Unsupervised Genetics

Data Computing

Computing project

There are many different kinds of cancer, often given the name of the tissue in which they originate: lung cancer, ovarian cancer, prostate cancer, and so on.

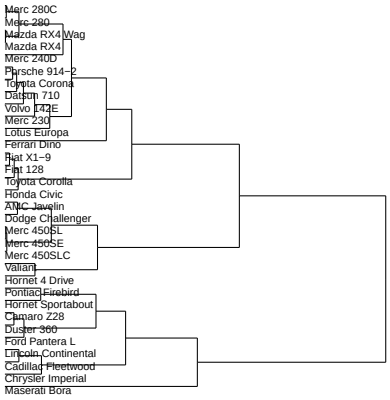
In this exercise, you are going to look for possible relationships among different cancer types. The basis for this will be the data in `NCI60`, which records the level of gene expression indicated by each of 41,078 probes against each of 60 different cell lines collected from different people with different kinds of cancer.

Looking at so many probes is a kind of shotgun approach enabled by *microarray* chips. It's assumed that only a few genes might be involved in any given type of cancer; by looking at the expression of many genes, it's hoped to be able to identify those few.

Clustering the cancer types

The point of clustering is to identify pairs or sets of cases that are similar. In *hierarchical clustering*, the pairs or sets are themselves grouped together in terms of similarity. The result is a kind of tree, a *dendrogram*, that shows which cases and sets of cases are similar.

As an example, here is a dendrogram constructed for the different cars in 'mtcars'. There are two major groups of cars. From the base of the tree, two branches identify two major groups of cars. Each of those groups is subdivided, with the process repeating until reaching a *leaf*: an individual case. The level of the horizontal line connecting two sub-branches indicates the distance between those sub-branches. For example, the Honda Civic, Toyota Corolla, and Fiat 128 are quite similar.



Glyph-ready data for constructing a dendrogram is in the usual tidy form, with the cases being the objects to cluster and the variables being the means to judge similarity between cases. The 'mtcars' data table is already in glyph-ready form; take a look! Note that the identifiers for the object, e.g. "Datsun 710," are not stored in a variable but in the *row names* of the data table.

mpg	cyl	disp	hp	drat
21.00	6	160.00	110	3.90
21.00	6	160.00	110	3.90
22.80	4	108.00	93	3.85
21.40	6	258.00	110	3.08
18.70	8	360.00	175	3.15
... and so on for 32 rows				

Table 1. `mtcars`

The process is as follows:

1. Find the distances from each case to every other case. The `dist()` function accomplishes this.

```
Dists <- dist(mtcars)
```

2. Apply the clustering algorithm to make the dendrogram.

```
Dendrogram <- hclust(Dists)
ddata <- dendro_data(Dendrogram)
```

3. Visualize the dendrogram

```
ggdendrogram(Dendrogram, rotate = TRUE) + geom_text(data = ddata$labels,
    aes(x = x, y = y, label = label), vjust = 0)
```

Wrangling the genetics data

The `NCI60` are not yet in a form suitable for clustering. There are several shortcomings.

1. The objects to be clustered — cell lines — are represented as variables instead of cases.
2. There are far too many probes to be able to identify "similarity" in a meaningful way. Recall that most of the probes are irrelevant.
3. The identifiers are not stored as the row names.

To start, put the data in a narrow form, like Table 2.

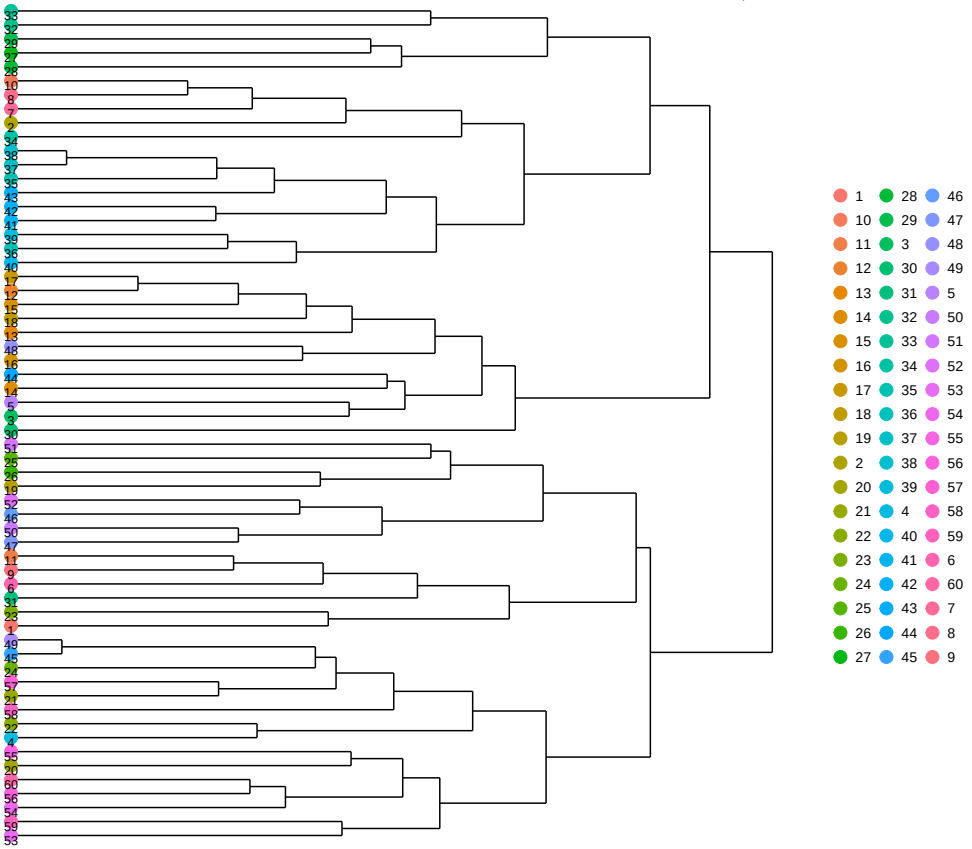
It happens that the same probe can appear multiple times on the gene chip. Combine all such repeats by grouping by `Probe` and `cellLine` and summarizing each group with `expression = mean(expression)`. Then use `ungroup()` on the result.[1]

In the narrow form, there's a simple way to pull out probes that might be of relevance to distinguishing among the different kinds of cancer. Presumably, for a relevant probe the expression will vary a lot among the cancer types. A very simple indicator of this is the *standard deviation* of each probe's expression. You can calculate this by grouping over `Probe` and summarising with `spread = sd(expression)`.

Identify a threshold for `spread` that identifies potential candidates for differential expression across cell lines and filter to keep only the probes above that threshold.

Wrangle the result into a wide format with the variables being the probes and the cases being the cell lines.

Cluster those cases and display your result. (You'll have to select out the `cellLine` variable in the data table you hand to `dist()`. The `dist()` function doesn't work with names like those in `cellLine`.)



Probe	cellLine	expression
3.8-1	BR.BT_549	-7.37
3.8-1	BR.HS578T	-7.30
3.8-1	BR.MCF7	-7.45
3.8-1	BR.MDA_MB_231	-7.51
3.8-1	BR.T47D	-7.74
... *and so on for 1,940,640 rows*		

Table 2. `NCI60` gathered into a narrow form. There is a row for each probe at each of the 60 cell lines.

[1] When you group on more than one variable, `summarise()` leaves the result partially grouped. This can interfere with later operations.

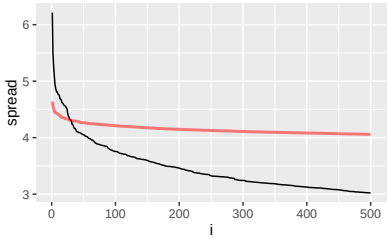


Figure 1: The spread of expression of probes across cell lines ordered from biggest to smallest for the leading 500 probes. For comparison, the red line shows the spread under the Null Hypothesis assumption that there is no relationship between probe and cell line.