

# **GS01 0163**

## **Analysis of Microarray Data**

Keith Baggerly and Bradley Broom  
Department of Bioinformatics and Computational Biology  
UT M. D. Anderson Cancer Center

`kabagg@mdanderson.org`  
`bmbroom@mdanderson.org`

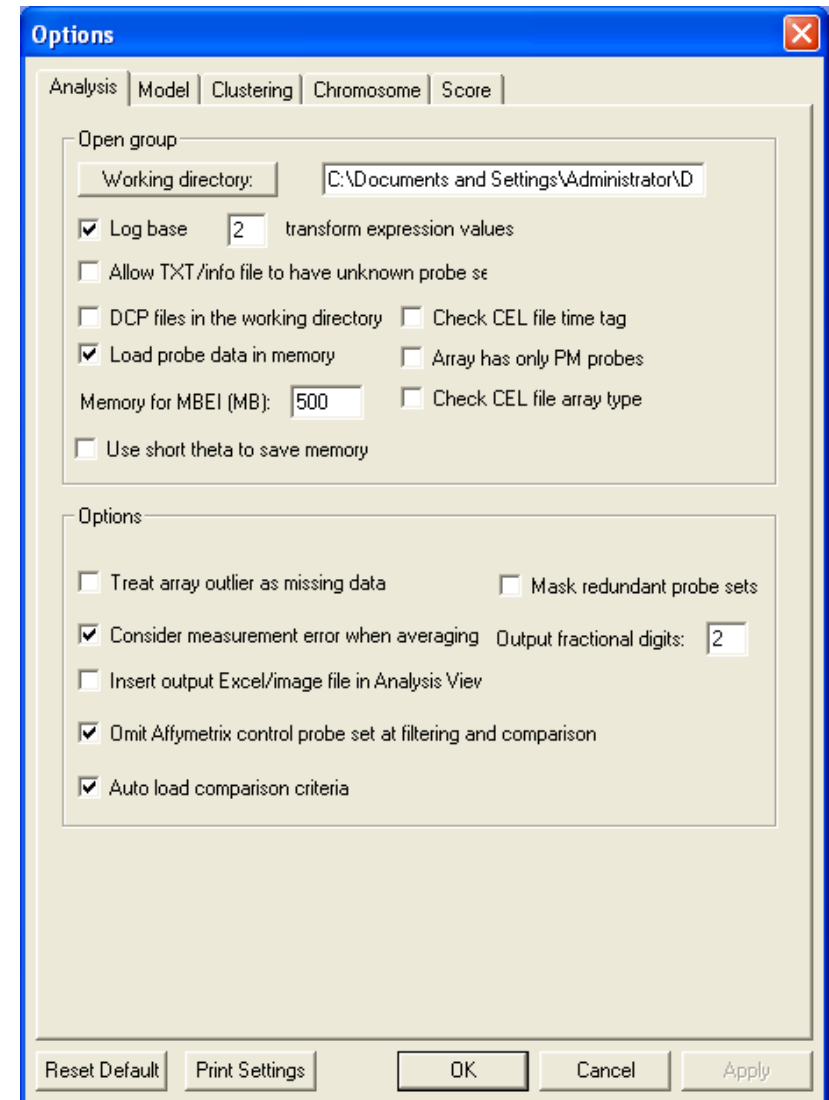
20 October 2009

# Lecture 15: Hierarchical Clustering

- So, why are we here?
- Clustering in dChip
- Measuring distances
- Hierarchical clustering
- When is a cluster valid?
- Clustering with fewer genes
- Simulating something

## Diversion: dChip for TCGA data

- The Affymetrix chip used for the TCGA expression data is the HT\_HG-U133A.
- You will need to download the CDF file for this chip from Affymetrix.
- Due to a bug in the current version of dChip, you must unclick the two “Check CEL file” options.



## So, why are we here?

We want to learn something about clustering microarray data.

It is a well-known fact that clustering was invented by Michael Eisen, Paul Spellman, Pat Brown, and David Botstein in one of the most widely cited papers of all time:

Cluster analysis and display of genome-wide expression patterns. PNAS 1998; 95:14863-14868.

## Digression

1. The ISI lists more than 2900 references to their paper.
2. You can tell they invented clustering, since their paper only has 16 references, 15 of which are to biologists and 1 to a computer scientist (Kohonen, for self-organizing maps). Their erratum, however, does give credit to John Weinstein in 1997 for coloring data matrices after clustering. So maybe Weinstein invented it.
3. They *are* responsible for choosing red-green colormaps, obviously being blithely unconcerned about the fact that this is the most common form of color-blindness.
4. The accuracy of well-known facts should always be questioned.

# Clustering in dChip

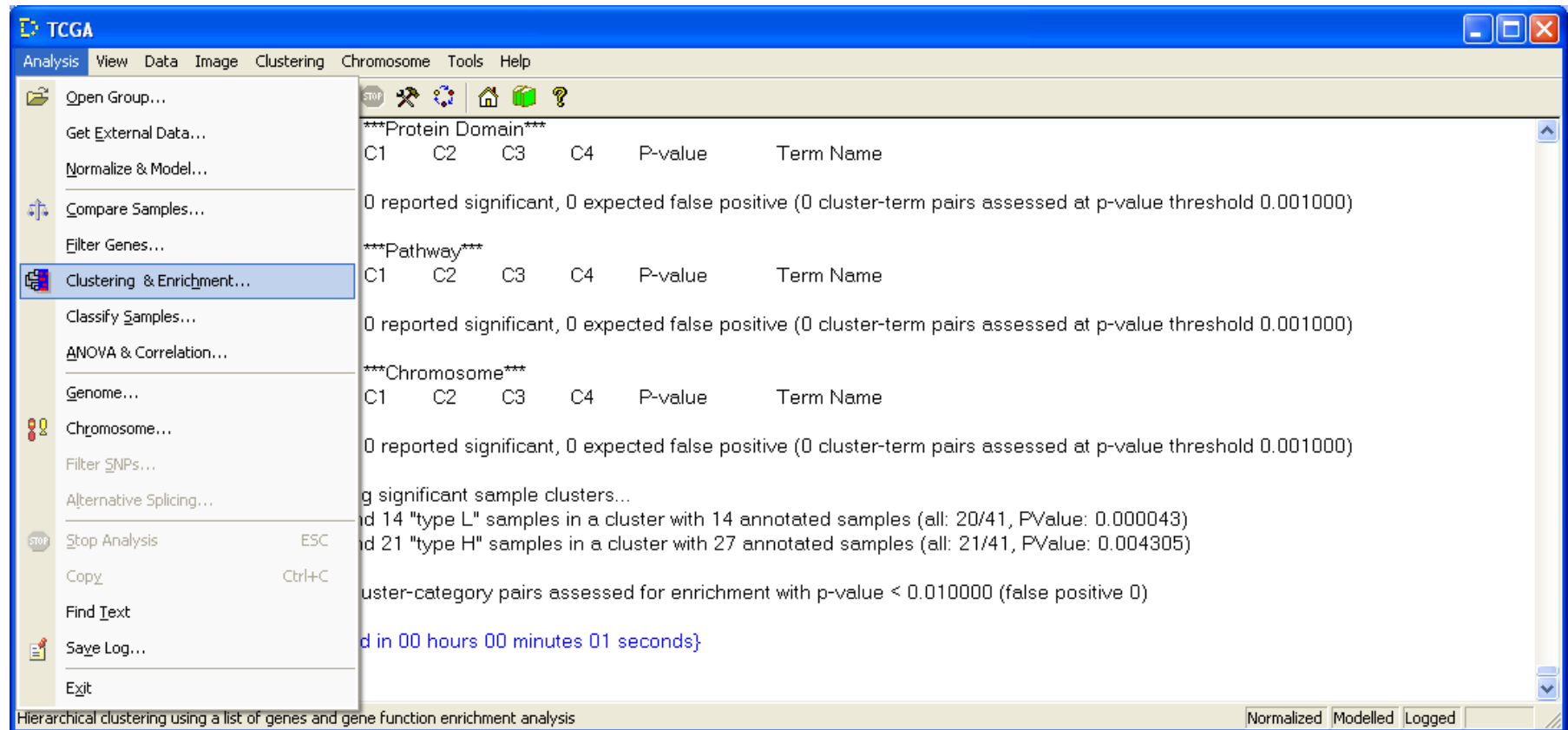
Let's continue with the ALL-MLL example we have been using for a while. Recall that, when last we visited this data set, we had:

1. Performed a comparison in `dChip` that identified 610 differentially expressed genes (604 with `dChip2006`)
2. Tried to find out if any functional categories of genes were over-represented on the list of differentially expressed genes.

Now we'd like to take a different approach to grouping the genes and see which ones have similar profiles across the samples.

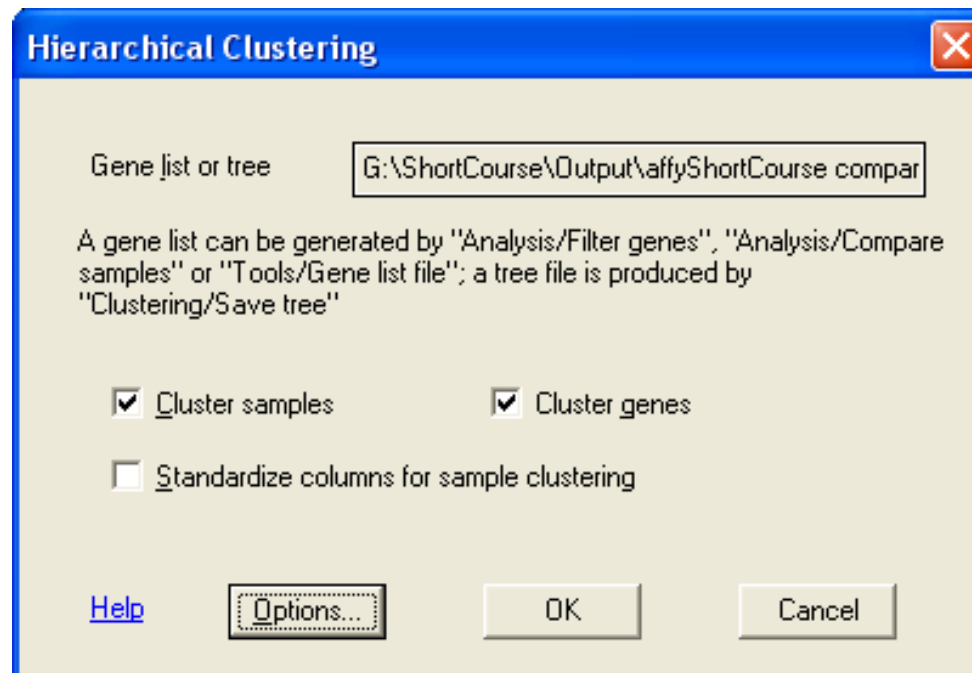
# Starting to use hierarchical clustering in dChip

On the main “Analysis” menu, select “Clustering & Enrichment” (was “Hierarchical clustering”).



# Starting to use hierarchical clustering in dChip

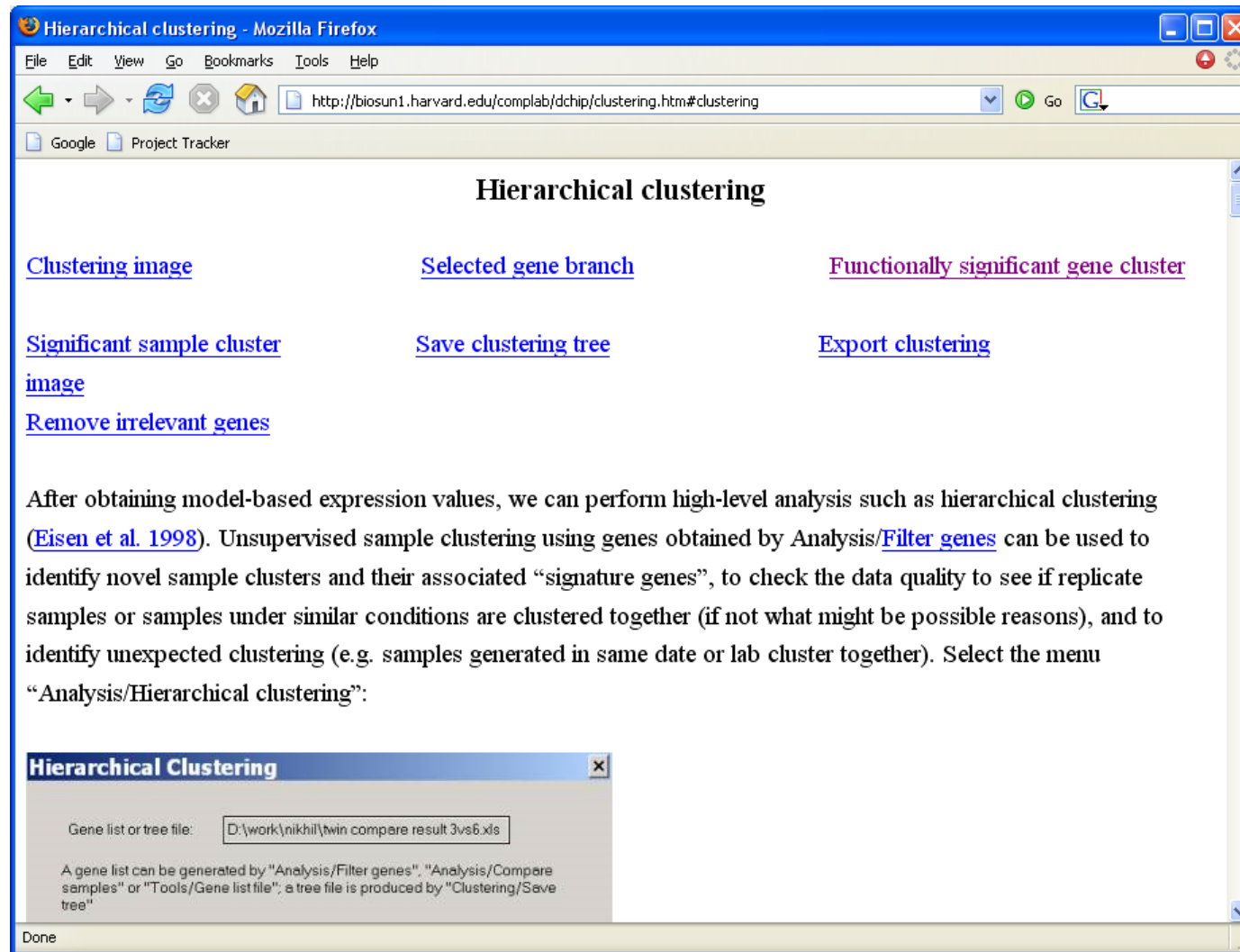
In the resulting dialog box, we can choose to cluster both genes and samples.



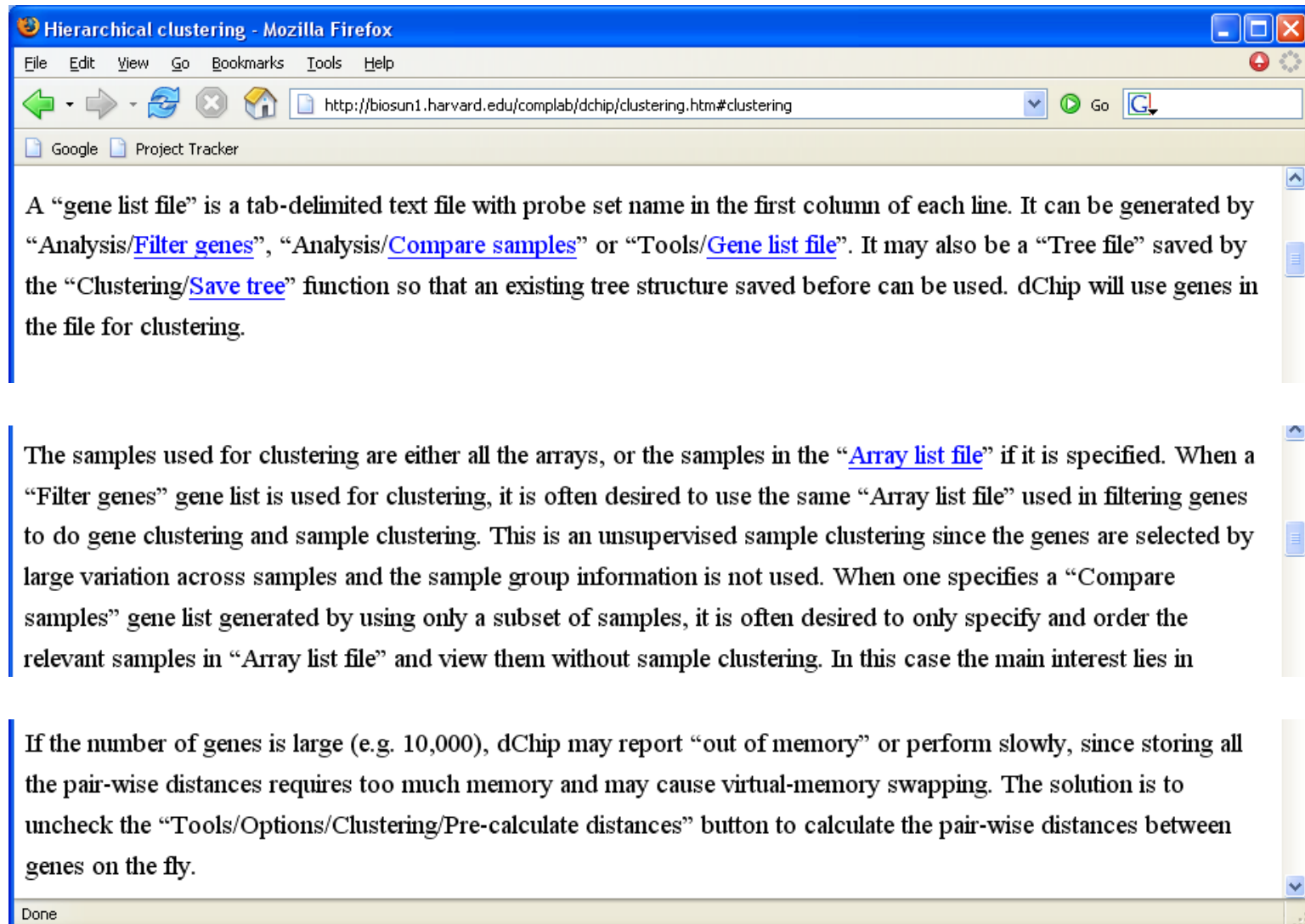
Whenever you're not sure about what to do in dChip, you can see what "Help" they provide.



# dChip help for hierarchical clustering

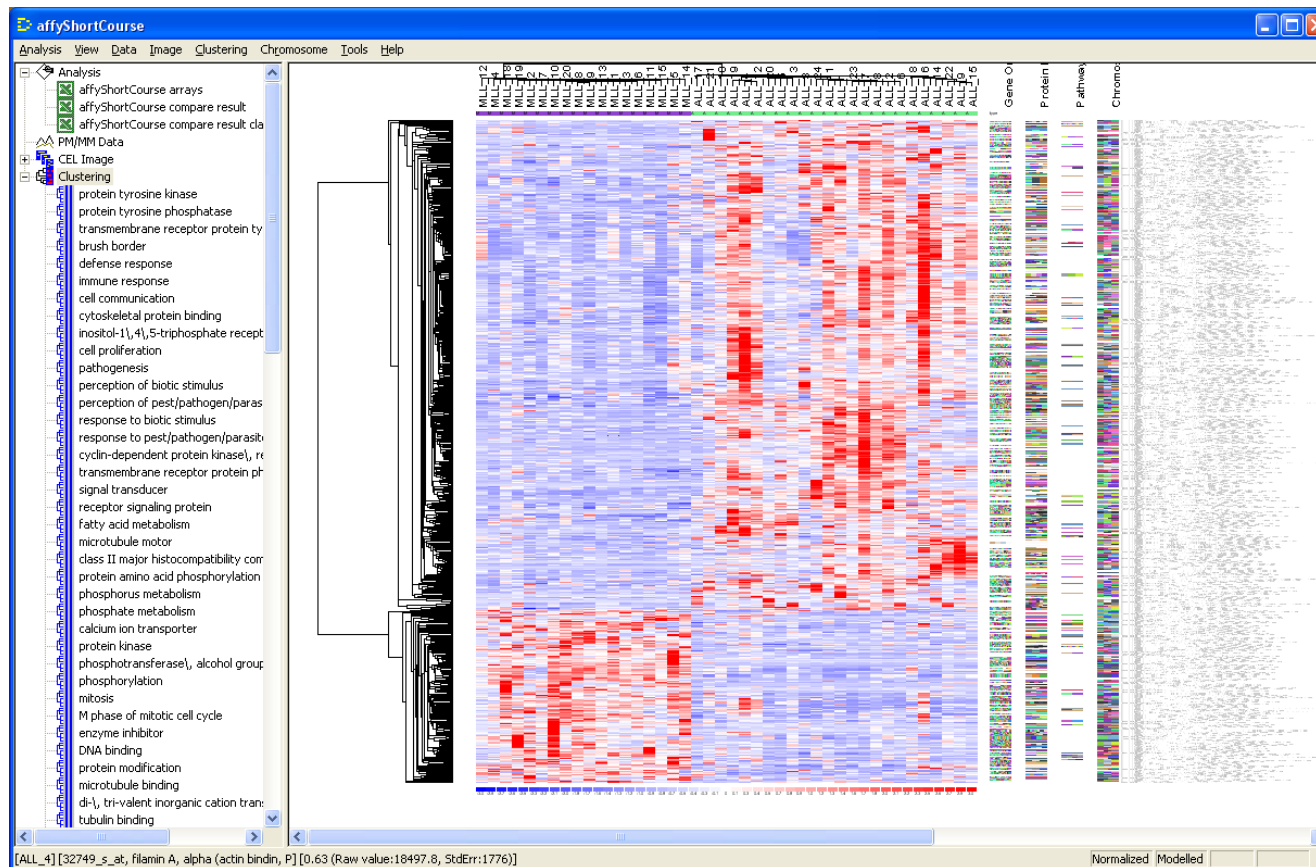


# dChip help for hierarchical clustering



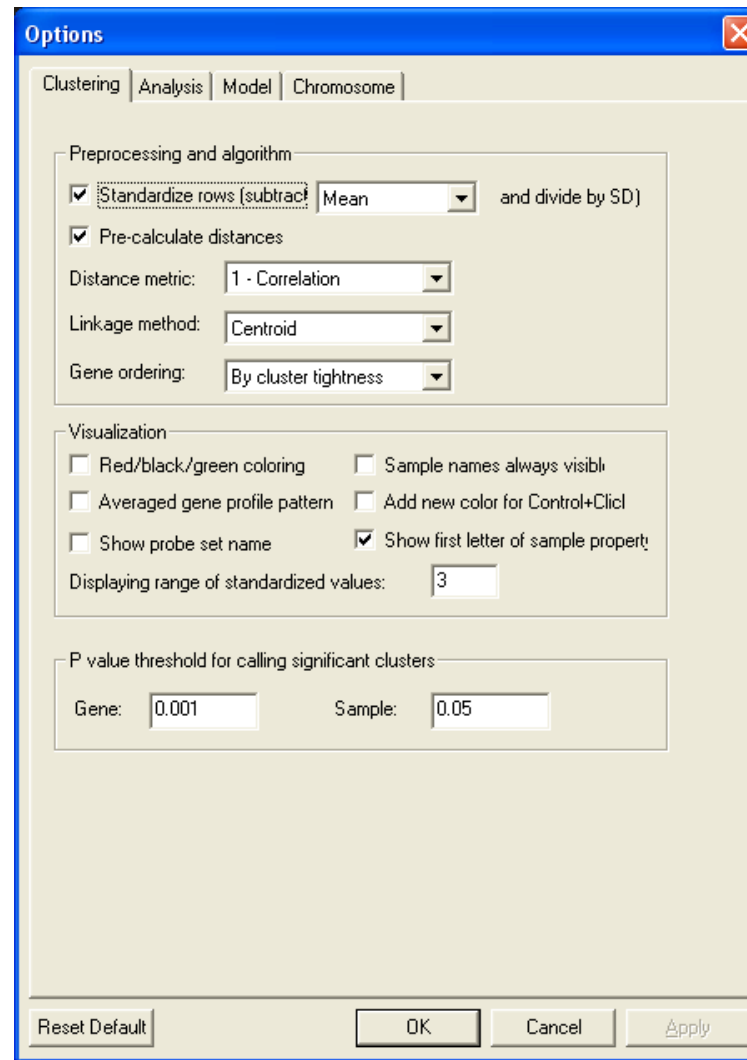
# dChip clustering results

Ignoring their advice, we go ahead and cluster samples using the 610 (604) genes selected from our previous sample comparison.



# dChip clustering options

Using “Tools” – > “Options”, we get:



The screenshot shows the "Options" dialog box with the "Clustering" tab selected. The dialog has four tabs: Clustering, Analysis, Model, and Chromosome. The "Clustering" tab contains the following settings:

- Preprocessing and algorithm:**
  - ☒ Standardize rows (subtract: Mean and divide by SD)
  - ☒ Pre-calculate distances
  - Distance metric: 1 - Correlation
  - Linkage method: Centroid
  - Gene ordering: By cluster tightness
- Visualization:**
  - ☐ Red/black/green coloring
  - ☐ Sample names always visible
  - ☐ Averaged gene profile pattern
  - ☐ Add new color for Control+Click
  - ☐ Show probe set name
  - ☒ Show first letter of sample property
  - Displaying range of standardized values: 3
- P value threshold for calling significant clusters:**
  - Gene: 0.001
  - Sample: 0.05

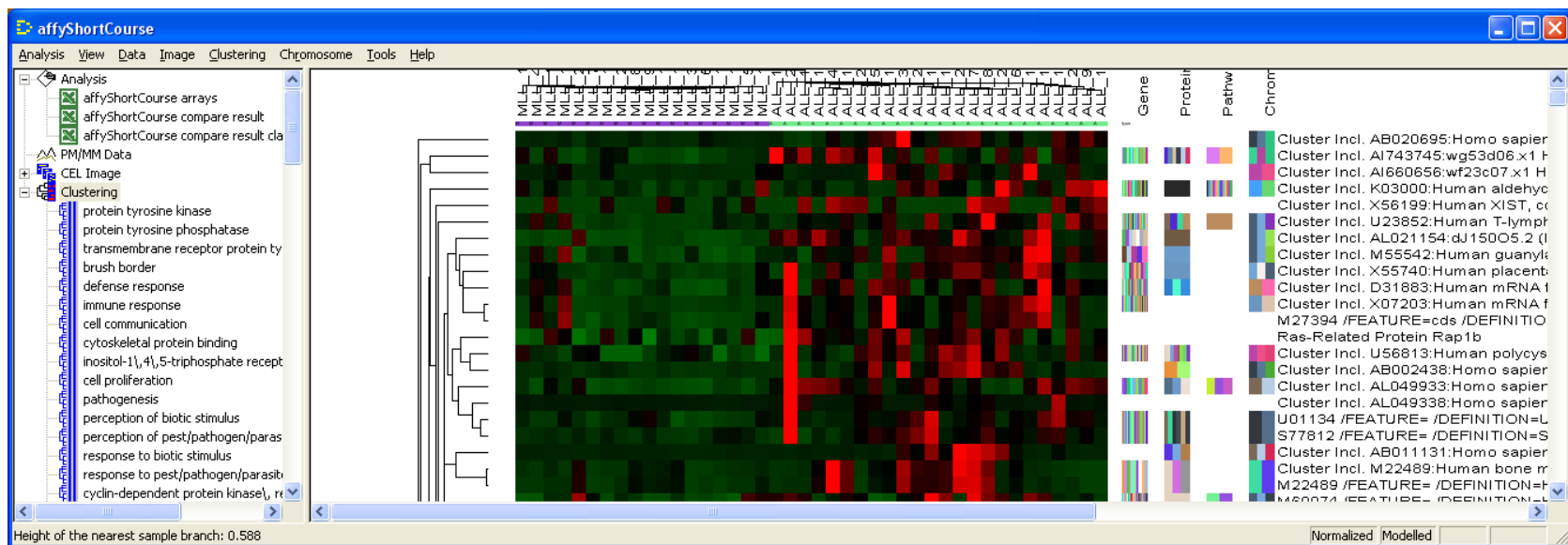
At the bottom of the dialog are four buttons: "Reset Default", "OK", "Cancel", and "Apply".

Checking the visualization option for red/black/green coloring gives the Eisen colormap.



# Exploring the clustering results

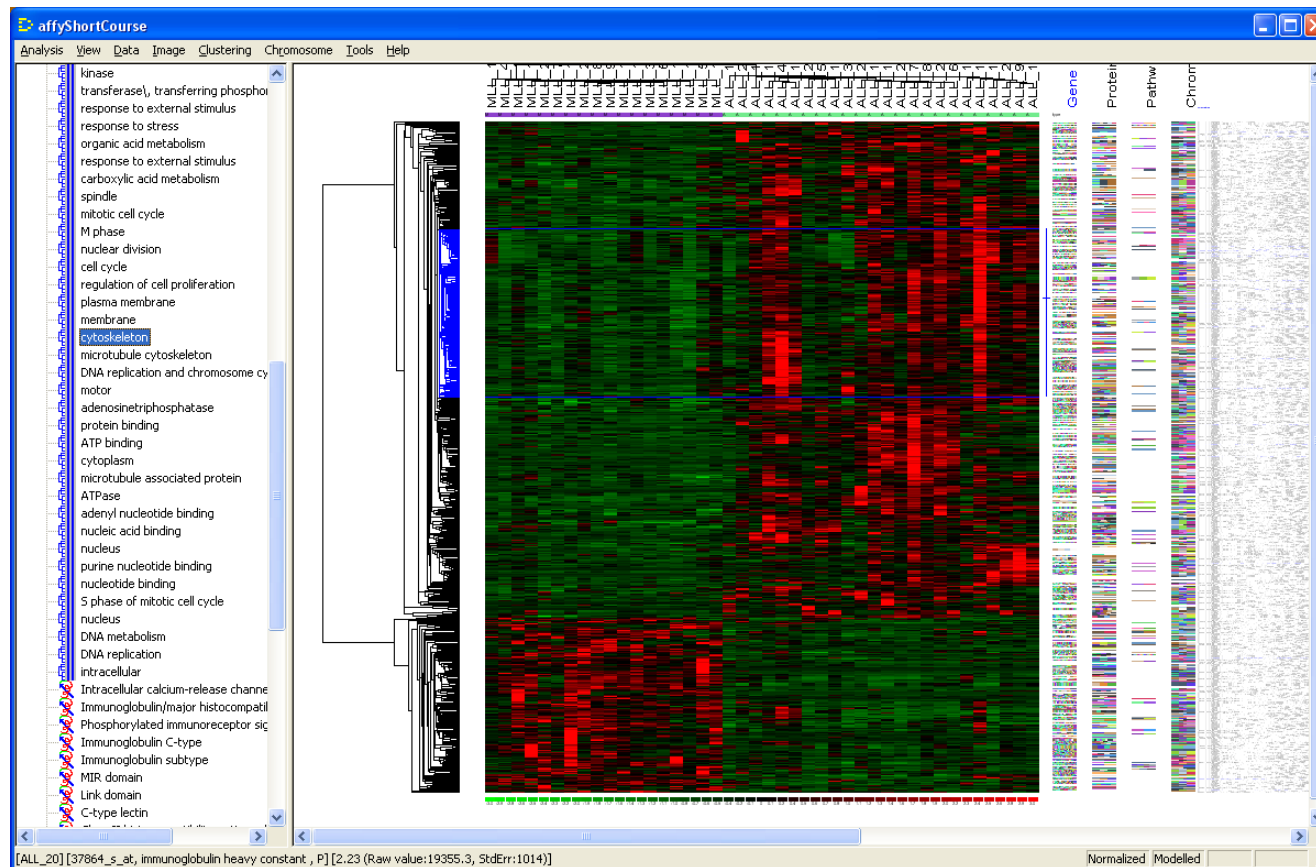
You can use the arrow keys to zoom in or out of the cluster diagram. You may need to zoom in a lot in order to be able to read the gene names.



Clicking on a gene name will open a web browser window to the Entrez Gene page for that gene.

# Exploring the clustering results

On the left, dChip lists “significant” GeneOntology categories, protein domains, and sample types. Click on the names to see where they can be found.





# Revisiting the clustering options

The screenshot shows a software window titled "Options" with a blue title bar and a close button. It contains four tabs: "Clustering", "Analysis", "Model", and "Chromosome". The "Clustering" tab is selected. The window is divided into three main sections: "Preprocessing and algorithm", "Visualization", and "P value threshold for calling significant clusters".

**Preprocessing and algorithm**

- ☒ Standardize rows (subtract Mean and divide by SD)
- ☒ Pre-calculate distances
- Distance metric: 1 - Correlation
- Linkage method: Centroid
- Gene ordering: By cluster tightness

**Visualization**

- ☐ Red/black/green coloring
- ☐ Sample names always visible
- ☐ Averaged gene profile pattern
- ☐ Add new color for Control+Click
- ☐ Show probe set name
- ☒ Show first letter of sample property
- Displaying range of standardized values: 3

**P value threshold for calling significant clusters**

- Gene: 0.001
- Sample: 0.05

At the bottom of the window are four buttons: "Reset Default", "OK", "Cancel", and "Apply".



# Revisiting the clustering options

- Choices for distance metric
  - 1 - correlation
  - 1 - absolute correlation
  - 1 - rank correlation
  - Euclidean
- Choices for linkage
  - centroid
  - average

Which should I choose? What do these mean?

## Measuring distances

Ideally, clustering methods tell us that some samples form a more coherent set than the data as a whole, where “more coherent” is generally taken to mean that the samples are closer together.

So, how do we define “closer”?

This requires the specification of a distance or “dissimilarity” matrix. Distances are calculated between each pair of samples. For this purpose, we view each sample as a vector in “gene-space”. The first distance measure most people think of is

Euclidean distance:  $\sqrt{\sum (x - y)^2}$

In the R language,

```
dEuclid <- dist(t(dataMatrix));
```

## Alternative definitions of distance

Maximum:  $\max(\text{abs}(x-y))$

Manhattan:  $\sum(\text{abs}(x-y))$

Minkowski:  $\sum(\text{abs}(x-y)^p)^{1/p}$

Canberra:  $\sum(\text{abs}(x-y) / (\text{abs}(x) + \text{abs}(y)))$

Binary:  $\sum(\text{xor}(x \neq 0, y \neq 0)) / \sum(x \neq 0 \mid y \neq 0)$

Correlation:  $(1 - \text{cor}(x, y)) / 2$

Absolute Correlation:  $(1 - \text{abs}(\text{cor}(x, y)))$

Rank Correlation:  $(1 - \text{cor}(\text{rank}(x), \text{rank}(y))) / 2$

Most clustering methods let you specify the distance measure, or construct any distance matrix you want and work with that matrix.

# Hierarchical clustering

Hierarchical clustering produces a dendrogram (a binary tree structure) that displays the distance relationships between clusters.

The most common implementation is agglomerative, which is an unnecessarily big word for bottom-up. The algorithm starts by joining the two samples that are closest together into a cluster. It then keeps repeating this process (joining the two closest clusters into a bigger cluster) until everything has been linked together.

There's only one problem: Distances were defined between individual vectors. How do you measure the distance between clusters of vectors in order to link them?

## Linkage rules

**Single:** Use the minimum distance between cluster members

**Complete:** Use the maximum distance between cluster members

**Average:** Use the mean distance between cluster members

**Median:** Use the median distance between cluster members

**Centroid:** Use the distance between the “average” member of each cluster

**Ward's:** Minimize the increase in variance of the cluster

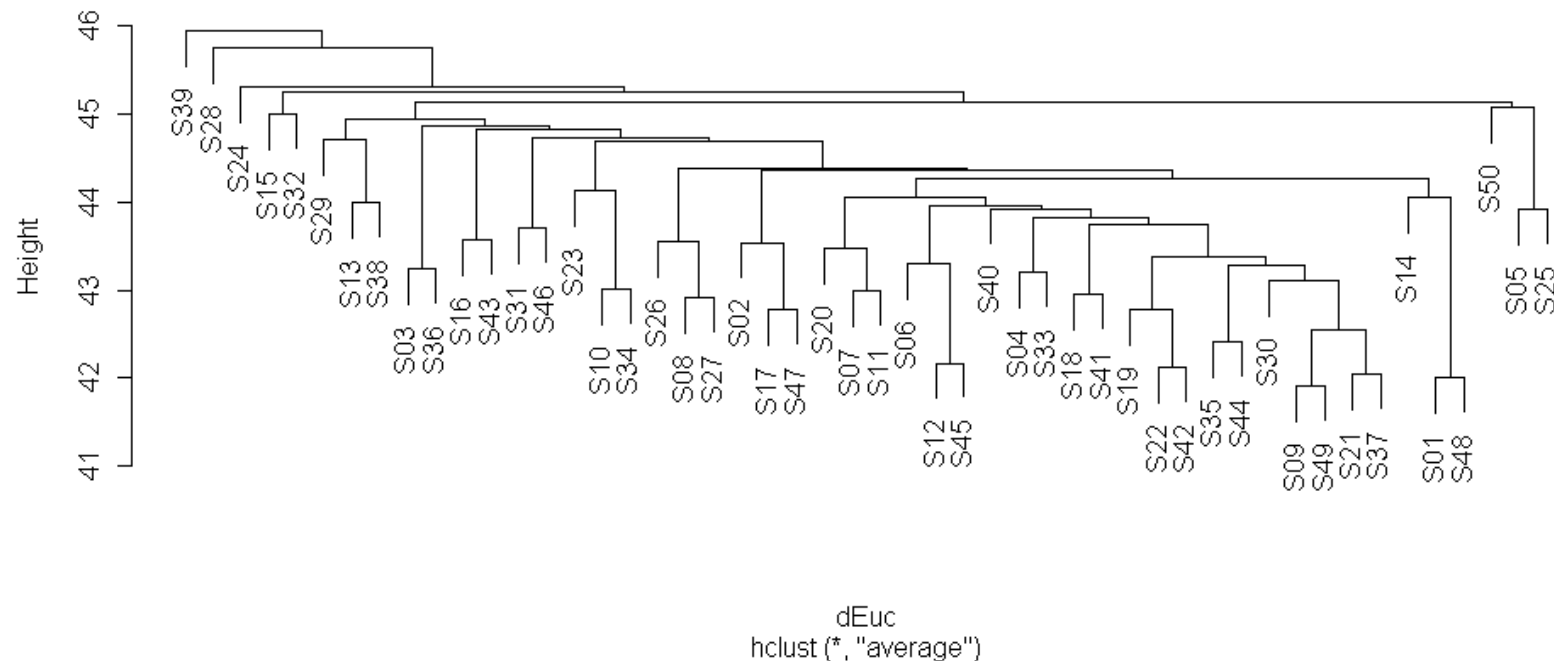
## Simulating nothing

One peculiarity of clustering algorithms is that they always produce clusters. This happens regardless of whether there is actually any meaningful clustering structure present in the data. So, let's simulate some unstructured data and see what happens. We'll write code in R for the simulations.

```
> n.genes <- 1000
> n.samples <- 50
> descr <- sapply(1:50,
                  function(n) sprintf('S%02d', n))
> dataMatrix <- matrix(rnorm(n.genes*n.samples),
+                       nrow=n.genes)
> colnames(dataMatrix) <- descr
```

# Clustering nothing

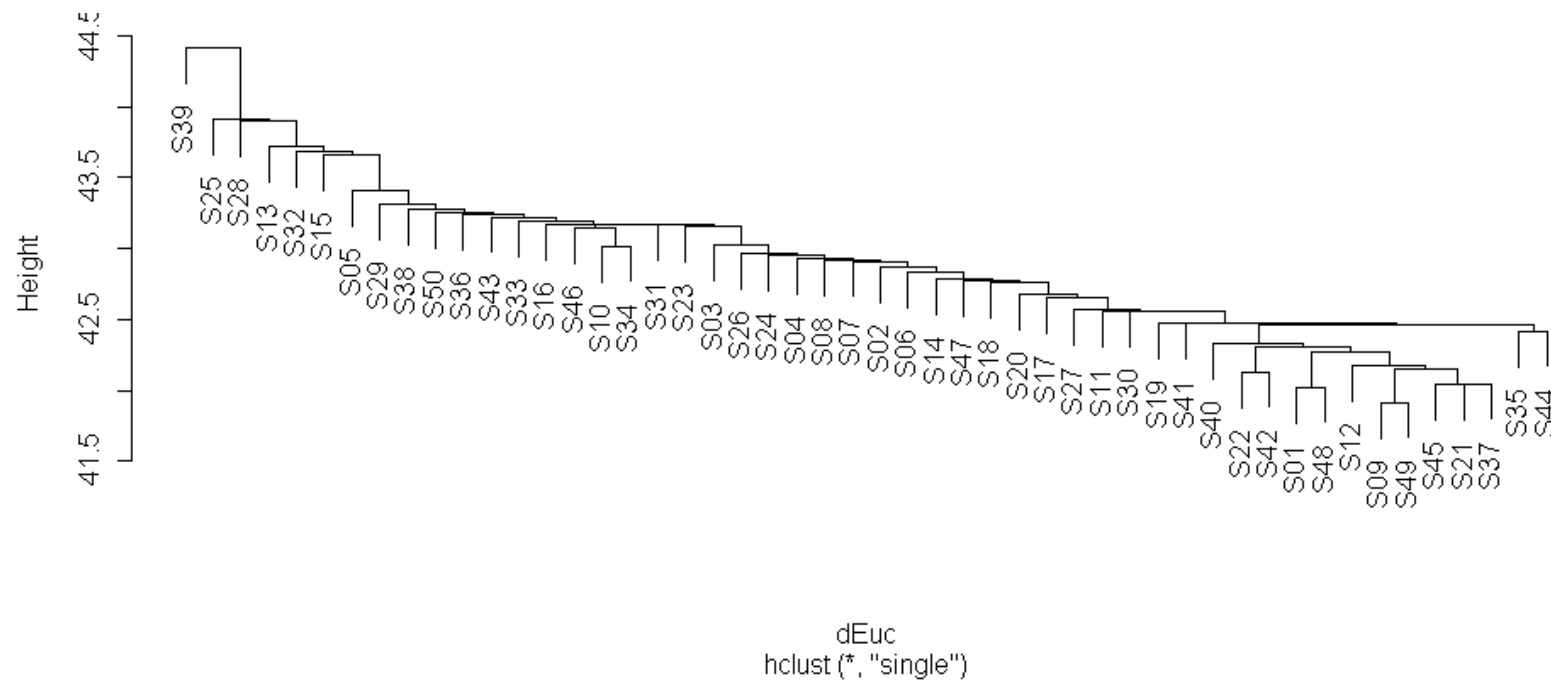
```
> dEuc <- dist(t(dataMatrix))  
> hAvgEuc <- hclust(dEuc, method='average')  
> plclust(hAvgEuc)
```



Euclidean distance, average linkage.

# Single linkage often produces “stringlike” clusters

```
> hSinEuc <-hclust(dEuc, method='single')  
> plclust(hSinEuc)
```

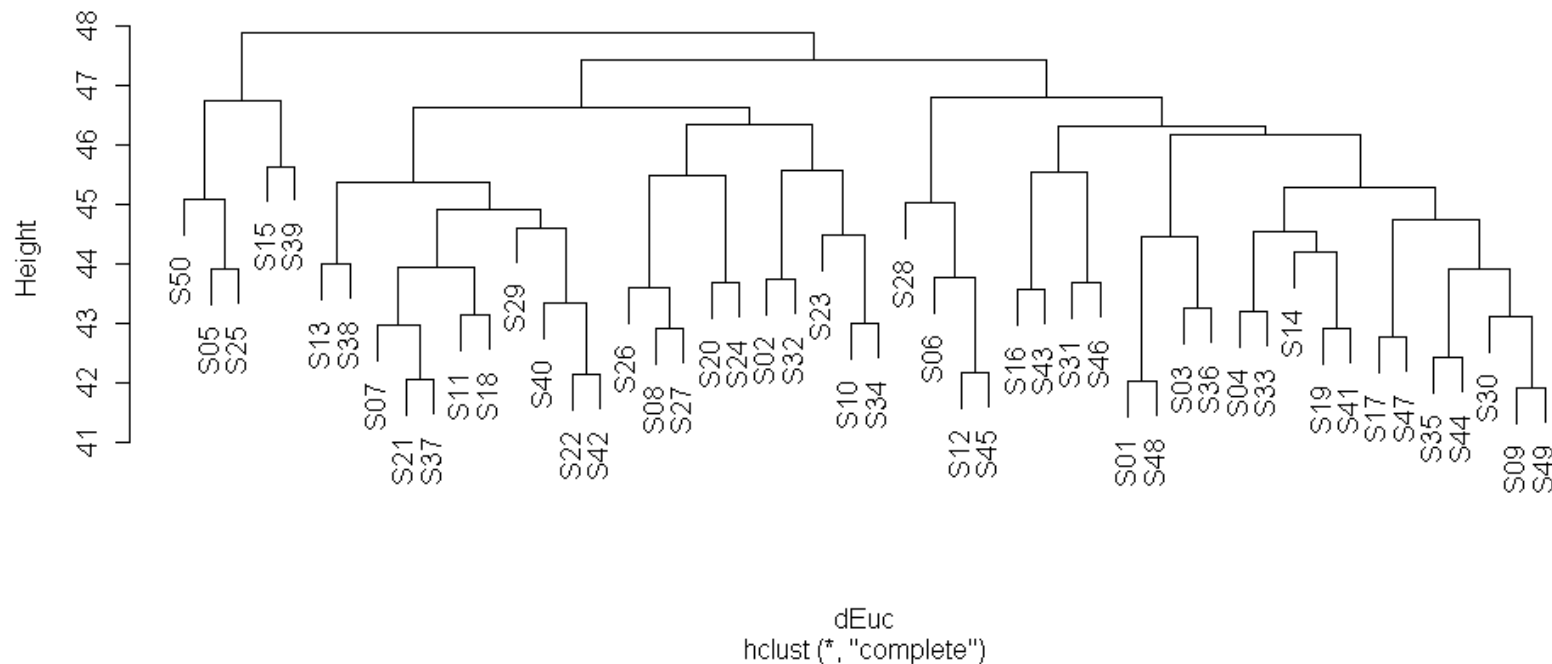


Euclidean distance, single linkage.



# Complete linkage tends to find compact clusters

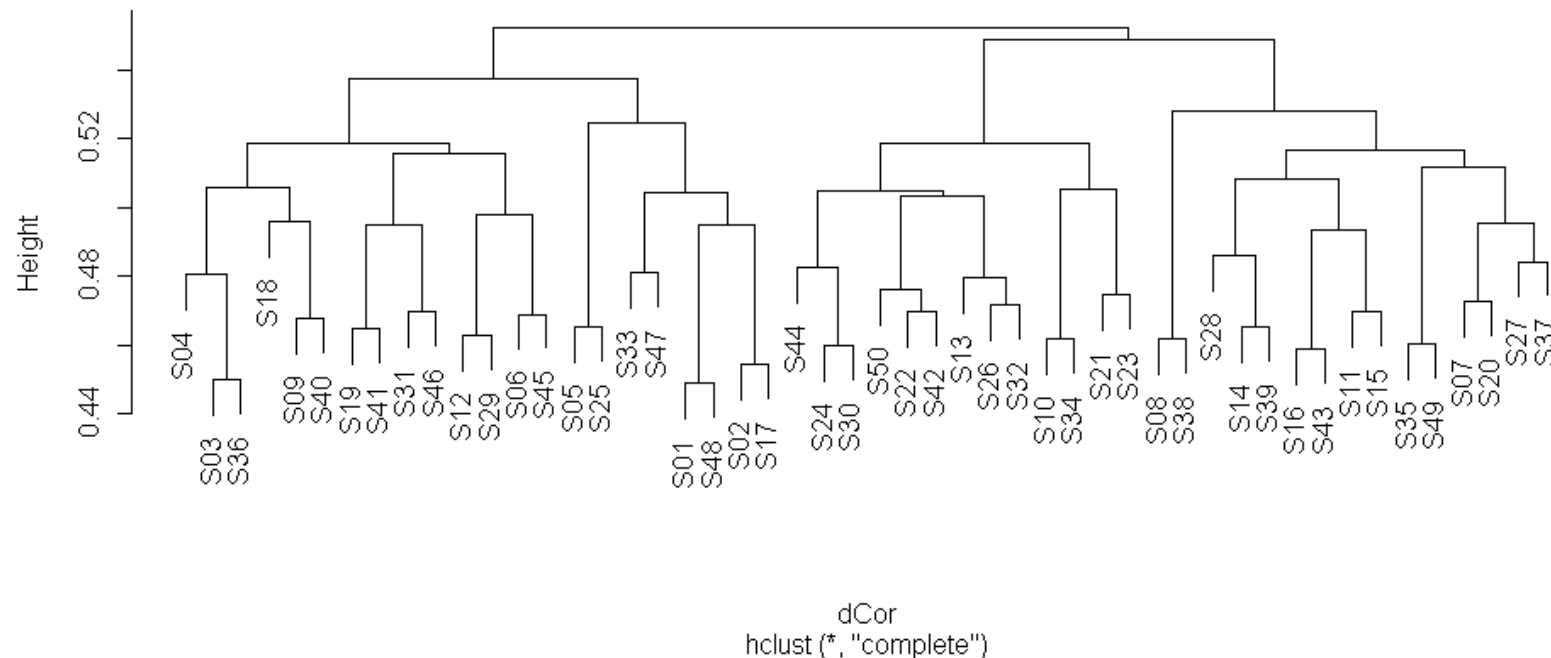
```
> hComEuc <- hclust(dEuc, method='complete')
> plclust(hComEuc)
```



Average linkage tends to produce clusters somewhere in between single and complete linkage.

# Clustering with correlation also finds structure

```
> dCor <- as.dist( (1-cor(dataMatrix))/2 )  
> hComCor <- hclust(dCor, method='complete')  
> plclust(hComCor)
```



Correlation distance, complete linkage.

## What's Stable?

We can flip branches around without affecting the underlying structure of the data, or changing the meaning of the clustering.

What things are left unchanged by such flips?

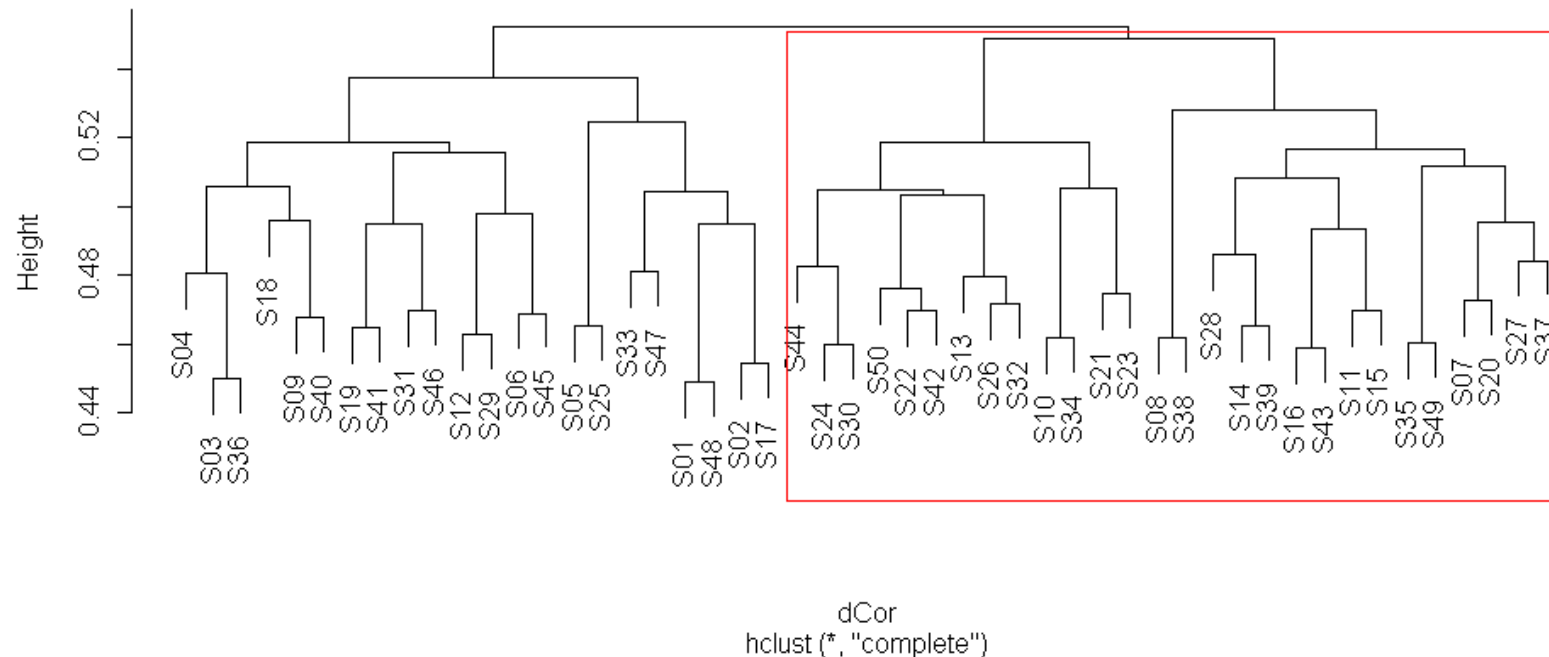
Say we flip a branch at height  $h_{flip}$ .

Membership of the sub-branches does not change, but the order can change across the boundary.

How do I define a “cluster”?

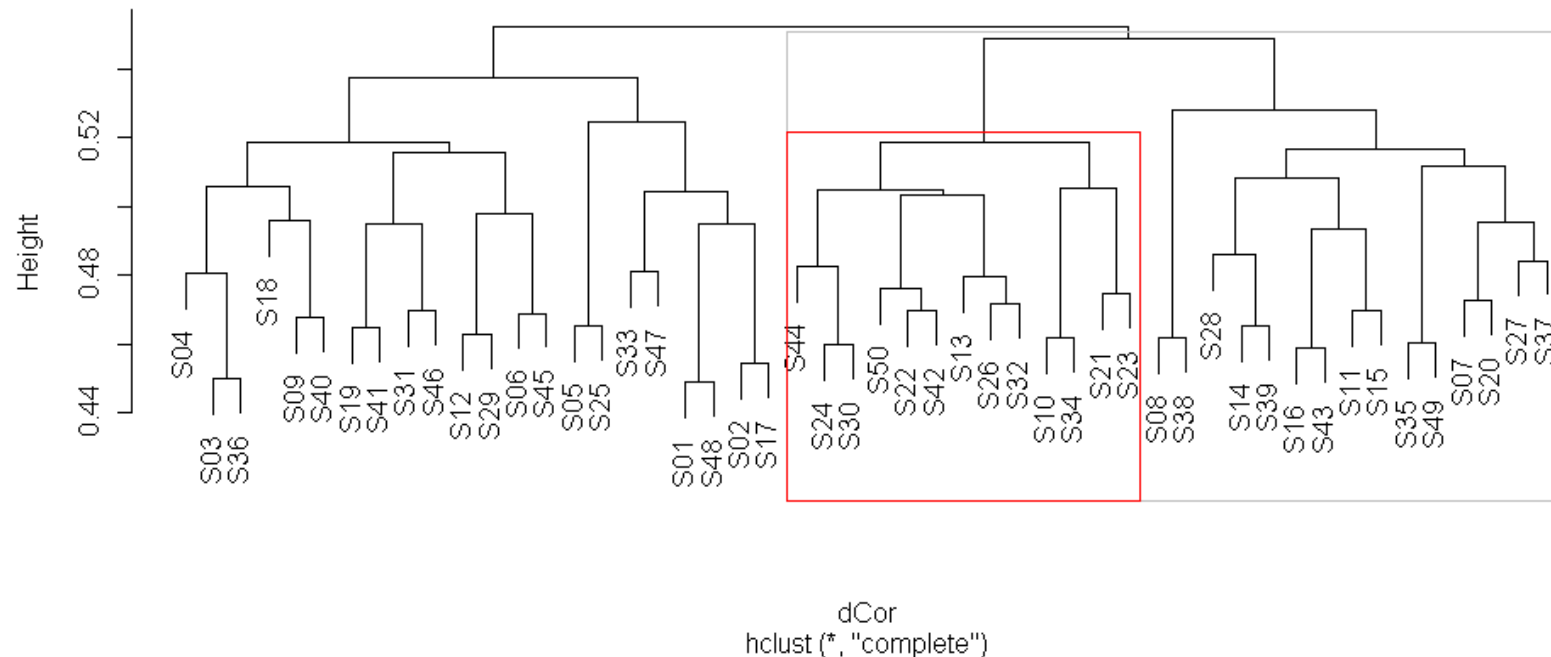
# What is a cluster?

If we cut the dendrogram at height  $h$ , then the sub-branches of each cut branch define clusters. Within a cluster, everything is closer than  $h$  to the rest. By varying the cut height, we can produce an arbitrary number of clusters.



# What is a cluster?

If we cut the dendrogram at height  $h$ , then the sub-branches of each cut branch define clusters. Within a cluster, everything is closer than  $h$  to the rest. By varying the cut height, we can produce an arbitrary number of clusters.



## When is a cluster valid?

In other words, where should we cut the tree in order to say that the branches at this point represent something real?

To convince you that this is a real problem, recall that we are using data that was simulated to be completely random.

Nevertheless, hierarchical clustering (with complete linkage and either Euclidean distance or correlation) apparently finds structure here.

## Bootstrap resampling

Testing cluster validity requires “perturbing” the data.

A cluster consists of pairs of items that are grouped together. If we repeatedly perturb the data, and the pairs still cluster together, this is a good sign that the cluster is “stable”. Samples that cluster in other groups are more questionable.

The simplest way to perturb the data is to “bootstrap” the individual genes, or rows of the data matrix.

The idea behind the bootstrap is to create a new data matrix (the same size as the original) by randomly selecting rows. Sampling is done with replacement, so some rows will be included multiple times and some rows will be omitted.

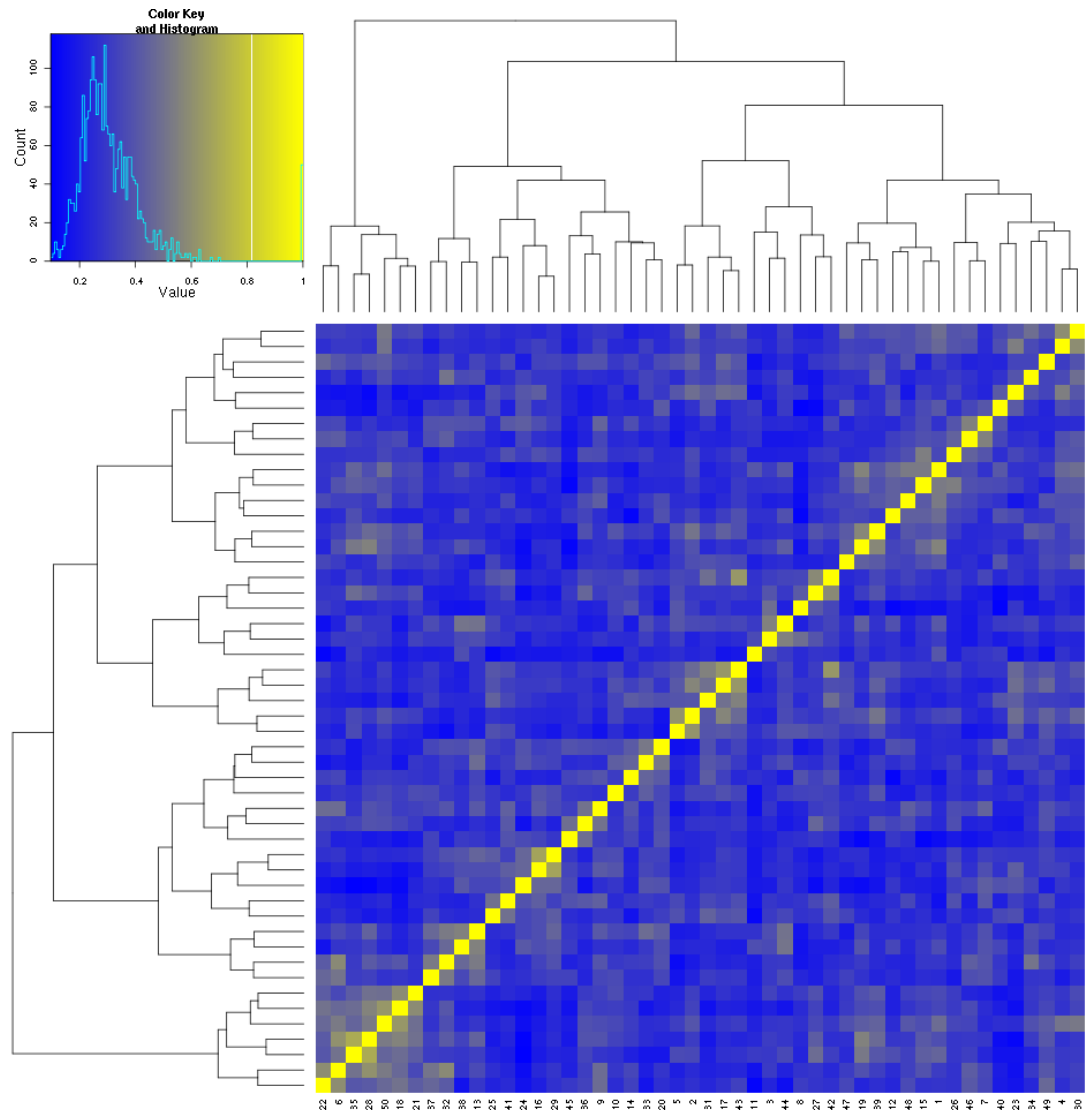
## Disturbing the universe

We can use the ClassDiscovery package to perform bootstrap resampling for clustering. In order to use  $k = 4$  groups, with hierarchical clustering by Pearson correlation distance and average linkage along with an outer loop of  $nTimes = 200$  bootstrap samples, we do the following:

```
> require(ClassDiscovery)
> bc <- BootstrapClusterTest(dataMatrix,
+                             cutHclust, k=4, method='average',
+                             metric='pearson', nTimes=200)
> image(bc)
```



# Sometimes it's good to find nothing...



## Additional Notes

We need to specify the number of clusters to bootstrap, since we record how many times samples are paired. This method extends directly to other clustering techniques.

The image is much more interpretable if the rows and columns of the matching matrix are reordered to match the ordering supplied by the clustering.

Instead of resampling the genes, we can “add noise” to the data from a normal distribution. The scale of noise to use is not obvious with real data.

We can also use “bootstrap subsampling”. Instead of reconstructing a sample of the same size as the number of genes on the array, make smaller samples to see how widely the clustering information is spread across the genes.

## Clustering with fewer genes

Many times, we cluster using a subset of genes. Maybe we think that other genes are just contributing noise, or maybe we want to focus on genes on a specific chromosome or genes in a specific pathway.

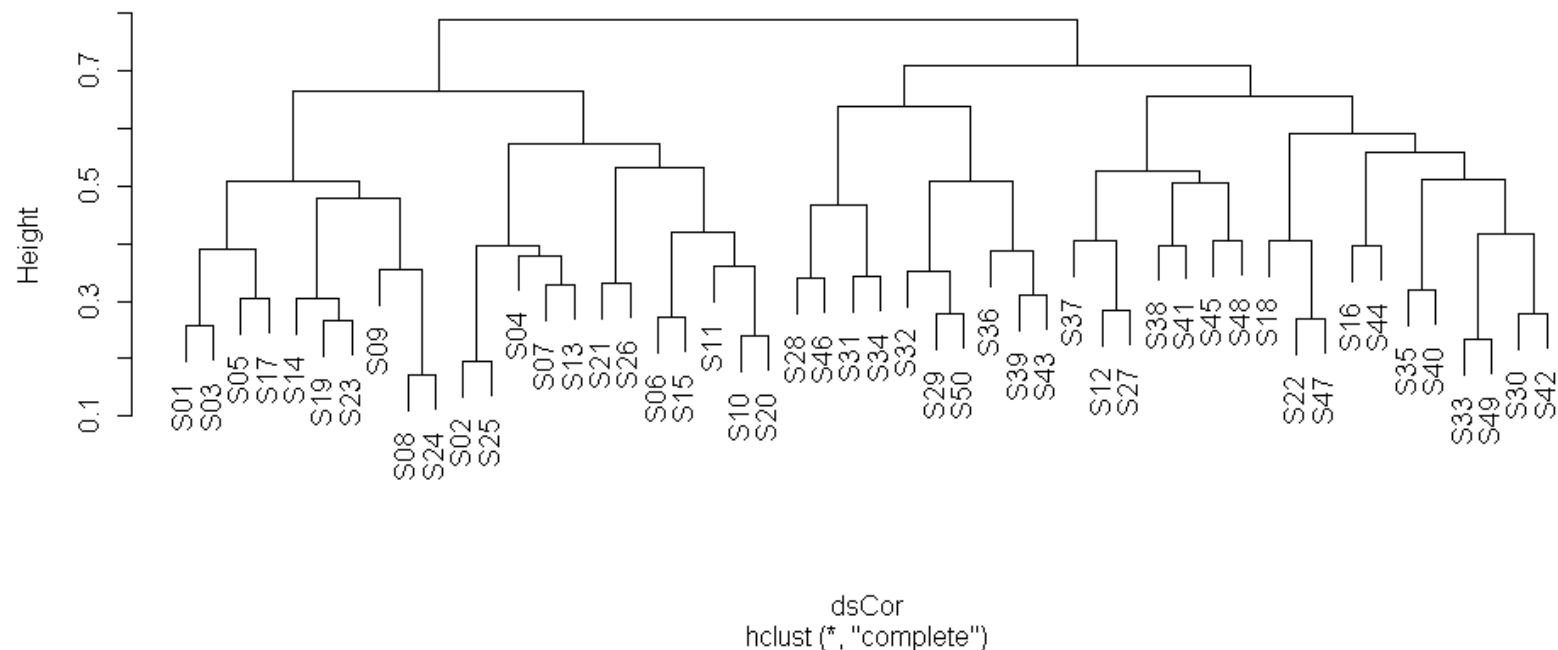
Occasionally, you see papers comparing two known classes of samples that perform the following analysis:

1. Find a list of differentially expressed genes.
2. Cluster data using only the differentially expressed genes.
3. Discover that you can successfully distinguish known classes.

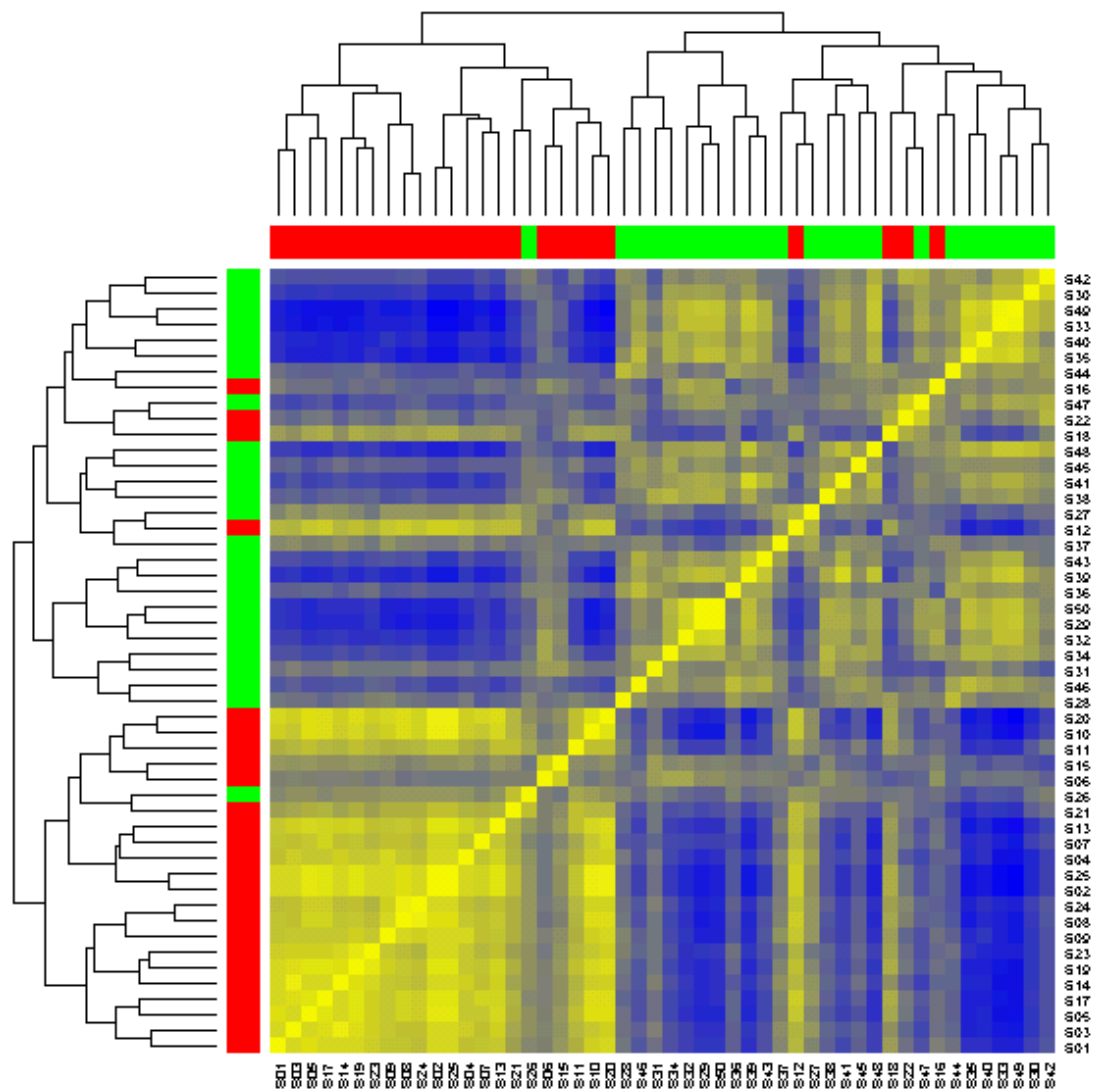
Should this be surprising?

# Bending reality to your will

Let's try this on our simulated data. We'll divide the 50 samples into two classes (the first 25 and the last 25). Next, we'll perform t-tests to see how well each gene separates the two classes, and cluster the data using the top 50 genes:



# Even the bootstrap doesn't save us...



## Filtering notes

Filters should not be related to a specific contrast if an overall view is desired.

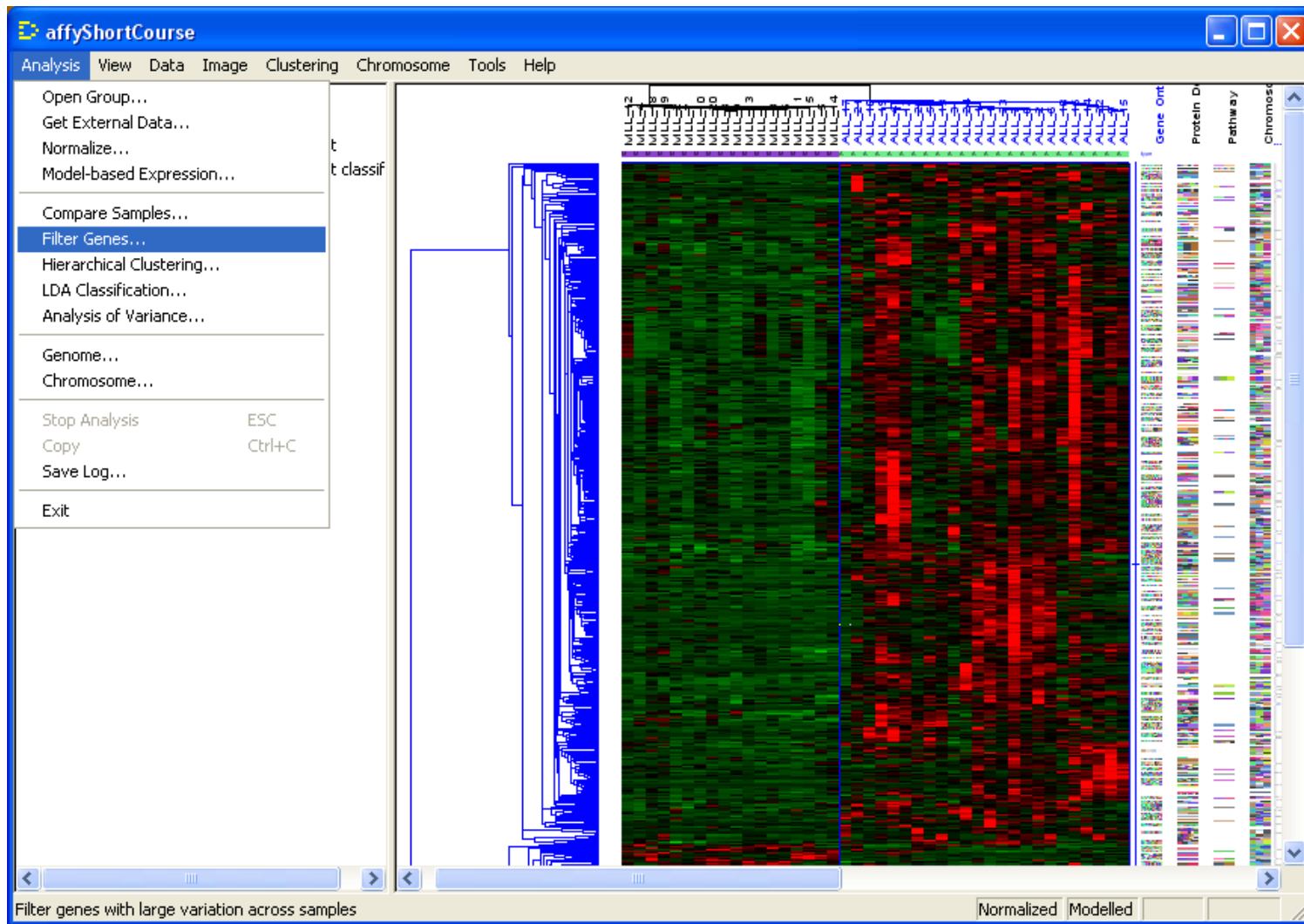
More natural filters exist:

- total variation
- all genes on a given chromosome
- all genes in a given ontology category

Filtering serves a practical purpose – it reduces the number of genes a lot. This is important because we may want to cluster the genes as well as the samples, and clustering thousands of things may make `dChip` (or R) complain. . . .

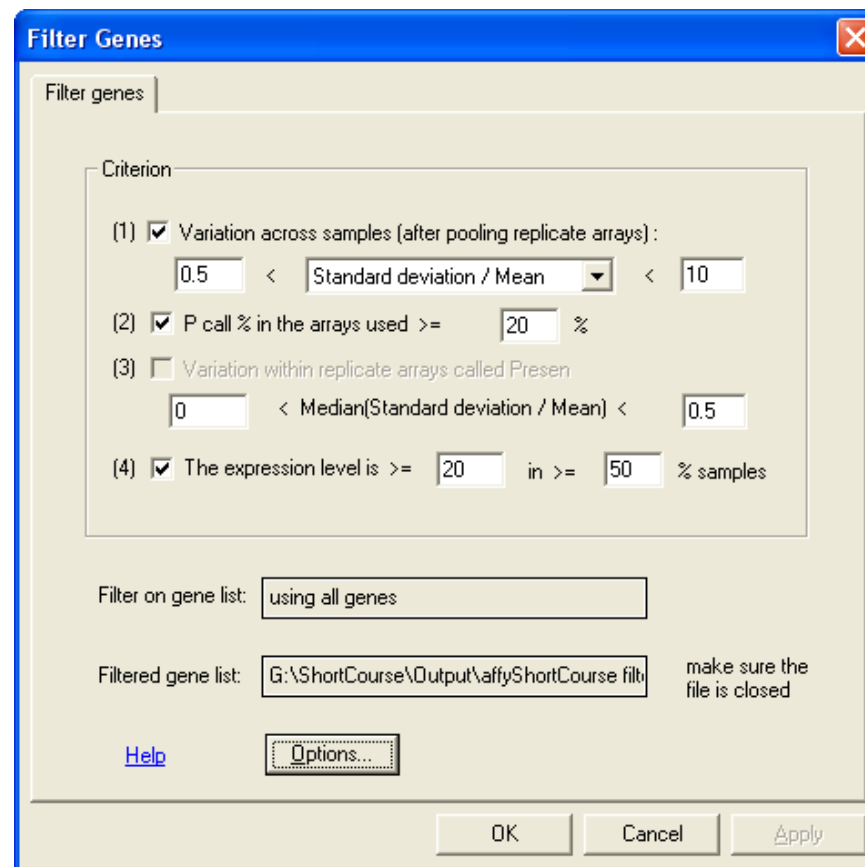
# Filtering in dChip

Use “Analysis” – > “Filter genes”.



# Filtering in dChip

Choose filtering parameters based on variation, expression, or present calls. (I have a bias against variation filters, but those are the default.)





# dChip Filter Results

The screenshot shows the 'affyShortCourse' application window. The left pane displays a tree view under 'Analysis' with the following items: 'affyShortCourse arrays', 'affyShortCourse compare result', 'affyShortCourse compare result classif', 'affyShortCourse filtered gene', 'PM/MM Data', 'CEL Image', and 'Clustering'. The right pane shows the output of the dChip filter process.

```

289/10821, PValue: 0.000399)
    Found 6 "4" genes in a cluster with 32 annotated genes (all:
399/10821, PValue: 0.000971)

    3966 cluster-Chromosome term pairs assessed for enrichment with p-
value < 0.001000

    Finding significant sample clusters...
    Found 24 "type ALL" samples in a cluster with 24 annotated samples
(all: 24/42, PValue: 0.000000)
    Found 18 "type MLL" samples in a cluster with 18 annotated samples
(all: 18/42, PValue: 0.000000)

    33 cluster-category pairs assessed for enrichment with p-value <
0.050000

Finished in 00 hours 00 minutes 02 seconds)

(Filter genes
Filtering genes...
Array list file used: None
1082 of 12626 probe sets satisfied the filtering criteria:
Variation across samples: 0.50 < Standard deviation / Mean <
10.00
P call % in the array used >= 20%
The expression level >= 20.00 in >= 50% samples

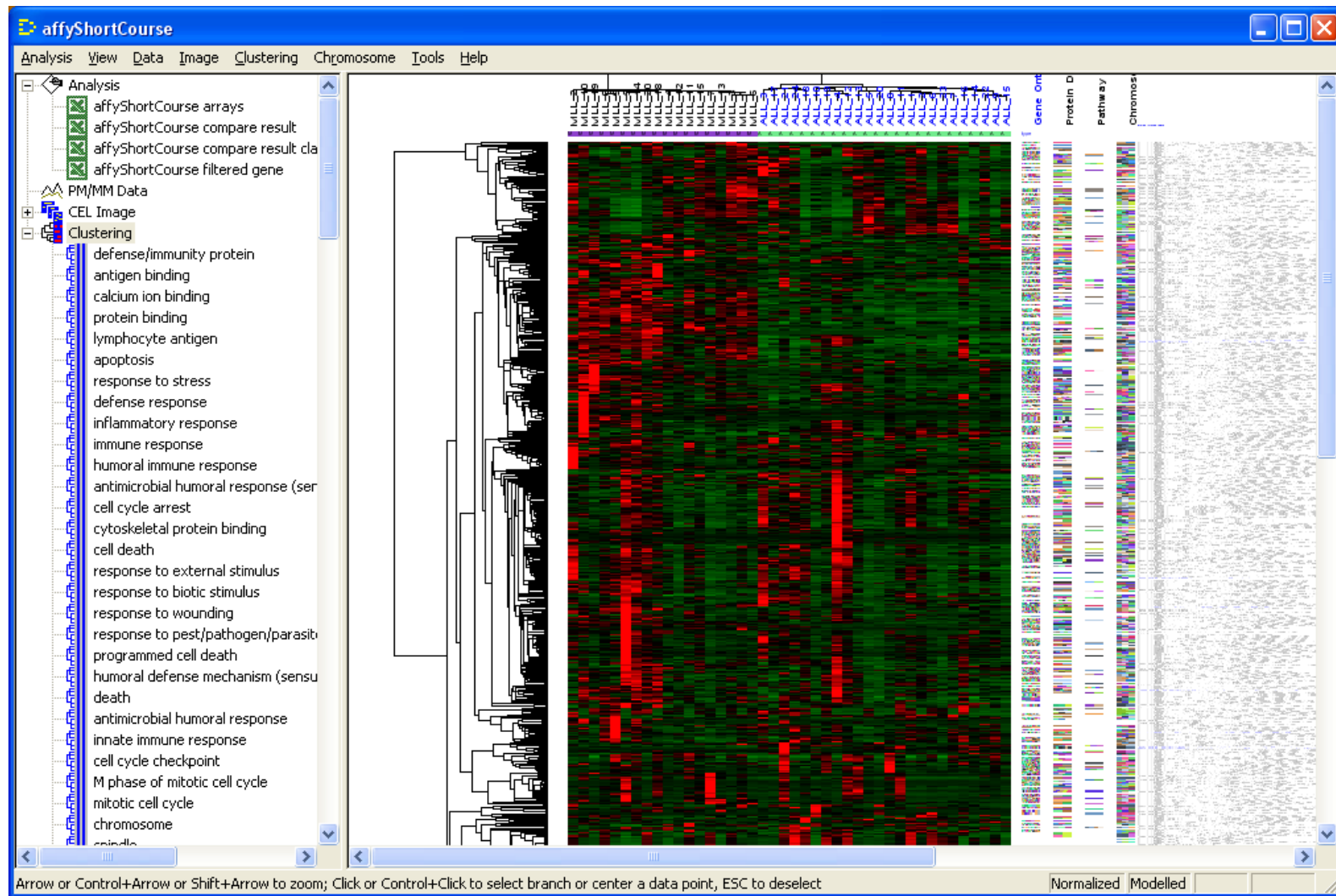
Filtered gene lists saved in G:\ShortCourse\Output\affyShortCourse
filtered gene.xls

Finished)
  
```

At the bottom of the window, there is a status bar with 'Analysis outputs' on the left and 'Normalized' and 'Modelled' tabs on the right.

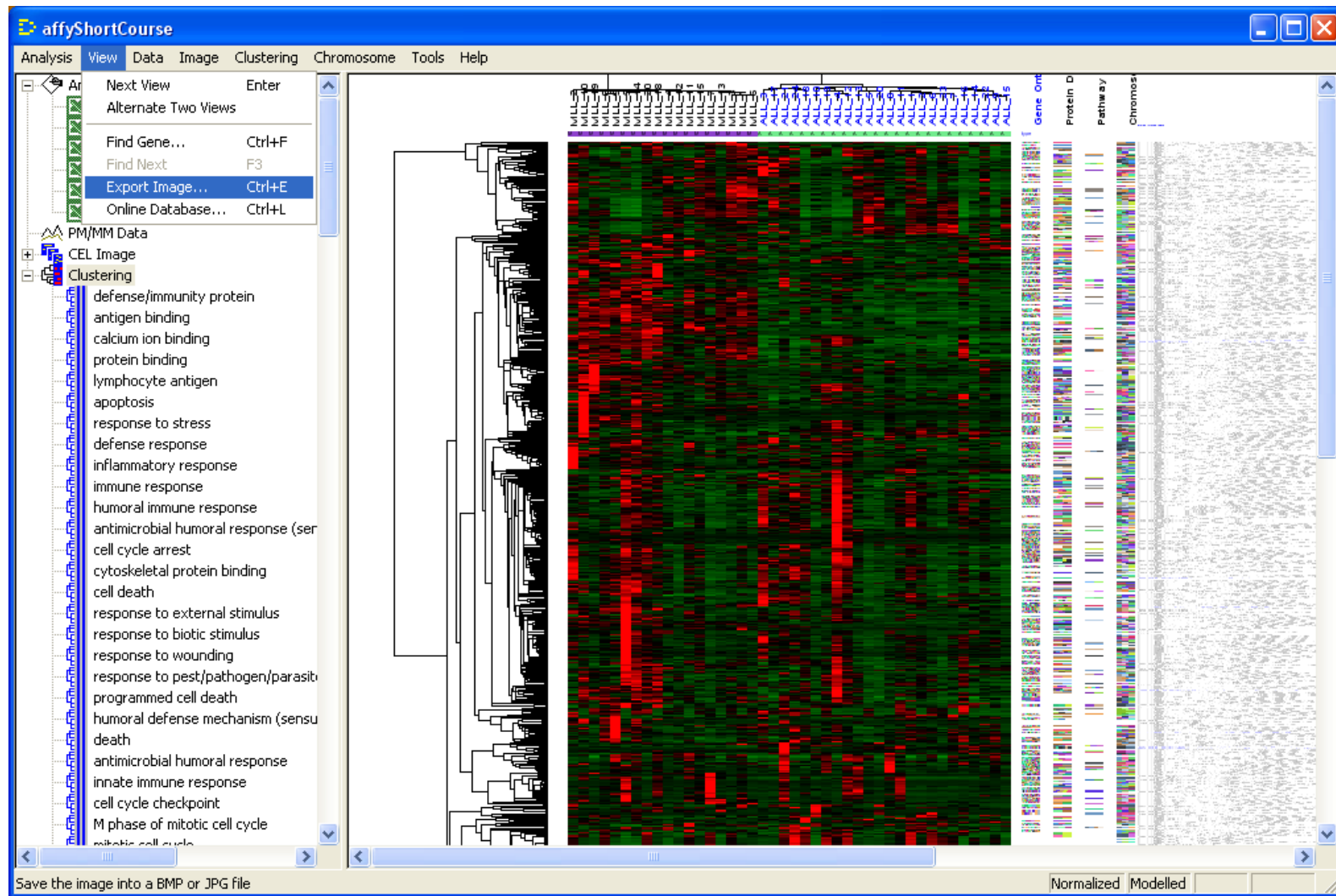
# dChip Clusters with Filtered Genes

We can distinguish ALL from MLL in an unsupervised setting.



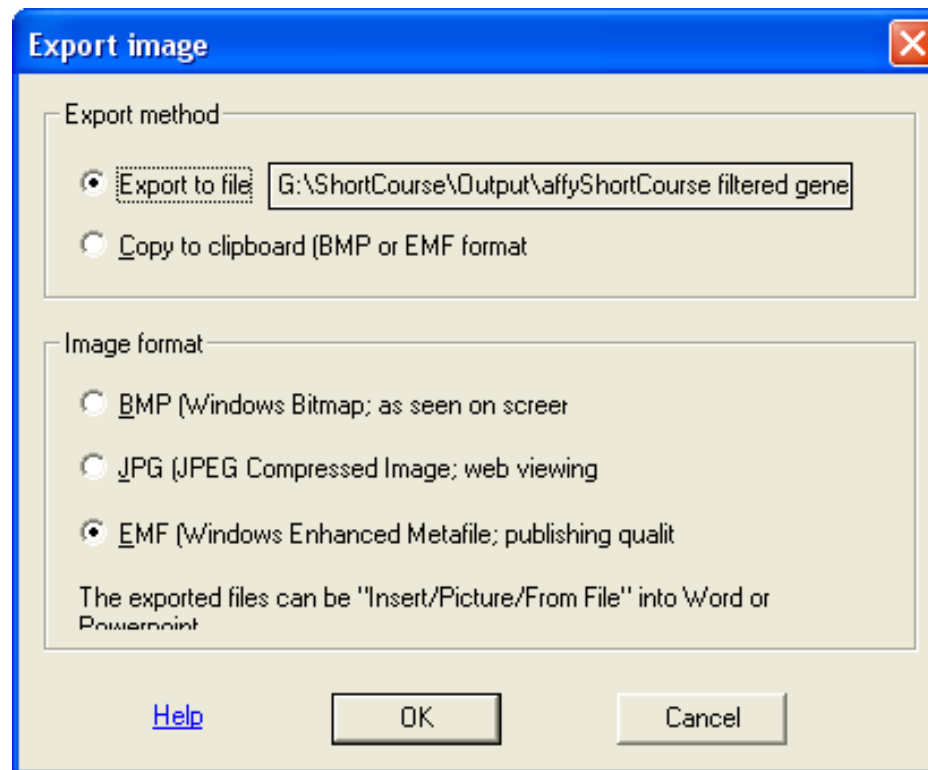
# Saving the cluster images

Use “View” — > “Export Image”.



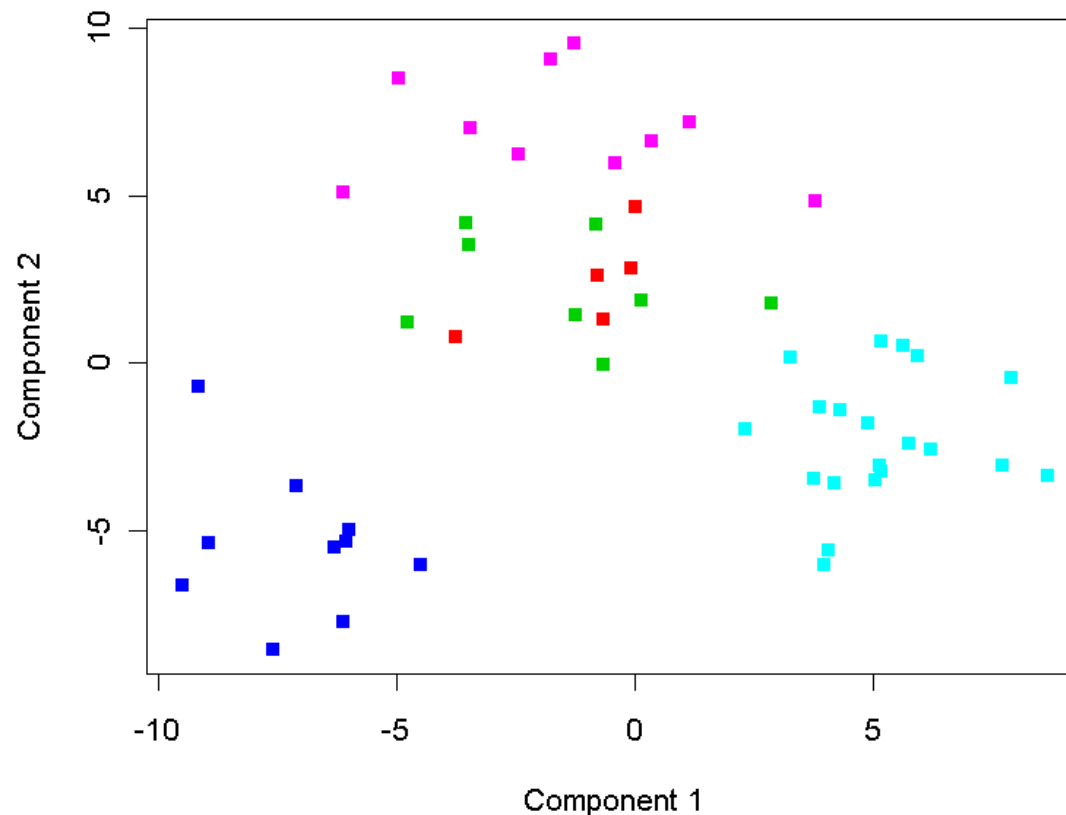
# Saving the cluster images

Choose an appropriate format. EMF is probably best if you ever want to zoom in enough to read the gene names.

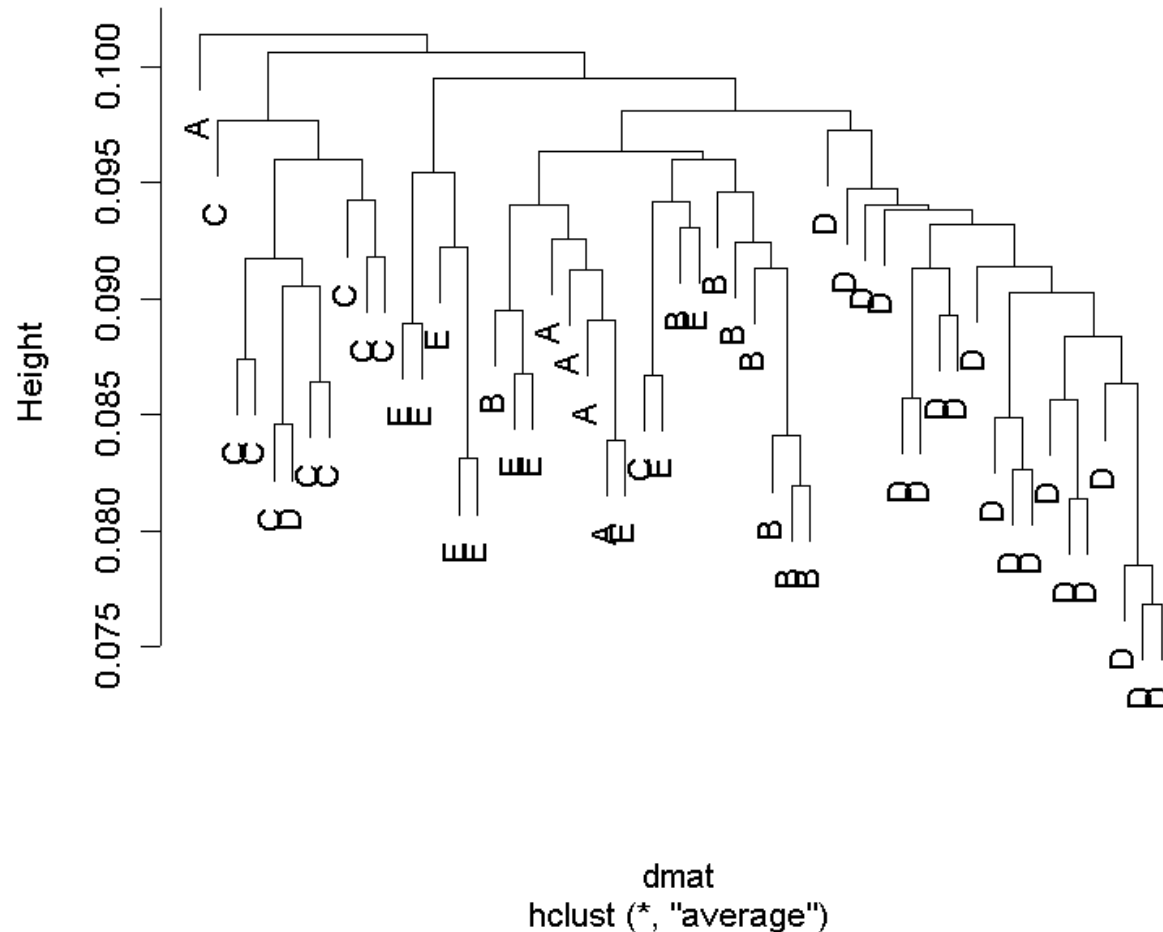


# Simulating something

Next, we simulated data with 1000 genes and 5 different sample classes containing different numbers of samples. Here's a two-dimensional picture of the truth:

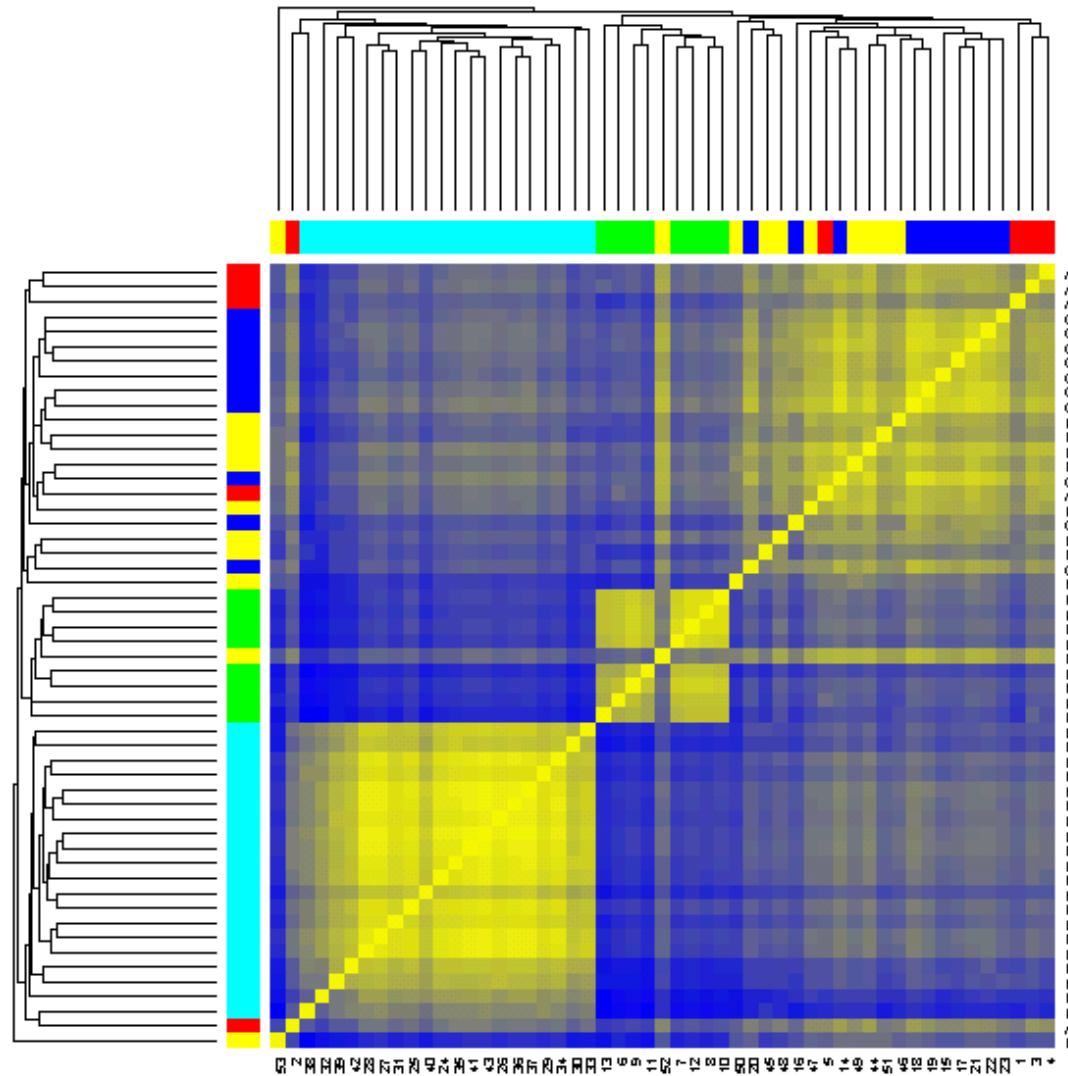


## Hierarchical clusters (correlation; average)

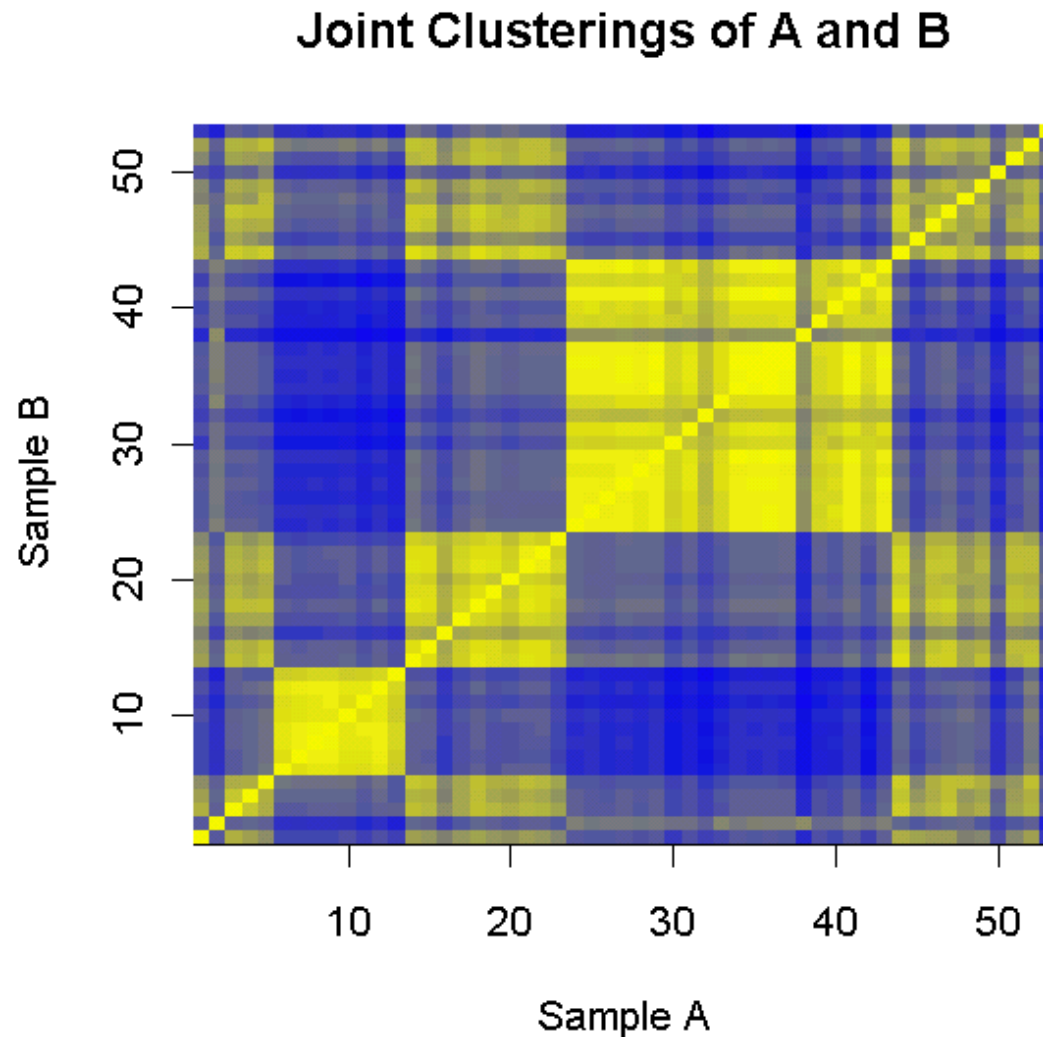


Three of the classes (B, C, D) are mostly correct. The other two classes are less concentrated.

# Bootstrap clusters



# Bootstrap clusters ordered by true groups

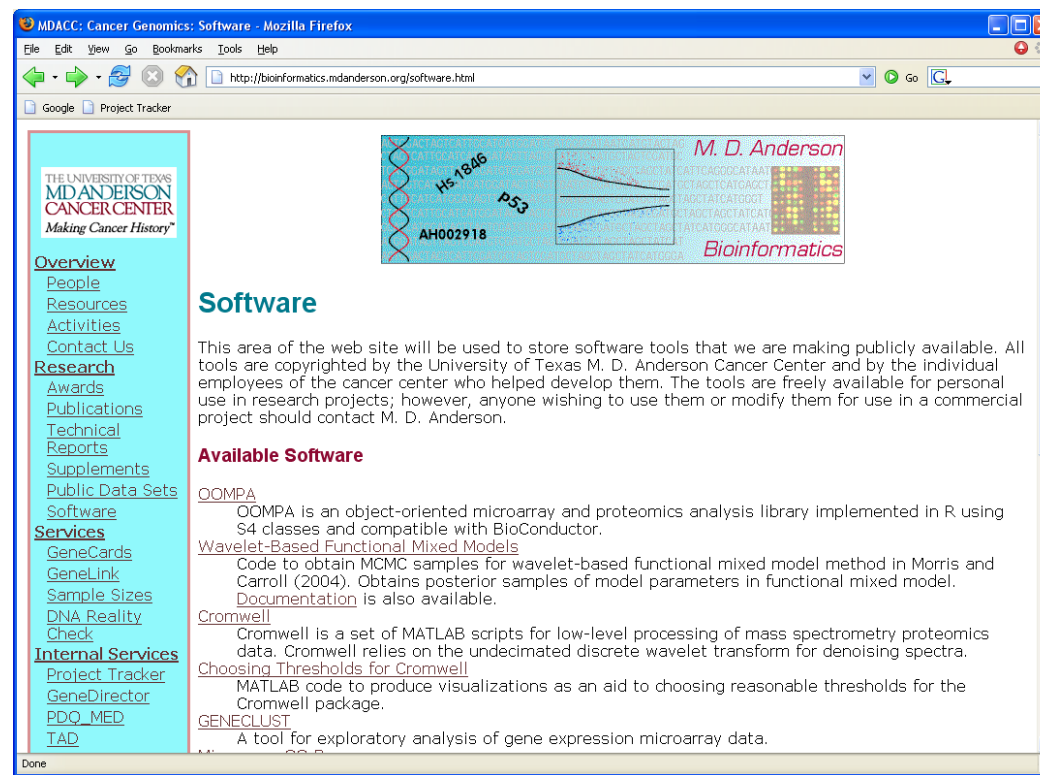




# R Libraries for Microarray Analysis

We have created R libraries that make it easier for statisticians to perform bootstrap validation of clusters.

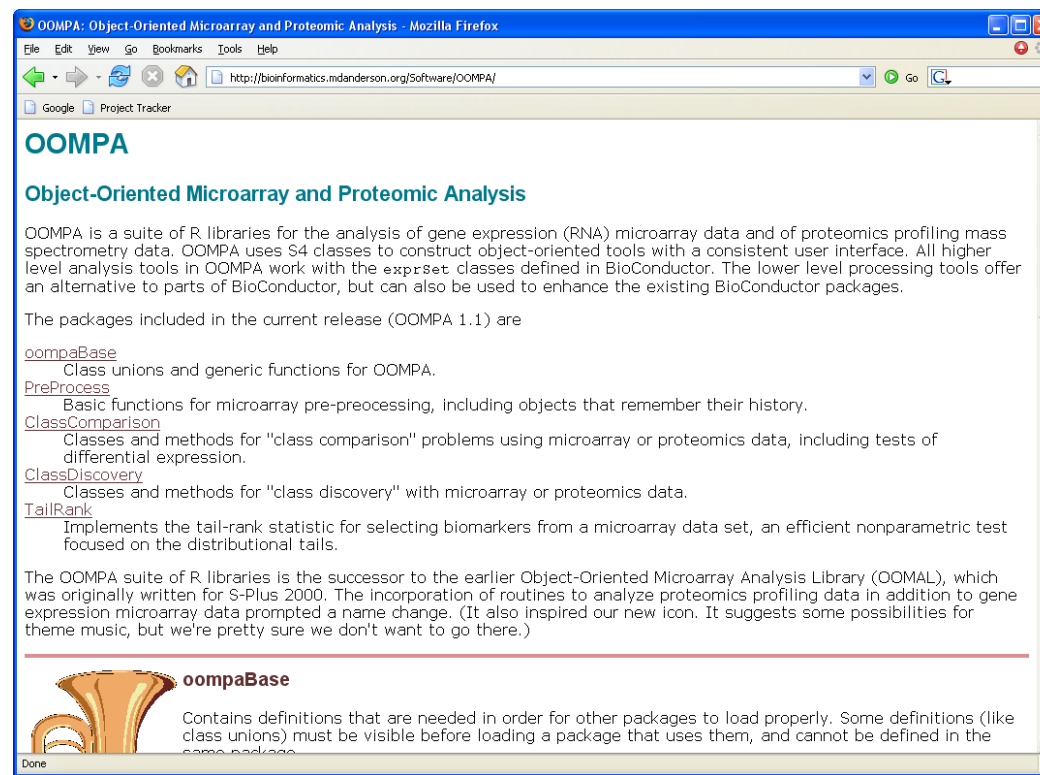
`http://bioinformatics.mdanderson.org/software.html`



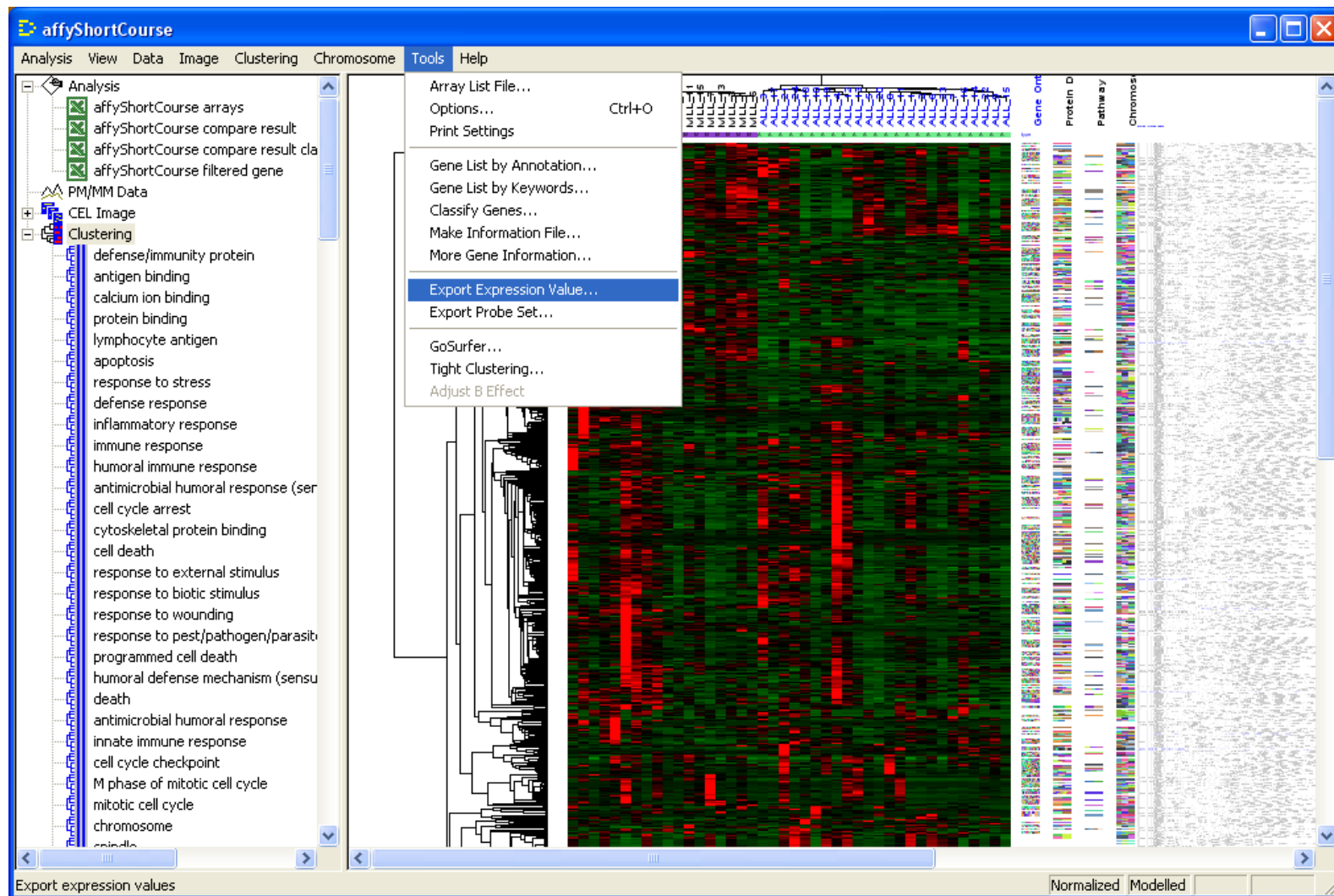
# OOMPA

Follow the link for Object-Oriented Microarray and Proteomic Analysis. Then get the libraries.

`http://bioinformatics.mdanderson.org/  
Software/OOMPA/`

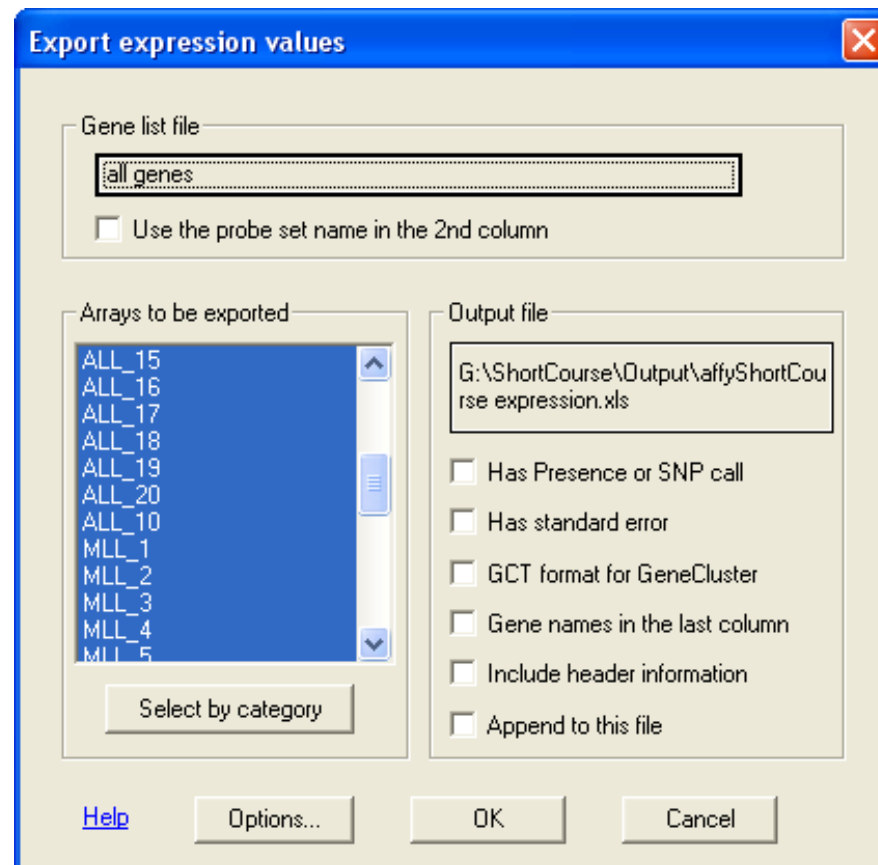


# Exporting the data from dChip



# Exporting the data from dChip

Change the “Gene list file” to “all genes”, or select the file from filtering genes or from comparing samples. Uncheck the boxes.



## Using the Exported Data

The exported data lives in a tab-separated values file with an “.xls” extension (so that Excel will open it easily). This can be read directly into R using a `read.table` command. If you prepared a sample information file, that can also be read into R using another `read.table` command. The bootstrap clustering routines can then be used on the real data.

Correlation distance, average linkage, 4 clusters, 200 bootstrap samples.

# Conclusions

1. Hierarchical clustering always finds clusters.
2. Bootstrap resampling can show that the clusters are fake.
3. Filtering to show a particular grouping can lead to incorrect results.
4. Hierarchical clustering with bootstrap validation may uncover real structure.