Matéria de Análise de Dados – Ciências Biomédicas

Aula 17: Machine Learning – Não Supervisionado

James Hunter

25 de Abril de 2017

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

Introdução

Neste unidade do curso, nós vamos tratar de machine learning não-supervisionado. Este quer dizer machine learning sem referência aos valores de uma variável dependente. Invés, nós vamos focar só nas variáveis independentes, as covariáveis, para ver como os casos se agrupam em clusters. Então podemos interpretar esses clusters para ver se tiveram significância.

Também, falei bastante de como machine learning pode aplicar-se para "big data", conjuntos enormes dos dados. Nós vamos usar um tal conjunto para aprender esse tipo de técnica. Nós vamos estudar a expressão de 22215 genes em 189 amostras de seis tipos de tecido. Os dados de expressão ocupam acima de 33 MB de espaço no memoria.

Nós vamos usar **análise de clusters** para determinar se podemos usando só os genes, podemos identificar os tecidos.

Os dados vêm do curso e livro **Data Analysis for the Life Sciences** de Rafael Irizzary e Michael Love.[1] Você pode obter os dados do site do curso de Irizzary e Love com o código seguinte:

```
library(devtools)
install_github("genomicsclass/tissuesGeneExpression")
```

```
library(tissuesGeneExpression)
data(tissuesGeneExpression)
dim(e) ## e contem os dados de expressão
```

```
## [1] 22215   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
```

Por causa de número baixo das amostras de placenta, vamos tirar essas amostras. Vai criar "noise" nos resultados mais tarde sem aumentar muito informação.

```
plac <- which(tissue == "placenta")
tissue <- tissue[-plac]
e <- as.data.frame(e) %>% select(-plac)
e <- as.matrix(e)
tab <- tab %>% slice(-plac)
table(tissue)
```

[1] Rafael Irizzary e Michael Love, **Data Analysis for the Life Sciences** (Leanpub, 2015). Você pode obter o livro *aqui*. O site do dados é da matéria PH525-4 (High Dimensional Analysis), de Harvard University no *EDX*.

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

```
table(tab$Tissue)
```

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

Distância

Quando falamos do processo de agrupamento o a formação de clusters, nós precisamos uma medida que podemos usar para julgar a relação entre os itens que estamos agrupando. A medida que faz o máximo sentido é a distância de um item com todos os outros. A dimensionalidade desta distância depende no número de covariáveis que estamos usando para julgar o agrupamento. A "distância" neste caso é uma abstração matemática, não uma distância física, limitada pelas duas dimensões do plano cartesiano. Cada amostra se define por 22.215 dimensões da matriz de expressão dos genes: $(E_{1,i}, \ldots, E_{22215,i})$ e cada gene se define em 189 dimensões: $(E_{g,1}, \ldots, E_{g,189})$.

Apesar do número alto de dimensões, podemos usar o mesmo calculo euclideano para determinar a distância entre pontos, exatamente como fazemos com pontos em espaço de duas dimensões. Lembrando que a distância entre dois pontos na mapa seria o seguinte:

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

No caso de um alto número de dimensões, o calculo simplesmente estende o euclideano a medir todas as diferenças entre A e B através de todas as dimensões que você está medindo. No caso das amostras, seria as 22.215 dimensões representadas pelos genes.

Distância em Algebra Linear

Esta expressão de diferença fica mais claro na linguagem das matrizes, álgebra linear. Neste notação, a letra "T" quer dizer que a matriz é *transposta* em que todos os elementos são reconfigurados de linhas em colunas e colunas em linhas. Por exemplo, uma matriz A com a forma

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

torna em A^T (A transposta) se as linhas e colunas são trocadas:

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

Agora que entendemos um pouco dos símbolos de álgebra linear, vamos escrever a distância entre qualquer dimensão dos genes e amostras:

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

com $\mathbf{Y}_i$ e $\mathbf{Y}_j$ sendo colunas i and j.

Vamos comparar a distância euclideana entre três amostras, duas dos *kidneys* – amostras 1 e 2 – e uma do *colon* – amostra 87.

```
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"
```

Seria muito difícil de expressar a unidade de destas distâncias porque têm 22.215 dimensões, mas existe. Uma conclusão que podemos observar é que os tecidos de *kidney* são mais próximos um ao outro que os dois são com o tecido de *colon*; uma conclusão que faz todo sentido.

Função `dist`

Quando queremos calcular a distância de uma vez, podemos usar a função `dist()`. Esta função calcula a distância entre linhas, não colunas. Então, precisamos transpôr a matriz `e` com a função `t()` para calcular a distância entre as colunas. O resultado seria um objeto de classe `dist`. Para trabalhar com este objecto, melhor forçar ele ao formato de uma matriz com `as.matrix()`[2].

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

Podemos confirmar com `dist()` as distâncias que calculamos antes.

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

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

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

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

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

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

Análise de Clusters

Agora que temos uma matriz de distâncias entre as amostras, podemos agrupar elas em clusters. Há duas variedades de algoritmos de clustering em R, clusters hierárquicos (*hierarchical clustering*) e clusters por k-médias (*k-means*). Você vai achar que análise de clusters é mais experimental que outras técnicas de estatística e análise de dados. A melhora maneira de usar clusters é de tentar vários modelos guardando os

[2]Se você tenta calcular a distância entre todos os genes, o resultado seria uma matriz de 22215 linhas x 22215 colunas, que vai provavelmente acabar com a memoria de seu computador. É importante em nossos exercícios que focamos em amostras, que só vai criar uma matriz de 189 x 189.

resultados de todos. Você pode assim olhar em todas as alternativas para ver se quais padrões emergem. Esses padrões podem formar um ponto de partida de uma nova hipótese cientifico e uma nova análise.[3]

Clusters hierarquicos O algoritmo começa criando um grupo para cada amostra e depois em cada iteração (ou loop) juntando os dois clusters mais próximos até existe um único cluster com todos as amostras. Nos algoritmos hierarquicos, não sabemos ao início quantos clusters nosso modelo vai produzir.

Clusters k-médias O algoritmo inicialmente aloca todas as amostras a um dos k grupos aleatoriamente. Depois, em cada iteração, o algoritmo determina o ponto central de todos os clusters – o *centroide* – e realoca as amostras para grupo com o centroide mais próximo até não é possível fazer mais realocações. O objetivo é para minimizar a variação dentro de cada cluster ("within cluster sum of squares"). Este dá para os cluster o máximo compacidade ou densidade. Neste tipo de clustering, decidimos ao início quantos clusters queremos e particionamos os dados para esse número de grupos.

Um dos pontos fortes de análise de clusters é a flexibilidade dela. Nós podemos agrupar as observações baseado nos features (variáveis) para descobrir sub-grupos das observações (o que estamos fazendo com os genes e as amostras) ou podemos agrupar as variáveis (genes) baseado nas observações para descobrir quais genes são muito parecidos.

Algoritmos de Clusters em R

Clusters Hierarquicos – `hclust()`

Para funcionar a análise `hclust` precisa uma matriz de distâncias e um método de aglomeração. Existem vários métodos possíveis. Mas, nós vamos trabalhar com o método padrão, `complete`, que maximiza a distância entre os pontos (amostras) que pertencem aos clusters diferentes.

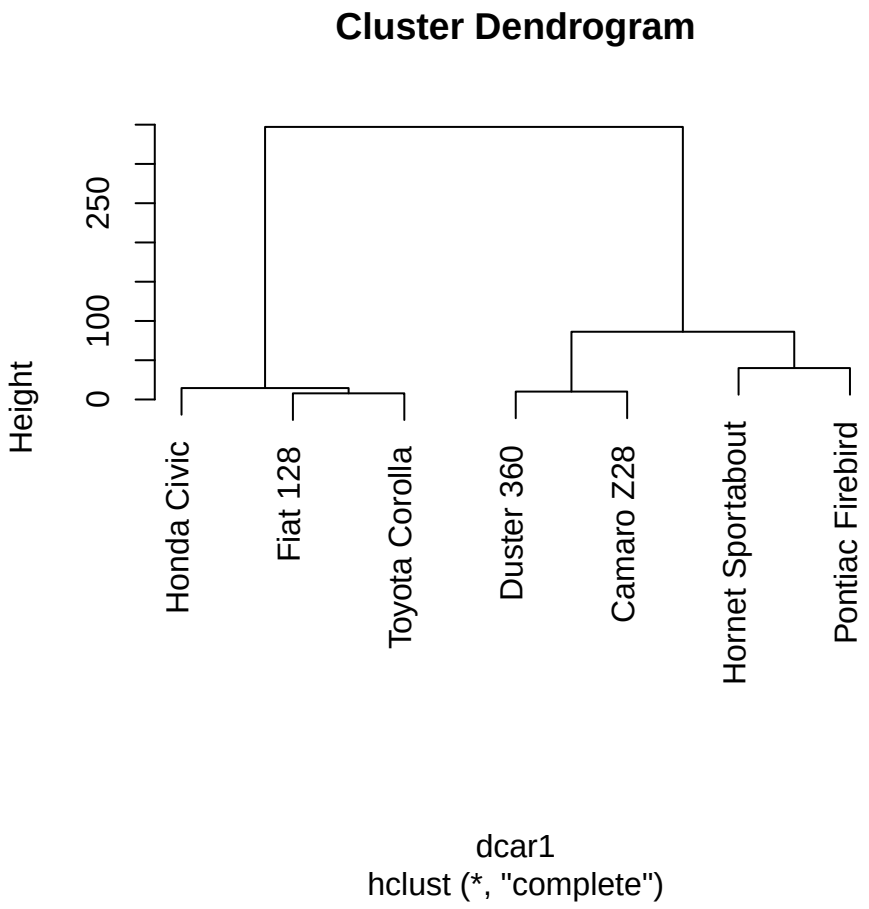
O resultado de um análise utilizando `hclust` é normalmente um *dendrograma*, um gráfico que representa os clusters que o modelo desenvolveu. Pode acessar este gráfico através a função `plot(x)`, onde x é o nome do objeto de cluster.

Antes de experimentar os clusters de nossos dados sobre genes e tecidos, vamos olhar num modelo muito mais simples: uma parte de conjunto (7 dos 32 carros originais) de dados `mtcars`, que trata de carros de 1974. Este conjunto faz como parte de base R. Os carros são

Hornet Sportabout, Duster 360, Fiat 128, Honda Civic, Toyota Corolla, Camaro Z28, Pontiac Firebird

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

[3] James, G., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning.* (Springer New York, 2013), p. 401.

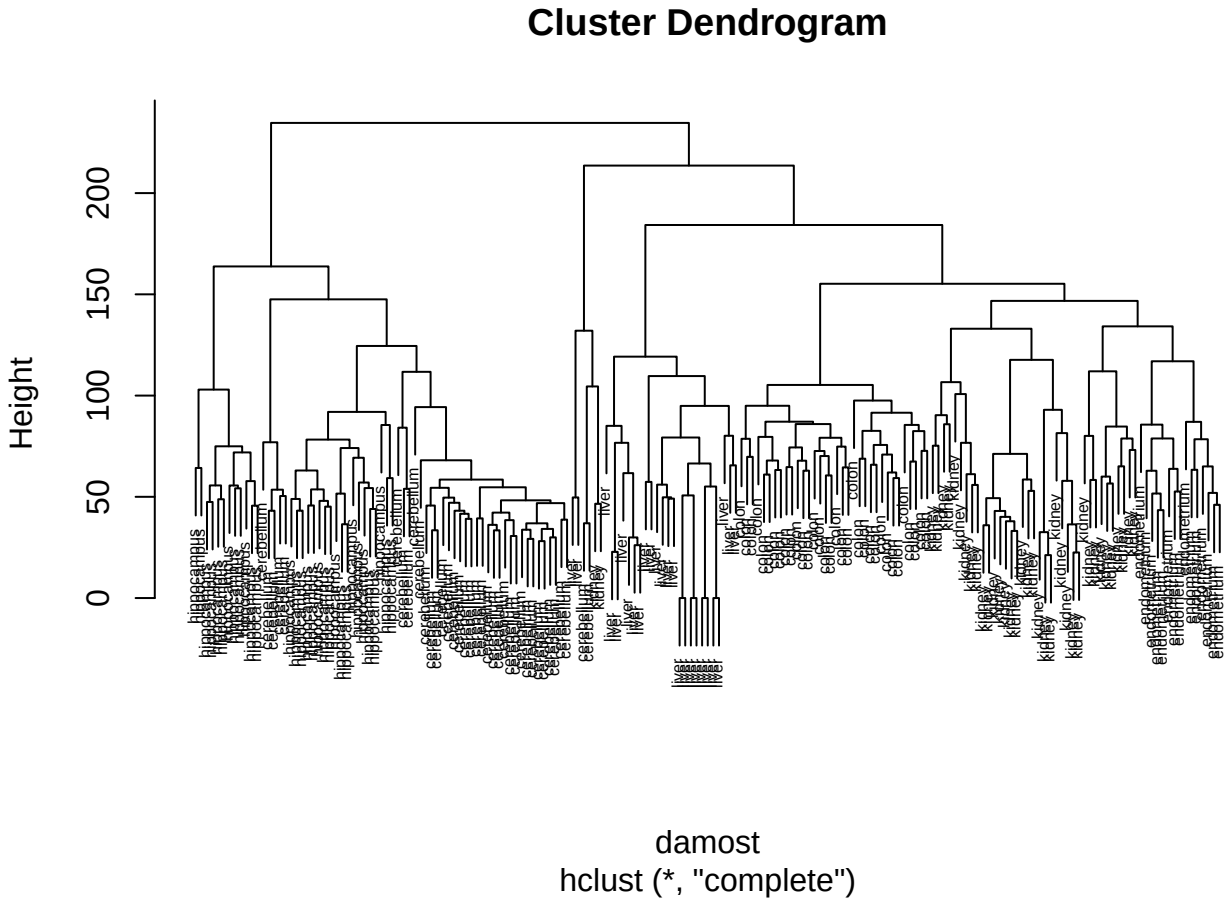


Nossos clusters mostram que sedãs econômicos (Honda Civic, Fiat 128 e Toyota Corolla) ficam juntos em um cluster separados pelos carros mais esportivos e poderosos que ficam no outro grupo ou cluster, que podemos chamar os carros "muscle". E nossa análise corretamente separa esses carros em 2 subgrupos, verdadeiro carros muscle (Duster e Camaro) dos carros "baby muscle", os compactos Hornet e Firebird.

`hclust` Aplicada aos Genes e Amostras

Para calcular a hierarquia dos clusters dos amostras segundo os genes, usaremos a matriz das distâncias `damost` que calculamos antes e aplicar a `hclust`. Anote bem que aqui precisamos as distâncias no formato (classe) de *distância* e não no formato de matriz.

```
damost <- (dist(t(e)))
hcamost <- hclust(damost)
plot(hcamost, labels = tissue, cex = 0.5)
```

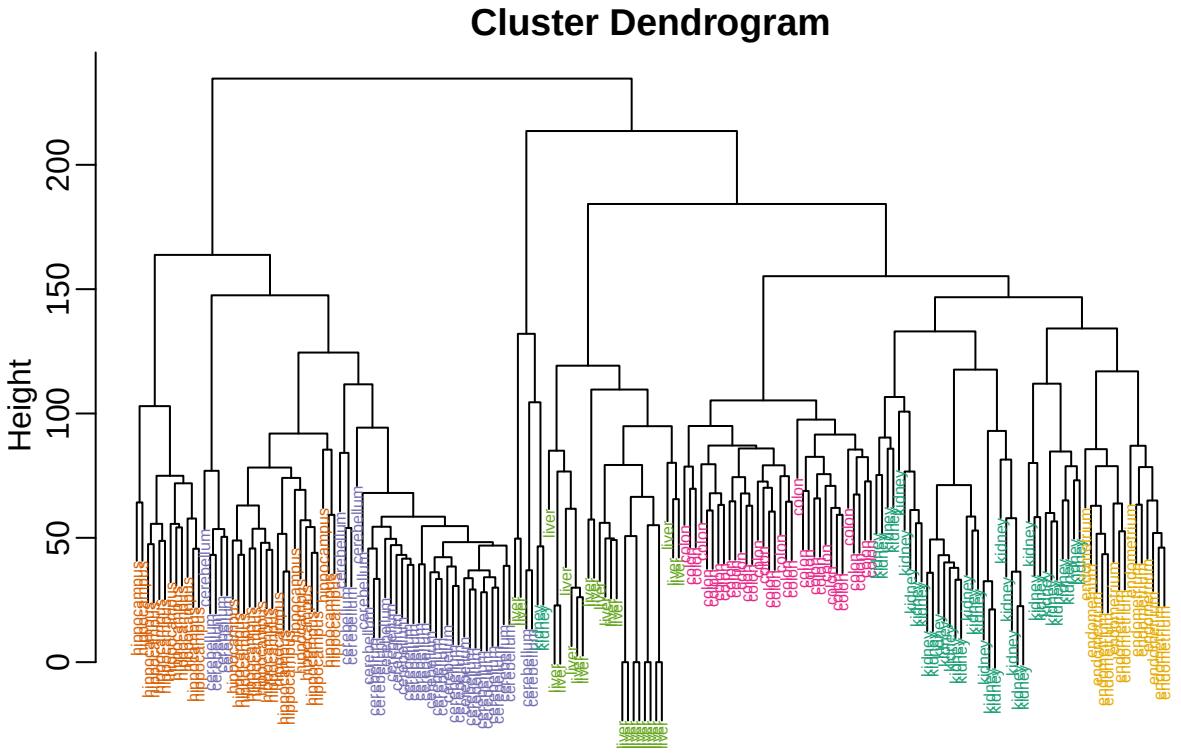


Os clusters parecem, mas por causa do número dos ramos da dendrograma, é muito difícil ler todos os rótulos dos tecidos para ver se o modelo agrupou as amostras segundo os tecidos ou não.

Podemos usar algumas funções de apoio que vem do pacote `rafalib` que Rafael Irizzary e Michael Love da Harvard criaram para facilitar as análises dos dados de alta dimensão biologicos. Os que vamos usar aqui são `mypar()` que estabelece bons parâmetros para gráficos e `myplclust()` que ajuda colocar cores nos plotagens de clusters hierarquicos.[4] Vamos olhar no novo gráfico com cores:

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

[4]Rafael Irizzary e Michael Love, `rafalib`, https://CRAN.R-project.org/package=rafalib.



Agora, os clusters por tecidos ficam mais claros. A análise descobriu com sucesso os tecidos e agrupou as amostras juntas por eles.

Para definir claramente os clusters, precisamos determinar uma altura em que podemos diferenciar os grupos. Se usamos todo a altura, teríamos só um cluster que não ajuda explicar a estrutura dos dados. Se usamos uma altura de 150, também teríamos clusters que combinam vários tecidos. Mas, se usamos uma altura de 120, temos uma arvore que claramente define os tecidos.

Para cortar a arvore nesta altura, usamos a função `cutree()` e especifica a altura de 120 (`h = 120`). Com este resultado, podemos construir uma tablela que define os tecidos caindo em cada cluster.

```
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
```

Aqui, podemos ver até mais claramente que cada cluster está definido por um tecido só, fora de cluster 9 que tem duas amostras vindo de cerebellum e duas do rim (*kidney*). Neste caso, porque envolve tão poucos casos, podemos ignorar este conflito dentro deste cluster. Mesmo se reduzimos o número de clusters, que podemos fazer com `cutree()` com o argumento `k =`, o modelo vai alinhar bem com a identidade dos tecidos. Vamos mostrar com 7 clusters, número igual ao número de tecidos. Neste caso, k representa o número de clusters que queremos mostrar.

```
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
```

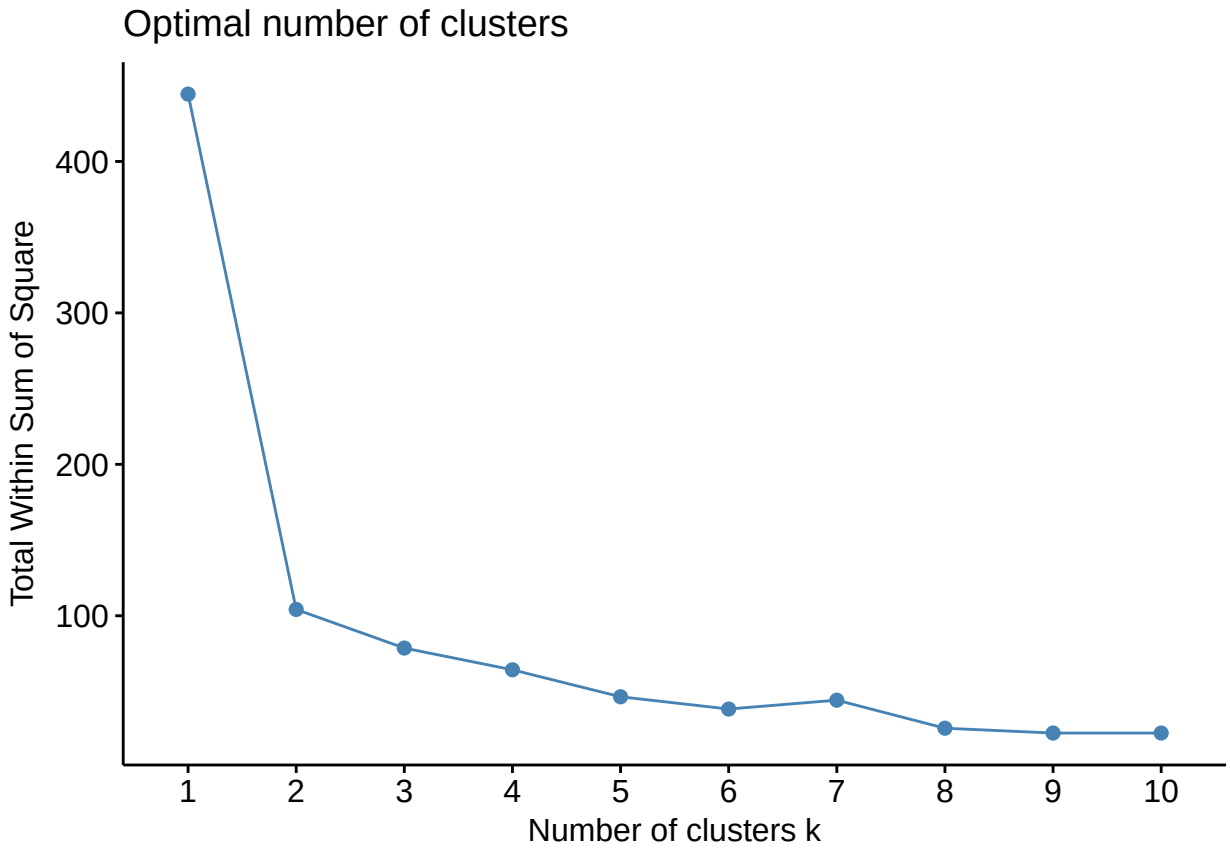
Aqui podemos ver que a situação é um pouco mais confusa. Clusters 2, 4, e 5 só têm um único tipo de tecido. Dos outros, cluster 6 comporta como cluster 9 na primeira tentativa, com poucos tecidos das diferentes regiões do corpo. Mas, cluster 1 mostram dois tecidos do abdome e cluster 3 mostra dois tecidos que fazem parte do cerebro.

Clusters k-Médias com `kmeans`

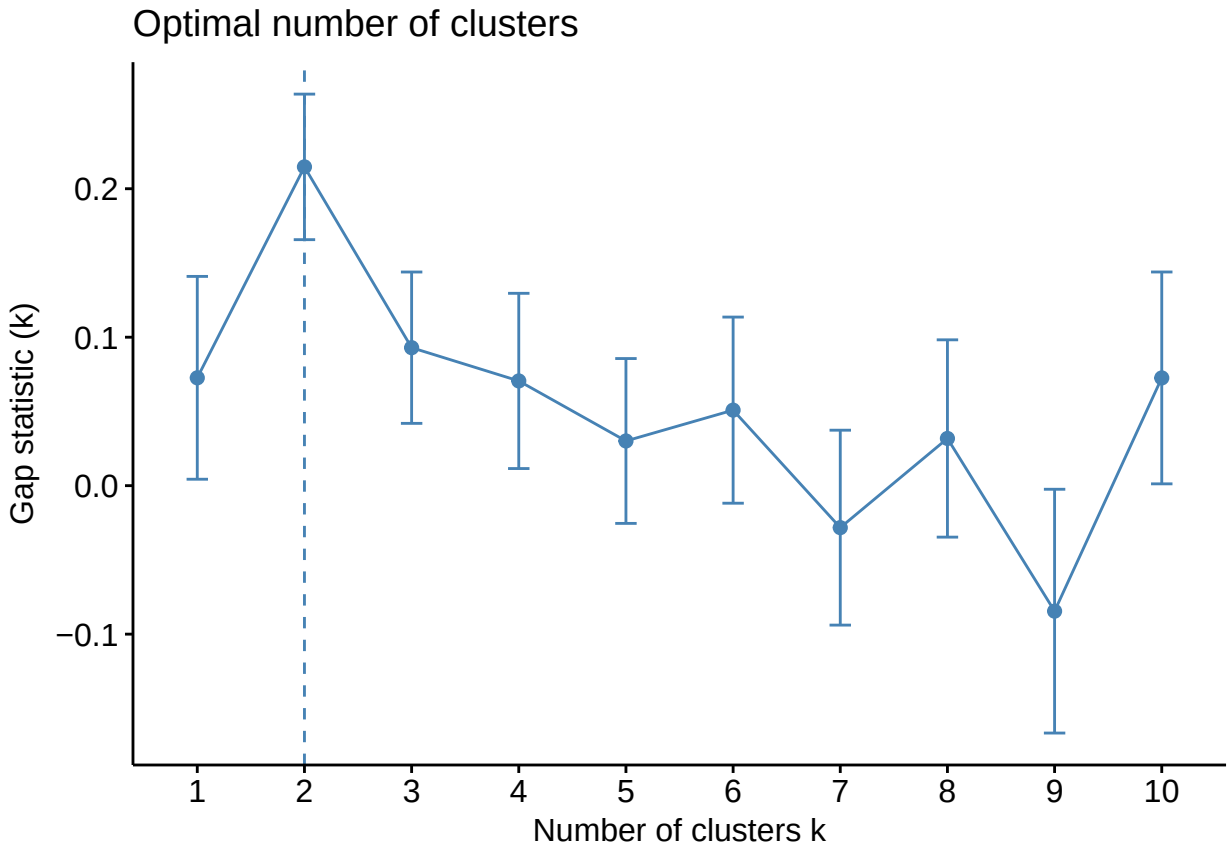
Com k-médias, nós escolhemos o número de clusters que queremos ver. Cada conjunto tem um número ótimo de de clusters que é o produto primeiro de estimação e depois de experimentação. Podemos fazer esta estimação melhor com o pacote `factoextra`, que é novo para nossa turma. Neste pacote, tem uma função, `fviz_nbclust()`, que usa vários índices para avaliar o número melhor de cluster para calcular. Vamos ver como esta funciona primeiro com um conjunto de dados simples[5], uma matriz simulada com 50 observações de 2 variáveis. Vamos criar os dados e depois calcular o número ótimo dos clusters. Eu recommendo o uso de 2 dos índices: `wss` que mede a variação dentro dos clusters e `gap_stat` que compara o wss a um valor esperado para essa variação.

```
## 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 # de clusters
fviz_nbclust(x, kmeans, method = "wss")
```

[5] James, et.al., p. 388.



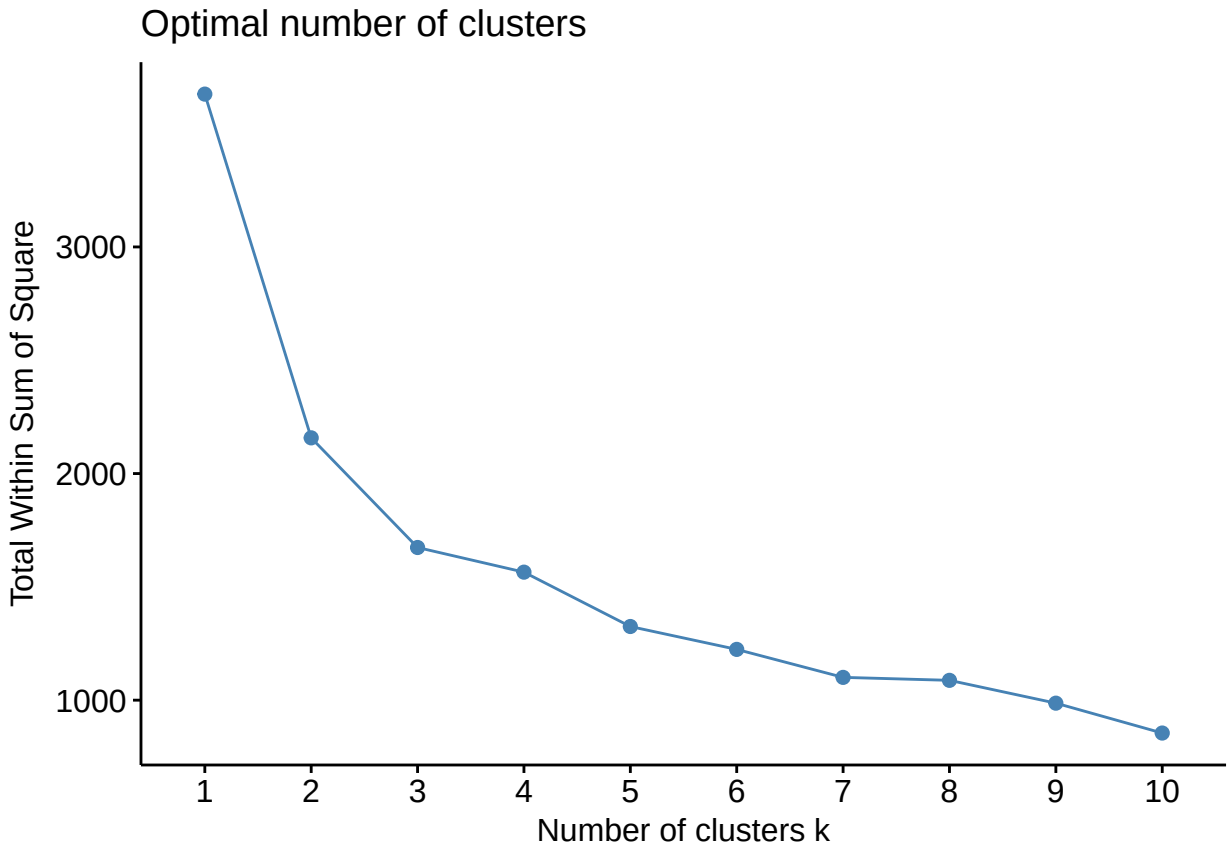

```
fviz_nbclust(x, kmeans, method = "gap_stat")
```



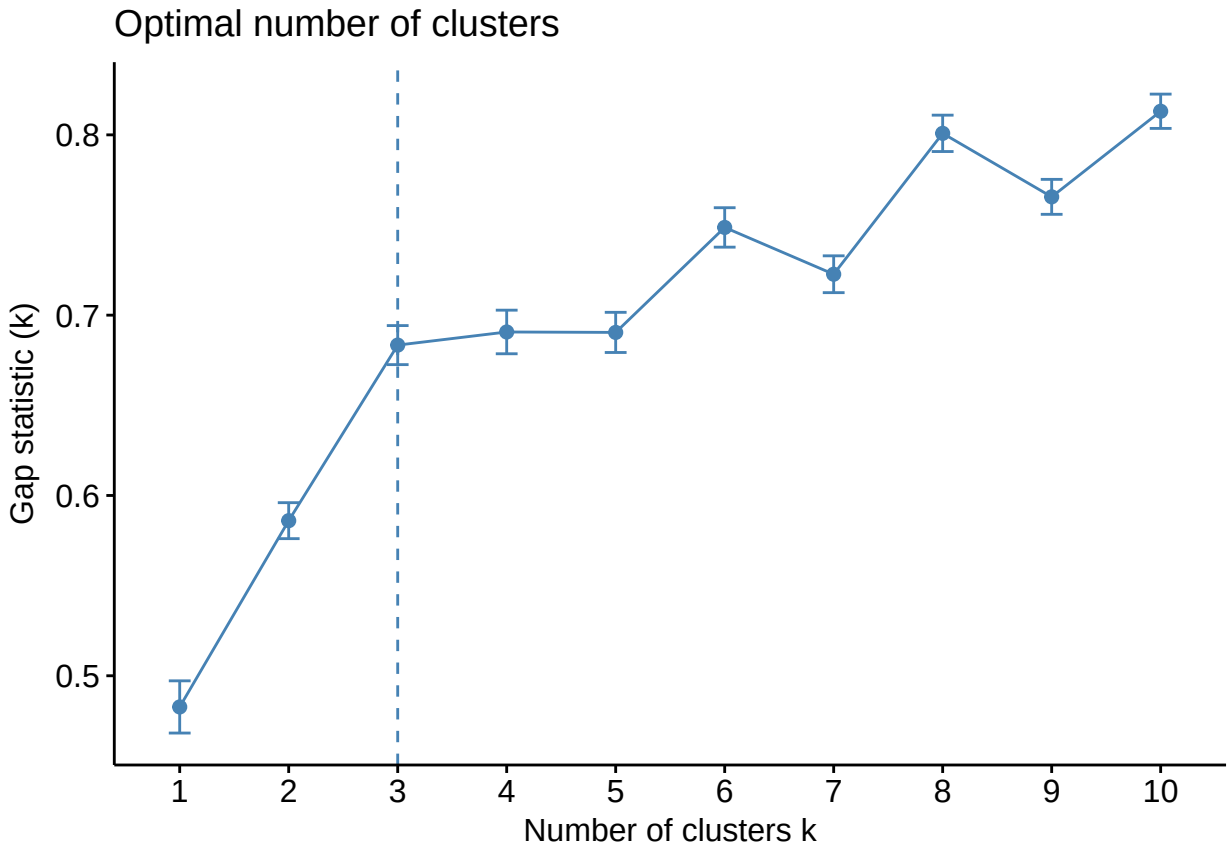
Esses duas medidas indicam 2 como o número melhor para o calculo dos clusters.

Vamos aplicar este método para calculo de número de clusters dos dados de amostras e genes. Neste caso, para encurtir um pouco o tempo de calculo, nós vamos limitar nossa análise a 50 genes, invés dos 22 mil no conjunto original.

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




```
fviz_nbclust(t(e[1:50,]), kmeans, method = "gap_stat")
```



Olhando no primeiro gráfico que mostra o total de soma de quadrados dentro dos clusters, podemos ver que tem um leve "cotovelo" em 3 clusters. A inclinação da linha muda de íngreme a rasa. Este ponto frequentemente mostra uma boa escolha do número de clusters. Também pode ver o mesmo padrão em 7 clusters.

Os calculos sugerem 3 clusters para a análise. Vamos experimentar e ver se o modelo pode identificar os tecidos. Lembre que vamos fazer isso com só 50 dos 22 mil genes. Então os resultados provavelmente não seriam tão bons que fossem obtidos com o conjunto inteiro.

A chamada de `kmeans()` deve ter um número alto de pontos iníciais aleatórios. Senão, o modelo pode ficar vitima de um valor inicial que pode criar um resultado instável. O argumento `nstart` instrui o algoritmo a testar um número de pontos iniciais e escolher aquele com o melhor soma dos quadrados ("wss") total dentro dos clusters. Um pequeno exemplo mostraria que um número alto como 50 reduzaria o valor do objetivo em comparação de um ponto incial. Vamos usar nossa matriz simulada `x` de novo para isso.

```
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"
```

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

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

Este aumento simples no número de pontos iniciais conseguiu tirar um fonte de inconsistência no modelo.

Agora, vamos testar o modelo das amostras, genes e tecidos.

```r
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"
```

```r
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
```

Neste modelo, cluster 2 é claramente representivo do figado, mas clusters 1 e 3 não diferenciam os tecidos muito logicamente. Podemos aumentar o número de clusters para próximo valor indicado no gráfico de `fviz_nbclust()`, que foi 7 clusters.

```r
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"
```

```r
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
```

Apesar de uma melhora forte no *wss*, o modelo ainda mostra uma mistura dos tecidos em clusters 1, 3, 6 e 7. Se aumentamos o número de clusters até 9, talvez podemos conseguir uma separação melhor.

```r
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"
```

```r
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
```

Aqui clusters 6, 7 e 9 unicamente tem amostras do tecido de figado, 5 é tecido de cerebellum, no 4 o

hippocampo é mais importante, e em 2 o cólon. Os outros ainda mostram uma mistura de tecidos. Mas, com o pequeno número de genes com que estamos trabalhando, esses resultados são aceitável.

Se experimentamos com todos os 22 mil genes e 8 clusters, vamos achar uma solução que separa bem os seis tipos de tecidos como no próximo modelo.

```
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

Um lugar que muitos pesquisadores em biologia e medicine já viram clusters hierarquicos é no contexto de "heatmaps", gráficos que combinam intensidade de expressão genomica (heatmap em si) com indicações da organização dos genes ou outros elementos que compõem o heatmap (dendrograma no margem do gráfico). Este heatmap é bem conhecido na literatura. Ele vem de uma microarray de expressão de genes.

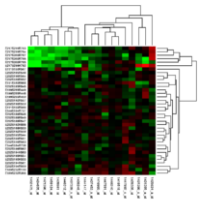


Figure 1: Heatmap de dados de microarray de Wikipedia

Com um subset dos genes, podemos criar nosso próprio heatmap utilizando os pacotes de R. Nós vamos escolher os genes que mostram a maiora variância sobre todos as amostras.

Passo 1: Estabelecer Uma Gama de Cores

A primeira tarefa é de decidir quais cores você quer usar no heatmap. Nós vamos usar aqui cores entre azul e verde vindo de um dos paletas do pacote `RColorBrewer`. Este fica nos pacotes que vocês já obtiveram. Você pode ver que as cores são descritas utilizando a anotação RGB. Com várias dessas funções, vou deixar vocês a descobrir os detalhes de operação porque são periférico ao proposito deste lição.

```
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

Para determinar quais genes têm a maior variância sobre todos as amostras, precisamos usar uma função que vem do sistema *Bioconductor*. `genefilter::rowVars` calcula a variância para cada linha num matriz ou data frame de um conjunto de dados biologicos. Para obter `genefilter`, precisa seguir os passos seguintes:[6]

```
source("https://bioconductor.org/biocLite.R")
biocLite()
BiocInstaller::biocLite("genefilter")
```

O índice (`idx`) que calculamos é os índices dos genes com a maior variância. Cada amostra tem uma cor designada basedo no tipo de tecido.

```
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

Apesar de R base tem uma função para criar um heatmap, o pacote `gplots` tem uma função que é mais flexível e mais elaborada: `heatmap.2`. Você precisaria download esse pacote.

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

O heatmap em si mostra que para muitos dos tecidos, esses genes podem diferenciar as amostras em geral corretamente para mostrar qual amostra foi tirado de qual tipo de tecido. Também na margem superior, é a barra que mostra os tecidos. A margem direta mostra os nomes dos 40 genes que estamos usando para este gráfioc e os nomes borrados no fundo são os nomes dos tecidos.

Conclusão

Além dos heatmaps, análise de clusters tem muito utilidade como uma técnica de machine learning. Já falei de uso nas análises exploratórias. E também estão muito úteis para análises em que você está procurando características próximas das celulas, tecidos, genes ou até fatores imunologicos.

[6] Nós vamos usar outras pacotes de Bioconductor na última semana da matéria. Então o download de Bioconductor vale a pena.

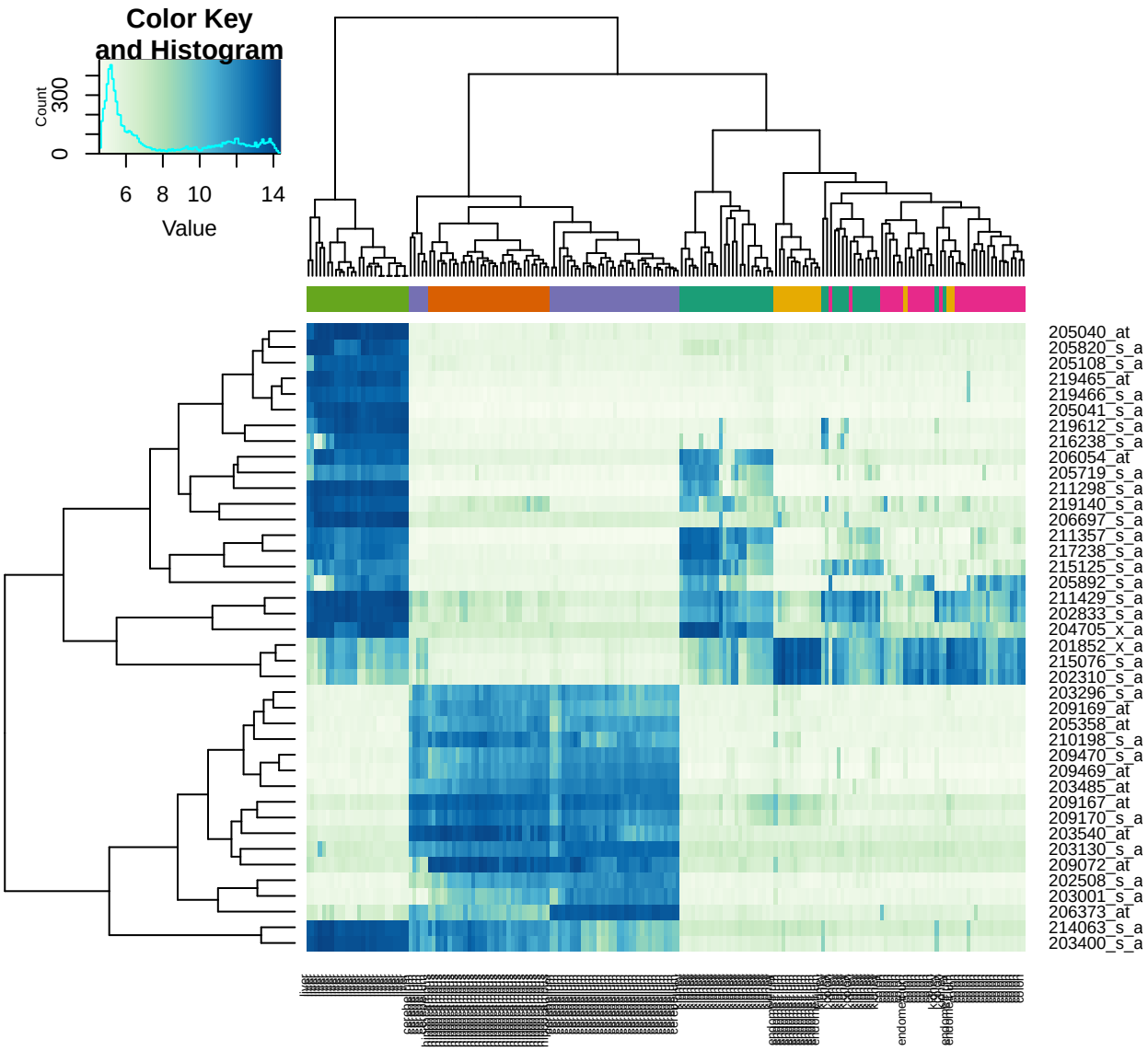


Figure 2: Heatmap created using the 40 most variable genes and the function heatmap.2.