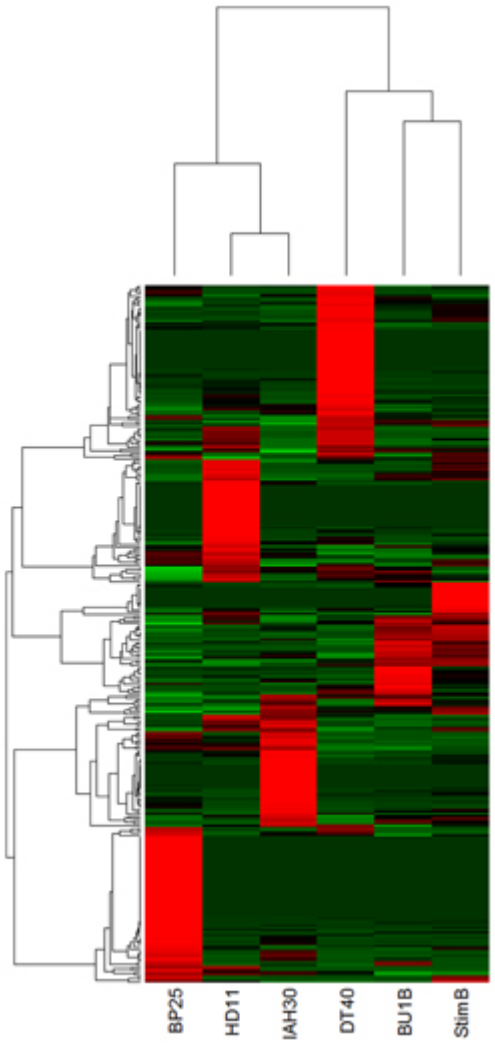
You probably don’t understand heatmaps

5TH APRIL 2015 / BIOMICKWATSON / 10 COMMENTS

If you work in any area of quantitative biology, and especially if you work with transcriptomic data, then you are probably familiar with heatmaps – used for as long as I have been in research, these figures cluster rows and columns of a data matrix, and show both dendrograms alongside a colour-scaled representation of the data matrix itself.

See an example from one of my publications below:



Pretty simple huh? Well actually, no, they're not, and unless you're a statistician or bioinformatician, **you probably don't understand how they work** 😉

There are two complexities to heatmaps – first, how the clustering itself works (i.e. how the trees are calculated and drawn); and second, how the data matrix is converted into a colour-scale image.

Clustering data

I don't really have time to explain cluster analysis, which actually refers to a huge range of methods. I can explain the most simple method though, which is hierarchical, agglomerative cluster analysis. In a nutshell, this works by first calculating the pairwise distance between all data points; it then joins the data points that are the least distant apart; then it joins the next least distant pair of points; etc etc until it has joined all points. The tree is a graphical representation of this process. At some point the process needs to join groups of points together, and

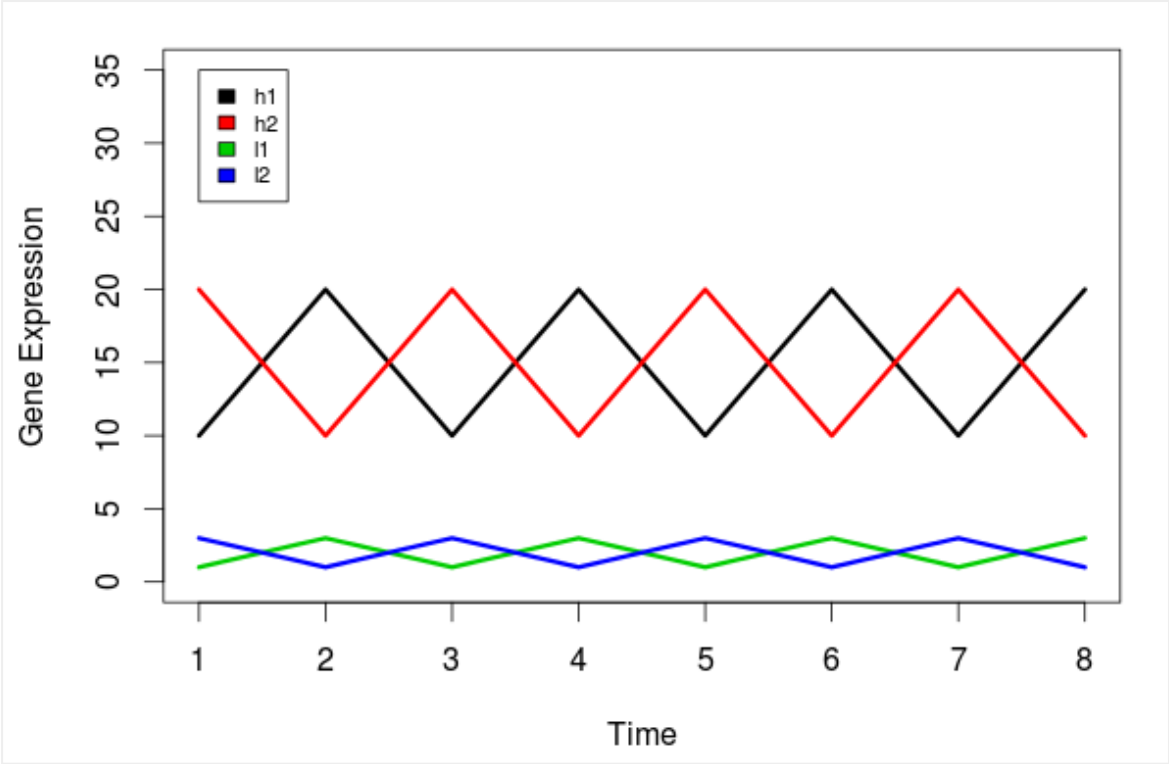
again there are many methods, but one of the most common method is to calculate the average pairwise distance between all pairs of points in the two groups.

Know your distance measure

Put simply, the distance measure is how *different* two data points are. It is orthogonal to the similarity measure, which measures how *similar* two data points are.

So how do we calculate distance? WIth the default methods for both the heatmap() and heatmap.2() functions in R, the distance measure is calculated using the dist() function, whose own default is euclidean distance. This measures the absolute distance between the points in space, and quite importantly, pays no attention to the "shape" of the "curve". To explain this, let's use an example. Imagine we have measured the gene expression of 4 genes over 8 time points:

```
h1 <- c(10,20,10,20,10,20,10,20)
h2 <- c(20,10,20,10,20,10,20,10)

l1 <- c(1,3,1,3,1,3,1,3)
l2 <- c(3,1,3,1,3,1,3,1)

mat <- rbind(h1,h2,l1,l2)

par(mar=c(4,4,1,1))
plot(1:8,rep(0,8), ylim=c(0,35), pch="", xlab="Time", ylab="Gene
Expression")

for (i in 1:nrow(mat)) {
lines(1:8,mat[i,], lwd=3, col=i)
}

legend(1,35,rownames(mat), 1:4, cex=0.7)
```



So we have two highly expressed genes and two lowly expressed genes. Crucially for this example, the two pairs of genes (high and low) have very different shapes.

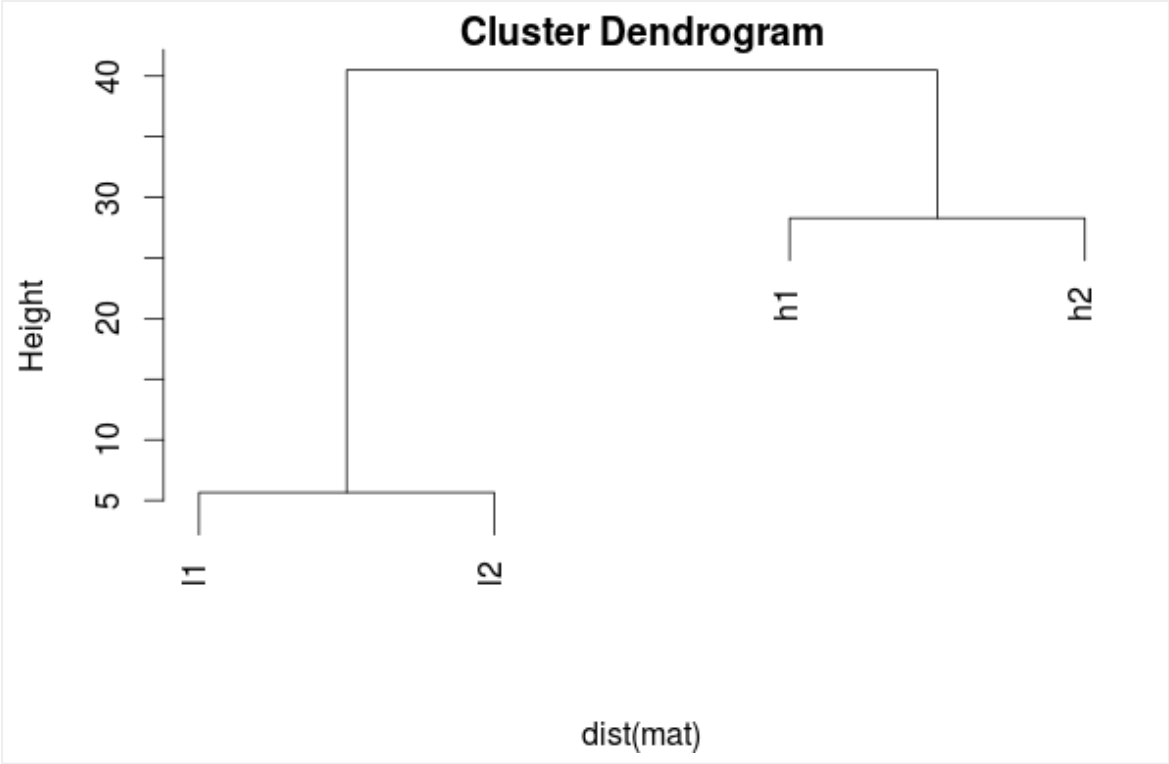
If we input this data into the dist() function, we get a distance matrix:

```
dist(mat)
```

```
          h1        h2        l1
h2 28.284271
l1 38.470768 40.496913
l2 40.496913 38.470768  5.656854
```

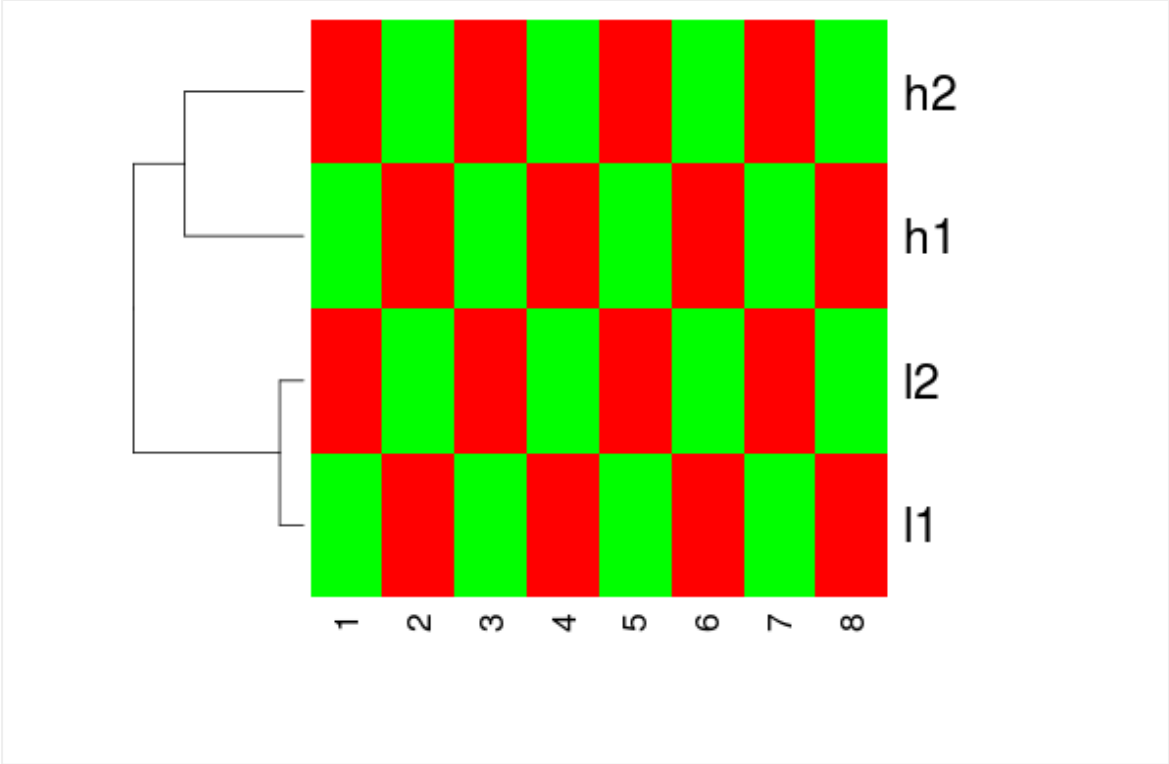
Here we can see that the *least distant* are l1 and l2 (distance = 5.65), so they will be clustered together; next least distant are h1 and h2 (distance = 28.28), so these will be clustered together next. Then finally the two groups will be joined. This is born out by a naive cluster analysis on the distance matrix:

```
hc <- hclust(dist(mat))
plot(hc)
```



and a naive heatmap (I've turned off the column tree as in gene expression profiling over time, we generally want the time points to be in the correct, original order):

```
heatmap(mat, Colv=NA, col=greenred(10))
```



There are multiple things going on here, so let's take this one thing at a time. The clustering has worked exactly as it was supposed to – by distance, l1 and l2 are the

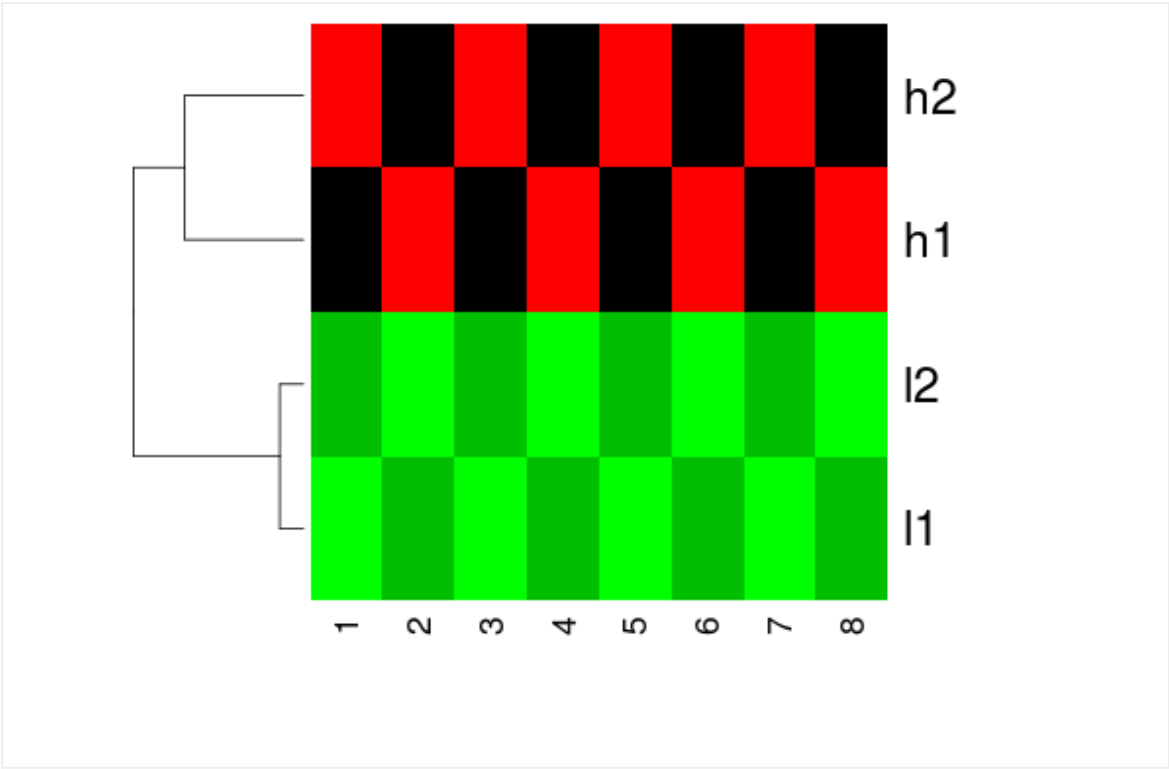
most similar so they are grouped together; then h1 and h2, so they are grouped together. But the heatmap looks terrible, the colours are all wrong. Why? Well, despite l1 and l2 being clustered together, their colours do not follow the same pattern; same goes for h1 and h2. Also, l1 and h1 have the same colours despite having VASTLY different expression profiles; same for l2 and h2.

Understand scaling

Think about the data, and then think about the colours in the heatmap above. Data points l2 and l2 have **exactly the same colours**, as do l1 and h1 – yet they have very different values. That's because the data points are *scaled* prior to being converted to colour, and the default in both heatmap() and heatmap.2() is to scale by row (other options are to scale by column, or no scaling at all). Scaling by row means that each row of the data matrix is taken in turn and given to the scale() function; the scaled data are then converted into colours.

Let's turn off scaling and see what happens. Here is the heatmap clustered by **euclidean distance** with scaling turned off:

```
heatmap(mat, Colv=NA, col=greenred(10), scale="none")
```



Well, this looks slightly better, but still not great! l1 and l2 are at least both green, and h1 and h2 are at least both red/black (though they still oppose one another).

What's happening here? Well, think about a heatmap and what green, red and black mean to you. Green usually refers to low; red usually refers to high; and black is somewhere in the middle. Well, without scaling, both l1 and l2 are "low", so they get given green colours; the highest points of h1 and h2 are "high", so they get red; the low points of h1 and h2 are in the middle, so they get black.

Improving things

Is any of this what you would expect? The answer, I think, is probably no. Usually, in gene expression profiling, we want to cluster together genes that have a similar profile, or similar shape, over time. When we apply a colour scale, as we do in a heatmap, we give low values green, high values red, and middle values black. Of course some genes are highly expressed, and some lowly expressed; do we want to give all highly expressed genes the colour "red", all lowly expressed genes the colour "green", regardless of shape? Well sometimes we might, but most often we don't. This is why the heatmap and heatmap.2 defaults are quite strange to us – they both scale the data by default, which is great if you want to cluster together data points with a similar shape; but they use euclidean distance, which is not what you want to use to cluster things points by shape.

How do we fix this? From the gene expression profiles, we know that h1 and l1 have a similar shape, and h2 and l2 have a similar shape, but dist() doesn't care about shape, it only cares about absolute distance.

How do we cluster on "shape"? Well, we need to use a different distance measure. To do this, we actually start with a similarity measure, the pearson correlation co-efficient. Without going in to too much detail, the pearson correlation coefficient produces a value between -1 and 1; a value of 1 signifies that the shapes of two profiles are identical; a value of -1 signifies that the shapes of two profiles are exactly opposite; and the scale between -1 and 1 represents less or more similar profiles.

The correlation matrix of the above data looks like this:

```
cor(t(mat))
```

```
   h1 h2 l1 l2
h1  1 -1  1 -1
h2 -1  1 -1  1
l1  1 -1  1 -1
l2 -1  1 -1  1
```

We can see that h1 and h2 have a correlation coefficient of -1 and therefore are very dis-similar; same for l1 and l2. However, h1 and l1 and h2 and l2 are perfectly positively correlated! This is what we expect! However, for clustering (and heatmaps) we need a distance measure, not a similarity measure, so we need to subtract all of these values from 1, which gives:

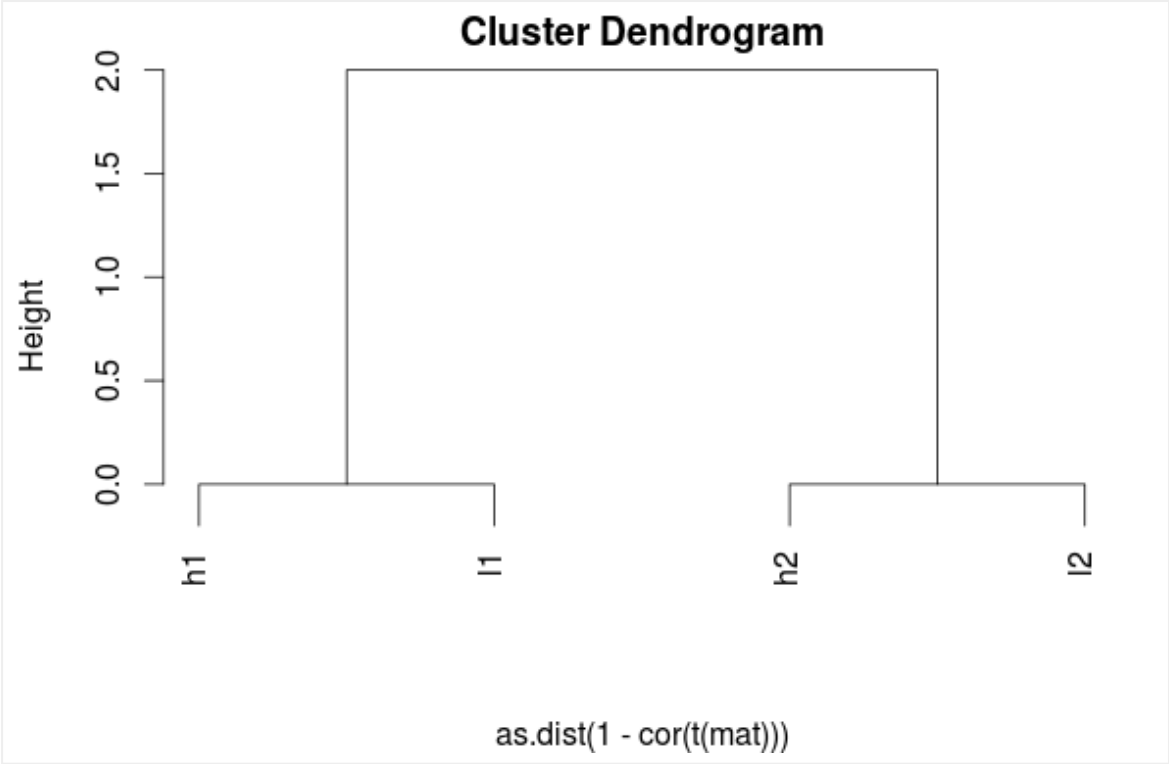
```
1 – cor(t(mat))
```

```
   h1 h2 l1 l2
h1  0  2  0  2
h2  2  0  2  0
l1  0  2  0  2
l2  2  0  2  0
```

Here the distance between h1 and l1 is 0; the distance between h2 and l2 is zero; the distance between h1 and h2 is 2 and the distance between l1 and l2 is also 2.

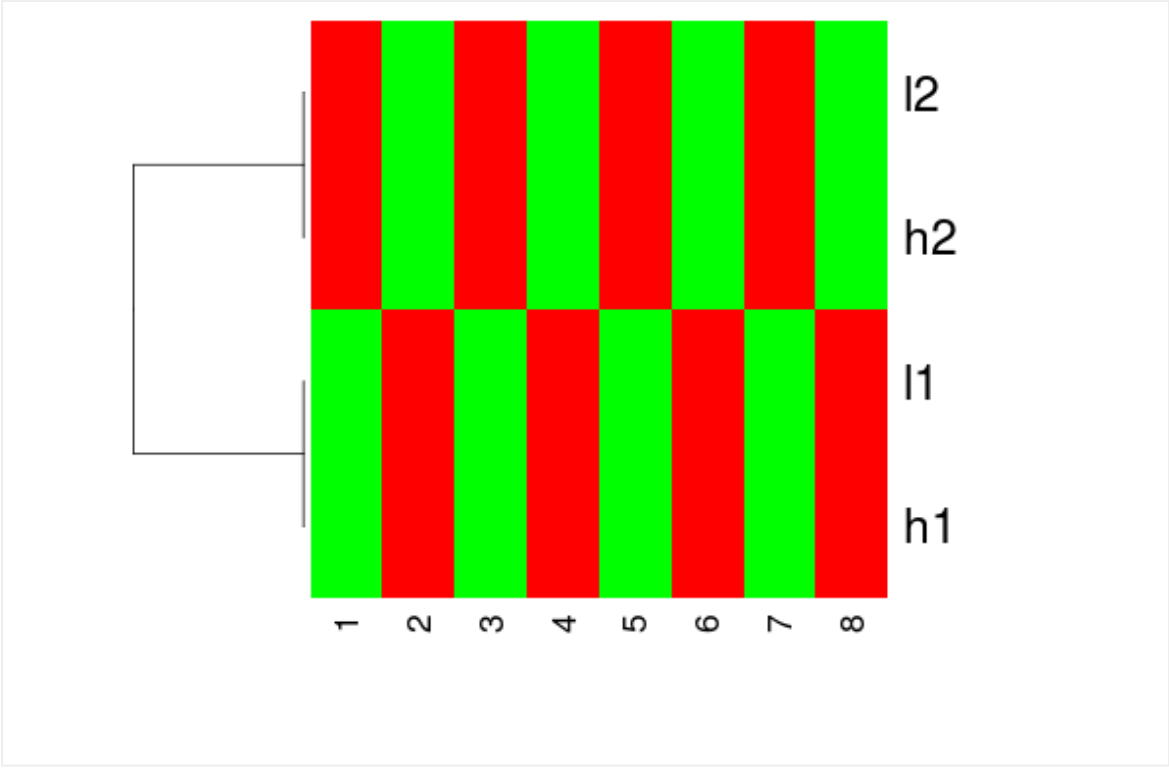
We can draw a naive cluster analysis of this data:

```
hc <- hclust(as.dist(1-cor(t(mat))))
plot(hc)
```



And a simple heatmap:

```
heatmap(mat, Rowv=as.dendrogram(hc), Colv=NA, col=greenred(10))
```



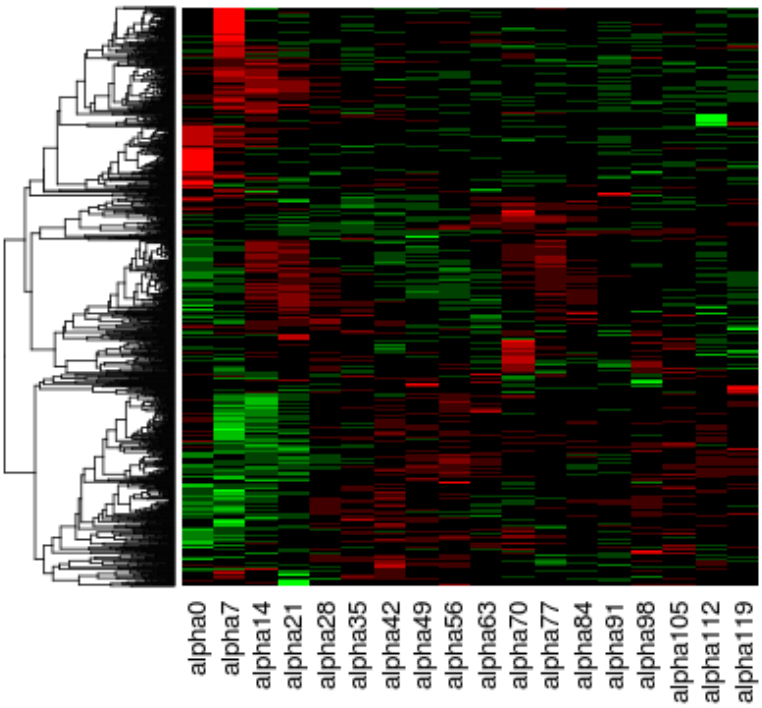
This is what we want – genes clustered on the shape of their expression profile; l1 and h1 clustered together, and they have the same colours; and l2 and h2

clustered together, and they have the same colours. I mean, it still looks awful, but that's because we used a silly example!

A proper heatmap

This example has been on the ARK-Genomics website for some time:

```
library(gplots)

# read the data in from URL
bots <- read.table(url("http://genome-
www.stanford.edu/cellcycle/data/rawdata/combined.txt"), sep="t",
header=TRUE)

# get just the alpha data
abot <- bots[,c(8:25)]
rownames(abot) <- bots[,1]
abot[1:7,]

# get rid of NAs
abot[is.na(abot)] <- 0

# we need to find a way of reducing the data.
# Sort on max difference and take first 1000
min <-apply(abot, 1, min)
max <- apply(abot, 1, max)
sabot <- abot[order(max – min, decreasing=TRUE),][1:1000,]

# cluster on correlation
hc <- hclust(as.dist(1 – cor(t(sabot))), method="average")

# draw a heatmap
heatmap(as.matrix(sabot),
Rowv=as.dendrogram(hc),
Colv=NA,
col=greenred(10),
labRow="")
```



I hope by now that you understand that heatmaps are quite complex visualisations; there are actually quite a few more complexities, but more of that in another post!

bioinformatics

PREVIOUS POST
The cost of sequencing is still going down

NEXT POST
Recreating a famous visualisation

Comments are closed.

RECENT POSTS

Come and do a PhD with us!

Come and work with us

The three technologies bioinformaticians need to be using right now

Massive scale assembly of genomes from metagenomes in ruminants

Voice is dead, will email be far behind?

META

Log in

Entries RSS

WordPress.org

March 2021

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

« Nov