Part 9: Clustering

Summary

Here, we will focus on the general problem of clustering, the challenge of dividing objects into K groups such that similar objects are assigned to the same group.

Why Cluster?

Making predictions is a very natural problem; most people can understand why it is useful to study methods for supervised learning. Applications for unsupervised learning can be less obvious.

There can be distinct groups present in data sets. This grouping can provide insight into the structure of the data, and guide further analyses (in the same way that stocks in different "sectors" may be best modeled separately).

Clustering can also help in the identification of outliers.

The following examples are from finance. Note that I am not making any claims as to the legitimacy and/or accuracy of their analyses; I am just providing examples of ways in which clustering techniques could be used.

Pavlidis, et al. (2006): "Financial Forecasting through Unsupervised Clustering and Neural Networks," *Operational Research*.

The authors use clustering to divide the predictor space into regions of greater homogeniety, and fit distinct (neural network) models in each of these.

"Although global approximation methods can be applied to model and forecast time series . . ., it is reasonable to expect that forecasting accuracy can be improved if regions of the input space exhibiting similar dynamics are identified and subsequently a local model is constructed for each of them."

Lee, et al. (2010): "An Effective Clustering Approach to Stock Market Prediction," *PACIS 2010 Proceedings*

The authors use clustering to divide the financial reports into homogeneous groups, and then take the cluster center as a "representative."

"When a financial report is released, we will transform it into a feature vector . . . [and] we assign [it] to the nearest representative feature vector. Then, we predict the direction of the stock price movement according to the class label of the nearest representative feature vector."

In this framework, the clustering serves as a "smoothing" operation on "nonstandard" data (the financial report).

Financial report		d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
Qualitative features	efficient	1	1	0	1	0	0	1	0	0
	growth	1	1	0	0	0	0	0	0	0
	advantage	1	1	1	0	0	0	0	0	0
	improvement	1	0	0	0	0	0	0	0	0
	deficient	0	0	0	1	1	0	0	0	0
	reorganize	0	0	0	1	1	0	0	1	1
	difficulty	0	0	0	1	1	1	1	0	0
	complaint	0	0	0	1	0	0	0	0	1
Quantitative features	operating margin	0.4	0.38	0.37	0.1	0.07	0.08	0.05	0.06	0.03
	ROE	0.3	0.28	0.27	0.01	0.04	0.04	0.07	0.02	0.05
	ROTA	0.25	0.23	0.22	0.02	0.05	0.07	0.04	0.01	0.04
	equity to capital	0.8	0.78	0.77	0.45	0.4	0.5	0.45	0.5	0.55
	receivables turnover	2.5	2.4	2.45	1.4	1.3	1.5	1.2	1.1	1.5
Class label		1	1	0	-1	-1	-1	0	-1	-1

Table 1. An example dataset.

Figure 1: From Lee, et al. (2010)

Miceli and Susinno (2003): "Using Trees to Grow Money," *Risk*

The authors cluster hedge fund performance time series in order to "visualize a taxonomy embedded in synchronous historical data." They state that "[i]n a universe with low transparency from an investor's point of view, and where operating strategies are protected from full disclosure . . . instruments of classification . . . would allow the investor to verify the declared strategies in the statements issued by hedge funds. In particular, nodes anomolous to a reference cluster help to identify funds that require a deeper investigation."

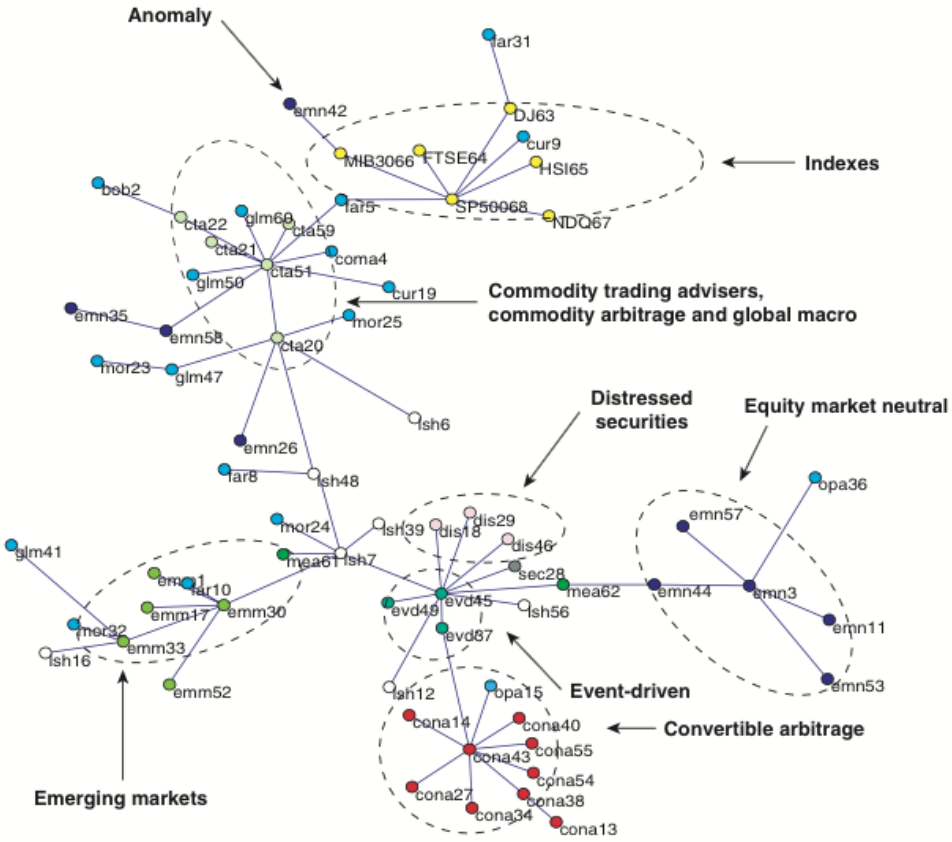


Figure 2: From Miceli and Susinno (2003)

Das (2003): "Hedge Fund Classification using K-means Clustering Method," *9th International Conference on Computing in Economics and Finance*.

The author proposes using clustering as a means of categorizing hedge funds, hence improving on the existing "myriad of classifications, some overlapping and some mutually exclusive."

Craighead and Klemesrud (2002): "Stock Selection Based on Cluster and Outlier Analysis," Nationwide Financial, Columbus, OH.

The authors use clustering to identify outliers that could potentially create problems with portfolio selection algorithms.

K-Means Clustering

The K-means clustering algorithm is a relatively simple and intuitive approach to dividing the sample $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ into K distinct groups.

Despite repeated extensions and enhancements of the method and the growth of other clustering methods, K-means remains popular and useful.

The algorithm is based on a very natural measure of within-cluster heterogeneity. Observations are assigned to clusters in an effort to minimize this measure.

It is assumed that K is fixed by the user; the choice of K is a tricky issue.

I will adopt the notation used by ISL, Section 10.3.1.

First, define C_k to be the set of indices that belong to cluster k, for $k = 1, 2, \ldots, K$.

Each observation is assigned to exactly one cluster. In other words, $C_k \subseteq \{1, 2, \ldots, n\}$ with

$$\bigcup_{k=1}^{K} C_k = \{1, 2, \ldots, n\} \quad \text{and} \quad C_k, C_\ell \text{ disjoint for } k \neq \ell.$$

With K-means, the similarity within cluster k is measured by

$$W(C_k) = \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \overline{x}_{kj})^2$$

where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $\overline{\mathbf{x}}_k$ is the cluster centroid, calculated as the sample mean of the cluster members.

Exercise: Under what circumstance(s) will $W(C_k)$ be small?

K-means clustering seeks to find the allocation of the observations that minimizes

$$\sum_{k=1}^{K} W(C_k)$$

Comparing every possible allocation is not realistic. (Note that there is no restriction on the sizes of the clusters.)

Instead, the following algorithm is utilized: (Algorithm 10.1 in ISL.)

Randomly assign a number from 1 to K to each fo the observations. These are the initial cluster assignments. Then repeat until no changes are made to cluster assignments:

1. Find the centroid for each of the K clusters.
2. Re-assign each observation to the cluster whose centroid is closest as measured by Euclidean distance.

Comment: Note that the criterion is guaranteed to not increase with each iteration of the algorithm, but the search could get caught in a local minimum (and not the global minimum).

To address this problem, it is standard to repeat the algorithm several times with different (randomly chosen) starting allocations. If you keep track of the criterion achieved from each repetition, you can easily determine which allocation is the best.

K-means in R

K-means is implemented in R using the function `kmeans()`.

The syntax is simple:

```
> kmout = kmeans(datamat, centers = 5, nstart = 10)
```

Here I have specified that I want $K = 5$ clusters, and that I want the basic algorithm repeated ten times in order to increase the chances of finding the global minimum.

Note that `datamat` must be formatted so that each of the vectors to be clustered are stored in its `rows`. Use the transpose function `t()`, if needed.

Example: Clustering Stock Time Series

In this example, we will cluster our data set of random-chosen NYSE stocks, with data taken from August 18, 2017 to September 29, 2017. This is a total of 30 trading days.

The data can be read in using the command

```
> stocksample = read.table("stocksample.txt", header=T,
+                  sep="\t", comment.char="")
>
> stocksamplescl = apply(stocksample[,5:34],1,scale)
```

Note that the data are formatted such that each row is a different stock; the columns are the different days. As before, each time series has been scaled so that it has a mean of zero and variance of one. It would be difficult to compare the series without this transformation.

The call to `kmeans()` is below. The matrix needs to be transposed:

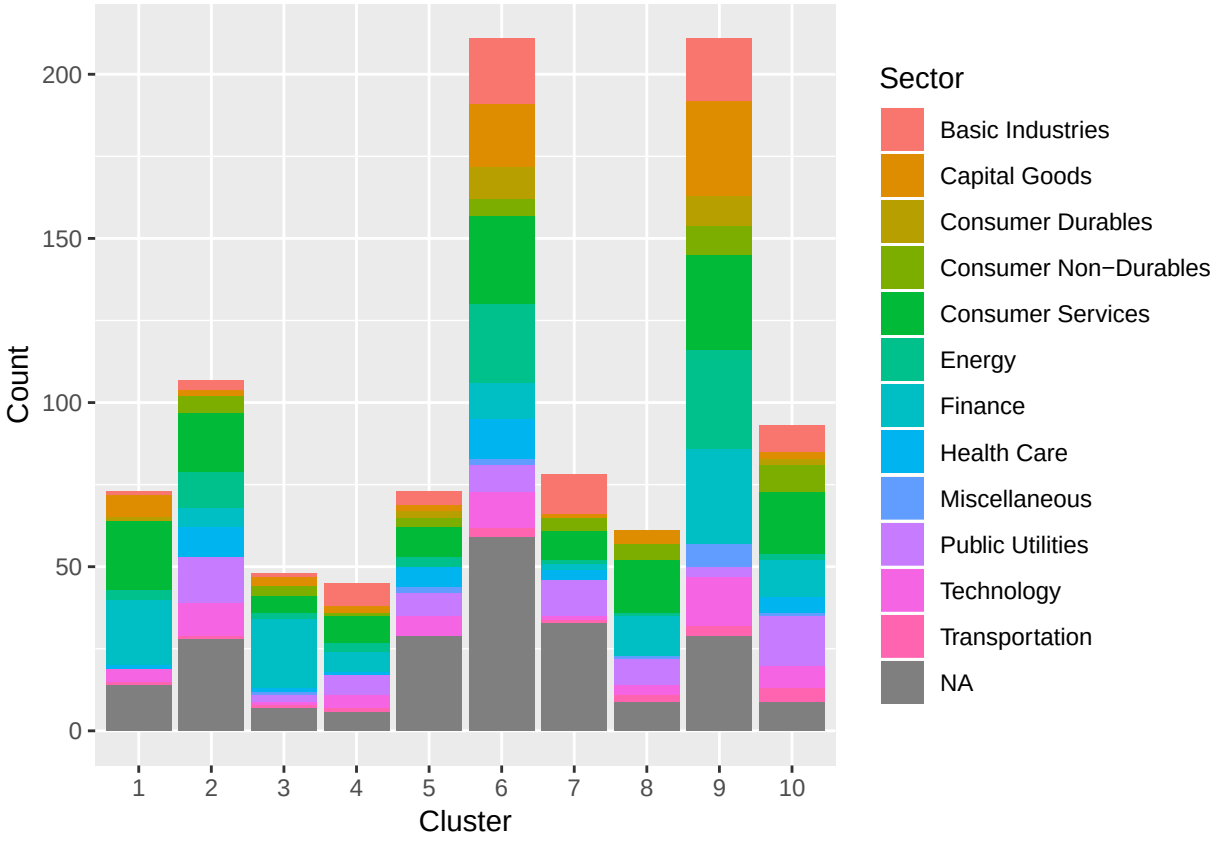
```
> kmout = kmeans(t(stocksamplescl), centers = 10,
+         nstart = 10)
```

Some comments on the output of `kmeans()`:

1. The rows of `kmout$centers` holds the centers of the K clusters. In our example, each center is a vector of length 30.
2. `kmout$cluster` specifies the cluster that each of the observations has been assigned to.
3. The value of $W(C_k)$ for each of the clusters can be found from `kmout$withinss`.
4. The size of each cluster is found from `kmout$size`.

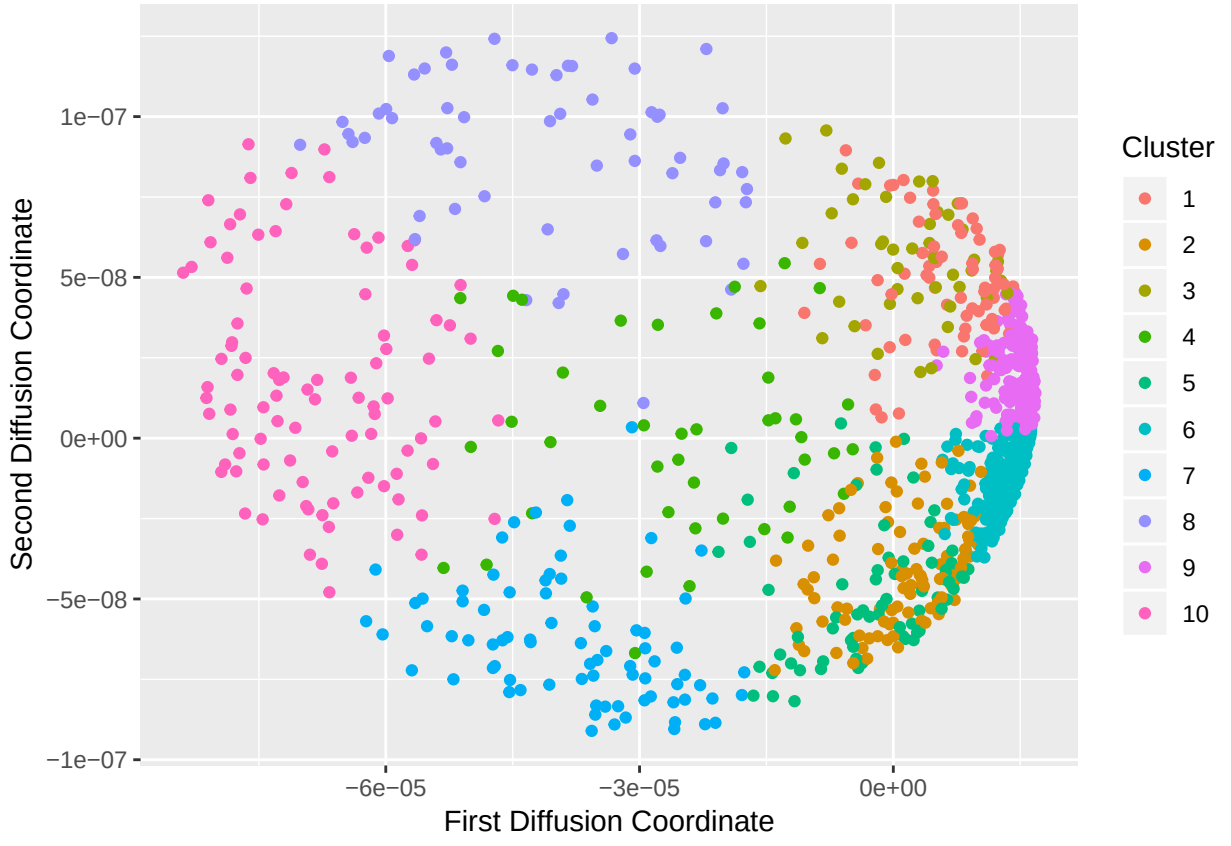
1 We can inspect how the clusters vary with respect to sector:

```
> stocksample$kmclust = factor(kmout$cluster)
>
> ggplot(stocksample, aes(x=kmclust,fill=sector)) +
+   geom_bar() +
+   labs(x="Cluster",y="Count", fill="Sector")
```


Count
200
150
100
50
0
1
2
3
4
5
6
7
8
9
10
Cluster
Sector
Basic Industries
Capital Goods
Consumer Durables
Consumer Non−Durables
Consumer Services
Energy
Finance
Health Care
Miscellaneous
Public Utilities
Technology
Transportation
NA

It is also interesting to compare K-means with the output of a dimension reduction technique seen previously:

```
> stockdistmat = dist(t(stocksamplescl))
> stockdiffmap = diffuse(stockdistmat, eps.val=50, t=10)
> stocksample$dmap1 = stockdiffmap$X[,1]
> stocksample$dmap2 = stockdiffmap$X[,2]
>
> ggplot(stocksample,aes(x=dmap1,y=dmap2,color=kmclust)) +
+   geom_point() +
+   labs(x="First Diffusion Coordinate",
+        y="Second Diffusion Coordinate", color="Cluster")
```


Second Diffusion Coordinate
1e−07
5e−08
0e+00
−5e−08
−1e−07
−6e−05
−3e−05
0e+00
First Diffusion Coordinate
Cluster
1
2
3
4
5
6
7
8
9
10

Exercise: Would it seem sensible to run K-means directly on the low-dimensional representation, e.g., on the diffusion coordinates?

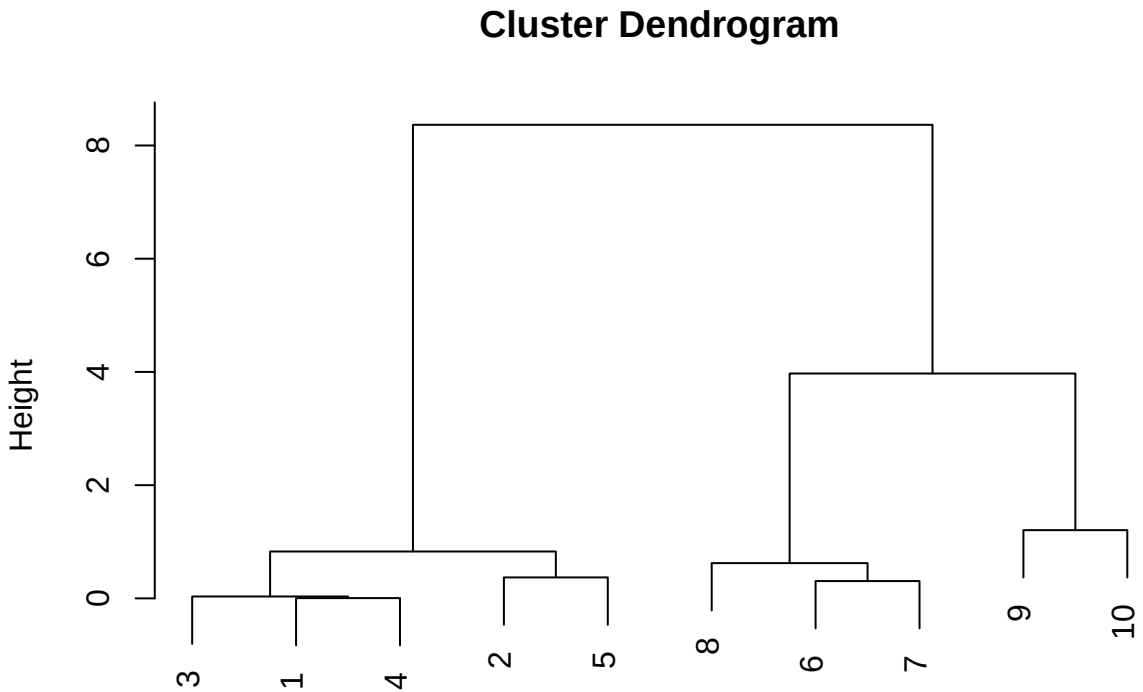
Hierarchical Clustering

Another class of clustering procedures, called hierarchical clustering, are characterized by the distinctive dendrograms they create to depict the relationships between observations.

An appealing feature of these methods is that the creation of the dendrogram does not require the specification of the number of clusters. Sometimes, the dendrogram my reveal an "obvious" division among the observations, and hence suggest the number of clusters needed.

This clustering method allows the user to specify a measure of dissimilarity. Often, Euclidean distance is used (just as in K-means).

It is useful to begin by understanding how to interpret the dendrogram. An example is given below.



Observations which are similar are "linked" at a low height in the dendrogram. For instance, to link observations 6 and 7, you need only go up to a height of approximately 0.5. These two observations are quite similar. But, in order to link observations 2 and 10, you need to go to a height of over 8. These are quite dissimilar.

The units for "height" match the units of the dissimilarity metric that the user provides.

Exercise: Which pair are more similar: 8 and 6, or 9 and 10?

Clusters can be formed by "cutting" the dendrogram at a user-chosen height. It may seem "clear" by looking at the dendrogram that cutting at around a height of 3 would be a good choice. The results is three clusters.

Note that in this case I generated the ten observations from the normal distribution with a standard deviation of 0.5. The mean for observations 1 thru 5 was 0, for observations 6 thru 8 was 5, and for observations 9 and 10 was 7. This explains the structure in the dendrogram.

Building the Dendrogram

The process of constructing the dendrogram is quite intuitive.

First, the two observations which are the most similar (as measured by the chosen dissimilarity metric) and joined together, at a height equal to their dissimilarity. In our example above, observations 1 and 4 were the first to be joined, at a height very close to zero.

This process then continues, with the most similar objects joined together at the appropriate height. **But:** Once observations are joined, they are **treated as a unit** from that point on. For instance, in the second step in the construction of the dendrogram above, observation 3 was joined with the **unit consisting of 1 and 4**.

This process continues until all observations are joined.

The obvious question is thus: **How is the dissimilarity of collections of observations calculated?** There are a few different approaches. Two of the standard ones are as follows:

With complete linkage, the distance between sets of observations, S_i and S_j, is defined to be the **maximal** dissimilarity between all pairs $x \in S_i$ and $y \in S_j$.

With single linkage, the distance between sets of observations, S_i and S_j, is defined to be the **minimal** dissimilarity between all pairs $x \in S_i$ and $y \in S_j$.

Less-commonly used options are centroid linkage, in which the distance between S_i and S_j is calculated as the dissimilarity between the centroids of the two groups, and average linkage, calculated as the average distance between all pairs of elements in S_i and S_j.

Hierarchical Clustering in R

The function `hclust()` performs hierarchical clustering in R.

Remember that this method is built around a user-chosen measure of dissimilarity. Hence, the primary input argument to `hclust()` is a symmetric n by n distance matrix. The diagonal of this matrix should consist of zeros.

The syntax for complete linkage clustering is as follows:

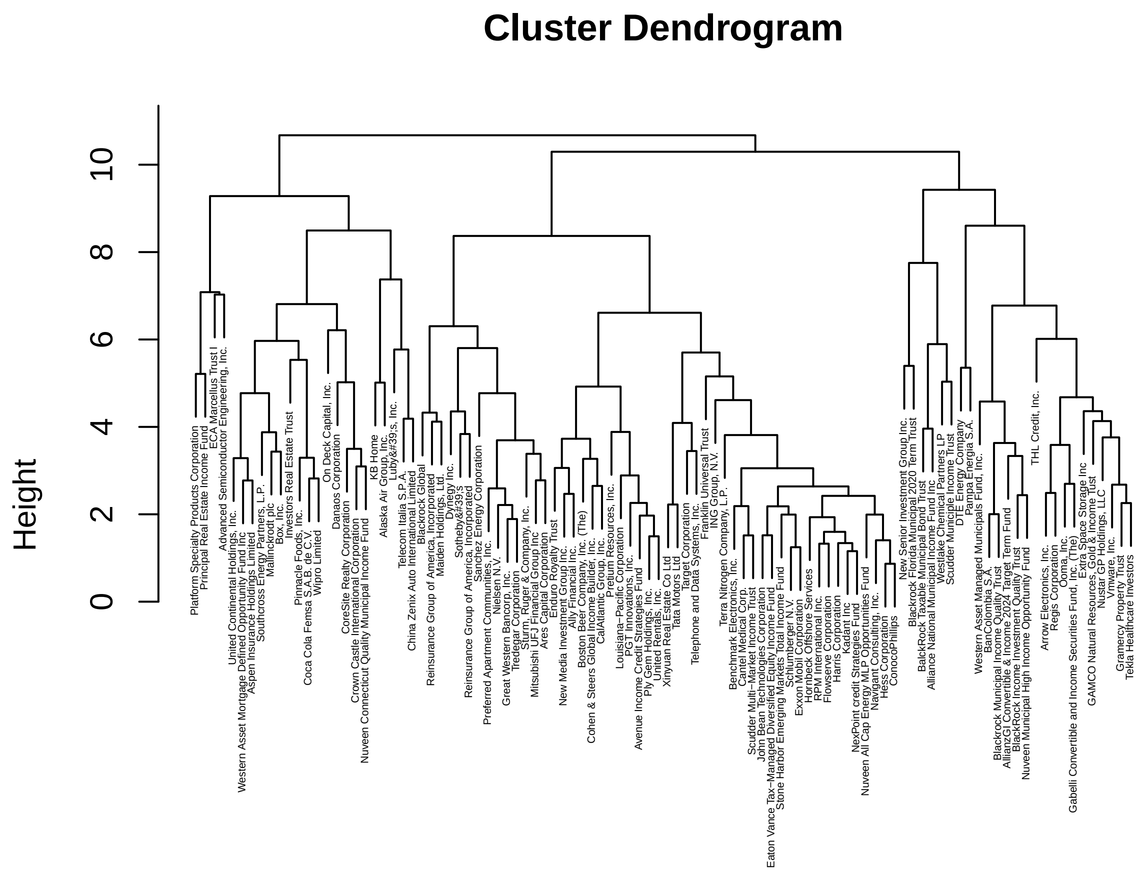
```
> hcout = hclust(dist(t(stocksamplescl[,1:100])),
+                method="complete")
```

(Note that only the first 100 stocks are used, since the dendrogram becomes difficult to read otherwise. There is no such practical limit in using this method, however.)

Other choices for `method` include `single`, `centroid`, and `average`.

The `plot()` function, when applied to the output of `hclust()`, produces the dendrogram:

```
> plot(hcout, labels=stocksample$name[1:100],
+           cex=0.35,sub="", xlab="")
```


Cluster Dendrogram
Height
0
2
4
6
8
10

Another useful function is `cutree()`. This will create return cluster membership for each of the observation, based on a desired number of clusters, or on a desired height.

For instance, to cut our dendrogram into six clusters, we specify `k=6`:

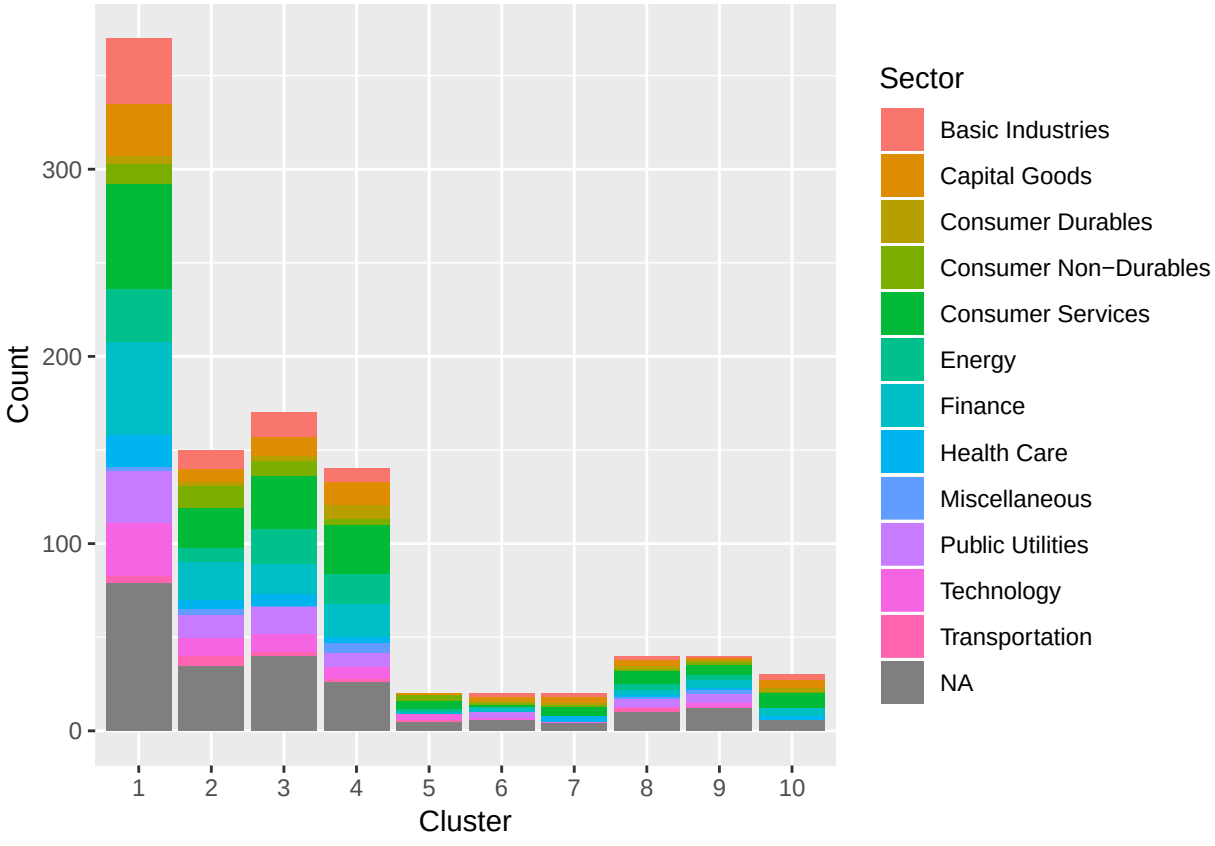
```
> cutree(hcout, k=6)[1:10]
 [1] 1 2 3 2 2 1 1 1 1 2
```

And to cut our our dendrogram at a height of 10, specify `h=10`:

```
> cutree(hcout, h=10)[1:10]
 [1] 1 2 3 2 2 1 1 1 1 2
```

1 We can again inspect how the clusters vary with respect to sector:

```
> stocksample$hcclust = factor(cutree(hcout,10))
>
> ggplot(stocksample, aes(x=hcclust,fill=sector)) +
+   geom_bar() +
+   labs(x="Cluster",y="Count", fill="Sector")
```


Count
300
200
100
0
1
2
3
4
5
6
7
8
9
10
Cluster
Sector
Basic Industries
Capital Goods
Consumer Durables
Consumer Non−Durables
Consumer Services
Energy
Finance
Health Care
Miscellaneous
Public Utilities
Technology
Transportation
NA