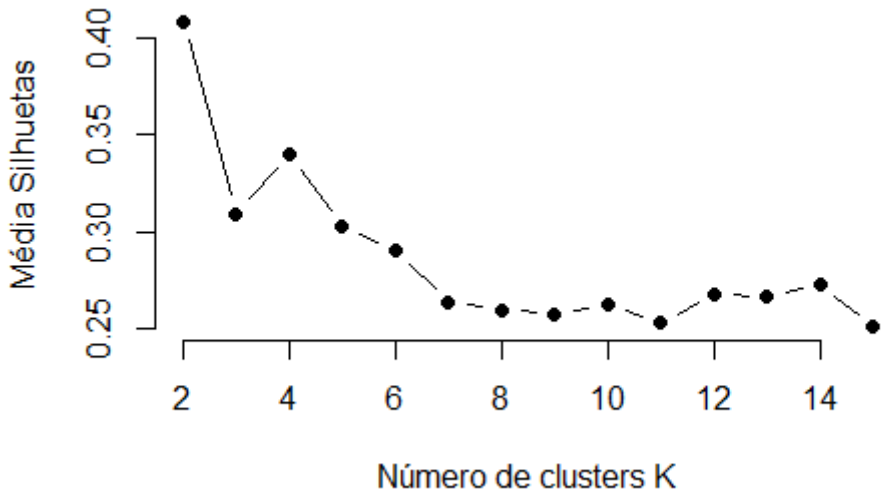
Average Silhoutte Method

A abordagem do método **average silhouette** mede a qualidade de um cluster. Ou seja, determina o quão bem cada objeto está dentro do seu cluster. Uma largura de silhueta média alta indica um bom agrupamento. O método calcula a silhueta média de observações para diferentes valores de k. O número ótimo de clusters k é aquele que maximiza a silhueta média em uma variedade de valores possíveis para k.

O código a seguir calcula essa abordagem para 1-15 clusters. Os resultados mostram que 2 clusters maximizam os valores médios de silhueta com 4 clusters entrando como segundo número ótimo de clusters.

Hide

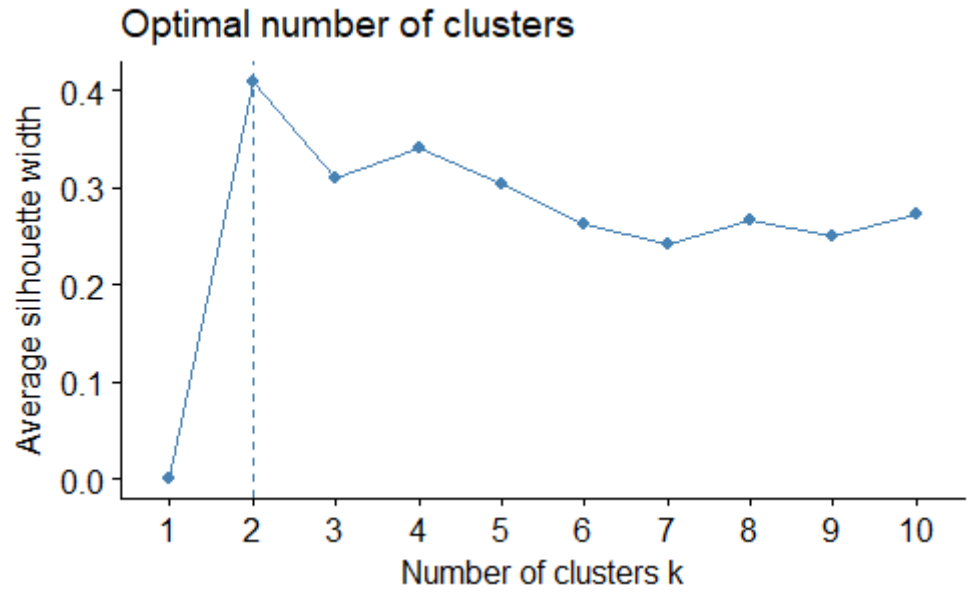
```
# Função para calcular a média da silhueta para K clusters
avg_sil <- function(k) {
  km.res <- kmeans(df, centers = k, nstart = 25)
  ss <- silhouette(km.res$cluster, dist(df))
  mean(ss[, 3])
}
# Clacular e plotar o wss para k = 2 a k = 15
k.values <- 2:15
# Extrair a silhueta média para 2-15 clusters
avg_sil_values <- map_dbl(k.values, avg_sil)
plot(k.values, avg_sil_values,
       type = "b", pch = 19, frame = FALSE,
       xlab = "Número de clusters K",
       ylab = "Média Silhuetas")
```



Semelhante ao método do “Elbow”“, este processo para calcular o”método da silhueta média" foi empacotado em uma única função (fviz_nbclust)

Hide

```
fviz_nbclust(df, kmeans, method = "silhouette")
```



Extraindo os Resultados

Com a maioria das abordagens sugeriram 4 grupos como o número de clusters ótimos, podemos realizar a análise final e extrair os resultados usando 4 clusters.

Hide

```
# Calcular o cluster k-means com k = 4
set.seed(123)
final <- kmeans(df, 4, nstart = 25)
print(final)
```

```
K-means clustering with 4 clusters of sizes 13, 16, 13, 8

Cluster means:
      Murder    Assault   UrbanPop        Rape
1 -0.9615407 -1.1066010 -0.9301069 -0.96676331
2 -0.4894375 -0.3826001  0.5758298 -0.26165379
3  0.6950701  1.0394414  0.7226370  1.27693964
4  1.4118898  0.8743346 -0.8145211  0.01927104

Clustering vector:
       Alabama         Alaska        Arizona       Arkansas     California       Colorado
 Connecticut       Delaware
             4              3              3              4              3              3
         2              2
       Florida        Georgia         Hawaii          Idaho       Illinois        Indiana
       Iowa         Kansas
             3              4              2              1              3              2
         1              2
      Kentucky      Louisiana          Maine       Maryland  Massachusetts       Michigan
  Minnesota    Mississippi
             1              4              1              3              2              3
         1              4
      Missouri        Montana       Nebraska         Nevada  New Hampshire     New Jersey
New Mexico       New York
             3              1              1              3              1              2
         3              3
North Carolina   North Dakota           Ohio       Oklahoma         Oregon   Pennsylvania   R
hode Island South Carolina
             4              1              2              2              2              2
         2              4
  South Dakota      Tennessee          Texas           Utah        Vermont       Virginia
Washington  West Virginia
             1              4              3              2              1              2
         2              1
     Wisconsin        Wyoming
             1              2

Within cluster sum of squares by cluster:
[1] 11.952463 16.212213 19.922437  8.316061
 (between_SS / total_SS =  71.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
"size"
[8] "iter"         "ifault"
```

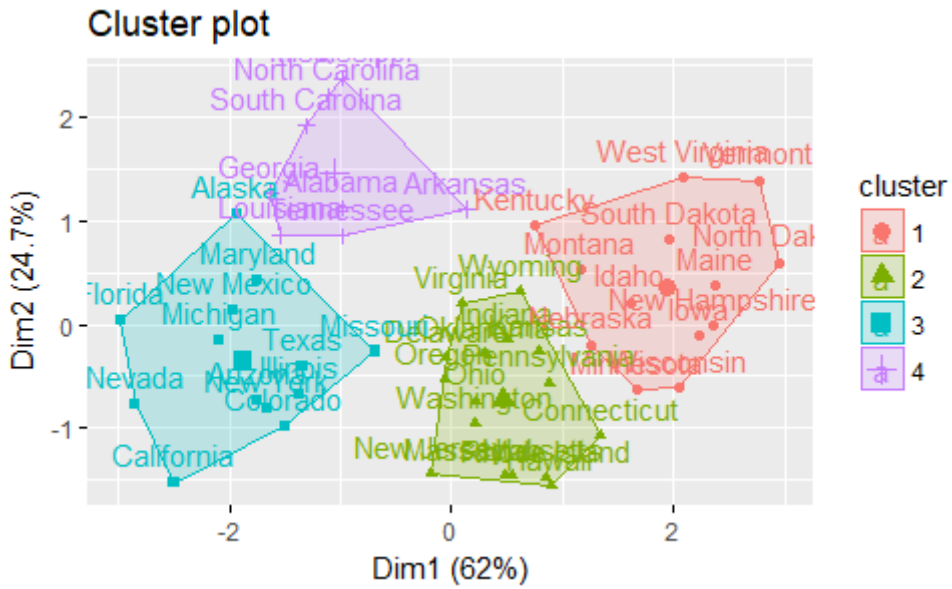
Hide

```
## K-means clustering com 4 clusters de tamanhos 13, 16, 13, 8
##
## Média do Cluster:
##       Murder    Assault   UrbanPop        Rape
## 1 -0.9615407 -1.1066010 -0.9301069 -0.96676331
## 2 -0.4894375 -0.3826001  0.5758298 -0.26165379
## 3  0.6950701  1.0394414  0.7226370  1.27693964
## 4  1.4118898  0.8743346 -0.8145211  0.01927104
##
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              4              3              3              4              3
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              3              2              2              3              4
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              2              1              3              2              1
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              2              1              4              1              3
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              2              3              1              4              3
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              1              1              3              1              2
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              3              3              4              1              2
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              2              2              2              2              4
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              1              4              3              2              1
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              2              2              1              1              2
##
## Within cluster sum of squares by cluster:
## [1] 11.952463 16.212213 19.922437  8.316061
##  (between_SS / total_SS =  71.2 %)
##
## Componentes disponíveis:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Podemos visualizar os resultados utilizando fviz_cluster

Hide

```
fviz_cluster(final, data = df)
```



E podemos extrair os clusters e adicionar aos nossos dados iniciais para fazer algumas estatísticas descritivas no nível do cluster:

Hide

```
USArrests %>%
  mutate(Cluster = final$cluster) %>%
  group_by(Cluster) %>%
  summarise_all("mean")
```

Cluster <int>	Murder <dbl>	Assault <dbl>	UrbanPop <dbl>	Rape <dbl>
1	3.60000	78.53846	52.07692	12.17692
2	5.65625	138.87500	73.87500	18.78125
3	10.81538	257.38462	76.00000	33.19231
4	13.93750	243.62500	53.75000	21.41250

4 rows

Comentários Adicionais

O agrupamento por método K-means é um algoritmo muito simples e rápido. Além disso, ele pode lidar eficientemente com conjuntos de dados muito grandes. No entanto, existem algumas fraquezas nessa abordagem.

Uma desvantagem potencial do agrupamento de K-means é que precisamos especificar inicialmente o número de clusters. O agrupamento hierárquico é uma abordagem alternativa que não exige que nos comprometamos com uma escolha particular de clusters. O agrupamento hierárquico tem uma vantagem adicional sobre o agrupamento de K-means, pois resulta em uma representação atrativa baseada nas árvores das observações, chamado dendrograma.

Uma desvantagem adicional de K-means é que é muito sensível a outliers e diferentes resultados podem ocorrer se alterarmos a ordem de nossos dados. A abordagem de Particionamento de grupos em torno da Medoids (PAM) é menos sensível aos outliers e fornece uma alternativa robusta aos k-means para lidar com essas situações.