Análise de Agrupamentos

Joás Ramos de Santana

Introdução

Podemos utilizar a Análise de Agrupamentos em várias áreas como marketing (segmentação de produtos mais importantes para realização de propaganda direcionada), vendas (segmentação de lojas, vendedores que precisam de treinamento, segmentação de clientes), fraude (verificar grupos de transações duvidosas) etc.

O objetivo da análise de cluster é agrupar as observações relativamente semelhante entre si em grupos de tal forma que esses grupos sejam consideravelmente diferentes um do outro. De forma mais técnica, queremos alocar as observações em uma quantidade relativamente pequena de agrupamentos **homogêneos internamente e heterogêneos entre si** representando o conjunto de observações a partir de determinadas variáveis (FÁVERO, 2017).

Começaremos utilizando a Análise de **Agrupamento Hierárquico** e depois ampliaremos o tema utilizando a metodologia **K-means** e **DBSCAN**.

O nosso objetivo é fazer uma análise de agrupamentos para os municípios do estado de Pernambuco com base em variáveis econômicas extraídas do IBGE-cidades.

Bibliotecas

Chamando as bibliotecas utilizadas.

```
library(tidyverse) # biblioteca para manipulacao de dados
library(cluster) # algoritmo de cluster
library(dendextend) # comparação de dendogramas
library(factoextra) # algoritmo de cluster e visualização
library(fpc) # algoritmo de cluster e visualização
library(gridExtra) # para a função grid arrange
```

Definindo o diretório de trabalho.

```
# definindo o diretório base com os dados
setwd('/home/joas/Data-science-projects/Análise de Cluster em Municípios de Pernambuco')
```

Método Hierárquico

Leitura dos Dados

As variáveis na base de dados são:

1. `municipio`: nome do município
2. `area` - Área Territorial - km² [2021]
3. `populacao` - População estimada - pessoas [2021]
4. `escolarizacao` - Escolarização 6 a 14 anos - % [2010]
5. `idhm` - IDHM – IDH Municipal [2010]
6. `receitas` - Receitas realizadas - R$ (×1000) [2017]
7. `pib_percapita` - PIB per capita - R$ [2019]

8. `salario_medio_mensal` - Salário médio mensal dos trabalhadores formais em relação ao salário mínino

Modificações realizadas na base:

- alteração dos nomes das colunas
- troca das vírgulas por ponto como separador decimal

Como o intuito não é mostrar o processo de ETL nos dados essas manipulações foram feitas anteriormente à importação da base aqui no *script*.

```
# LEITURA DOS DADOS
# separado por vígula (sep = ","); com cabeçalho (header = T)
municipios <- read.table("dados/municipios-pe.csv", sep = ",", header = T)
head(municipios) # visualizando as primeiras linhas
```

```
##               municipio    area população escolarizacao idhm  receitas
## 1          Abreu e Lima  126384    100698          97.2  679 154773.90
## 2 Afogados da Ingazeira  377696     37546          97.3  657  84524.29
## 3               Afrânio 1490594     19981          98.7  588  48071.21
## 4             Agrestina  200369     25240          96.4  592  61063.17
## 5            Água Preta  533332     37386          94.0  553  71553.72
## 6           Águas Belas  885988     43923          96.1  526  84026.50
##   pib_percapita salario_medio_mensal
## 1      17756.99                 1.9
## 2      12464.16                 1.7
## 3       8663.84                 1.8
## 4      11341.74                 1.4
## 5       6974.69                 1.6
## 6       8337.93                 1.7
```

Precisamos deixar apenas as variáveis métricas, por isso transformaremos os nomes dos municípios em índice das linhas.

```
# transformando a primeira coluna em índice de linhas
rownames(municipios) <- municipios[,1]
municipios <- municipios[,-1] # excluindo a primeira coluna
```

Antes de prosseguir vamos separar para uma análise prévia apenas os municípios que pertencem à Região Metropolitana de Recife (RMR). Essa separação serve para mostrar a separação hierárquica mais claramente.

```
# separando os municípios da RMR
RMR <- municipios[c('Recife', 'Abreu e Lima', 'Araçoiaba',
                    'Cabo de Santo Agostinho', 'Camaragibe','Goiana',
                    'Igarassu','Ilha de Itamaracá', 'Ipojuca', 'Itapissuma',
                    'Jaboatão dos Guararapes', 'Moreno', 'Olinda','Paulista',
                    'São Lourenço da Mata'),]

RMR <- data.frame(RMR) # salvando em um dataframe
```

Análise dos dados - Dendogramas

Outra exigência para fazer uma análise de cluster é a padronização das variáveis quando estão em unidades de escala diferentes, que é o nosso caso.

Não interpretamos os dados padronizados, depois de formados os clusters, voltamos para a tabela original e faremos a interpretação do que os clusters querem dizer.

```
#Padronizar variaveis (neste caso é necessário pois tem variável em escalas diferentes)
RMR.padronizado <- scale(RMR)
```

Calculando a matriz distância pelo método da **distância euclidiana**, que é utilizada por padrão. Mas podemos utilizar a distancia máxima, *manhattan*, *canberra*, binária ou *minkowski*.

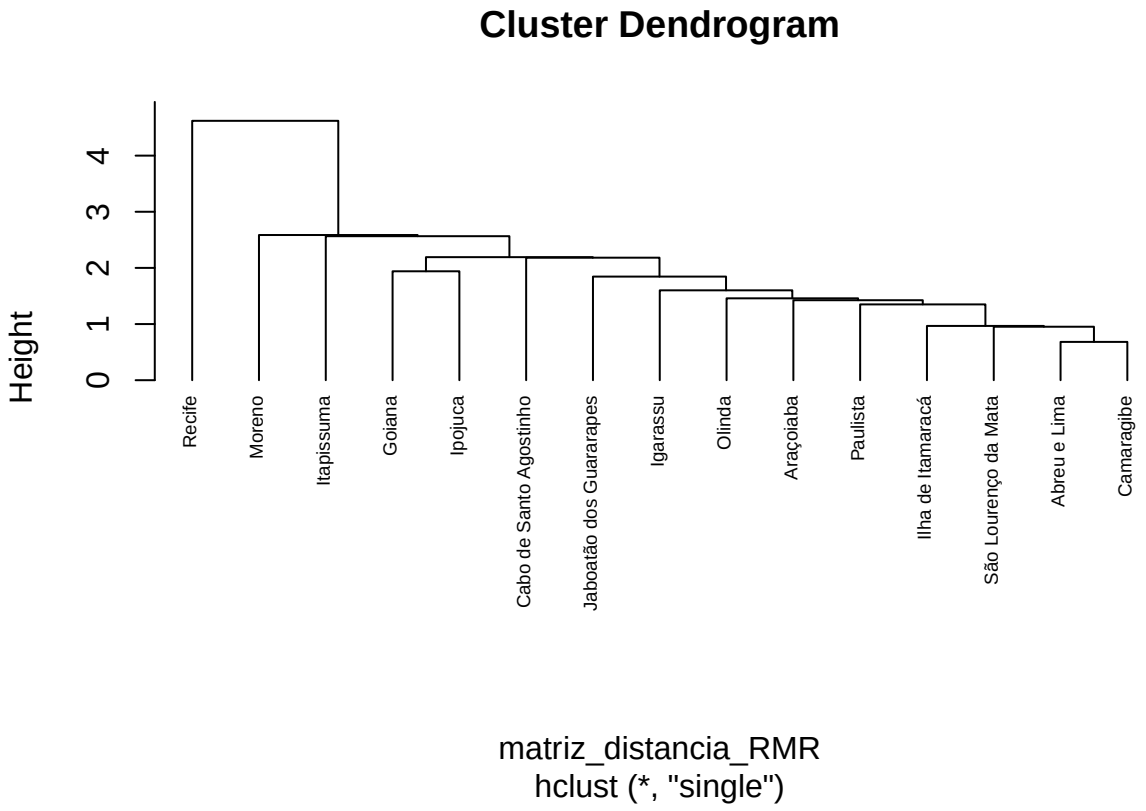
```
#CALCULANDO MATRIZ DE DISTANCIAS
matriz_distancia_RMR <- dist(RMR.padronizado, method = "euclidean")
```

Agora vamos calcular os clusters por quatro métodos: “average”, “single”, “complete” e “ward.D”.

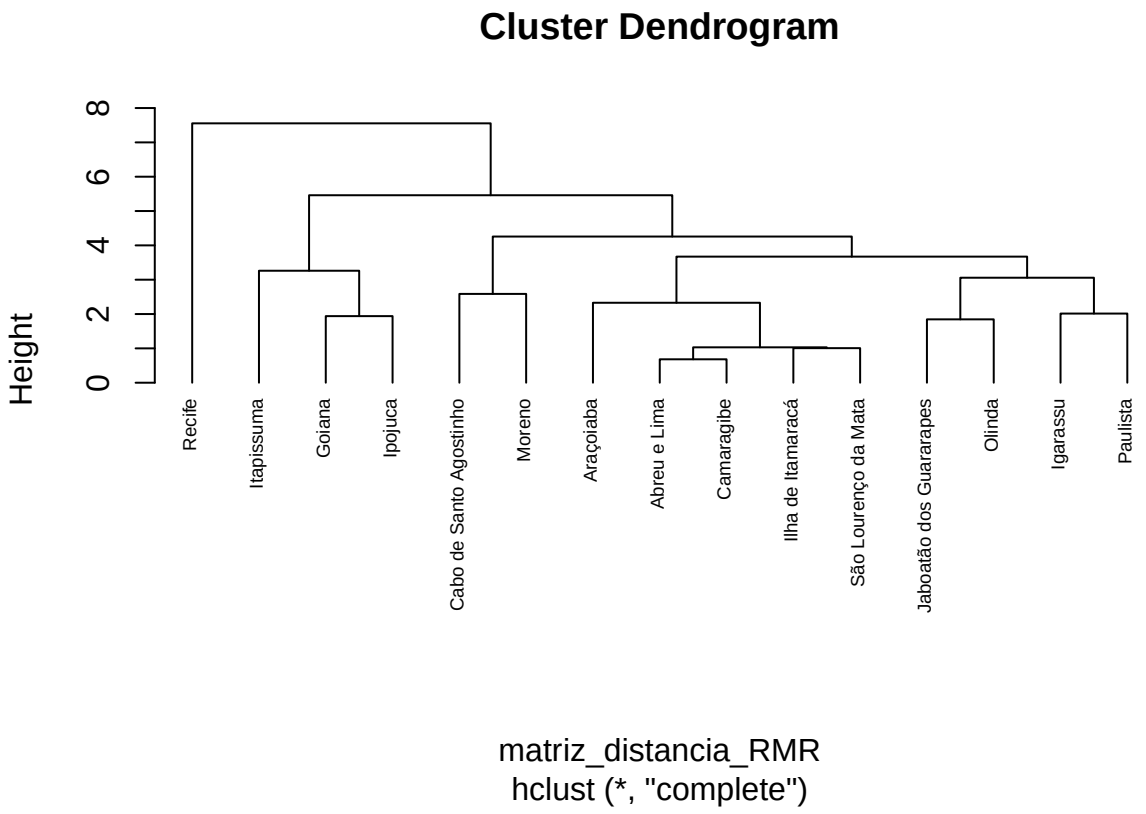
```
# DEFININDO O CLUSTER A PARTIR DO METODO ESCOLHIDO
# ch = cluster hierárquico
ch1 <- hclust(matriz_distancia_RMR, method = "single" )
ch2 <- hclust(matriz_distancia_RMR, method = "complete" )
ch3 <- hclust(matriz_distancia_RMR, method = "average" )
ch4 <- hclust(matriz_distancia_RMR, method = "ward.D" )
```

Plotando os dendogramas:

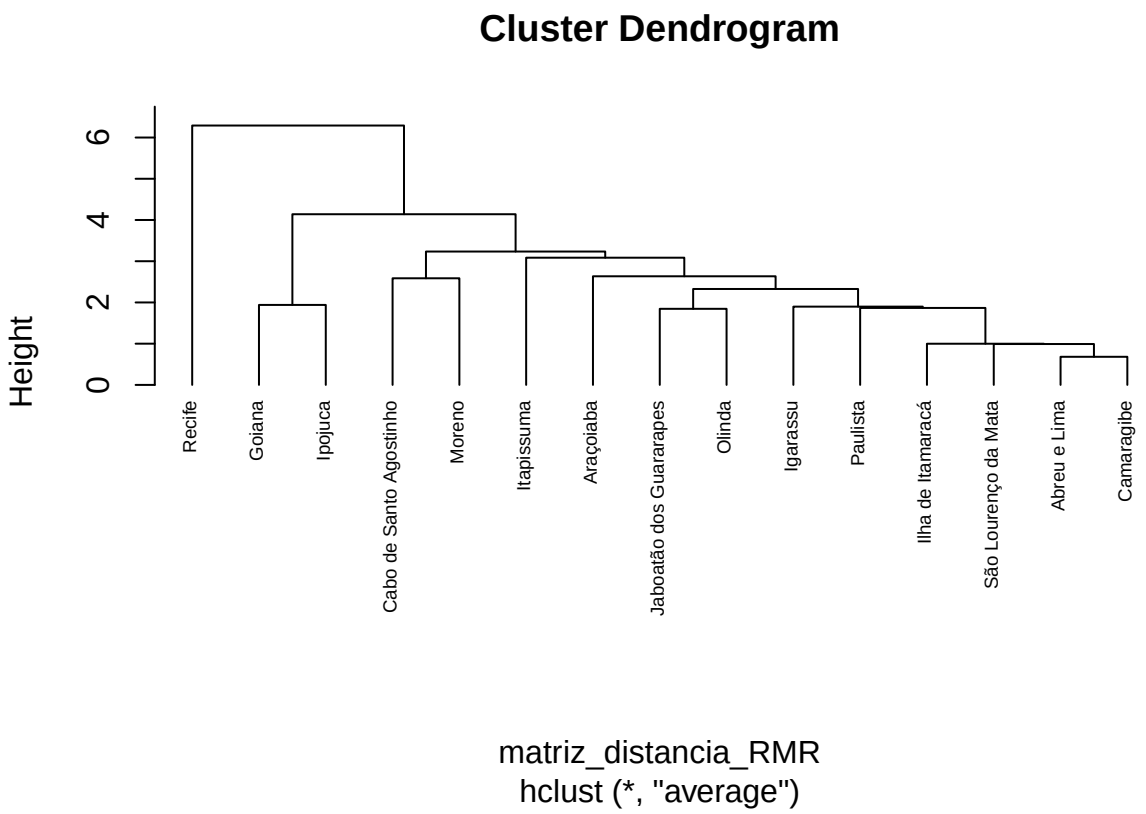
```
#DESENHANDO O DENDOGRAMA
plot(ch1, cex = 0.6, hang = -1)
```



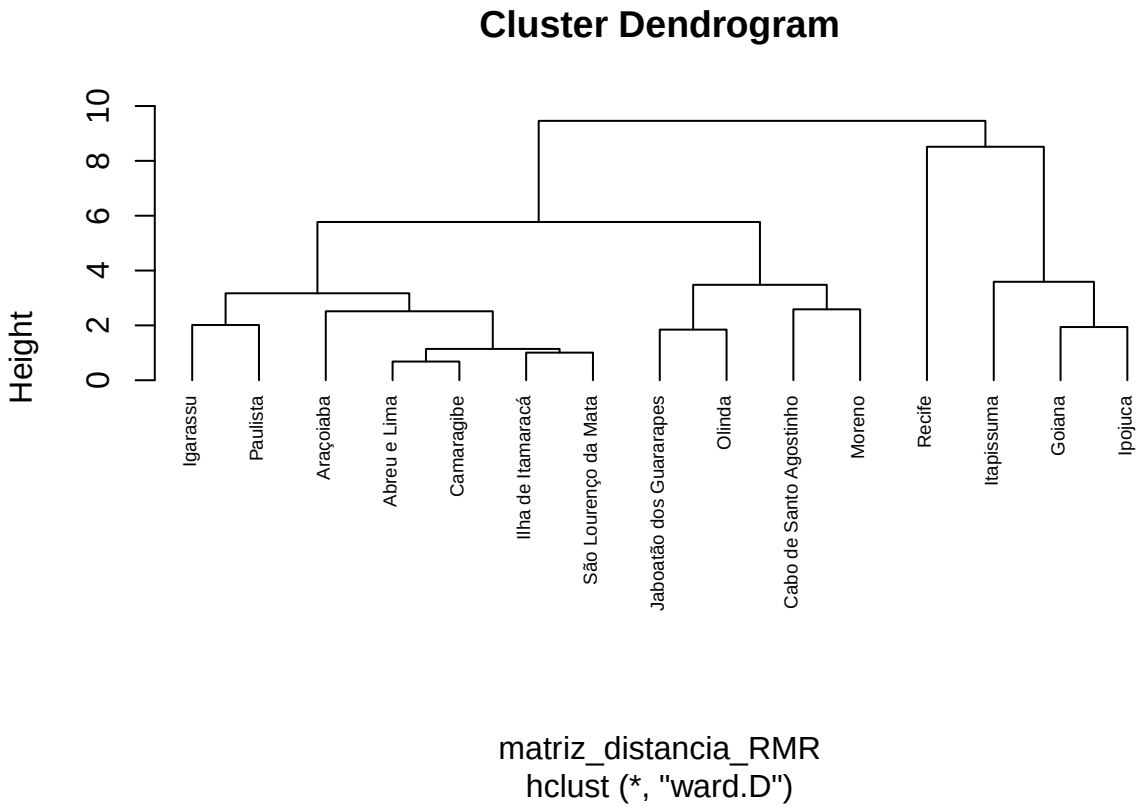

```
plot(ch2, cex = 0.6, hang = -1)
```




```
plot(ch3, cex = 0.6, hang = -1)
```

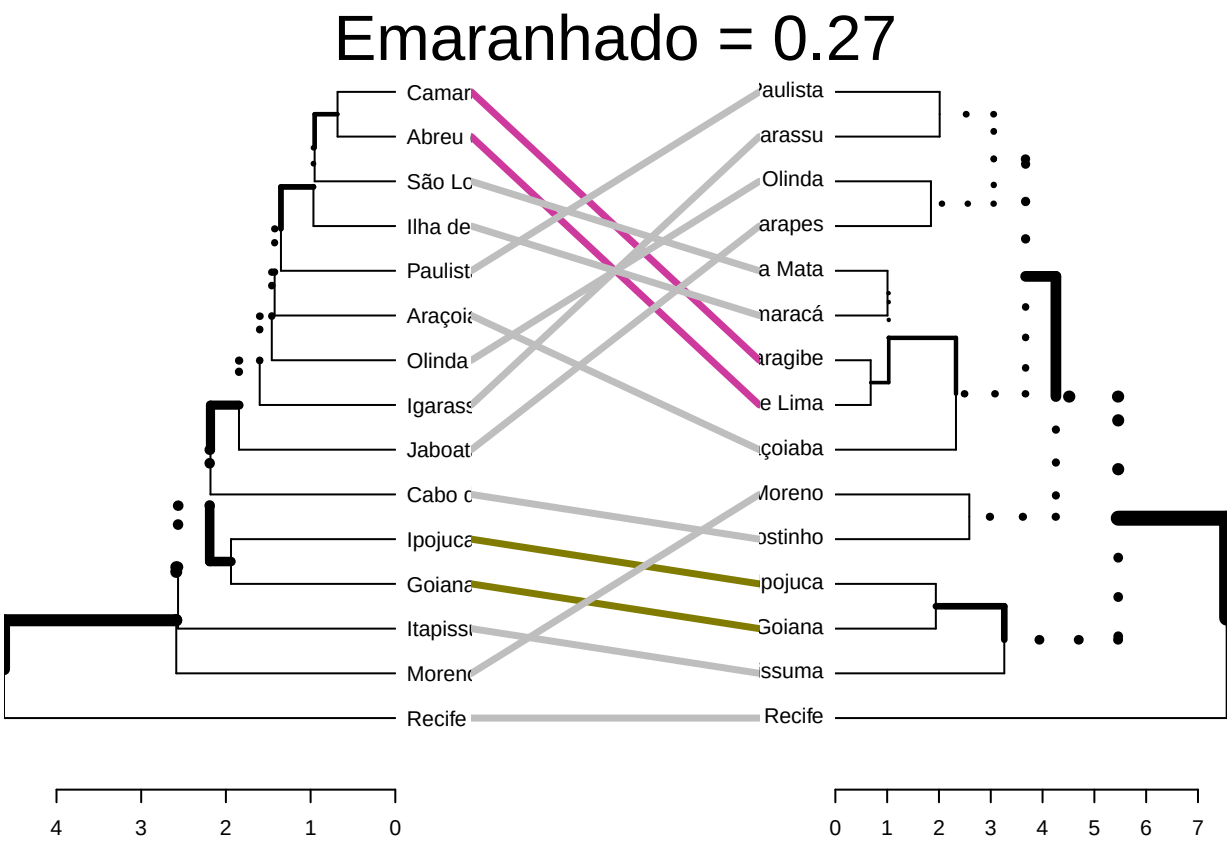


```
plot(ch4, cex = 0.6, hang = -1)
```



Como são gerados dendogramas diferentes, podemos fazer uma comparação de dendogramas. Vamos comparar os métodos *single* com o método *complete*.

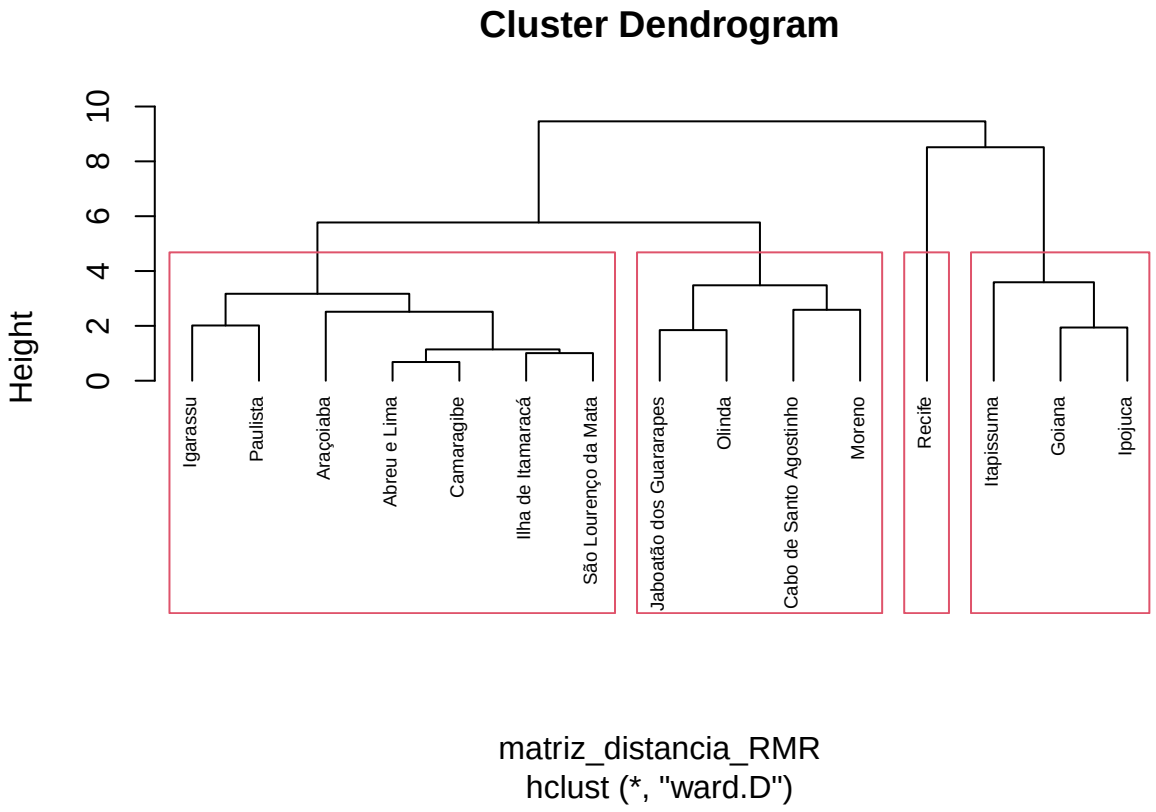
```
# COMPARANDO DENDOGRAMAS
# comparando o método single com complete
dend_ch1 <- as.dendrogram(ch1)
dend_ch2 <- as.dendrogram(ch2)
dendogramas <- dendlist(dend_ch1, dend_ch2) # guardando em uma lista
#EMARANHADO
tanglegram(dend_ch1, dend_ch2, main = paste("Emaranhado =",
                                            round(entanglement(dendogramas),2)))
```



A comparação mostra onde cada elemento mudou de grupo. Apesasr de poucos elementos a visualização já se mostra difícil.

Um método direto para separarmos a quantidade de grupos, à primeira vista, é traçar uma reta horizontal quando os grupos demoram a se unirem e ver em quantos grupos cortaram. Para exemplo, vamos separar em 4 grupos o dendograma para o método *Wald* (`ch4`).

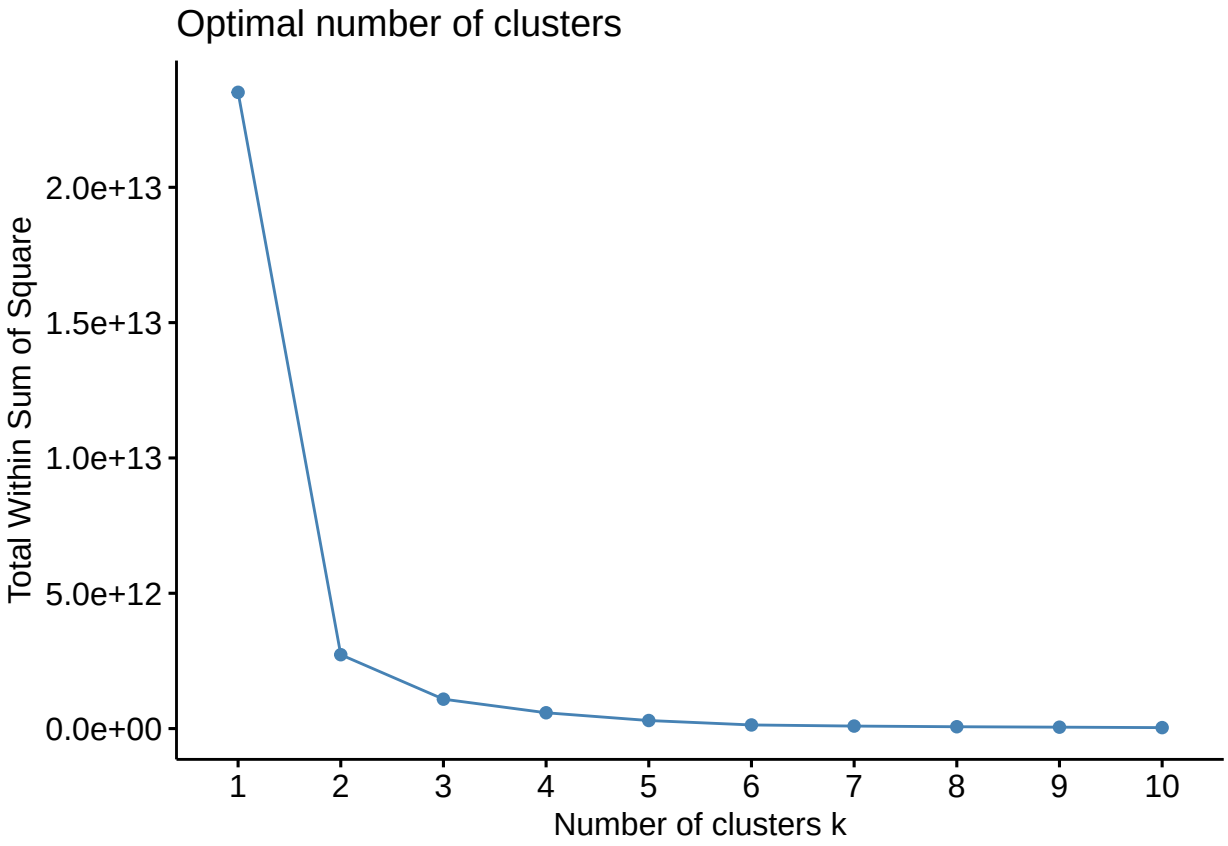
```
# separando em 4 clusters
plot(ch4, cex = 0.6, hang = -1)
rect.hclust(ch4, k = 4)
```



Observamos Recife isoladamente, dois grupos menores com 2 e 3 municípios neles e um grupo com a maior parte dos municípios.

Para uma escolha melhor da quantidade de grupos podemos utilizar o **método Elbow**.

```
# DEFININDO O NÚMERO DE CLUSTERS COM MÉTODO ELBOW
# passamos os dados padronizados;
# o tipo de agrupamento (hcut - método hierárquico);
# e o método de estimação (wss = Weighted Sum of Squares)
fviz_nbclust(RMR, FUN = hcut, method = "wss")
```



Pelo gráfico podemos escolher 3 ou 4 grupos como número de clusters. Vou definir 4.

Com esses 4 grupos definidos vamos salvar os dados dos clusters, incluir essa nova informação dos dados originais para podermos fazer uma análise descritiva para cada grupo. Vamos utilizar o método de Wald para as análises (`ch4`).

```
# vamos salvar os grupos cortados
grupo_municipios <- cutree(ch4, k = 4) # cortando ch4 em 4 grupos
table(grupo_municipios)
```

```
## grupo_municipios
## 1 2 3 4
## 1 7 4 3
```

Podemos ver a quantidade de municípios em cada grupo. Vamos juntar esses dados com a tabela original RMR.

```
#transformando em data frame a saida do cluster
grupo_municipios2 <- data.frame(grupo_municipios)
#juntando com a base original
base_municipios_fim <- cbind(RMR, grupo_municipios)
```

Assim, teremos um dataframe com a base original e na última coluna está a qual grupo cada município pertence.

Com essa tabela em mãos podemos fazer uma análise descritiva acerca dos grupos. Vamos gerar uma tabela com a média de cada variável para cada grupo.

```
media_grupo_RMR <- base_municipios_fim %>%
  group_by(grupo_municipios) %>%
  summarise(qtde_municipios = n(), # número de indivíduos de cada grupo
```

```
            populacao_media = mean(população),
            escolarizacao_media = mean(escolarizacao),
            idhm_medio = mean(idhm),
            receita_media = mean(receitas),
            pib_percapita_medio = mean(pib_percapita),
            salario_medio_mensal = mean(salario_medio_mensal))
media_grupo_RMR
```

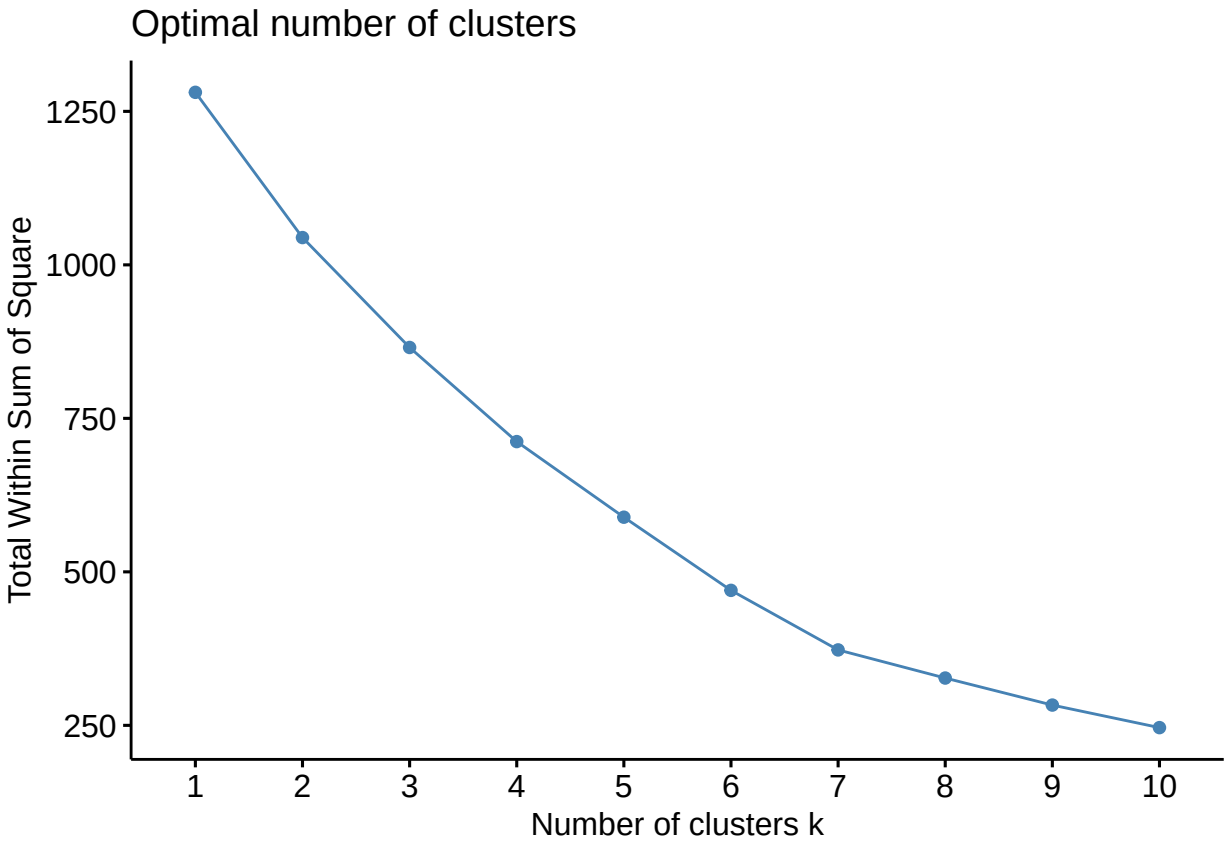
```
## # A tibble: 4 x 8
##   grupo_municipios qtde_munici~1 popul~2 escol~3 idhm_~4 recei~5 pib_p~6 salar~7
##              <int>         <int>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1                1             1  1.66e6    97.1    772   4.86e6  33232.    3.2
## 2                2             7  1.26e5    97.4    667.  2.07e5  13681.    1.79
## 3                3             4  3.45e5    96.5    698.  7.21e5  23366.    1.95
## 4                4             3  6.89e4    97.2    634.  3.52e5 109931.    2.73
## # ... with abbreviated variable names 1: qtde_municipios, 2: populacao_media,
## #   3: escolarizacao_media, 4: idhm_medio, 5: receita_media,
## #   6: pib_percapita_medio, 7: salario_medio_mensal
```

Temos, então, uma tabela com todos os dados médios para os clusters feitos. Podemos ver que Recife ficou isolado, por ser a capital os dados são, em média, maiores do que dos outros municípios. Mas podemos ver que o grupo com 7 municípios são os que apresentam menores valores de desenvolvimento, então eles formam um grupo por esse motivo.

Análise para todos os municípios de PE

Agora faremos a análise hierárquica para todos os municípios de PE. Veremos uma das limitações para o uso de dendogramas.

```
# padronizando as variáveis
municipios.padronizado <- scale(municipios)

# calculando a matriz de distância utilizando o método euclidiano
matriz_distancia_municipios <- dist(municipios.padronizado,method = "euclidean")

# analisando o número de grupos pelo método Elbow
fviz_nbclust(municipios.padronizado, FUN = hcut, method = "wss") # 7 grupos
```

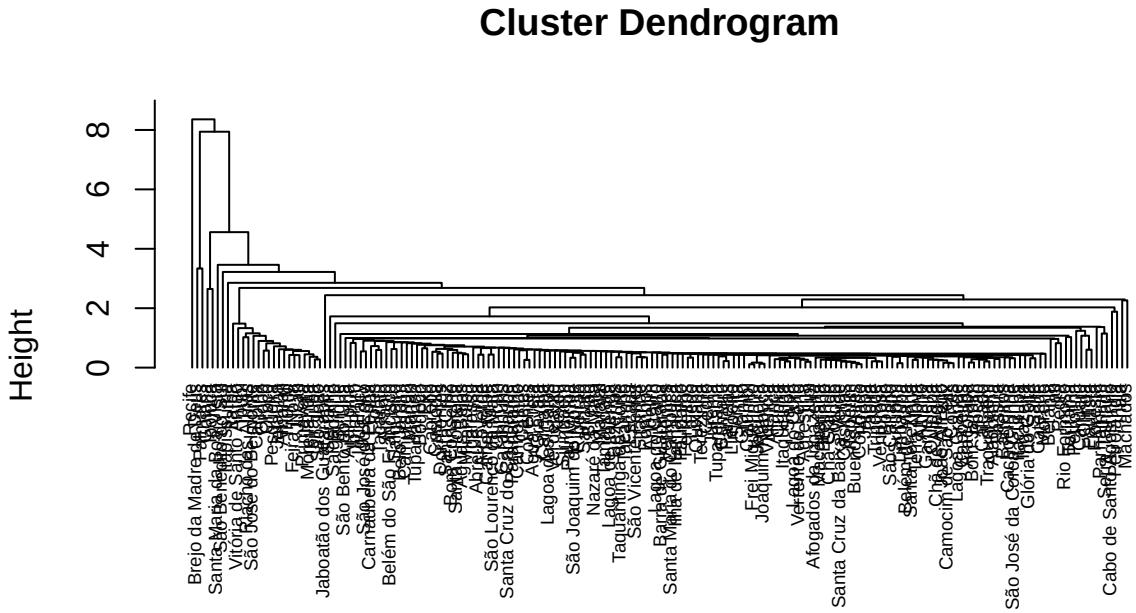


Para este caso vamos calcular o cluster utilizando apenas dois métodos: *single* e *ward*.

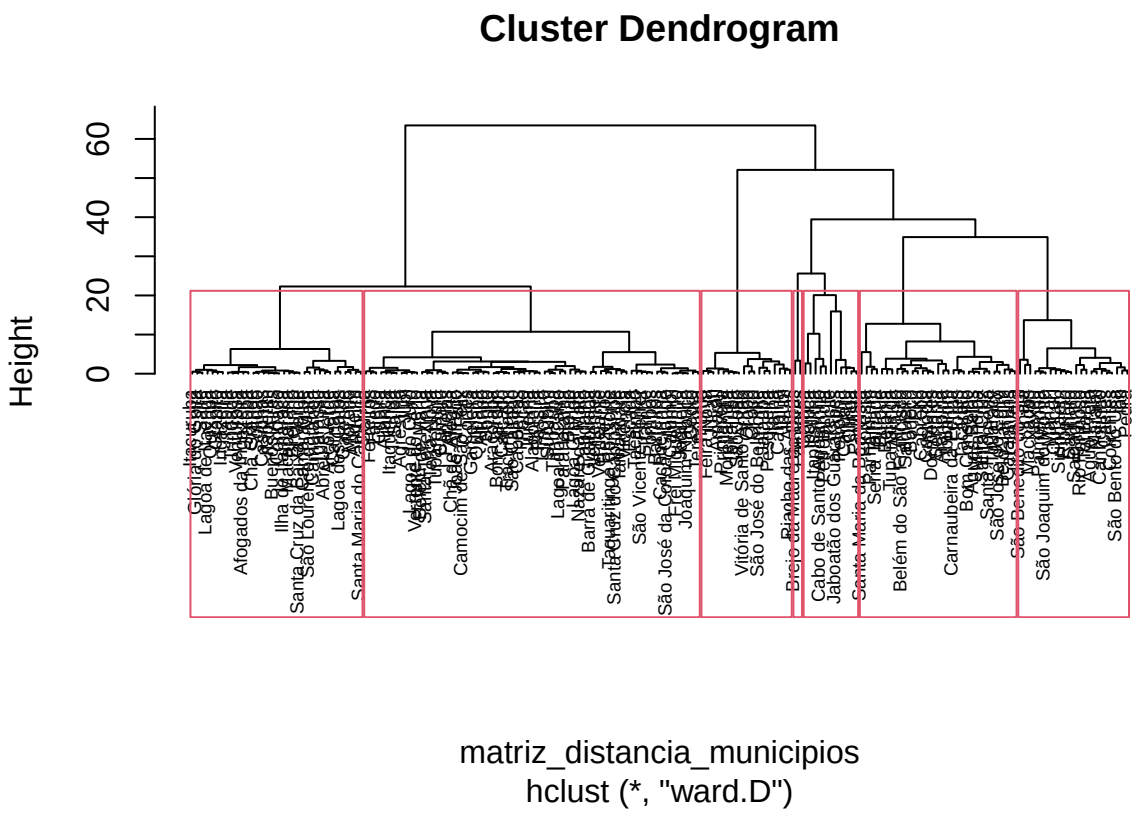
```
# definindo o cluster apenas para os métodos single e ward.
hc1 <- hclust(matriz_distancia_municipios, method = "single" )
hc4 <- hclust(matriz_distancia_municipios, method = "ward.D")
```

Desenhando os dendogramas.

```
# dendogramas
plot(hc1, cex = 0.6, hang = -1)
```




```
plot(hc4, cex = 0.6, hang = -1)
rect.hclust(hc4, k = 7) # cortando o último dendograma em 7 grupos
```



```
# número de municípios por grupo
grupo_municipios_total <- cutree(hc4, k = 7) # cortando hc14 em 4 grupos
table(grupo_municipios_total)
```

```
## grupo_municipios_total
##  1  2  3  4  5  6  7
## 34 31 66 22 18  2 11
```

Podemos ver que, principalmente, na abordagem do método *single* a análise do dendograma ficou muito confusa. Isto é uma desvantagem desse método. Mais a frente veremos que o uso do método K-means auxilia na visualização dos grupos para um número tão grande de variáveis.

Método Não-hierárquico

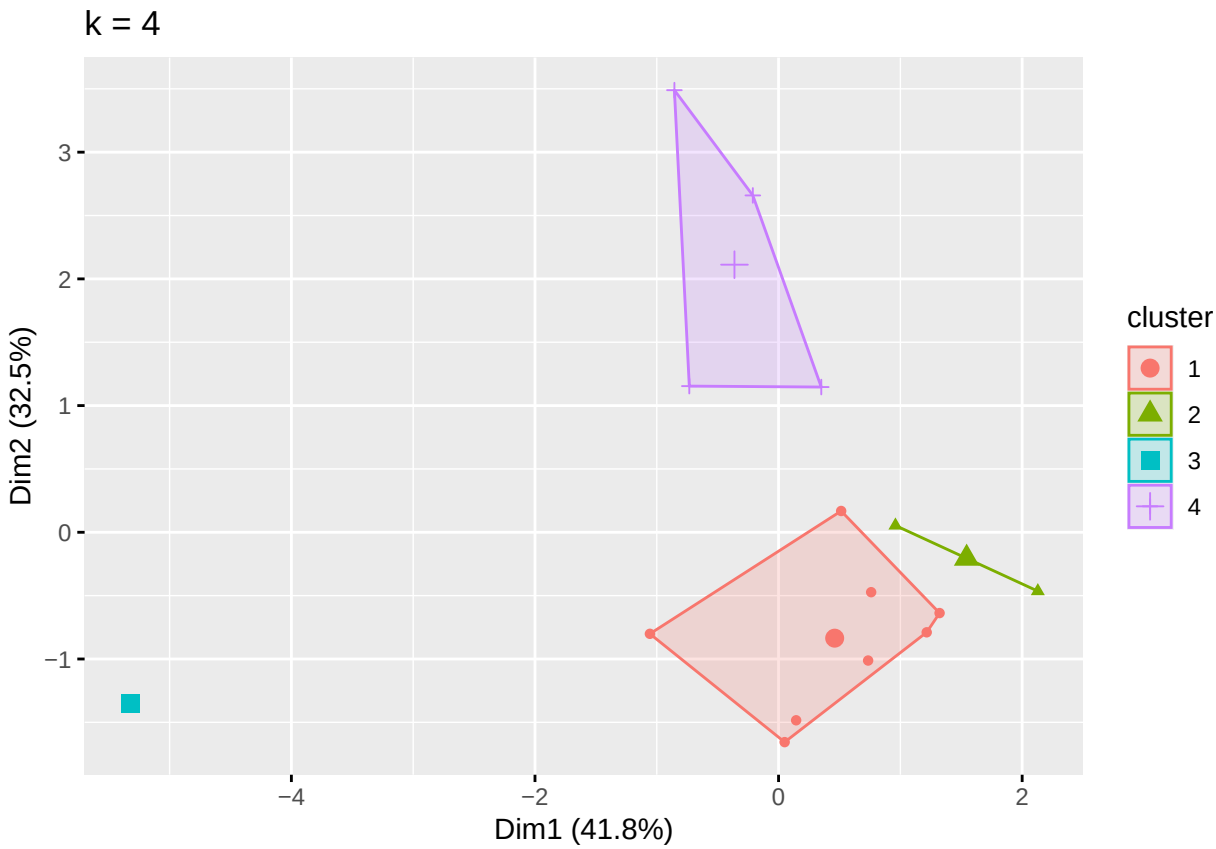
K-MEANS

Agora vamos ver essa mesma análise utilizando métodos não-hierárquicos. Começaremos pelo **k-means**. Ainda precisamos utilizar os dados padronizados, passaremos 4 centróides.

```
# Calcular o Cluster - utilizamos os dados padronizados
RMR.cluster <- kmeans(RMR.padronizado, centers = 4)
```

Agora vamos gerar o gráfico com os grupos.

```
# gerando os gráficos
fviz_cluster(RMR.cluster, geom = "point", data = RMR.padronizado) + ggtitle("k = 4")
```

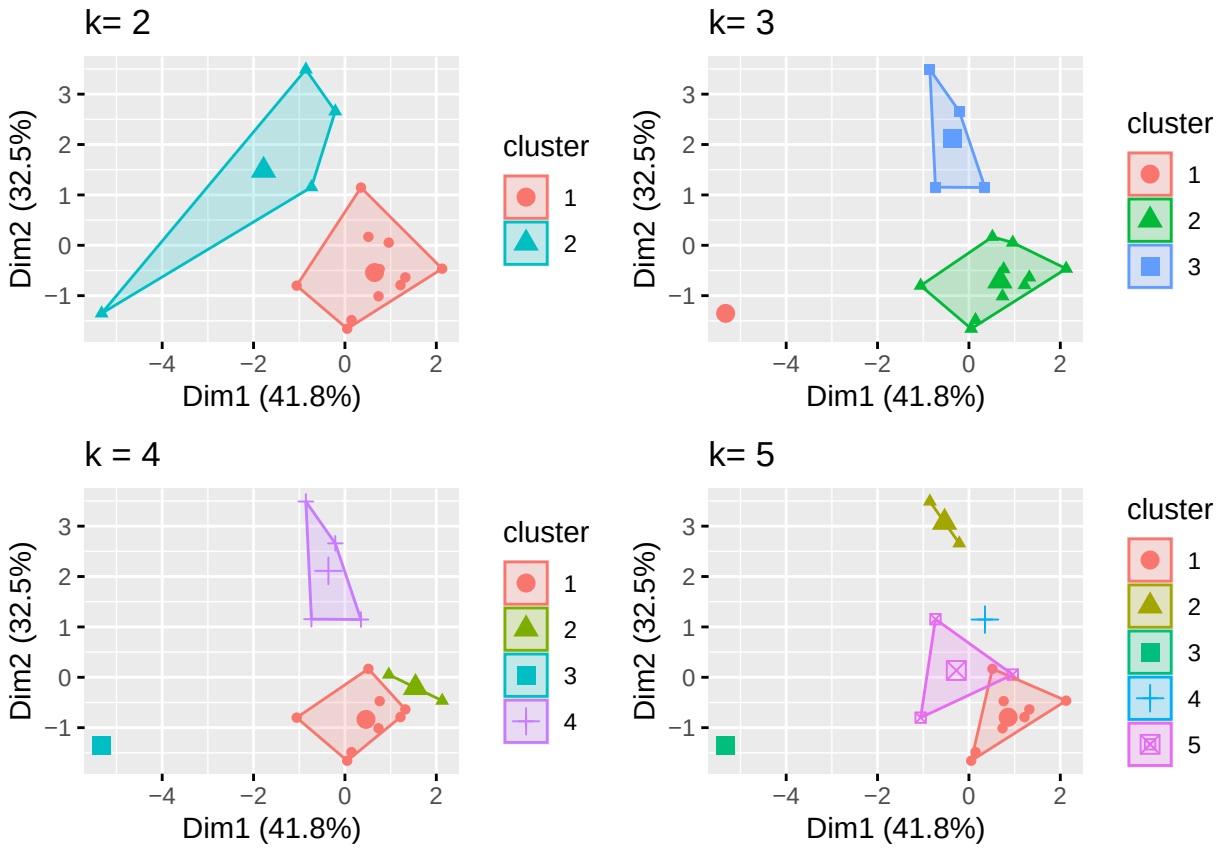


O gráfico retorna duas dimensões, `Dim1` e `Dim2`. `Dim1` está pegando 41.8% da variabilidade dos dados, enquanto `Dim2` está pegando 32.5% da variabilidade dos dados. Se a área de algum cluster está grande e com

muitas observações é porque há uma grande variabilidade de dados indicando que pode criar outros grupos. Podemos testar com mais centróides caso seja necessário.

Para um exemplo vamos rodar com outras combinações de grupos:

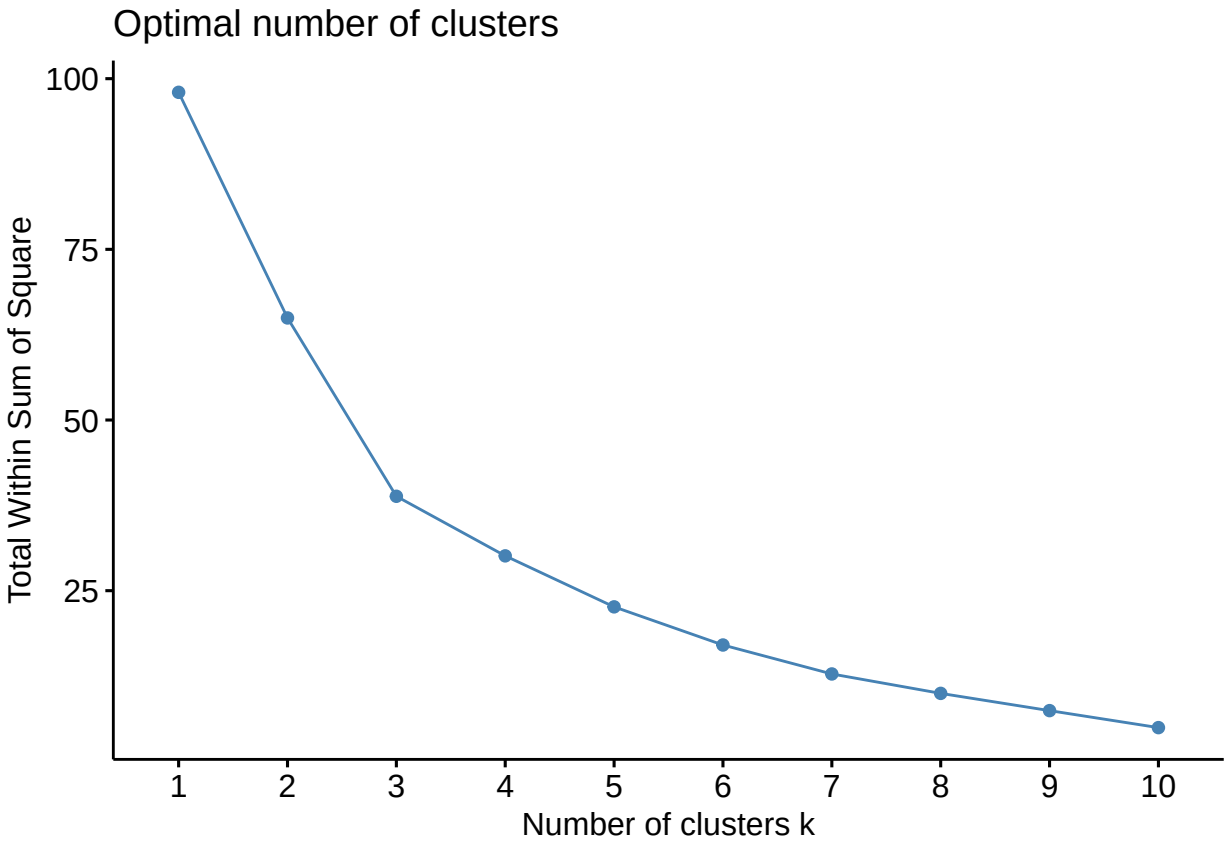
```
# criando mais clusters - testando com mais grupos
RMR.cluster2 <- kmeans(RMR.padronizado, centers = 2)
RMR.cluster3<- kmeans(RMR.padronizado, centers = 3)
RMR.cluster5 <- kmeans(RMR.padronizado, centers = 5)

# criando e salvando os gráficos
G1<-fviz_cluster(RMR.cluster2,geom="point",data=RMR.padronizado)+ggtitle("k= 2")
G2<-fviz_cluster(RMR.cluster3,geom="point",data=RMR.padronizado)+ggtitle("k= 3")
G3<-fviz_cluster(RMR.cluster,geom="point",data=RMR.padronizado)+ggtitle("k = 4")
G4<-fviz_cluster(RMR.cluster5,geom="point",data=RMR.padronizado)+ggtitle("k= 5")

# Imprimindo os gráficos em uma mesma figura
grid.arrange(G1, G2, G3, G4, nrow = 2)
```



Desta maneira, podemos ter uma noção da formação dos clusters.

Para verificarmos a quantidade de grupos pelo método Elbow

```
fviz_nbclust(RMR.padronizado, kmeans, method = "wss")
```

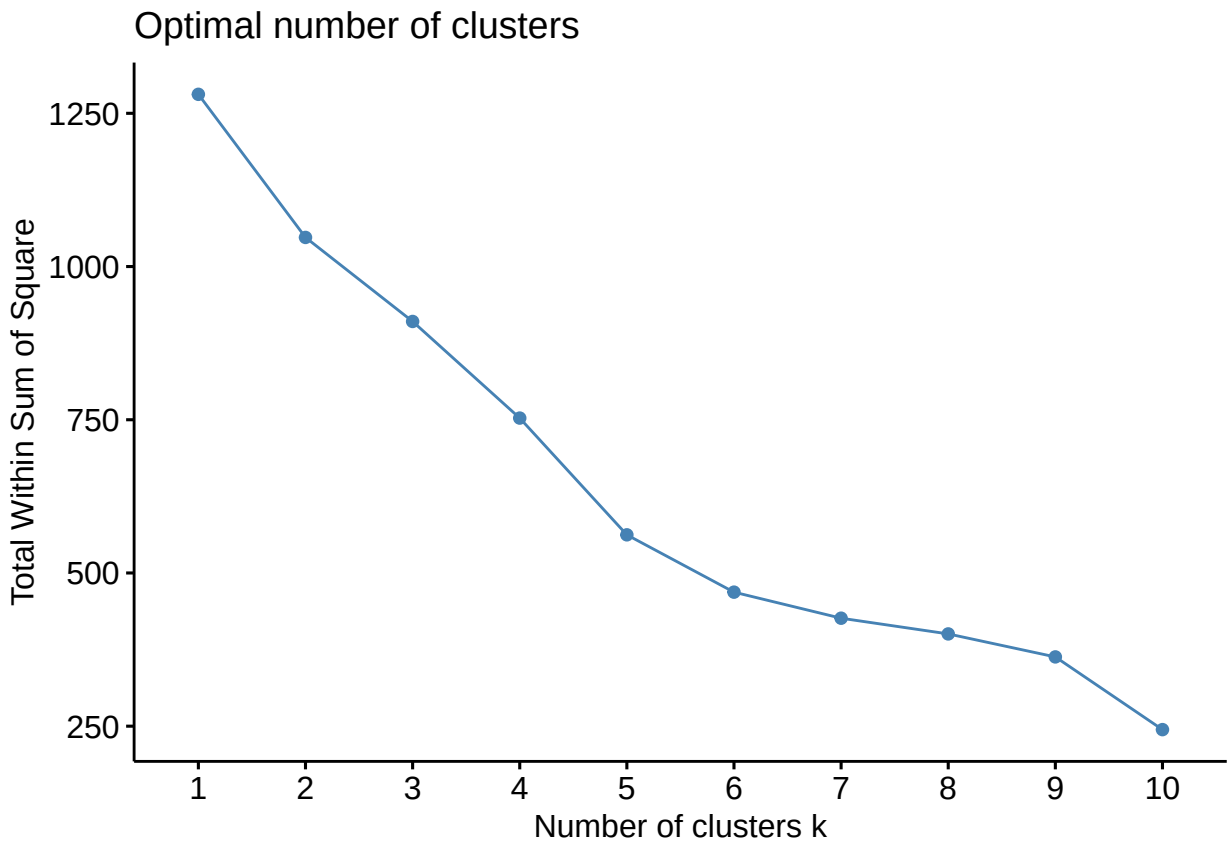


A escolha fica entre 3 ou 4 grupos como melhores separações.

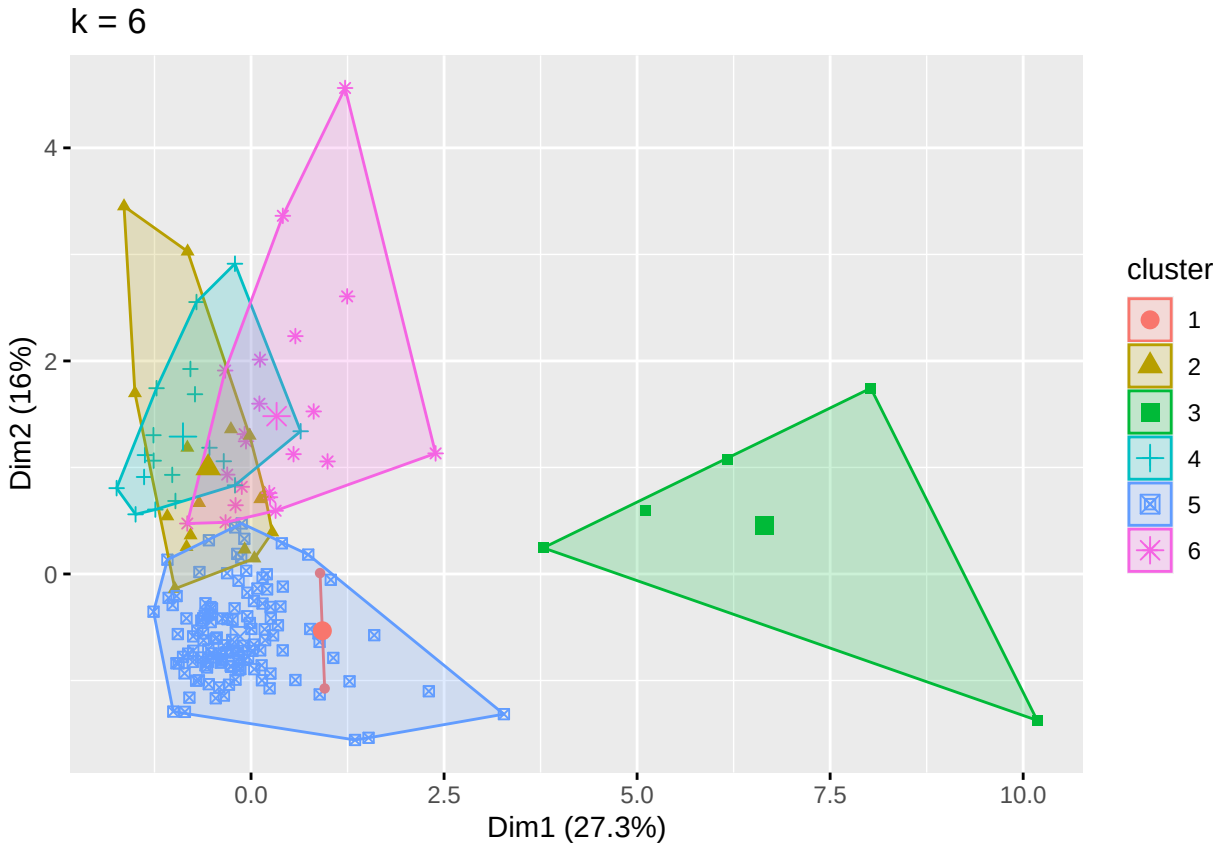
Método K-means para todos os municípios de PE

Faremos a mesma análise incluindo todos os municípios de PE para efeito de comparação.

```
# verificando a quantidade de grupos pelo método Elbow
fviz_nbclust(municipios.padronizado, FUN = kmeans, method = "wss") # 6 grupos
```



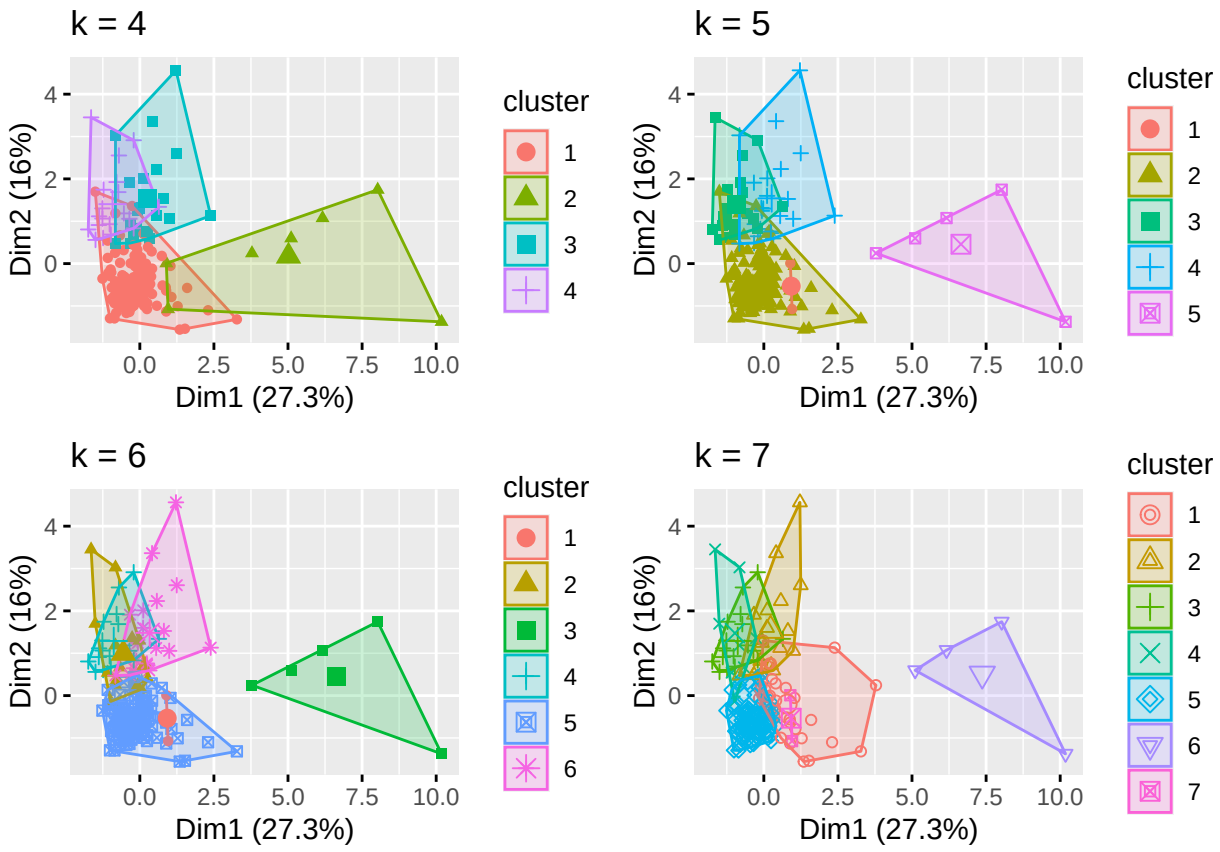

```
# Calcular o Cluster - utilizamos os dados padronizados
municipios.cluster <- kmeans(municipios.padronizado, centers = 6)

# Visualizando os clusters
fviz_cluster(municipios.cluster, geom = "point",
             data = municipios.padronizado) + ggtitle("k = 6")
```




```
# observamos a sobreposição de vários grupos

# Dim1 está pegando 27.3% da variabilidade dos dados
# Dim2 está pegando 16% da variabilidade dos dados
```

Podemo ver que há grupos se sobrepondo. Gerando para mais centróides.

```
# criando mais clusters - testando com mais grupos
municipios.cluster4<- kmeans(municipios.padronizado, centers = 4)
municipios.cluster5 <- kmeans(municipios.padronizado, centers = 5)
municipios.cluster7 <- kmeans(municipios.padronizado, centers = 7)

# criando e salvando os gráficos
G1 <- fviz_cluster(municipios.cluster4, geom = "point",
                   data = municipios.padronizado) + ggtitle("k = 4")
G2 <- fviz_cluster(municipios.cluster5, geom = "point",
                   data = municipios.padronizado) + ggtitle("k = 5")
G3 <- fviz_cluster(municipios.cluster, geom = "point",
                   data = municipios.padronizado) + ggtitle("k = 6")
G4 <- fviz_cluster(municipios.cluster7, geom = "point",
                   data = municipios.padronizado) + ggtitle("k = 7")

# Imprimindo os gráficos em uma mesma figura
grid.arrange(G1, G2, G3, G4, nrow = 2)
```



Método DBSCAN

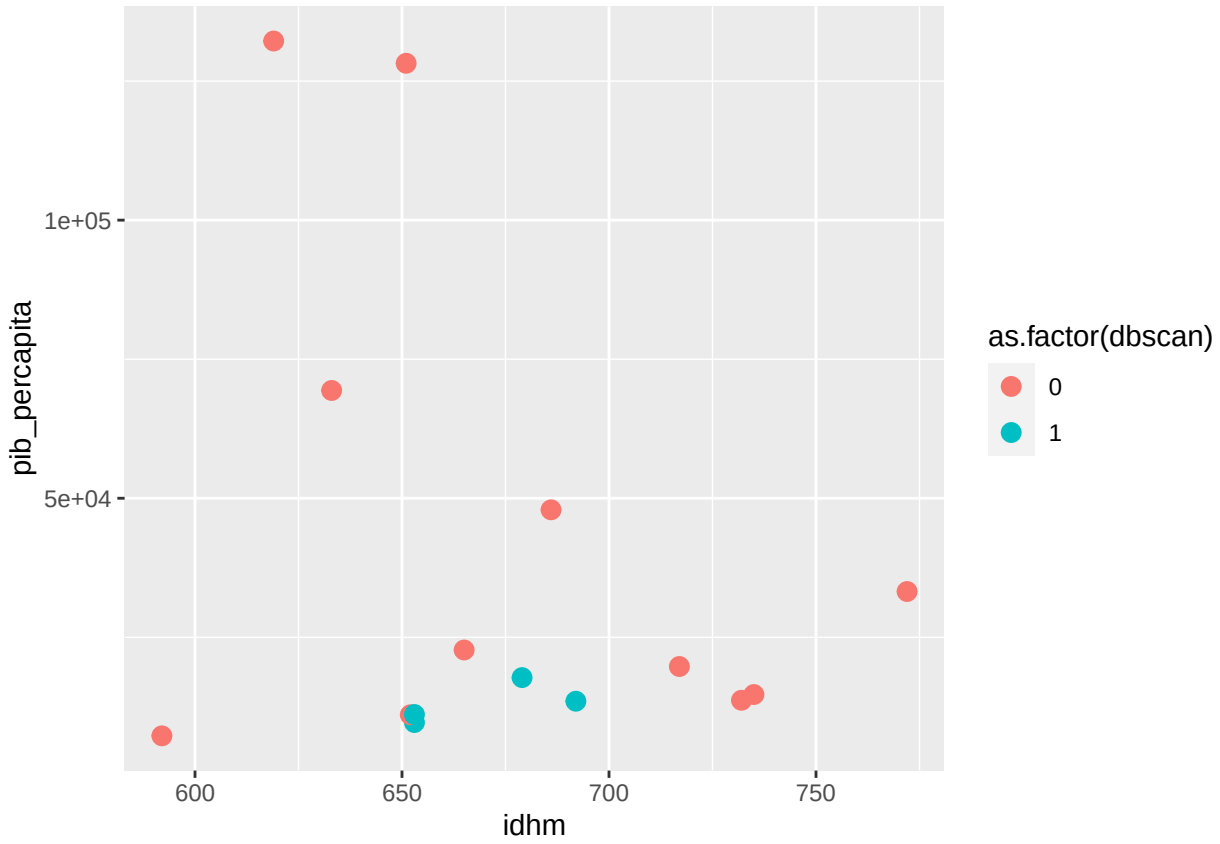
O método DBSCAN é uma clusterização espacial baseada em densidade. Talvez para nossos dados não seja o ideal, mas vamos fazer uma pequena análise para podermos ter uma comparação com as anteriores.

```
# Calcular o Cluster - o parâmetro eps é a largura da borda
# nesse caso colocamos 0.56 de maneira subjetiva, o ideal é ir ajustando
# MinPts é o mínimo de pontos dentro do círculo,
# colocamos 3 para ser o mínimo para formar um grupo
# esse método é muito sensível a esses parâmetros
dbscan <- fpc::dbscan(RMR.padronizado,eps = 1, MinPts = 3)
```

```
# para sabermos onde cada observação está em cada grupo
base_municipios_fim$dbscan <- dbscan$cluster
```

Visualizando graficamente apenas para duas variáveis.

```
#visualizando em cores os clusters
base_municipios_fim %>% ggplot() +
  geom_point(aes(x = idhm,
                 y = pib_percapita,
                 color = as.factor(dbscan)),
             size = 3)
```


1e+05
5e+04
pib_percapita
600
650
700
750
idhm
as.factor(dbscan)
0
1