Unsupervised Learning II

36-600

The Setting

- The setting for *unsupervised learning* is that you have...
  - ...a collection of p measurements (recorded in columns of a data frame)...
  - ...for each of n objects (recorded in rows of a data frame)
- "Unsupervised" simply means that none of the variables is a response variable: we are not trying to predict the value of a measurement
- One can think of unsupervised learning as being an extension of EDA
  - in EDA, the goal is to visualize *projected* data to build intuition and to visually assess potential associations between variables
  - in unsupervised learning, we implement statistical algorithms to uncover potential structure in the data in their native space
- A main, overriding issue with unsupervised learning is that *there are no universally accepted mechanisms for model assessment or selection, i.e., there is not necessarily going to be a unique right answer!*

Hierarchical Clustering

- The algorithm for hierarchical clustering is, like that for K-means, straightforward:

Algorithm 10.2 *Hierarchical Clustering*

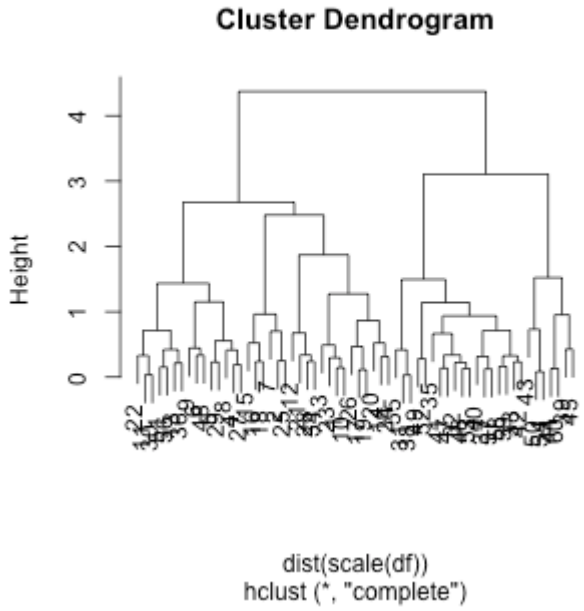
1. Begin with n observations and a measure (such as Euclidean distance) of all the $\binom{n}{2} = n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.
2. For $i = n, n-1, \ldots, 2$:
   (a) Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.
   (b) Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

- Note the "[t]reat each observation as its own cluster"
  - this is "bottom-up" or *agglomerative* clustering
  - a primary limitation of agglomerative clustering is that all clusters lie within other clusters (hence the name "hierarchical")

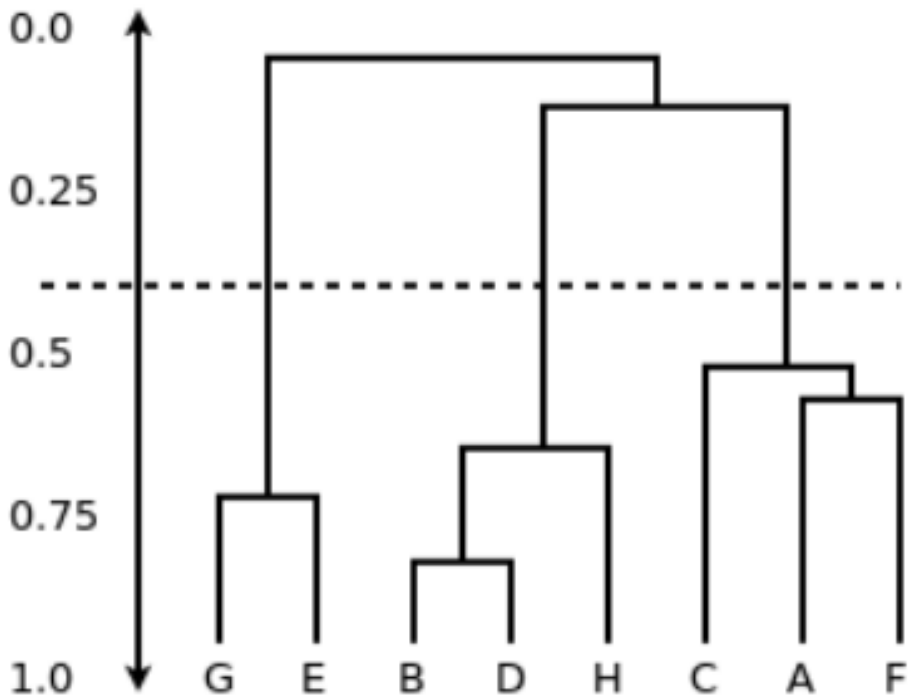
Hierarchical Clustering: Example

- Here's an example of a dendrogram for complete-linkage hierarchical clustering
    - the "height" along the vertical axis at which two clusters fuse indicates dissimilarity...the greater the vertical distance between merge points, the greater the dissimilarity between clusters

```
hc.out = hclust(dist(scale(df)),method="complete")  # we use the same data as we do for K-means
plot(hc.out)
```



Hierarchical Clustering: Tree Cutting



(Credit goes to Erik Velldal for the above image.)

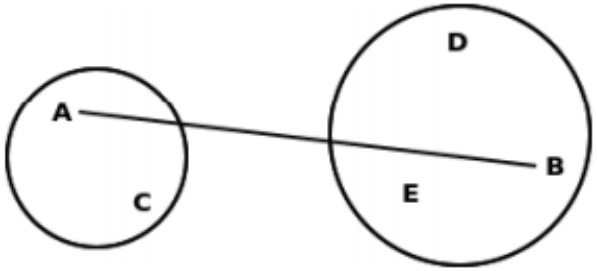
Hierarchical Clustering: Linkage

- In agglomerative clustering, clusters are built up piece by piece by linking them together
  - here is no unique algorithm for how that linking is done
  - commonly used methods are complete and average linkage

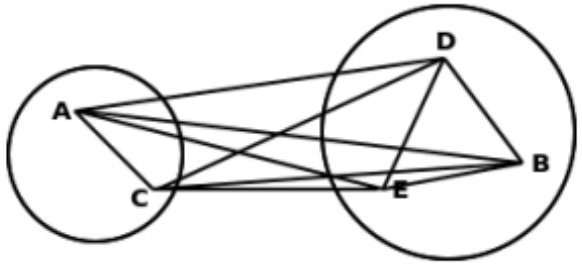
Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

Hierarchical Clustering: Linkage

- Complete linkage:

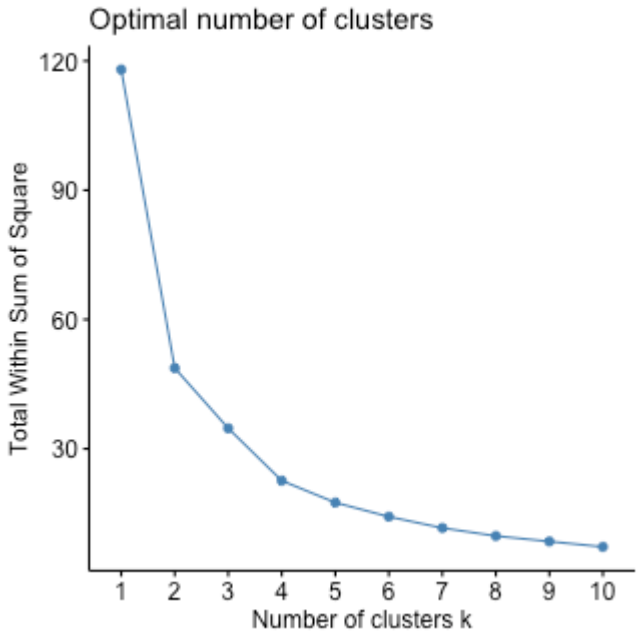


- Average linkage:



Hierarchical Clustering: the Elbow Method

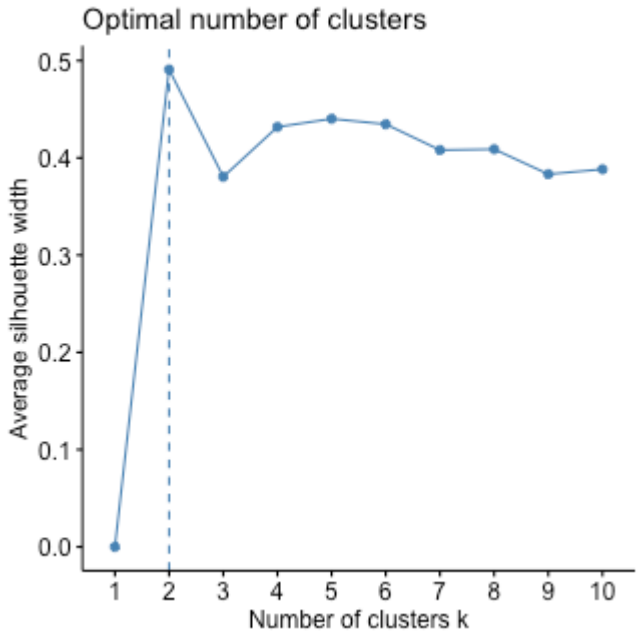
```
suppressMessages(library(factoextra))
fviz_nbclust(scale(df),FUN=hcut,method="wss")
```



- Note that in the examples we show in this set of slides, we will utilize `factoextra` package functions exclusively.

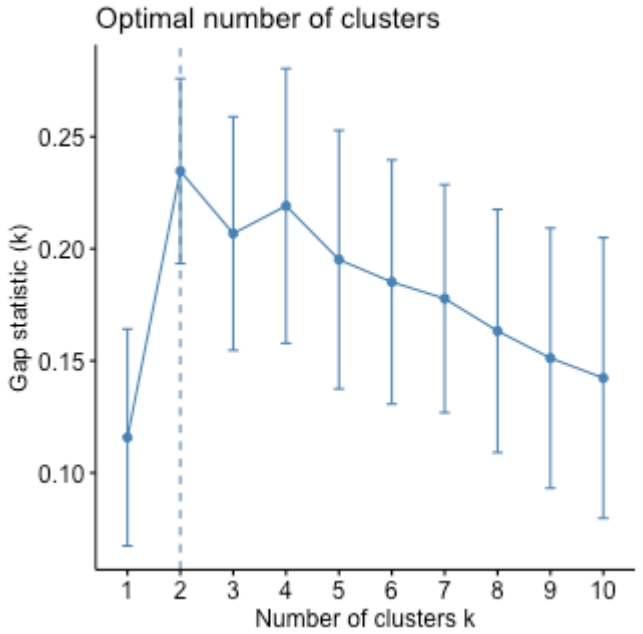
Hierarchical Clustering: the Silhouette Method

```
fviz_nbclust(scale(df),FUN=hcut,method="silhouette")
```



Hierarchical Clustering: the Gap Statistic

```
library(cluster)
gs <- clusGap(scale(df),FUN=hcut,nstart=20,K.max=10,B=50)
fviz_gap_stat(gs)
```

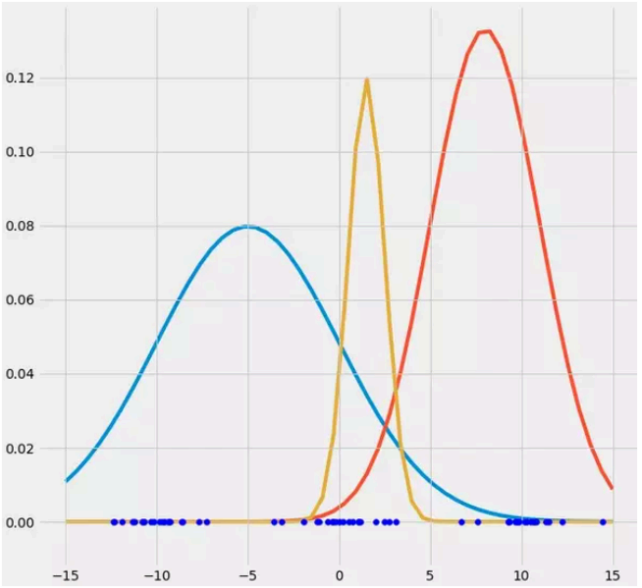


K-Means and Hierarchical Clustering: Wrap-Up

- So: which of these should you use?
- Quoting ISLR:
  - "we recommend performing clustering with different choices of [methods and parameters], and looking at the full set of results in order to see what patterns consistently emerge"
- That said:
  - in K-means, you specify the number of clusters before running the algorithm, as opposed to hierarchical clustering, where you specify the number of clusters afterwards by cutting across a dendrogram
  - dendrograms can be *really* hard to read when the sample size is large
  - all data are assigned to clusters in these algorithms

About That Last Point...

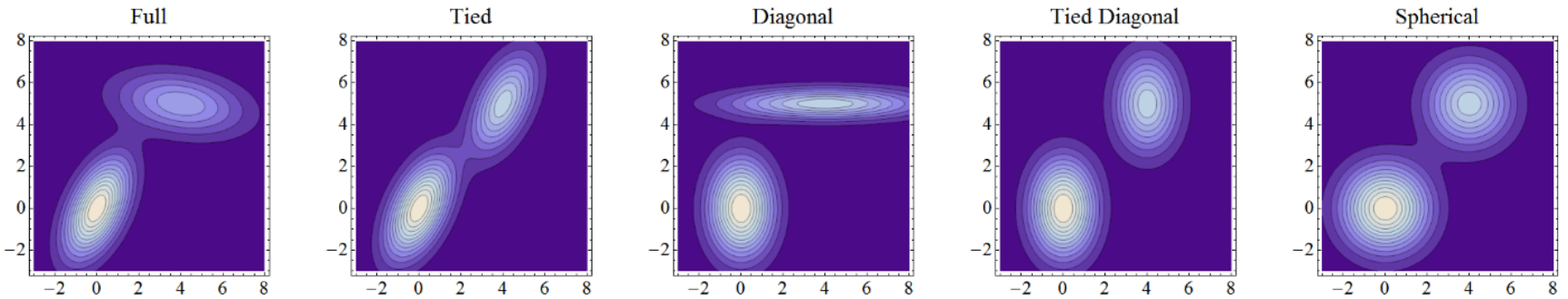
- Perhaps we would like to implement an algorithm that does not necessarily force every datum into a cluster
  - one example of such an algorithm is the Gaussian Mixture Model



(Courtesy this web page)

GMM: Example

- Here we will use the `ClusterR` package to determine the probabilities that data belong to one of a pre-defined number of Gaussian-shaped clusters
    - a limitation of this model is that the Gaussians are *diagonal*: the longest axis lies along the x or y axis, and is not allowed to rotate away from those axes



GMM: Example

```
suppressMessages(library(ClusterR))
gmm.out <- GMM(df,gaussian_comps=2)
pred    <- predict_GMM(df,gmm.out$centroids,gmm.out$covariance_matrices,gmm.out$weights)
gmm.out$centroids
```

```
##              [,1]       [,2]
## [1,] -0.07563208 0.04640021
## [2,]  2.05763239 2.33456223
```

```
names(pred)
```

```
## [1] "log_likelihood" "cluster_proba"  "cluster_labels"
```

- In this output:
  - `log_likelihood` is the natural logarithm of the probability density function values at each data point
  - `cluster_proba` is a matrix of probabilities: each row represents a data point, and each column the probability that the data point is in that cluster
  - `cluster_labels` is the predicted cluster...basically, it is the number of the column with the highest probability

GMM: Example

- Our data are drawn from two normal distributions, thus we expect that we should see that they map to one of the two normals in the GMM with high probability
    - the plot below shows the probability of a datum belonging to cluster 1