Unsupervised Genetics

Daniel Kaplan

November 12, 2014

Unsupervised learning is an important part of machine learning. In unsupervised learning, you specify a set of variables and the machine looks for patterns.

One common type of pattern is clusters. You're going to investigate that with the `NCI60` data.

Recall that the `NCI60` data gives the level of expression for each of 41078 probes against each of 61 cell lines, collected from different people and different kinds of cancer.

As is often the case, most of the probes have little to say. The first step in unsupervised learning can be to apply a rule that you think might identify the important probes. Here, a starting rule is that important probes should have an expression that varies a lot across the cell lines.

To get you started, here are some data transfiguration step to create glyph-ready data for clustering. Read through the statements and figure out what they mean. (You can ignore the `mutate()` lines that fix a flaw in the original data.)

```
Long <- NCI60 %>%
  mutate( Probe=as.character(Probe )) %>%
  gather(value=expression, key=cellLine, -Probe)
# Average multiple entries when a probe appears multiple times
Long <- Long %>%
  mutate( cellLine=as.character(cellLine )) %>%
  group_by( cellLine, Probe ) %>%
  summarise( expression=mean(expression, na.rm=TRUE ))
# 50 biggest variance probes
Keepers <- Long %>%
  group_by( Probe ) %>%
  summarise( var=sd(expression, na.rm=TRUE)) %>%
  filter( row_number(desc(var)) <= 50 )
```

Once the high-variance probes have been identified, you can filter them out of the `NCI60` data.

```
ForClustering <- NCI60 %>%
  inner_join( Keepers %>% select(-var) ) %>%
  mutate( Probe=as.character(Probe) )
```

```
## Joining by: "Probe"
```

Make sure you understand what are the cases and variables of `ForClustering`.

You can cluster the rows easily. First, get rid of the `Probe` variable. The next, incomprehensible-looking expression, saves the contents of `Probe` so that it will appear in the plotted output.

```
rownames( ForClustering ) <-
  paste(ForClustering$Probe,1:nrow(ForClustering))
ForClustering <- ForClustering %>%
  select( -Probe )
```

Compute the distances between all pairs of rows:

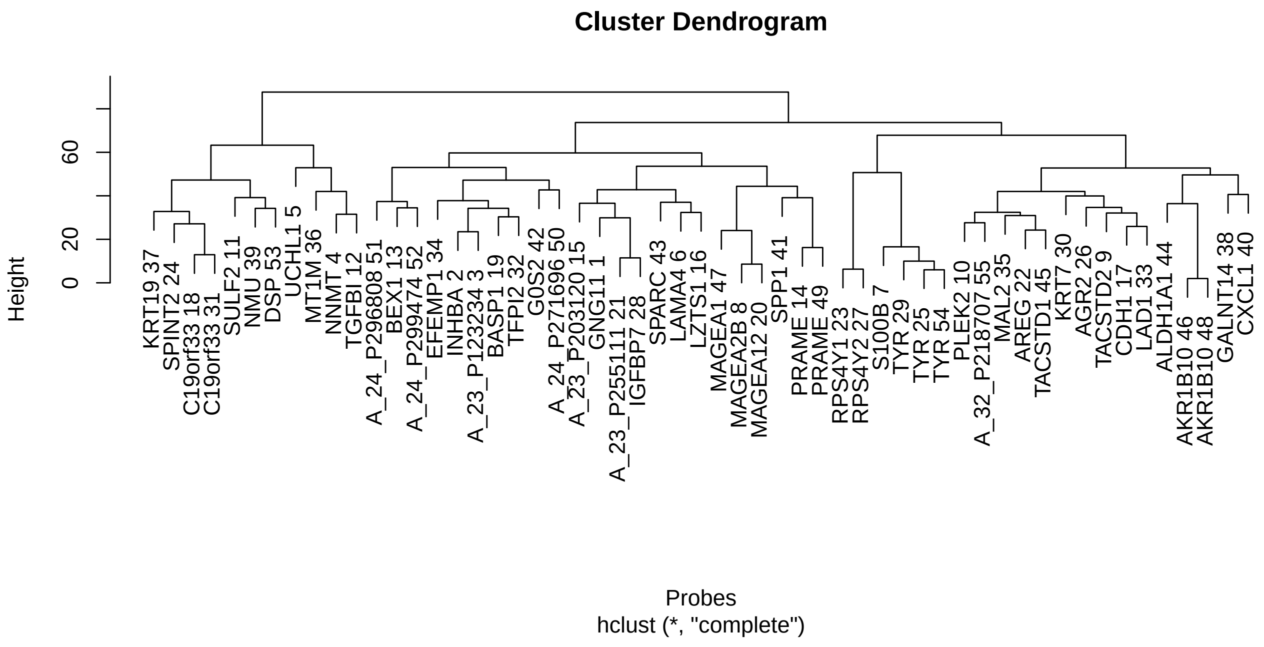
```r
PairDistances <- dist( ForClustering )
```

Construct the clusters:

```r
Clusters <- hclust(PairDistances)
```

And plot . . .

```r
plot( Clusters, xlab="Probes" )
```



You might prefer, however, to cluster the cell lines. If the probes contain information about the cell line, it might be that you'll see related cell lines being clustered closely together.

The cases are the rows, which are the individual probes. The next statement will flip the data table over so that the rows become the columns and the columns the rows.

```r
Flipped <- t(ForClustering) %>% data.frame()
```

Now you can find the pair-wise distances and cluster. In interpreting the tree, keep in mind that the cancer type of each cell line is encoded in the first two or three letters of the line's name: OV is ovarian, RE is renal, LC is lung cancer, BR is breast, CNS is brain, CO is colon, ME is melanoma.

```r
Finally <- dist( Flipped )
ClustFinally <- hclust( Finally )
plot( ClustFinally)
```