R / Bioconductor: Curso Intensivo

Leonardo Collado Torres
Licenciatura en Ciencias Genómicas, UNAM
www.lcg.unam.mx/~lcollado/index.php

Cuernavaca, México
Oct-Nov, 2008

Nuestro problema a reproducir

- Para empezar la clase de hoy vamos a continuar un poco sobre la línea de microarreglos.
- Vamos a usar el paquete ALL que contiene la información para nuestro ejercicio.
- Para entender un poco más, lean el abstract del artículo del cual vienen los datos y nuestro objetivo es reproducir la figura 2 de este otro artículo[1].

[1] Si quieren aprender más sobre los orígenes de Bioconductor pueden leer u hojear este artículo :)

Explorando la info

- Cargemos nuestra información y explorenla :)

```
> library("ALL")
> data(ALL)

> ALL
```

- Por ejemplo, podemos ver los resultados de las 128 muestras para una prueba molecular.

```
> ALL$mol.biol[1:10]

 [1] BCR/ABL  NEG      BCR/ABL  ALL1/AF4
 [5] NEG      NEG      NEG      NEG
 [9] NEG      BCR/ABL
6 Levels: ALL1/AF4 ... p15/p16
```

Datos para la figura

- En nuestra práctica solo nos interesan los que tienen una translocación entre los cromosomas 9 y 22 (BCR/ABL) o entre los cromosomas 4 y 11 (ALL11/AF4). Así que filtramos nuestros datos:

```
> eset <- ALL[, ALL$mol.biol %in%
+     c("BCR/ABL", "ALL1/AF4")]
```

- Nos quedamos con 47 muestras donde cada una tiene información de 12 625 genes. Es demasiada información para correr un `heatmap`.

Genes Diferencialmente Expresados

- Ahora usamos a limma para encontrar los genes diferencialmente expresados.

```
> library(limma)
> f <- factor(as.character(eset$mol.biol))
> design <- model.matrix(~f)
> fit <- eBayes(lmFit(eset, design))
```

Figura 1

- Ya con esta información podemos reproducir la figura 1 del artículo

```
> topTable(fit, coef = 2)
```

	ID	logFC	AveExpr
1016	1914_at	-3.076231	4.611284
7884	37809_at	-3.971906	4.864721
6939	36873_at	-3.391662	4.284529
10865	40763_at	-3.086992	3.474092
4250	34210_at	3.618194	8.438482
11556	41448_at	-2.500488	3.733012
3389	33358_at	-2.269730	5.191015
8054	37978_at	-1.036051	6.937965
10579	40480_s_at	1.844998	7.826900
330	1307_at	1.583904	4.638885

Figura 1

	t	P.Value
1016	-27.49860	4.663431e-31
7884	-19.75478	1.033405e-24
6939	-19.61497	1.401149e-24
10865	-17.00739	5.695120e-22
4250	15.45655	2.809128e-20
11556	-14.83924	1.428253e-19
3389	-12.96398	2.638314e-17
8054	-10.48777	5.126800e-14
10579	10.38214	7.206177e-14
330	10.25731	1.079482e-13

	adj.P.Val	B
1016	5.887581e-27	56.32653
7884	5.896502e-21	44.23832
6939	5.896502e-21	43.97298

Figura 1


```
10865 1.797522e-18 38.64615
4250  7.093049e-17 35.10692
11556 3.005283e-16 33.61391
3389  4.758388e-14 28.76471
8054  8.090730e-11 21.60216
10579 1.010866e-10 21.27732
330   1.362846e-10 20.89145
```

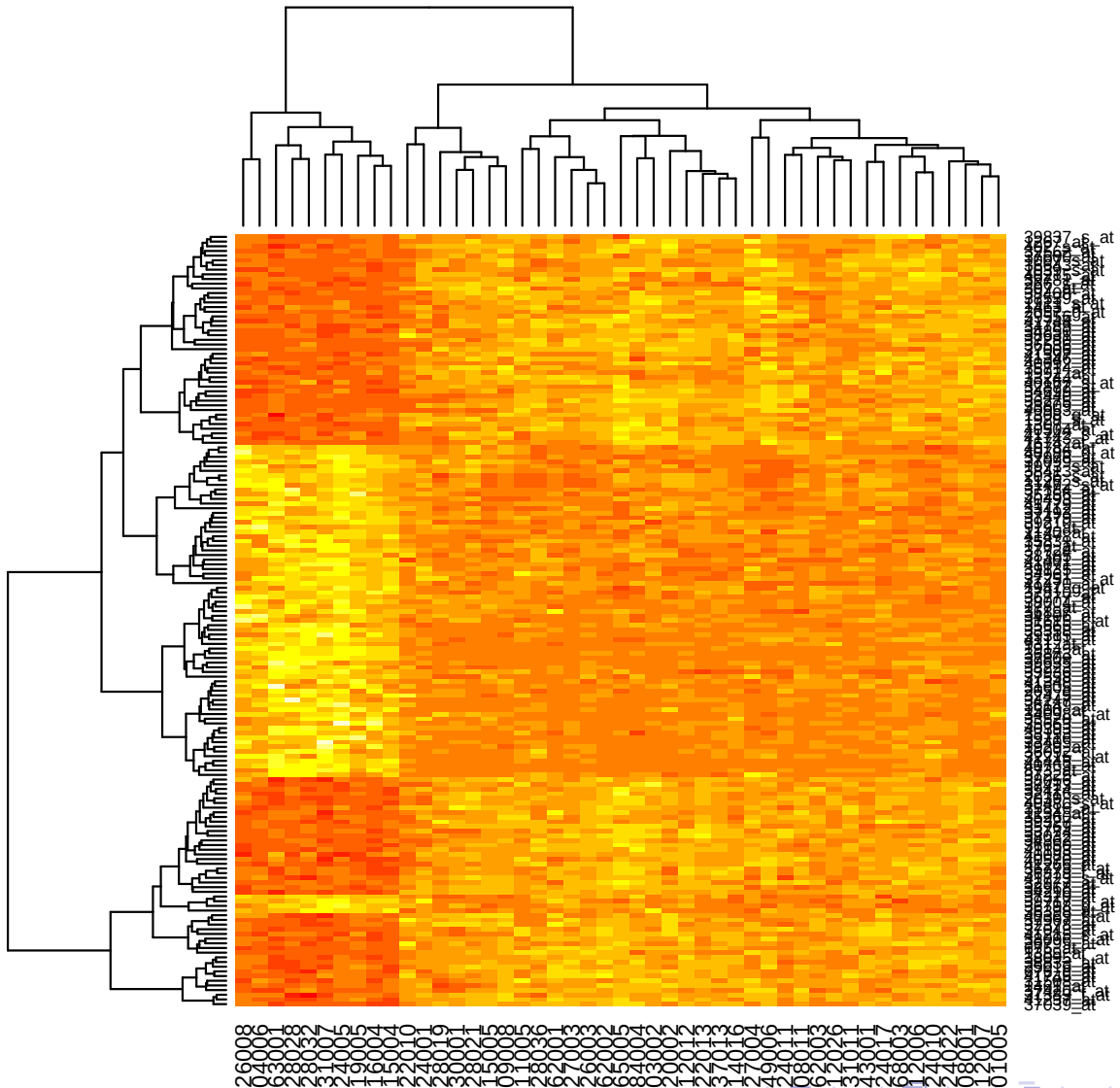
Seleccionamos nuestros genes

- Solo queremos los genes con un valor p menor a 0.05.
- Nos quedamos con los 165 genes que cumplen esto y hacemos el heatmap :)

```
> selected <- p.adjust(fit$p.value[,
+     2]) < 0.05
> esetSel <- eset[selected, ]

> heatmap(exprs(esetSel))
```

heatmap básico



heatmap: arreglamos los colores

- Ahora personalizamos la gráfica para que se parezca más a la del artículo.
- Primero, usamos los mismos colores:

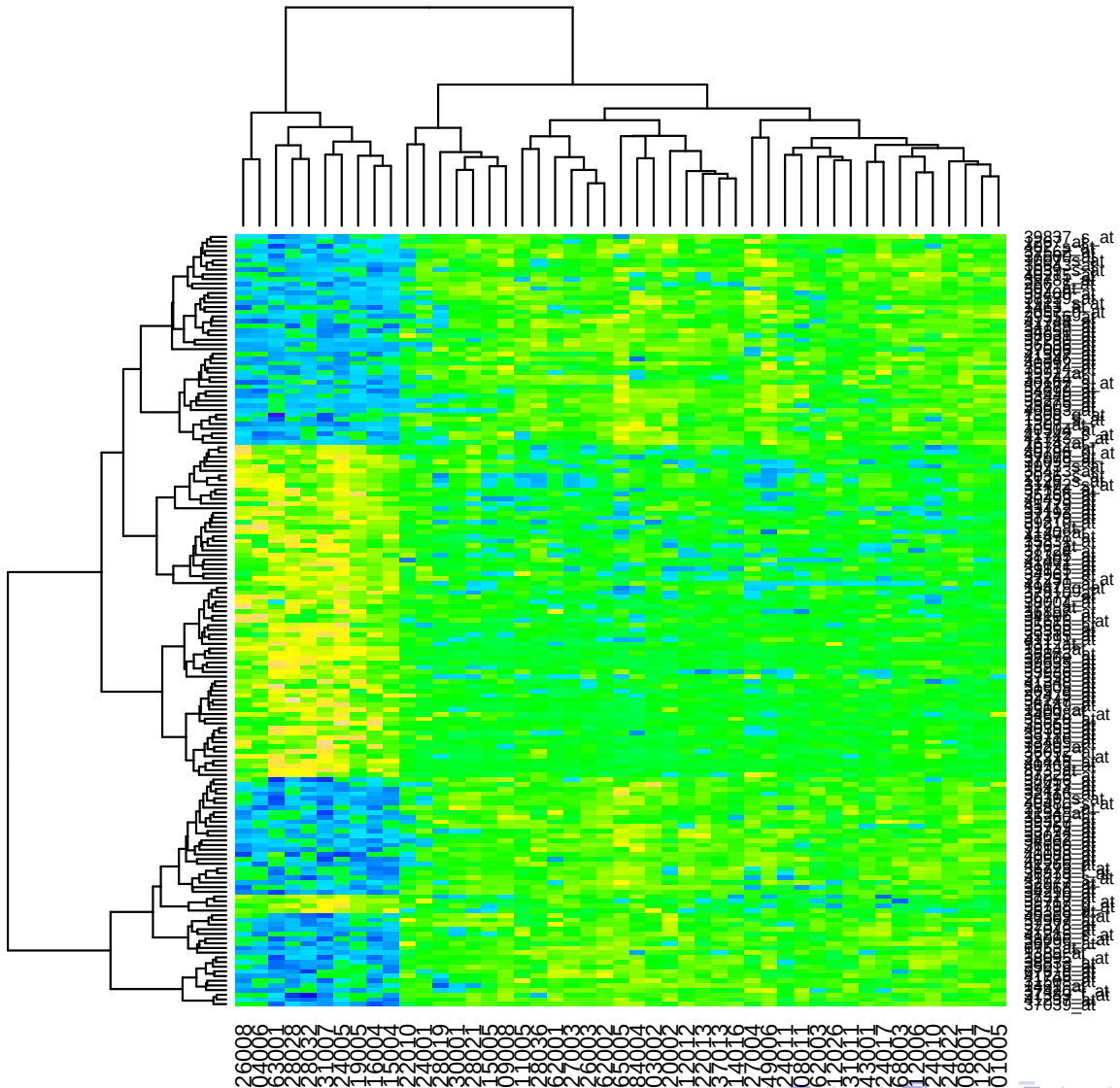
```
> heatmap(exprs(esetSel), col = topo.colors(100))
```

- Sin embargo, a esta gráfica le falta la barrita roja con azul. En el artículo la usan para enfatizar una diferencia entre 10 pacientes y los otros 37.
- Para hacer la barrita usamos el argumento ColSideColors y creamos una función para mapear la información de la prueba molecular.

heatmap: arreglamos los colores


```
> color.map <- function(mol.biol) {
+     if (mol.biol == "ALL1/AF4")
+         "#FF0000"
+     else "#0000FF"
+ }
> patientcolors <- unlist(lapply(esetSel$mol.bio,
+     color.map))
```

heatmap: con colores similares

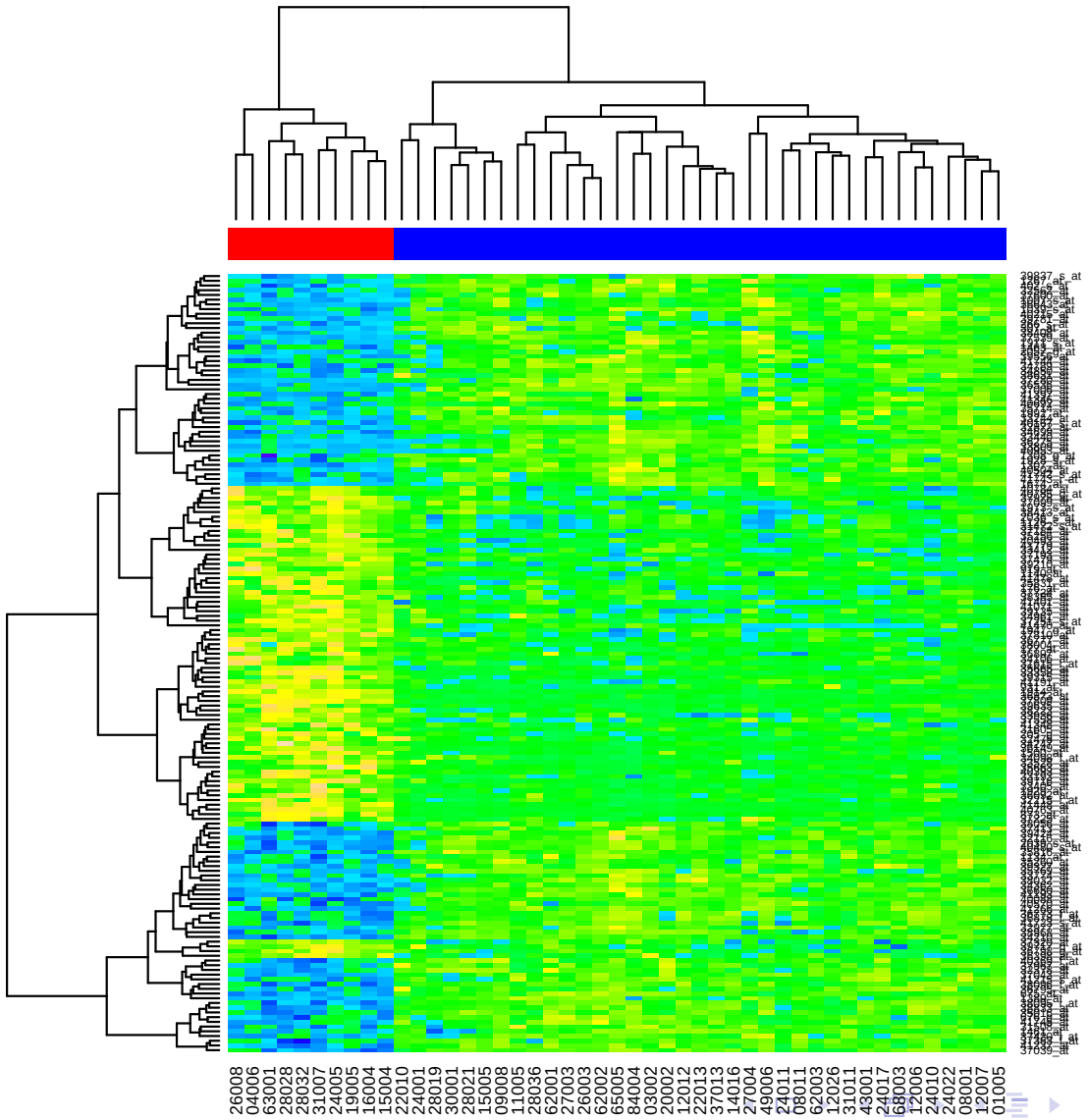


Agregamos la barra

- Ya con nuestra función, hacemos una gráfica MUY similar a la del artículo.
- Para que los nombres no se vean tan apachurrados, usamos el argumento cexRow.

```
> heatmap(exprs(esetSel), col = topo.colors(100),
+     ColSideColors = patientcolors,
+     cexRow = 0.5)
```

heatmap prácticamente idéntico



Créditos

- Listo!
- En sí el ejercicio lo tomé de Peter Cock: Using R to draw a Heatmap from Microarray Data.
- Me pareció sencillo e interesante :). Allí les explica de la función heatmap.2 del paquete gplots con la cual pueden manipular más a su `heatmap`. Por ejemplo, le pueden añadir una leyenda explicando la relación entre los valores y los colores.

Definición

- Hacer un *clustering* es el proceso por el cual clasificas objetos en diferentes grupos llamados *clusters* con el fin de que cada grupo comparta un rasgo común. Generalmente agrupas tus objetos dada una medida de distancia.
- El *clustering* de datos es muy usado en análisis estadísticos, en campos como el *machine learning*, *data mining*, reconocimiento de patrones, análisis de imágenes y pues en la bioinformática :D.
- Hay muchos métodos y formas de agrupar tus datos. Usamos `R` por su eficiencia para manejar estructuras de datos y funciónes para el clustering, por los ambientes eficientes que ofrece para probar algoritmos y por la cantidad de paquetes y funciones relacionadas disponibles.

Info de apoyo

- En sí, el *clustering* es muy complejo ya que el método a usar depende de tus datos y el problema que buscas resolver.
- En esta página pueden encontrar una lista de diferentes paquetes en `R` relacionados al clustering con breves explicaciones.
- Si les interesa esto del *machine learning*, chequen esta página homóloga.
- En fin, wiki también puede serles útil.

Transformaciones de Datos

- Bueno, antes de usar los métodos de clustering, hay que decidir si transformamos nuestros datos previamente o no.
- Centrar y estandarizar:
  - Substraes la media a cada dato.
  - Divides tus datos por la desviacion estándar.
  - Tus datos tendrán media 0 y desviación estándar 1.
- Centrar y escalar tus datos usando scale.
  - Substraes la media a cada dato.
  - Divides tus datos por la raíz de la media cuadrada.
  - Tus datos tendrán media 0 y desviación estándar 1.
- Transformar con `log`.
- Cambiar los valores de tus datos por su *rank*.
- Sin transformar :)

Calcular la distancia

- El siguiente paso es escoger el método de distancia a usar. Hay muchos y cada uno tiene ventajas y desventajas.
- La opción básica es el método Euclidiano. Esta ya la conocen desde la prepa aunque noten que no sirve para correlaciones negativas y no es invariante de escala.
- Algo que nos dejó muy en claro Arturo Medrano es que no importa que uses, vas a recuperar cluster. Pero son los buenos? Pues el problema principal es escoger bien que usar de acuerdo a tus datos.
- Hay dos distancias basadas en correlación: la Pearson y la Spearman. Su problema principal es que es sensible a los *outliers*.

Calcular la distancia

- En fin, otras distancias son la binaria, Manhattan, Máxima, Minowski, etc. En R pueden obtener más información con `?dist`.

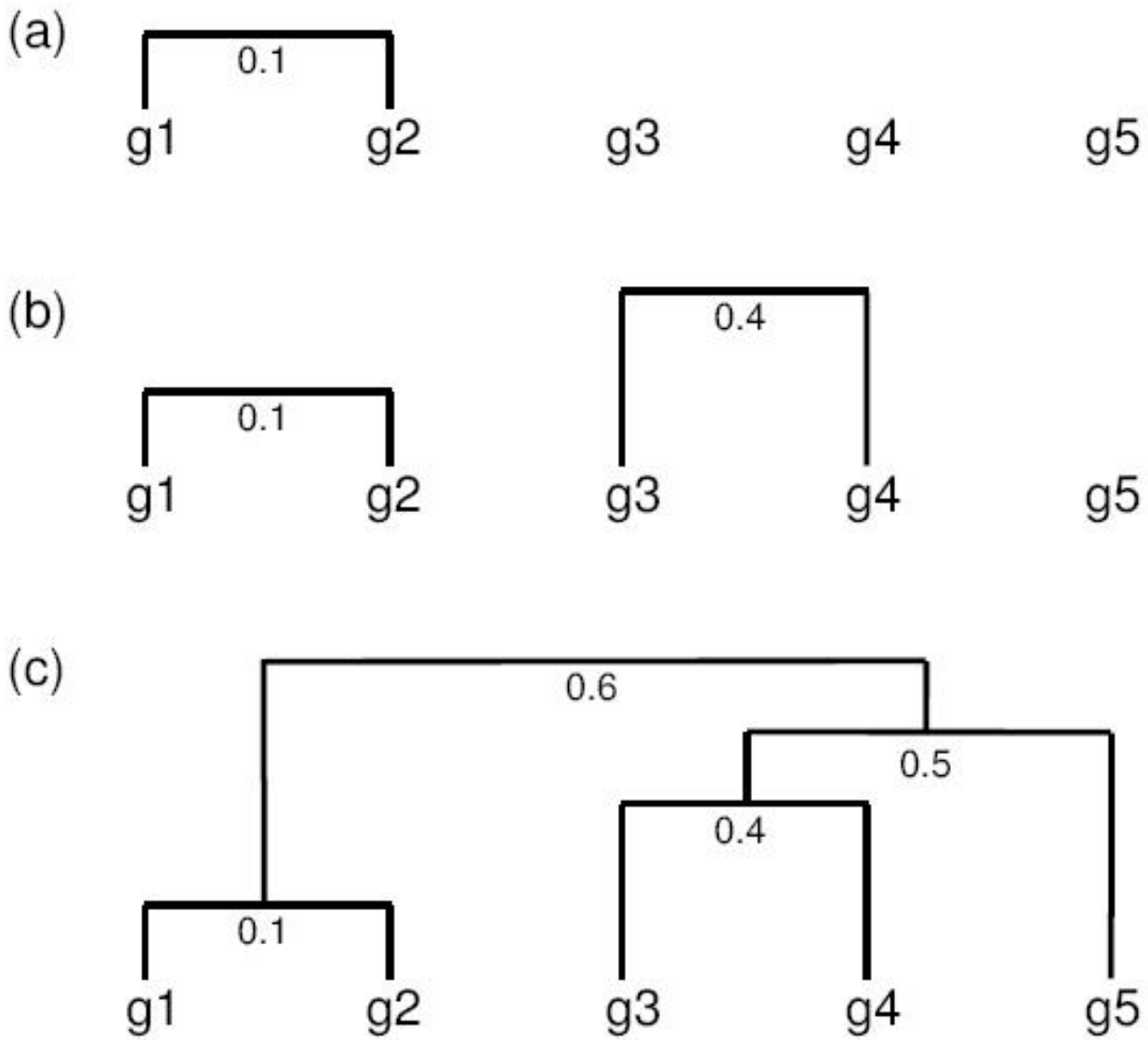
Métodos de clustering

- Ya entrando al clustering, hay que diferenciar los métodos. Básicamente se dividen en:
  - Clustering jerarquíco: aglomerativo, divisorio.
  - Clustering no jerarquíco: k-means, PCA: *principal component analysis*, etc.

Clustering jerarquíco

- La idea básica del clustering aglomerativo es la siguiente:
  - Identificar los clusters con la menor distancia
  - Unirlos a los nuevos clusters
  - Calcular la distancia entre clusters
  - Regresar al primer paso hasta que tengas un solo cluster con todos tus datos

Jerárquico y aglomerativo



Funciones en R

- En fin, en R podemos usar las funciones hclust y agnes si queremos ir de abajo hacia arriba.
- Para ir en el sentido contrario, está la función diana.

No jerárquico

- De los métodos de clustering no jerárquico, el más usado es el k-means. Este funciona así:
  - Escoger un número k de clusters
  - Asigna al azar los datos a los k clusters
  - Calcula un nuevo centroide para cada uno de los k clusters
  - Calcula la distancia entre todos los datos hacia los k centroides
  - Asigna cada dato al centroide más cercano
  - Repite el proceso hasta que las asignaciones sean estables

Funciones en R

- En R podemos hacer el k-means con la función kmeans del paquete Stats.
- Otras opción algo similar es con la función pam.
- El PCA se hace con la función prcomp.
- En fin, hay muchos paquetes disponibles para hacer tipos de clusters.

Un ejercicio simple

- Vamos a hacer un ejercicio simple con datos de GEO.
- Pueden bajar el archivo `GSE1110clean.txt` o simplemente usar el siguiente comando[2]:

```
> mydata <- read.delim("http://www.lcg.unam.mx/~l
+     header = T, sep = "\t")
```

- Hay que volver nuestro objeto una matriz y añadir bien los nombres de las líneas.

```
> rownames(mydata) <- mydata[, 1]
> mydata <- as.matrix(mydata[, -1])
```

[2] Acuérdense de que los códigos están en la página del curso

Filtrando nuestros datos

- Si se fijan, `mydata` tiene 22 810 líneas y 22 columnas; no podemos hacer un clustering de este tamaño, pues `R` nos va a protestar por limitacions de memoria. Vamos a filtrar nuestra información para quedarnos solo con las líneas de intensidad alta o de alta variabilidad.

```
> mydata <- mydata[apply(mydata >
+     100, 1, sum)/length(mydata[1,
+     ]) > 0.5 & apply(log2(mydata),
+     1, IQR) > 1.5, ]
```

- Tgirke, quien creó este ejercicio, hizo una función especial para escoger colores, así que leemos su código con la función source.

```
> source("http://faculty.ucr.edu/~tgirke/Documents
```

El clustering como tal

- Ahora tenemos que centrar y escalar nuestros datos.

```
> mydatascale <- t(scale(t(mydata)))
```

- Vamos a hacer clusters por correlación de Pearson para nuestras líneas y por Spearman para nuestras columnas.

```
> hr <- hclust(as.dist(1 - cor(t(mydatascale),
+     method = "pearson")), method = "complete")
> hc <- hclust(as.dist(1 - cor(mydatascale,
+     method = "spearman")), method = "complete")
```

- Si se fijan, usamos as.dist para interpretar los resultados de nuestras correlaciones como distancias. Pues ese el tipo de objeto que necesita hclust como entrada.

Visualizando clusters!

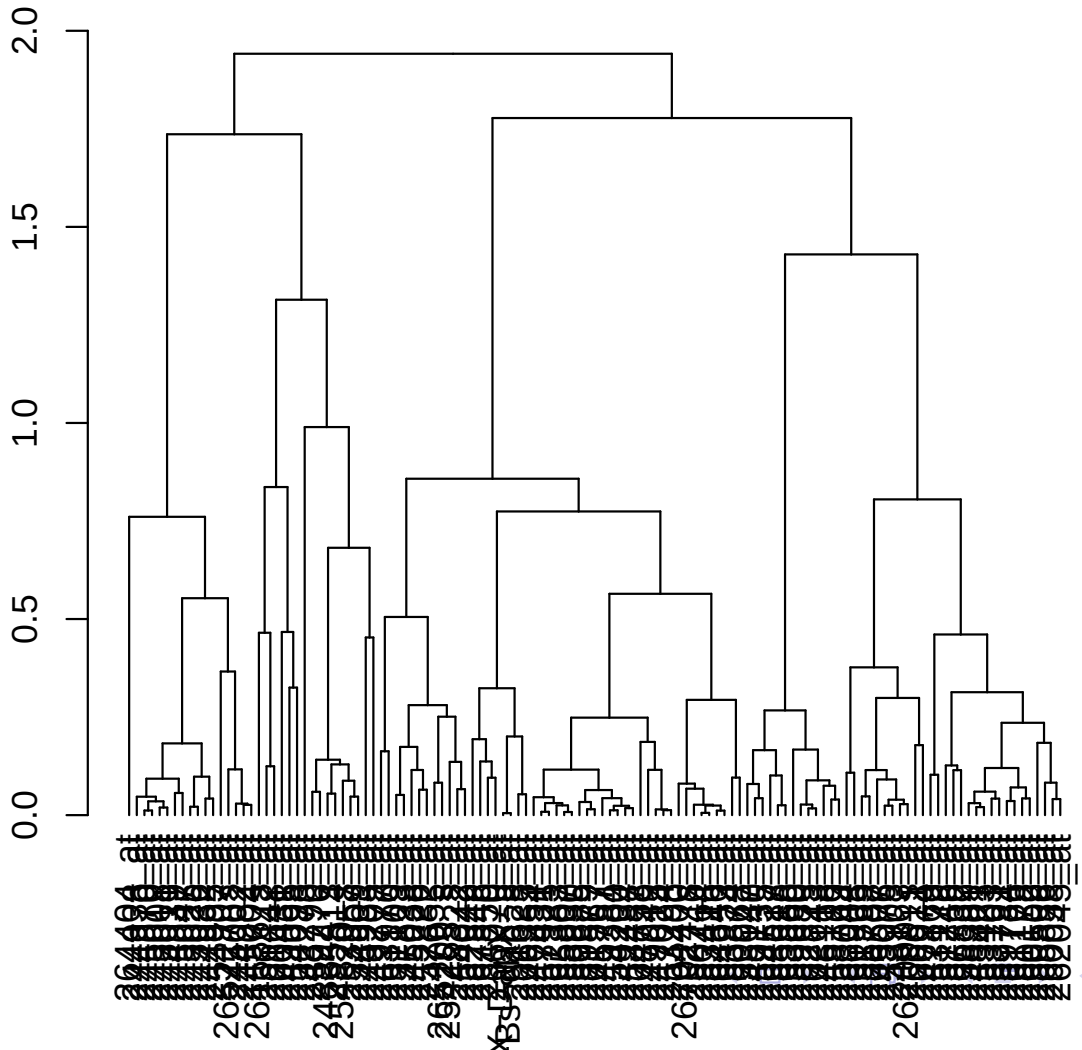
- Hay varias formas de visualizar los resultados de un clustering. La más común es por dendogramas usando plot, o si tienes 2, con un heatmap.

```
> plot(as.dendrogram(hr))
> plot(as.dendrogram(hc))
> heatmap(mydata, Rowv = as.dendrogram(hr),
+         Colv = as.dendrogram(hc), col = my.colorFct
+         scale = "row")
```

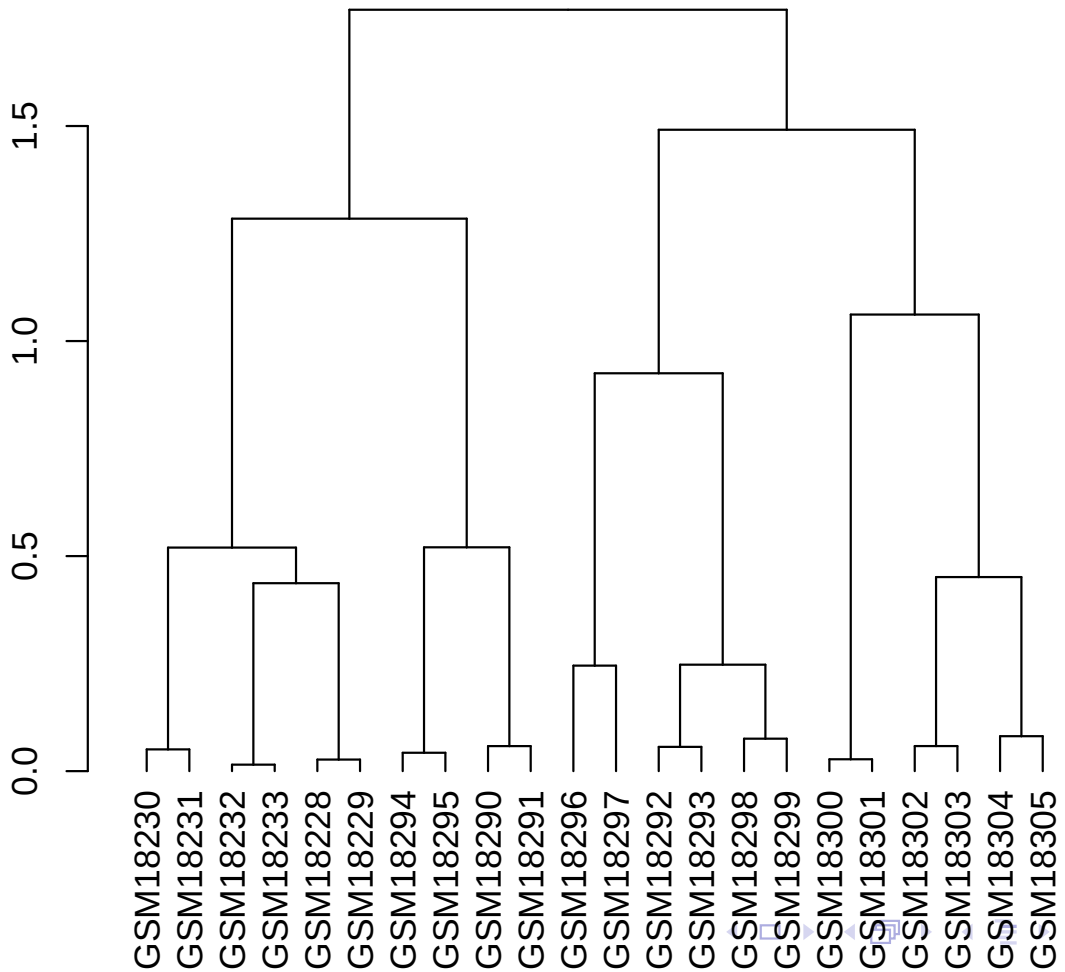
- Acuérdense de que los resultados de funciones como hclust son objetos con muchos attributos. Chequenlos con la función attributes

```
> attributes(hr)
```

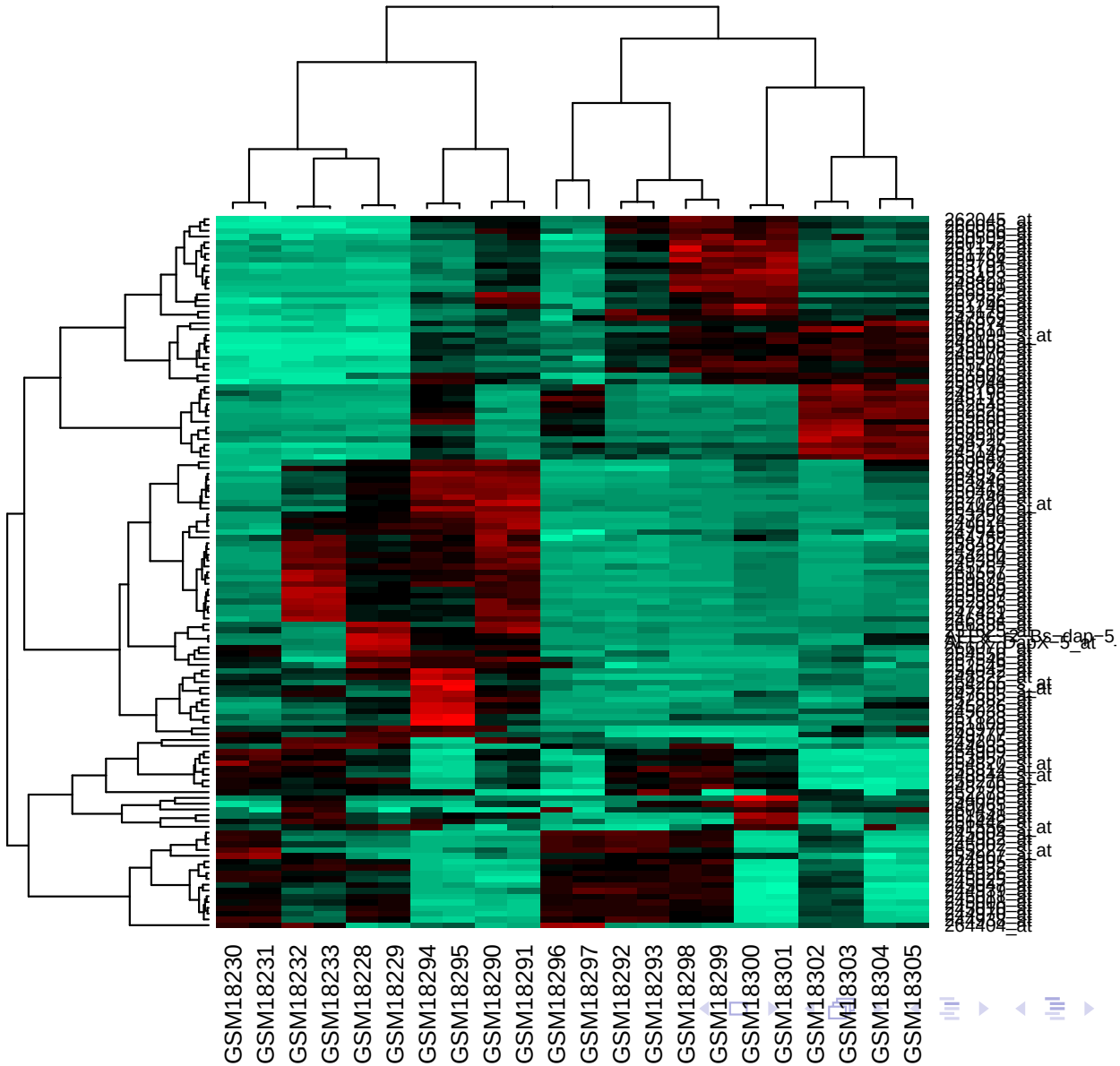
Dendograma: hr



Dendograma: hc



heatmap


GSM18230
GSM18231
GSM18232
GSM18233
GSM18228
GSM18229
GSM18294
GSM18295
GSM18290
GSM18291
GSM18296
GSM18297
GSM18292
GSM18293
GSM18298
GSM18299
GSM18300
GSM18301
GSM18302
GSM18303
GSM18304
GSM18305

Creando nuestro heatmap final!

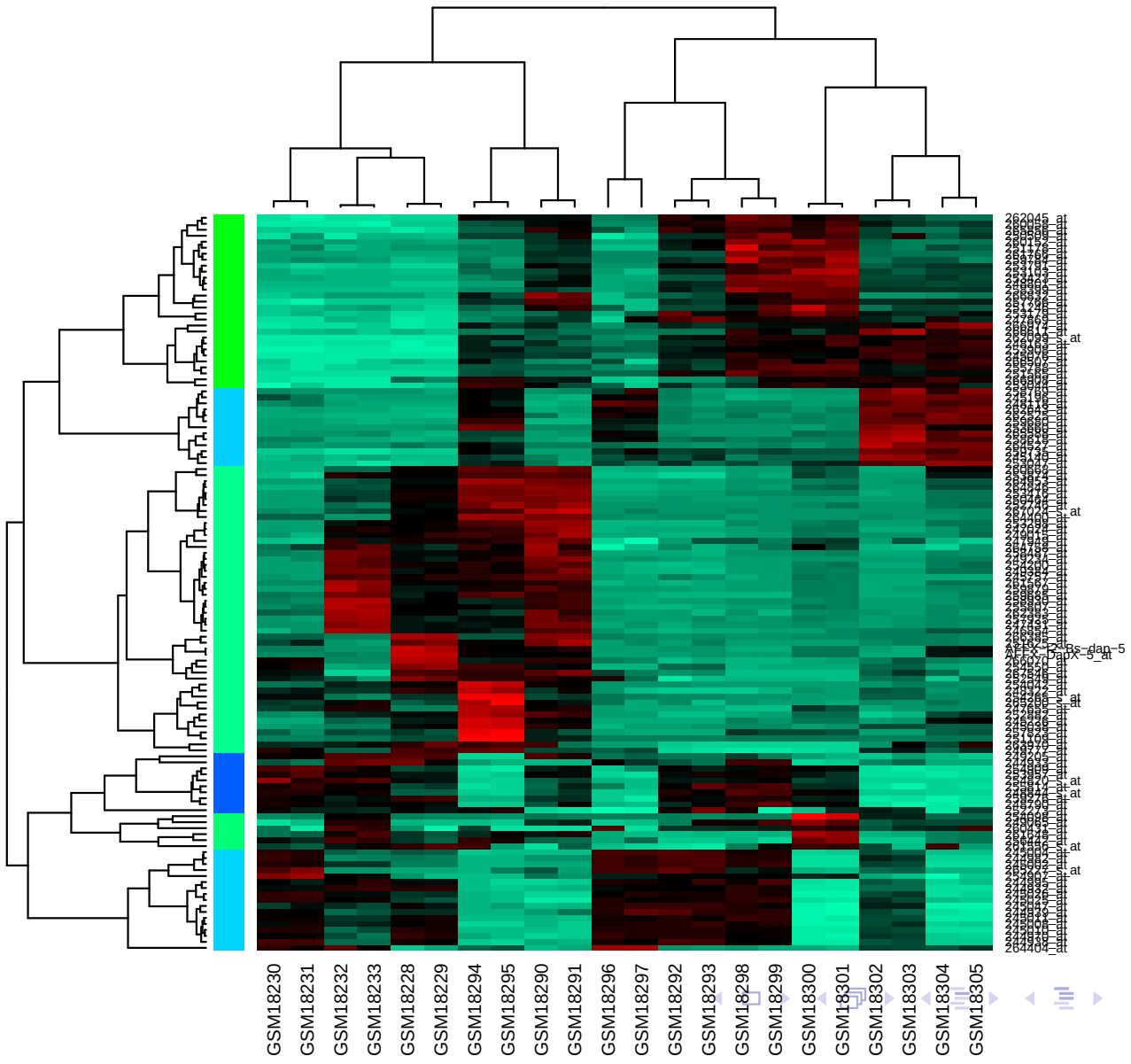
- Ahora vamos a cortar el árbol a una altura específica y le pondremos una barrita con colores del lado izquierdo para ver los diferentes clusters.

```
> mycl <- cutree(hr, h = max(hr$height)/1.5)
> mycolhc <- sample(rainbow(256))
> mycolhc <- mycolhc[as.vector(mycl)]
```

- Ya con nuestra barrita preparada, hacemos el `heatmap` final:

```
> heatmap(mydata, Rowv = as.dendrogram(hr),
+         Colv = as.dendrogram(hc), col = my.colorFct
+         scale = "row", RowSideColors = mycolhc)
```

Heatmap final



Fin Clustering

- Listo! Ya saben lo más básico del clustering. Claro, una cosa es encontrar los grupos y la otra es encontrar una explicación biológica a dichos grupos.
- Sé que algunos tendrán curiosidad de aprender más al respecto, así que los invito a seguir esta página.
- Por ejemplo, acabamos de hacer solo el principio de la sección de ejercicios.

Ligando R con otros lenguajes

- Ya vimos en la clase pasada como conectarnos a MySQL desde R. También puedes juntar a R con:
  - Excel, al hacer archivos separados por tab o comas.
  - Perl, usando llamadas al sistema o con un paquete como el RSPerl.
  - Python, usando RPy. En sí Python no lo hemos visto en la LCG, pero sé que existe un repositorio similar a BioPerl y Bioconductor llamado BioPython.
  - Con páginas HTML usando Rpad.
  - En fin, en este link que es parte de los links del material de apoyo pueden encontrar más información al respecto.

Fin del Curso!

- Tareas de R como tal ya no tendrán este semestre, aunque nos vemos la próxima semana para checar sus avances de proyecto en la parte de R y dentro de 2 semanas para evaluar su proyecto completo.
- Espero que les haya gustado el curso y que si usen R[3]. El próximo semestre verán R avanzado con Alejandra Medina :) Me enseñan pls!
- Les recuerdo que luego ustedes tendrán que participar en la impartición de cursos similares a las siguientes generaciones. Además, puede que en su laboratorio no muchos o nadie sepa de R, así que les deseo suerte en esta siguiente etapa de compartir sus conocimientos.

[3] Por favor! Ya no hagan gráficas de pie!!! :P