MAD-CB

Machine Learning – 3

It's tough to make predictions, especially about the future. – Yogi Berra

Machine Learning Não Superivisionado

- Machine learning sem referência aos valores de uma variável dependente
- Focar só nas variáveis independentes para ver como os casos se agrupam em clusters

Técnica Apta para “Big Data”

- Análise de clusters e outras técnicas não supervisionados ajuda a reduzir as complexidades de grandes conjuntos de dados (“big data”)
- O conjunto que usaremos hoje é a análise de amostras vindo de um estudo de câncer
- Tem 22215 genes estudados através 189 amostras
- Nosso interesse: quais tipos de tecido cada amostra representa
  - O processo de cluster vai reproduzir o agrupamento dos tecidos na vida real?
- Os dados de expressão dos genes ocupam acima de 33MB de memoria

Dados e Muito da Apresentação Graças a . . .

- Livro **Data Analysis for the Life Sciences**
- Curso Harvard EDx PH525-4 (High Dimensional Analysis)
- Ambos de Rafael Irizzary e Michael Love

Carregar os Dados

```
library(tissuesGeneExpression)
data(tissuesGeneExpression)
paste("Dimensões dos Dados:", dim(e)[1], "x", dim(e)[2]) ## e contem os dados de ex
```

```
## [1] "Dimensões dos Dados: 22215 x 189"
```

```
table(tissue) ## tissue[i] conta qual tecido se representa por e[,i]
```

```
## tissue
##  cerebellum       colon endometrium hippocampus      kidney       liver
##          38          34          15          31          39          26
##    placenta
##           6
```

Tirar Amostras de Placenta

- Número baixo pode distorcer os resultados dos demais
  - Sem aumentar informação útil
- Nova tabela dos tecidos

```
## tissue
##  cerebellum       colon endometrium hippocampus      kidney       liver
##          38          34          15          31          39          26
```

Distância

Conceito de Distância

- Para processo de agrupamento o a formação de clusters . . .
- Necessário medida para julgar relação entre itens sendo agrupados
- Distância de um ao outro
- Quantas dimensões depende do número de covariáveis
- “Distância” neste caso é uma abstração matemática
  - Não uma distância física, limitada pelas duas dimensões do plano cartesiano

Distância em Dados de Genes e Tecidos

- Cada amostra se define por 22.215 genes
  - $(E_{1,i}, \ldots, E_{22215,i})$
- Cada gene se define por 189 amostras
  - $(E_{g,1}, \ldots, E_{g,189})$

Distância Euclideana

- Podemos usar os calculos euclideanos para determinar distância entre os pontos
  - Apesar de altas dimensões
- Simplificado: distância entre 2 pontos numa mapa

$$\text{dist}(A_{x,y}, B_{x,y}) = \sqrt{(A_x - B_x)^2 - (A_y - B_y)^2}$$

Distância Euclideana Estendida

- Pode medir todas as diferenças entre os pontos
- Usando todas as dimensões
- Ex.: as 22.215 dimensões de genes
- Notação de matrizes (álgebra linear) facilita o entendimento

Distância em Algebra Linear

- Letra "T" refere a uma matriz que é *transposta*
  - Todos os linhas tornam colunas; toda coluna, uma linha

Matriz A e a Transposta

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Distância em Qualquer Dimensão

$$\text{dist}(i, j) = (\mathbf{Y}_i - \mathbf{Y}_j)^\top (\mathbf{Y}_i - \mathbf{Y}_j)$$

- Com $\mathbf{Y}_i$ e $\mathbf{Y}_j$ sendo colunas i e j

Distância entre 3 Amostras – 2 dos Rins e 1 do Cólon

- Amostras 1 e 2 – rins
- Amostra 87 – cólon

```
a1 <- e[,1]
a2 <- e[,2]
a87 <- e[,87]
dist12 <- sqrt(sum((a1-a2)^2))
paste("distância 1 e 2", round(dist12,3))
```

```
## [1] "distância 1 e 2 85.855"
```

```
dist187 <- sqrt(sum((a1-a87)^2))
paste("distância 1 e 87", round(dist187,3))
```

```
## [1] "distância 1 e 87 122.892"
```

```
dist287 <- sqrt(sum((a2-a87)^2))
paste("distância 2 e 87", round(dist287,3))
```

```
## [1] "distância 2 e 87 115.477"
```

Conclusão sobre Distância

- Amostras 1 e 2 são mais perto porque são os 2 de rim
- Amostra 87 mais longe porque vem de outro tipo de tecido

Função `dist`

- Calcula a distância entre linhas de uma matriz
  - *Não* colunas
- Para medir a distância entre amostras (colunas)
  - Deve transpôr a matriz com a função `t()`
- Resultado é da classe `dist`
- Facilita o trabalho com ele se forçar ao formato de uma matriz
  - com `as.matrix()`

```
damost <- as.matrix(dist(t(e)))
```

Pergunta - O Que Aconteceria se Não Fez a Transposição?

- Calcular a diferença entre todos os genes

Pergunta - O Que Aconteceria se Não Fez a Transposição?

- Calcular a diferença entre todos os genes
- Matriz de 22215 linhas x 22215 colunas

Pergunta - O Que Aconteceria se Não Fez a Transposição?

- Calcular a diferença entre todos os genes
- Matriz de 22215 linhas x 22215 colunas
- Ouch! Estourar memoria

Confirmar com `dist()` as distâncias

```
damost[1,2]
```

```
## [1] 85.8546
```

```
damost[1,87]
```

```
## [1] 122.8919
```

```
damost[2,87]
```

```
## [1] 115.4773
```

Análise de Clusters

Pontos Gerais

- Técnica mais experimental que outras
  - Precisa tentar soluções e parâmetros
  - Guardar resultados
  - Achar padrões consistentes
  - Usar resultados para iniciar novas pesquisas

Algoritmos de Clusters

- *Clusters Hierárquicos* (“hierarchical clustering”)
- *Clusters por k-médias* (“k-means”)

Clusters Hierárquicos

- Algoritmo começa criando um grupo para cada amostra
- Em cada iteração (loop)
  - Juntar os dois clusters mais próximos
- Até existe um único cluster com todos as amostras
- Ao início, não sabemos quantos clusters nosso modelo vai produzir

Clusters por k-médias

- Você decide quantos clusters (k) que precisa ou deseja
- Algoritmo inicialmente aloca todas as amostras a um dos k grupos aleatoriamente
- Em cada iteração (loop)
  - Algoritmo determina o ponto central de todos os clusters – o *centroide*
  - Realoca as amostras para grupo com o centroide mais próximo
- Até não pode fazer mais realocações
- Objetivo é para minimizar a variação dentro de cada cluster
  - "within cluster sum of squares"
  - Estabelece máxima compacidade

Algoritmos de Clusters em R

Clusters Hierarquicos – `hclust()`

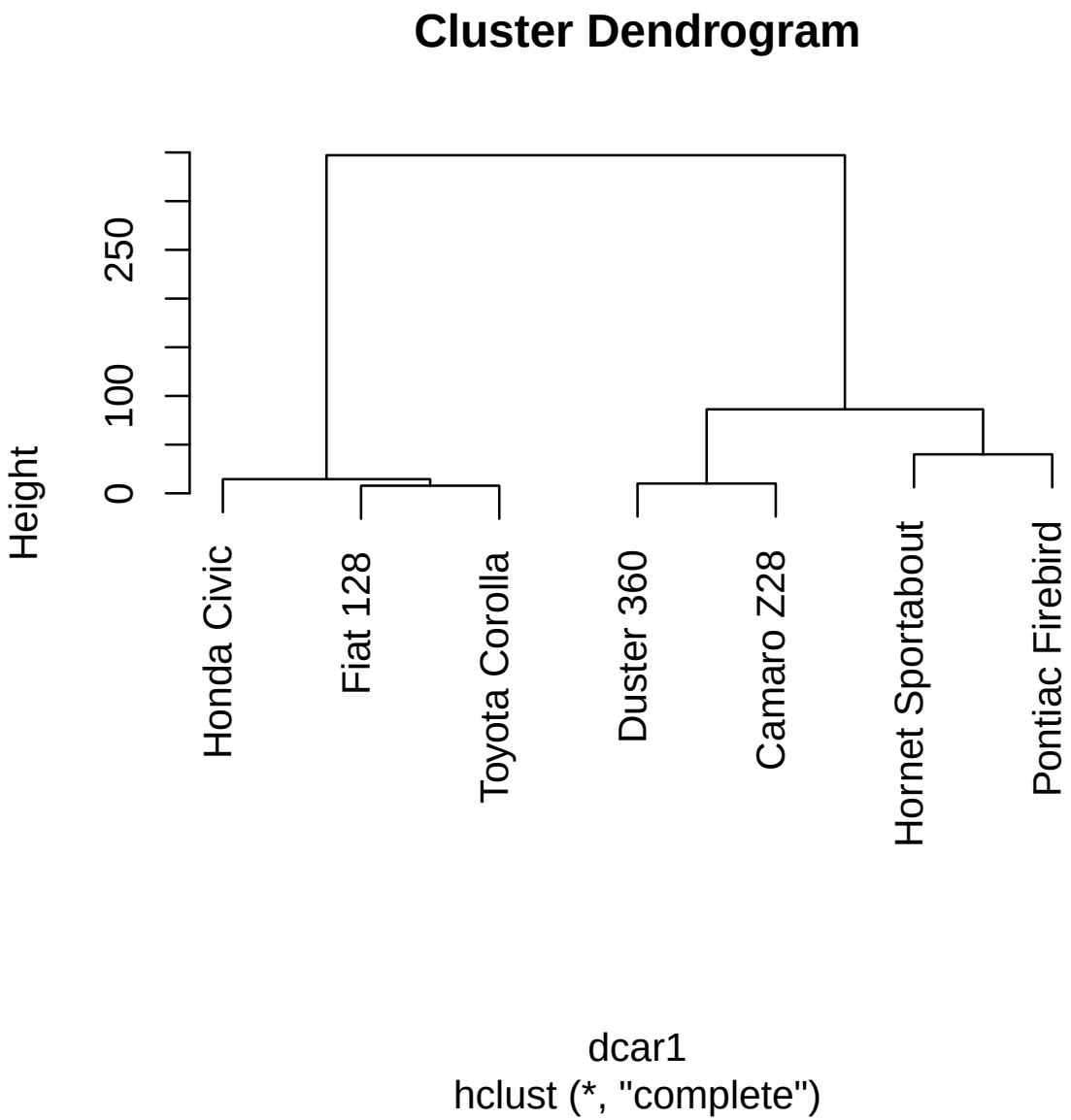
- Precisa usar uma matriz de distâncias e um método de aglomeração
- Vários métodos possíveis
  - Trabalhamos com método padrão, `complete`, que maximiza distância entre as amostras de clusters diferentes
- Resultado é uma *dendrograma*
  - Gráfico que representa os clusters que o modelo desenvolveu
- Pode acessar este gráfico através a função `plot(x)`, onde x é o nome do objeto de cluster
  - Que `hclust` criou

Exemplo Mais Simples

- Parte de conjunto `mtcars` em R
  - Carros de 1974
  - 7 dos 32 originais
- Lista dos carros:
  - Hornet Sportabout, Duster 360, Fiat 128, Honda Civic, Toyota Corolla, Camaro Z28, Pontiac Firebird

```r
car1 <- mtcars[c(5,7,18,19,20,24,25),]
dcar1 <- dist(as.matrix(car1))
hccars <- hclust(dcar1)
hccars
```

```
##
## Call:
## hclust(d = dcar1)
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 7
```


Cluster Dendrogram
Height
0
100
250
Honda Civic
Fiat 128
Toyota Corolla
Duster 360
Camaro Z28
Hornet Sportabout
Pontiac Firebird
dcar1
hclust (*, "complete")

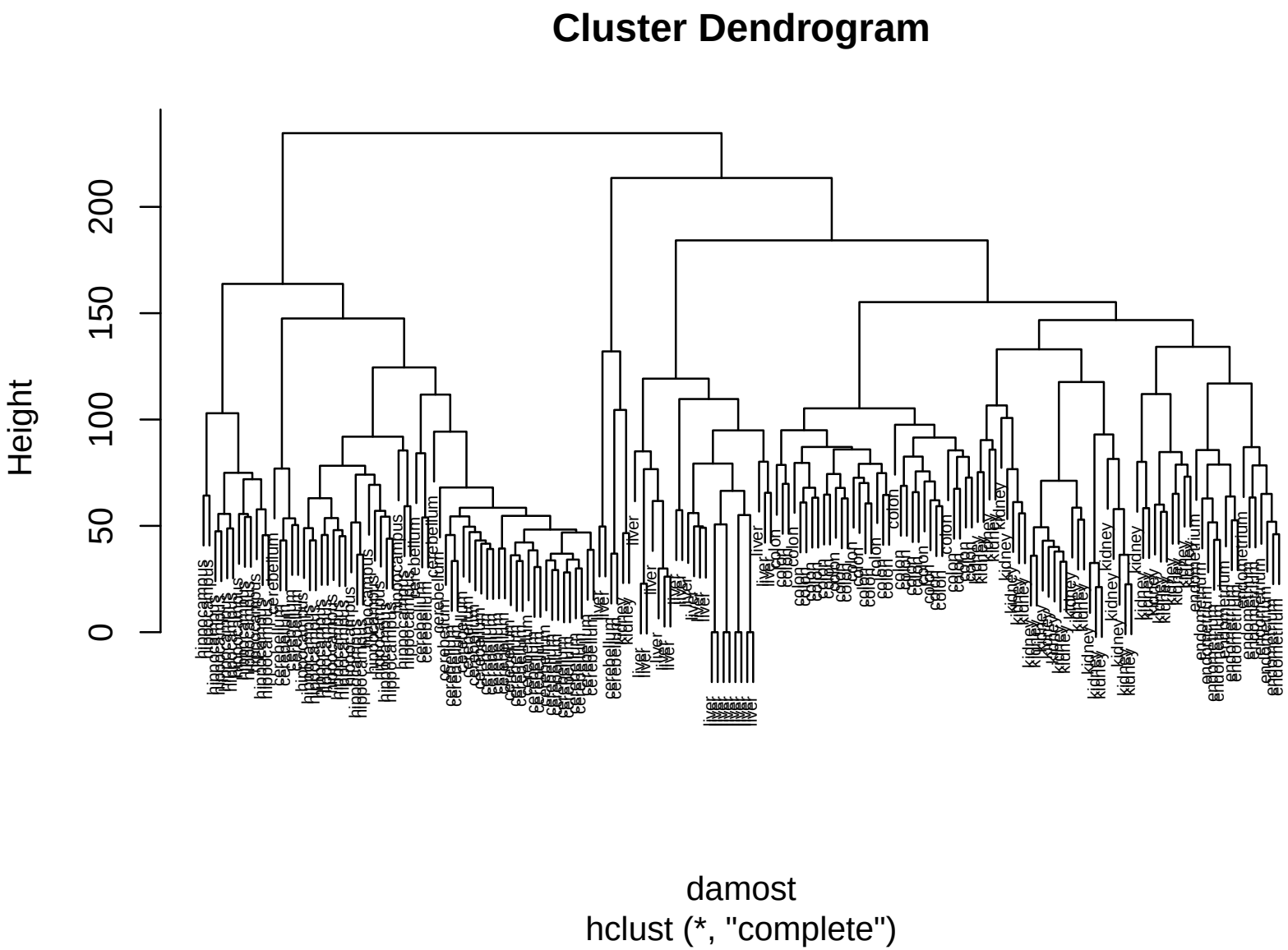
Resultados dos Carros

- Sedãs econômicos ficam juntos
- Separadas dos carros mais esportivos e poderosos
- 2 subgrupos dos esportivos
  - Verdadeiro carros muscle (Duster e Camaro)
  - “Baby muscle”, os compactos Hornet e Firebird

hclust Aplicada aos Genes e Amostras

```
damost <- (dist(t(e)))
hcamost <- hclust(damost)
```

```
plot(hcamost, labels = tissue, cex = 0.5)
```

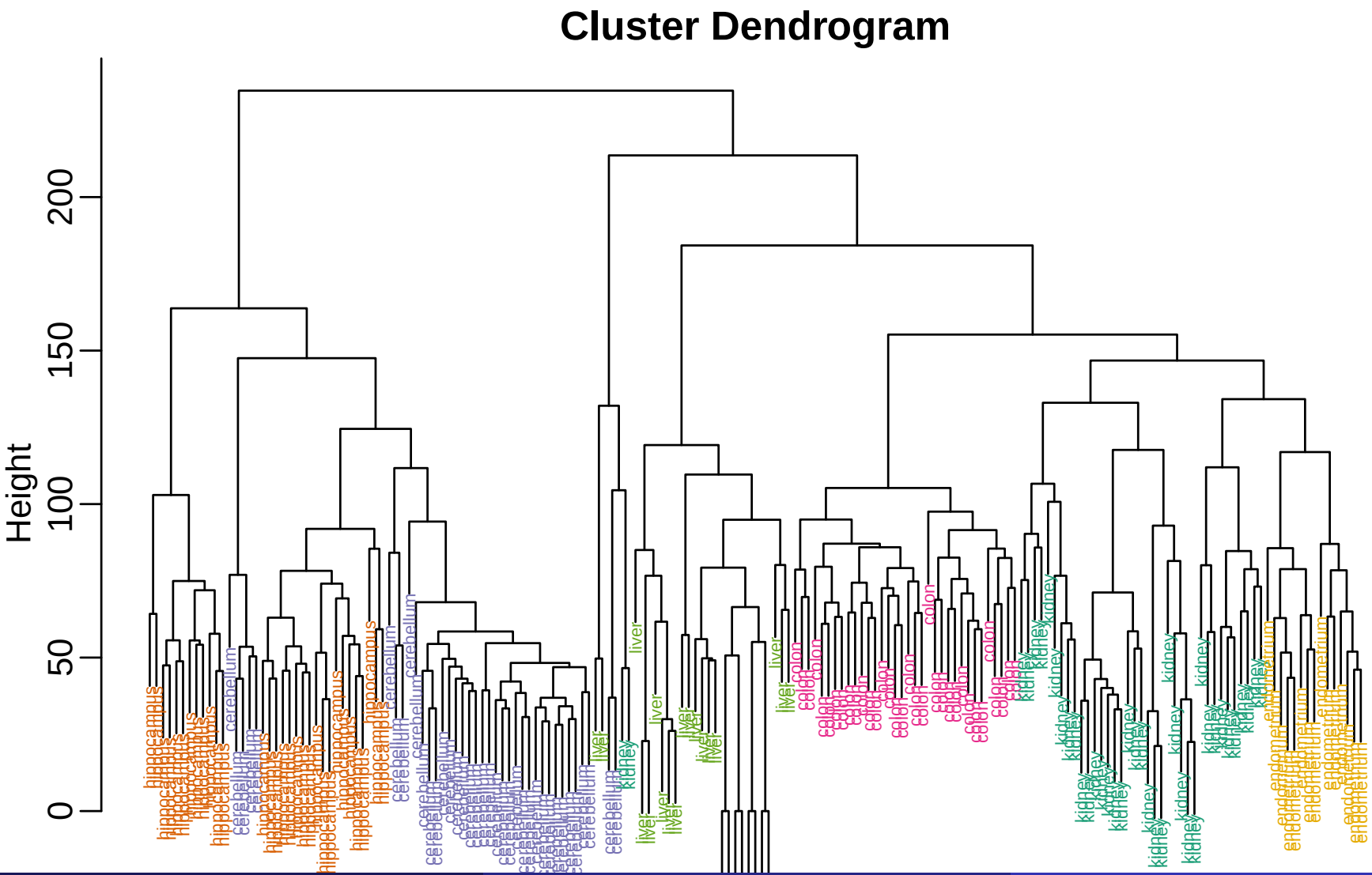


Problemas de Leitura

- Clusters parecem mas difícil ler eles
  - Densidade de ramos de dendrograma
- Utilizar algumas funções do pacote `rafalib` de Irizzary e Love
  - Para facilitar análises de dados de alta dimensão
- `mypar()` – bons parâmetros para gráficos
- 'myplclust()' – colocar cores nos plotagens de clusters hierárquicos

Dendrograma com Cores

```
mypar() # sempre chamada sem argumentos
myplclust(hcamost, labels=tissue, lab.col=as.fumeric(tissue), cex=0.5)
```



Definição dos Clusters

- Precisamos determinar uma altura em que podemos diferenciar os grupos
- Se usamos uma altura de 120, temos uma arvore que claramente define os tecidos
- Função `cutree()` e especifica a altura de 120 (`h = 120`)
- Tabela que define os tecidos caindo em cada cluster

```r
amostclust <- cutree(hcamost, h = 120)
table(true = tissue, cluster = amostclust)
```

```
##                cluster
## true             1  2  3  4  5  6  7  8  9 10 11 12
##   cerebellum     0  0  0  0 31  0  0  0  2  0  0  5
##   colon          0  0  0  0  0  0 34  0  0  0  0  0
##   endometrium    0  0  0  0  0  0  0  0  0  0 15  0
##   hippocampus    0  0 12 19  0  0  0  0  0  0  0  0
##   kidney         9 18  0  0  0 10  0  0  2  0  0  0
##   liver          0  0  0  0  0  0  0 24  0  2  0  0
```

Tabela Mostra

- Cada cluster definido por um tecido só
  - Fora de cluster 9 que tem duas amostras vindo de cerebellum e duas do rim
  - Pode ignorar este conflito dentro deste cluster – poucos casos

`cutree()` com Outro Número de Clusters (k)

- Usar o argumento `k =`
- Testar com `k = 7`

7 Clusters

```
amostclust2 <- cutree(hcamost, k = 6)
table(true = tissue, cluster = amostclust2)
```

```
##                cluster
## true             1  2  3  4  5  6
##   cerebellum     0  0 36  0  0  2
##   colon          0  0  0 34  0  0
##   endometrium   15  0  0  0  0  0
##   hippocampus    0 12 19  0  0  0
##   kidney        37  0  0  0  0  2
##   liver          0  0  0  0 24  2
```

Clusters k-Médias com `kmeans`

- Escolhemos o número de clusters que queremos ver
- Cada conjunto tem um número ótimo de de clusters
- Pode estimar um número bom com pacote `factoextra`
- Função `fviz_nbclust()` usa vários índices para avaliar o número melhor de clusters

Experimentar com Conjunto Simples

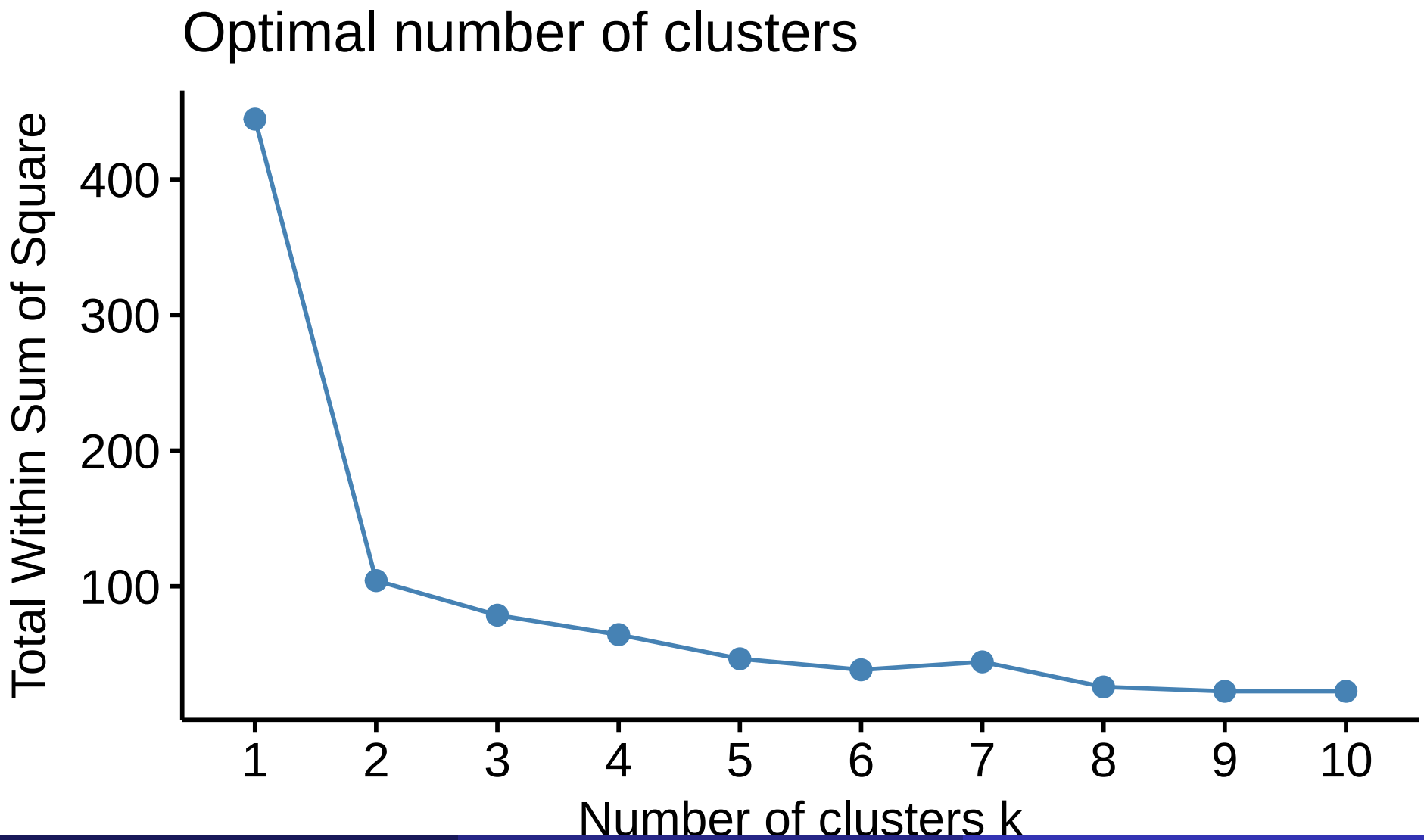
- Dados simulados
- 50 observações de 2 variáveis

Criar Conjunto

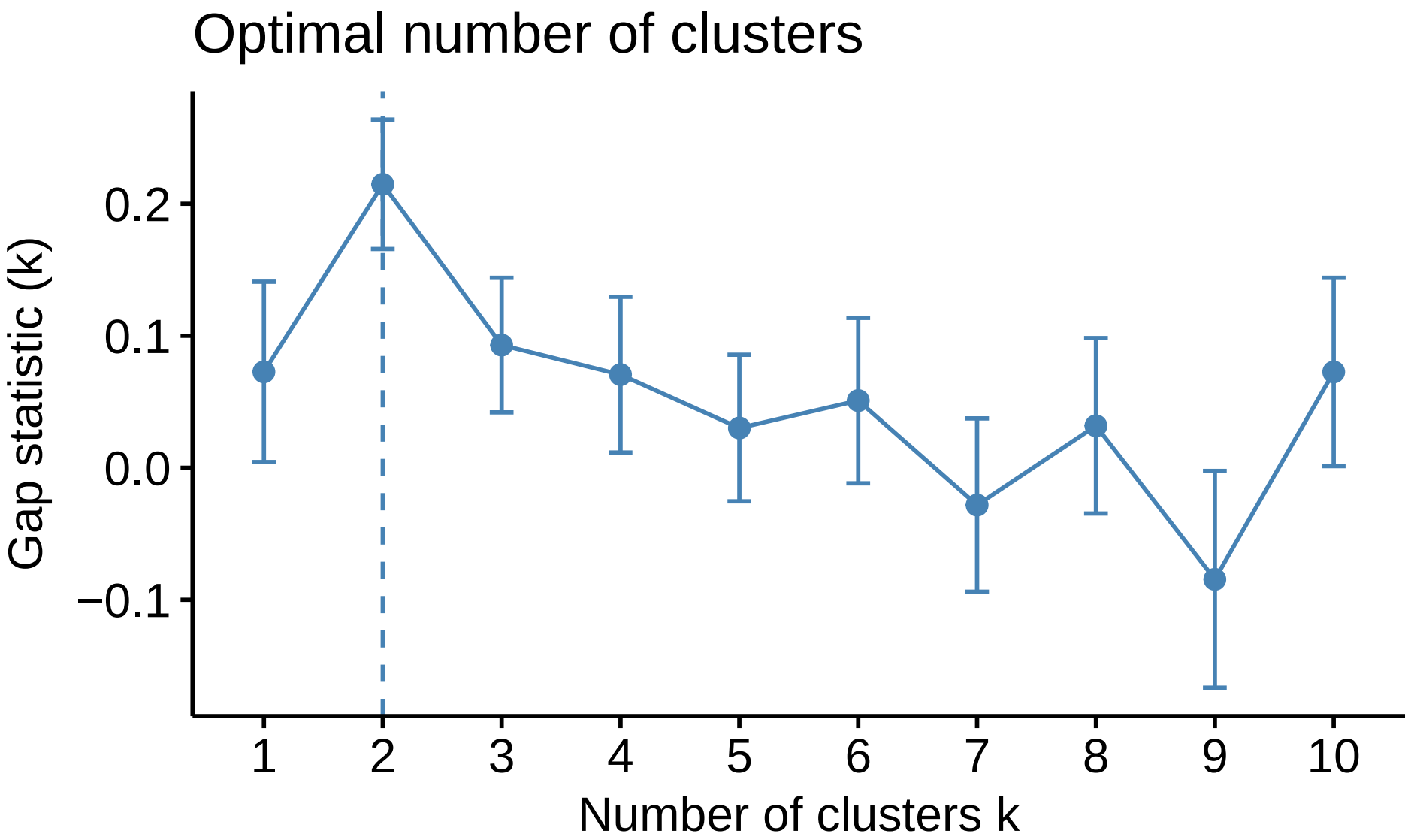
```
## Criar conjunto
set.seed(42)
x=matrix(rnorm(50*2), ncol=2)
x[1:25,1]=x[1:25,1]+3
x[1:25,2]=x[1:25,2]-4
```

Calcular Número de Clusters

```
set.seed(42)
fviz_nbclust(x, kmeans, method = "wss")
```

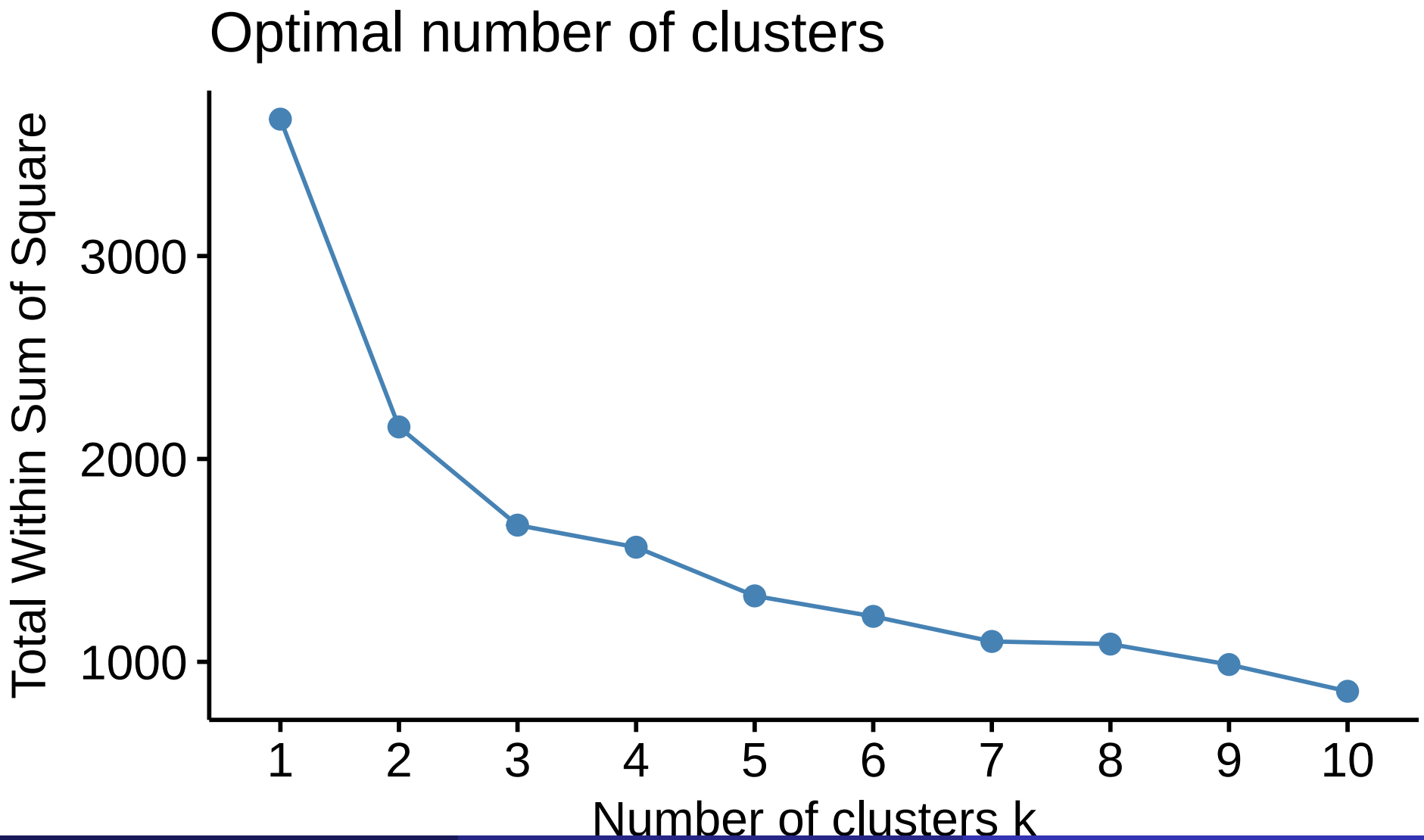


```r
set.seed(42)
fviz_nbclust(x, kmeans, method = "gap_stat")
```

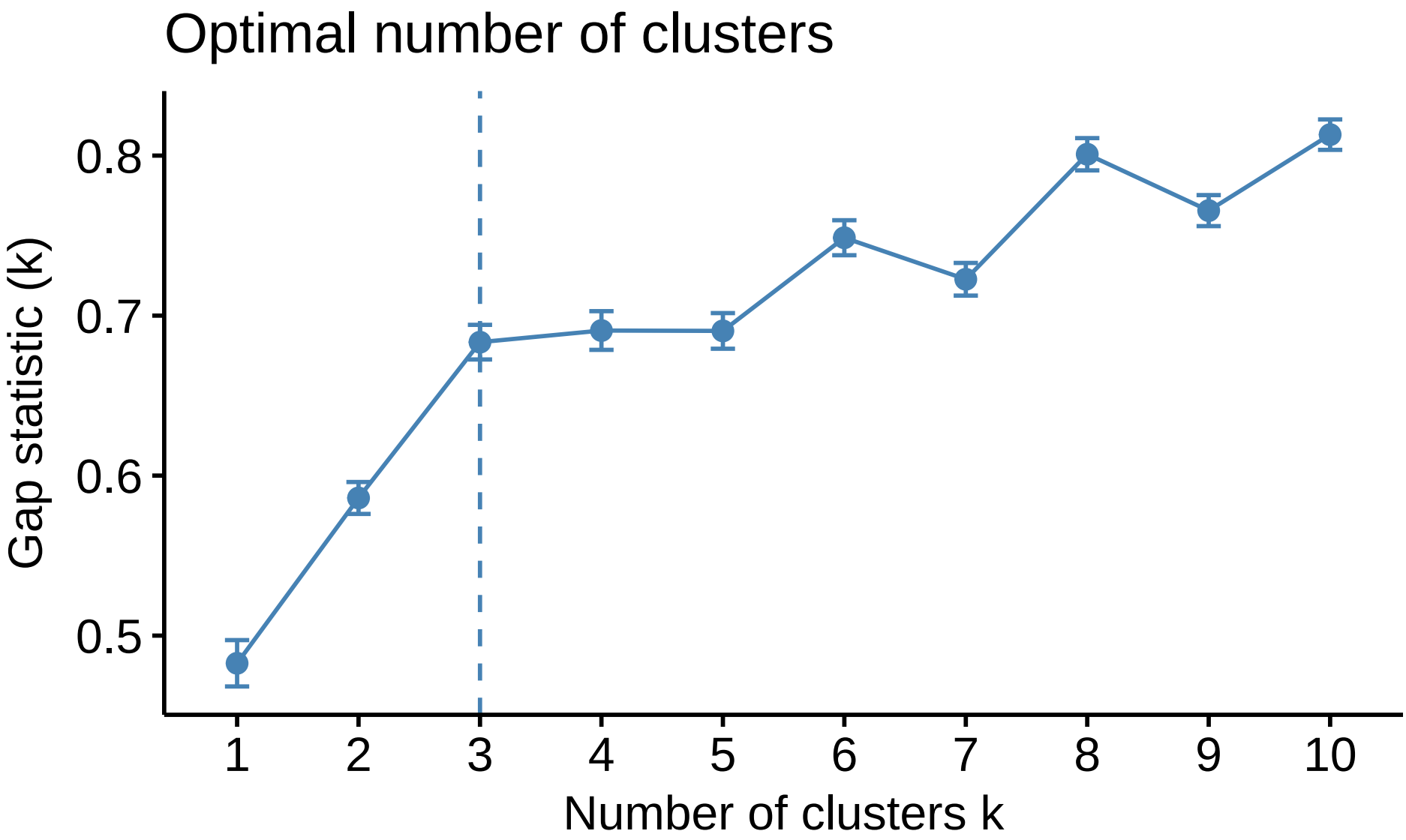


Calculo de Clusters de Amostras e Genes (50 Genes)

```
set.seed(42)
fviz_nbclust(t(e[1:50,]), kmeans, method = "wss")
```



```r
set.seed(42)
fviz_nbclust(t(e[1:50,]), kmeans, method = "gap_stat")
```



Resultado de Estimação de Número de Clusters

- "Cotovelo" em 3 clusters
  - Inclinação da linha muda de íngreme a rasa
  - Outro em 7 clusters
- Cotovelo mostra uma boa escolha do número

`nstart` = – Pontos Iniciais Aleatórios

- Modelo precisa um número alto de pontos iniciais
  - Testa todos que são especificados e escolhe o melhor
  - Com pouco, risco de chegar a conclusão rápido demais
  - Um número alto de `nstart` reduziria a soma dos quadrados total ("wss")
  - Dentro dos clusters

Pequeno Exemplo de nstart =

```r
set.seed(42)
resx1 <- kmeans(x, 3, nstart = 1)
resx50 <- kmeans(x, 3, nstart = 50)
paste("wss com nstart = 1:", round(resx1$tot.withinss, 3))
```

```
## [1] "wss com nstart = 1: 78.674"
```

```r
paste("wss com nstart = 50:", round(resx50$tot.withinss, 3))
```

```
## [1] "wss com nstart = 50: 76.58"
```

Testar o Modelo de 50 Genes

```
set.seed(42)
kamost3 <- kmeans(t(e[1:50,]), centers = 3, nstart = 50)
paste("Soma dos quadrados dentro dos clusters:", round(kamost3$tot.withinss, 3))
```

```
## [1] "Soma dos quadrados dentro dos clusters: 1673.701"
```

```
table(true = tissue, cluster = kamost3$cluster)
```

```
##                cluster
## true             1  2  3
##   cerebellum     4 34  0
##   colon         34  0  0
##   endometrium   15  0  0
##   hippocampus    0 31  0
##   kidney        36  3  0
##   liver          0  0 26
```

Modelo com k Aumentado a 7 – Próximo Cotovelo

```
set.seed(42)
kamost7 <- kmeans(t(e[1:50,]), centers = 7, nstart = 50)
paste("Soma dos quadrados dentro dos clusters:", round(kamost7$tot.withinss, 3))
```

```
## [1] "Soma dos quadrados dentro dos clusters: 1064.251"
```

```
table(true = tissue, cluster = kamost7$cluster)
```

```
##              cluster
## true           1  2  3  4  5  6  7
##   cerebellum   2  0  4  0 30  0  2
##   colon        0  0 34  0  0  0  0
##   endometrium  0  0  1  0  0 14  0
##   hippocampus 29  0  0  0  0  0  2
##   kidney       1  0  0  0  0 36  2
##   liver        0  2  0 24  0  0  0
```

Tentativa com $k = 9$

```
set.seed(42)
kamost9 <- kmeans(t(e[1:50,]), centers = 9, nstart = 50)
paste("Soma dos quadrados dentro dos clusters:", round(kamost9$tot.withinss, 3))
```

```
## [1] "Soma dos quadrados dentro dos clusters: 880.651"
```

```
table(true = tissue, cluster = kamost9$cluster)
```

```
##              cluster
## true           1  2  3  4  5  6  7  8  9
##   cerebellum   0  4  2  2 30  0  0  0  0
##   colon        0 33  0  0  0  0  0  1  0
##   endometrium  9  1  0  0  0  0  0  5  0
##   hippocampus  0  0  2 29  0  0  0  0  0
##   kidney      18  0  2  0  0  0  0 19  0
##   liver        0  0  0  0  0 17  7  0  2
```

Modelo com Todos os Genes

```
set.seed(42)
kamost8 <- kmeans(t(e), centers = 9, nstart = 50)
paste("Soma dos quadrados dentro dos clusters:", round(kamost8$tot.withinss, 3))
```

```
## [1] "Soma dos quadrados dentro dos clusters: 516630.921"
```

```
table(true = tissue, cluster = kamost8$cluster)
```

```
##                cluster
## true             1  2  3  4  5  6  7  8  9
##   cerebellum     0  2  0  0  5  0  0  0 31
##   colon          0  0  0  0  0  0  0 34  0
##   endometrium    0  0 15  0  0  0  0  0  0
##   hippocampus    0  0  0 31  0  0  0  0  0
##   kidney        18  2  0  0  0  0 19  0  0
##   liver          0  2  0  0  0 24  0  0  0
```

Heatmaps

- Encontra clusters hierárquicos mais frequentemente em heatmaps
- Gráficos que combinam
  - Intensidade de expressão genomica (heatmap)
  - Indicações da organização dos genes ou outros elementos (dendrogramas nas margens)

Passo 1: Estabelecer Uma Gama de Cores

```
hmcol <- colorRampPalette(brewer.pal(9, "GnBu"))(100)
head(hmcol)
```

```
## [1] "#F7FCF0" "#F5FBEE" "#F3FAEC" "#F1F9EA" "#EFF9E9" "#EDF8E7"
```

Passo 2: Escolhe os Genes a Colocar no Heatmap

- Precisamos usar uma função que vem do sistema *Bioconductor*
  - Como fazer o download do Bioconductor e os pacotes fica na versão escrita
- Pacote `genefilter`

Genes com a Máxima Variância

- Escolher 40 genes
- Designar uma cor para cada amostra baseado nos tipos de tecido
- `idx` será os índices dos genes com a variância máxima determinada por `rowVars`

```r
suppressPackageStartupMessages(library(genefilter))
rv <- rowVars(e) # calcular as variâncias de todos os genes
idx <- order(-rv)[1:40] # pôr em ordem e escolhe o top 40
## Designar cores `cols` para cada amostra baseado no tecido
cols <- palette(brewer.pal(8, "Dark2"))[as.fumeric(tissue)]
```

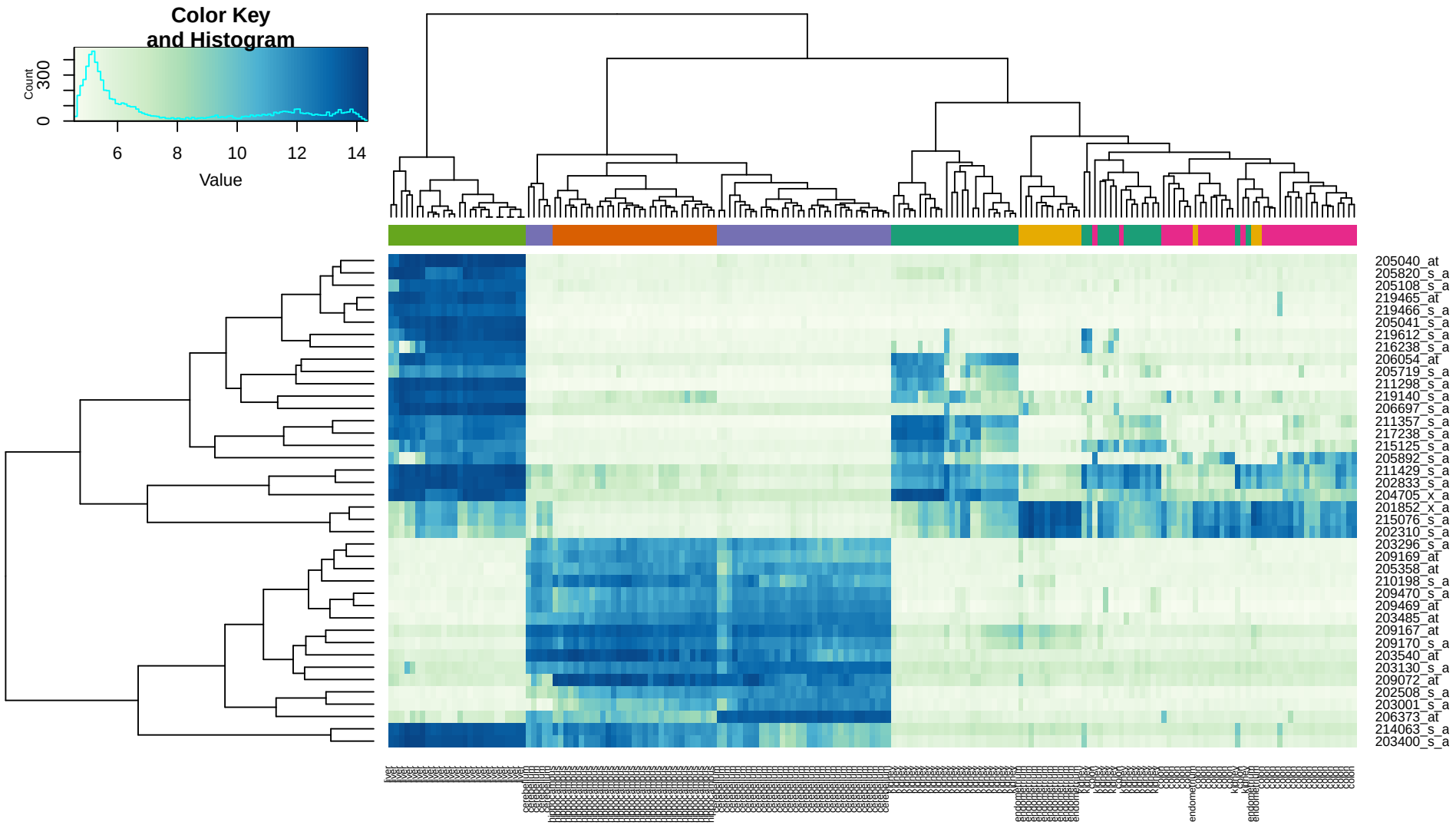
```
headTail(cbind(colnames(e), cols))
```

```
##                    V1    cols
## 1    GSM11805.CEL.gz #1B9E77
## 2    GSM11814.CEL.gz #1B9E77
## 3    GSM11823.CEL.gz #1B9E77
## 4    GSM11830.CEL.gz #1B9E77
## ...              <NA>    <NA>
## 180 GSM323527.CEL.gz #7570B3
## 181 GSM323565.CEL.gz #7570B3
## 182 GSM323566.CEL.gz #7570B3
## 183 GSM323567.CEL.gz #7570B3
```

Passo 3: Fazer o Heatmap

- Função de heatmaps de pacote `gplots` mais flexível que a de R base (`heatmap.2`)
- Precisa download `gplots`

```
suppressPackageStartupMessages(library(gplots))
heatmap.2(e[idx,], labCol=tissue,
          trace="none",
          ColSideColors=cols,
          col=hmcol)
```


Color Key
and Histogram
Count
0
300
6
8
10
12
14
Value
205040_at
205820_s_a
205108_s_a
219465_at
219466_s_a
205041_s_a
219612_s_a
216238_s_a
206054_at
205719_s_a
211298_s_a
219140_s_a
206697_s_a
211357_s_a
217238_s_a
215125_s_a
205892_s_a
211429_s_a
202833_s_a
204705_x_a
201852_x_a
215076_s_a
202310_s_a
203296_s_a
209169_at
205358_at
210198_s_a
209470_s_a
209469_at
203485_at
209167_at
209170_s_a
203540_at
203130_s_a
209072_at
202508_s_a
203001_s_a
206373_at
214063_s_a
203400_s_a