HT2015: SC4
Statistical Data Mining and Machine Learning

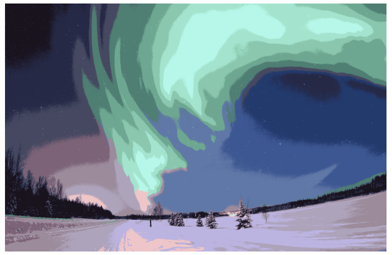
Dino Sejdinovic
Department of Statistics
Oxford

http://www.stats.ox.ac.uk/~sejdinov/sdmml.html

Clustering

Clustering

- Many datasets consist of multiple heterogeneous subsets.
- **Cluster analysis**: Given an unlabelled data, want algorithms that automatically group the datapoints into coherent subsets/clusters.
- Examples:
  - market segmentation
  - discovering communities in social networks
  - inferring population structures from genetic data
  - image segmentation / edge detection



Types of Clustering

- **Model-based** clustering:
  - Each cluster is described using a probability model.
- **Model-free** clustering:
  - Defined by **similarity/dissimilarity** among instances within clusters.

Types of Clustering

- **Exclusive** / **Partition-based** clustering methods:
  - Any instance belongs to one of K clusters.
  - The number of clusters is usually fixed beforehand or investigated for various values of K as part of the analysis.
- **Overlapping** clustering methods:
  - An instance may fall into several clusters.
- **Probabilistic** clustering methods:
  - An instance belongs to each cluster with a certain probability.
- **Hierarchical** clustering methods:
  - Allocate points into clusters, then clusters into super-clusters forming a hierarchy.
  - Typically the hierarchy forms a binary tree (a **dendrogram**) where each cluster has two “children” clusters.

K-means

Partition-based methods seek to divide data points into a pre-assigned number of clusters $C_1, \dots, C_K$ where for all $k, k' \in \{1, \dots, K\}$,

$$C_k \subset \{1, \dots, n\}, \qquad C_k \cap C_{k'} = \emptyset \ \ \forall k \neq k', \qquad \bigcup_{k=1}^{K} C_k = \{1, \dots, n\}.$$

For each cluster, represent it using a **prototype** or **cluster centroid** μ_k.
We can measure the quality of a cluster with its **within-cluster deviance**

$$W(C_k, \mu_k) = \sum_{i \in C_k} \|x_i - \mu_k\|_2^2.$$

The overall quality of the clustering is given by the total within-cluster deviance:

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

The overall objective is to choose both the cluster centroids and allocation of points to minimize the **objective function**.

K-means

$$W = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 = \sum_{i=1}^{n} \|x_i - \mu_{c_i}\|_2^2$$

where $c_i = k$ if and only if $i \in C_k$.

- Given partition $\{C_k\}$, we can find the optimal prototypes easily by differentiating W with respect to μ_k:

$$\frac{\partial W}{\partial \mu_k} = 2 \sum_{i \in C_k} (x_i - \mu_k) = 0 \qquad \Rightarrow \mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

- Given prototypes, we can easily find the optimal partition by assigning each data point to the closest cluster prototype:

$$c_i = \operatorname*{argmin}_k \|x_i - \mu_k\|_2^2$$

But joint minimization over both is computationally difficult.

K-means

The K-means algorithm is a widely used method that returns a **local optimum** of the objective function W, using iterative and alternating minimization.

1. Randomly initialize K cluster centroids $\mu_1, \dots, \mu_K$.
2. **Cluster assignment:** For each $i = 1, \dots, n$, assign each x_i to the cluster with the nearest centroid,

$$c_i := \operatorname*{argmin}_k \|x_i - \mu_k\|_2^2$$

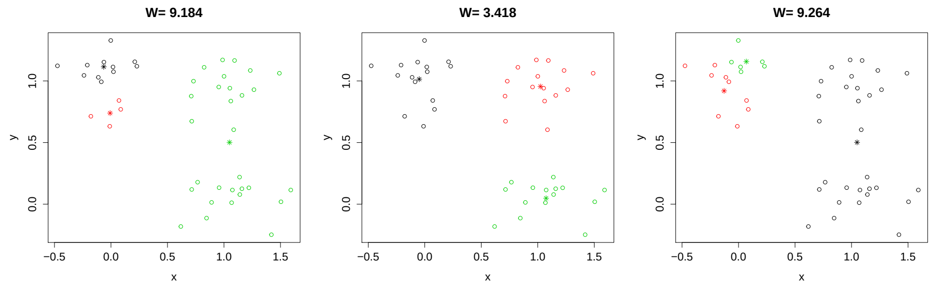
   Set $C_k := \{i : c_i = k\}$ for each k.
3. **Move centroids:** Set $\mu_1, \dots, \mu_K$ to the averages of the new clusters:

$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

4. Repeat steps 2-3 until convergence.
5. Return the partition $\{C_1, \dots, C_K\}$ and means $\mu_1, \dots, \mu_K$.

K-means

- **The algorithm stops in a finite number of iterations.** Between steps 2 and 3, W either stays constant or it decreases, this implies that we never revisit the same partition. As there are only finitely many partitions, the number of iterations cannot exceed this.
- **The K-means algorithm need not converge to global optimum.** K-means is a heuristic search algorithm so it can get stuck at suboptimal configurations. The result depends on the starting configuration. Typically perform a number of runs from different configurations, and pick the end result with minimum W. Multiple initializations effective especially for $K < 10$.

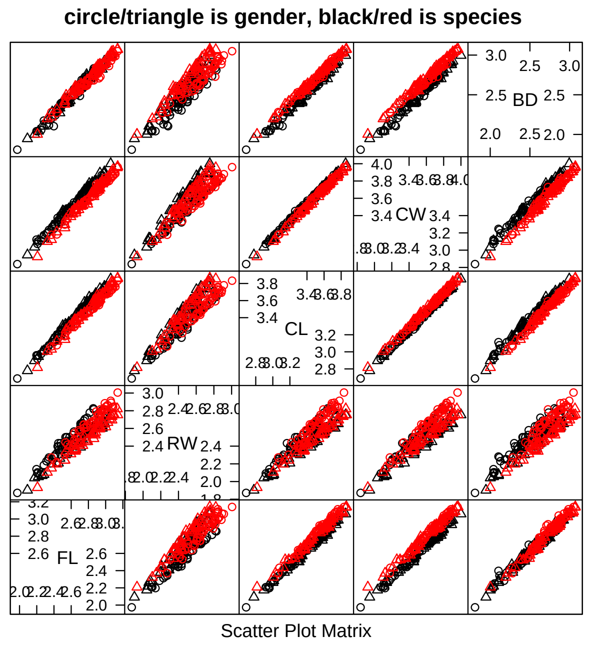


K-means on Crabs

Looking at the Crabs data again.

```
library(MASS)
library(lattice)
data(crabs)

splom(~log(crabs[,4:8]),
    col=as.numeric(crabs[,1]),
    pch=as.numeric(crabs[,2]),
    main="circle/triangle is gender, black/red is species")
```

K-means on Crabs

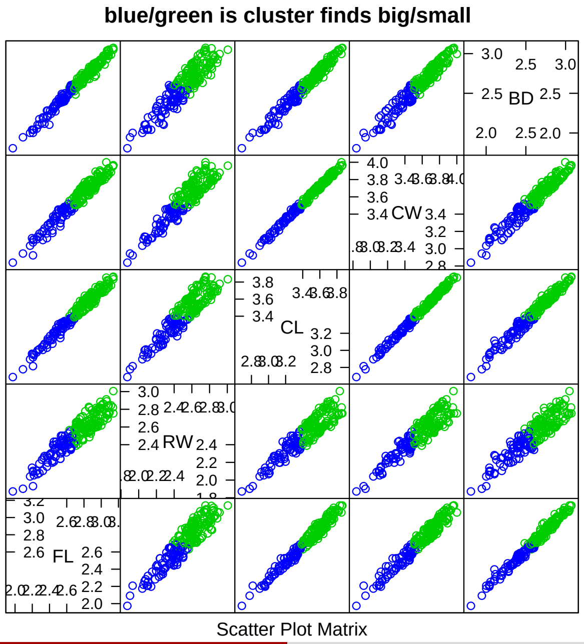


K-means on Crabs

Apply K-means with 2 clusters and plot results.

```
cl <- kmeans( log(crabs[,4:8]), 2, nstart=1, iter.max=10)

splom(~log(crabs[,4:8]),
      col=cl$cluster+2,
      main="blue/green is cluster finds big/small")
```

K-means on Crabs



K-means on Crabs

'Whiten' or 'sphere'[1] the data using PCA.

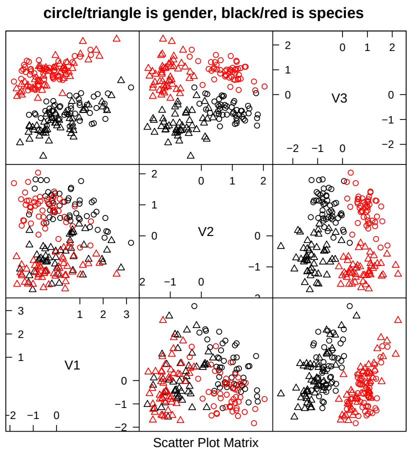
```
pcp <- princomp( log(crabs[,4:8]) )
spc <- pcp$scores %*% diag(1/pcp$sdev)
splom( ~spc[,1:3],
    col=as.numeric(crabs[,1]),
    pch=as.numeric(crabs[,2]),
    main="circle/triangle is gender, black/red is species")
```

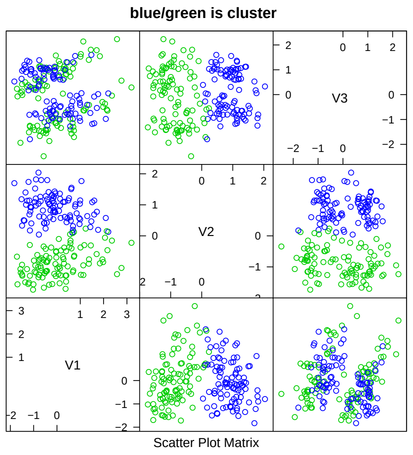
And apply K-means again.

```
cl <- kmeans(spc, 2, nstart=1, iter.max=20)
splom( ~spc[,1:3],
      col=cl$cluster+2, main="blue/green is cluster")
```

[1] Apply a linear transformation so that covariance matrix is identity.

K-means on Crabs

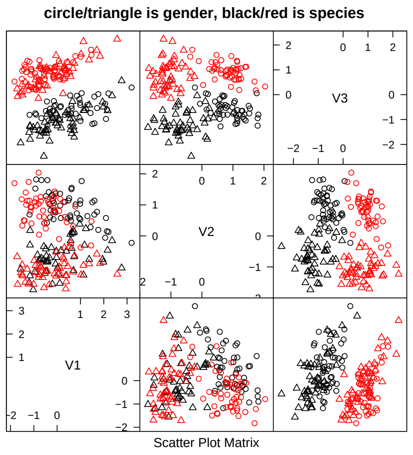


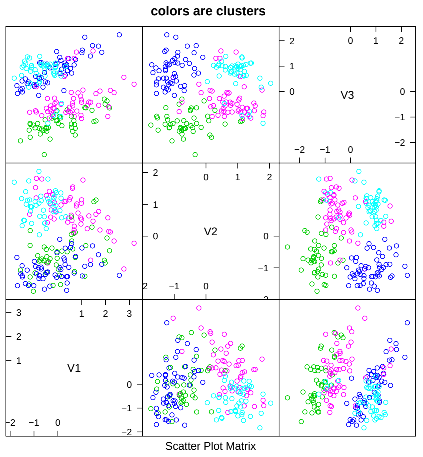


Discovers gender difference...
Results depends crucially on sphering the data first.

K-means on Crabs

Using 4 cluster centers.





K-means Additional Comments

- **Good practice initialization.** Randomly pick K training examples (without replacement) and set $\mu_1, \mu_2, \ldots, \mu_K$ equal to those examples
- **Sensitivity to distance measure.** Euclidean distance can be greatly affected by measurement unit and by strong correlations. Can use Mahalanobis distance instead:

$$\|x - y\|_M = \sqrt{(x-y)^\top M^{-1}(x-y)}$$

  where M is positive semi-definite matrix, e.g. sample covariance.
- **Determination of K.** The K-means objective will always improve with larger number of clusters K. Determination of K requires an additional **regularization** criterion. E.g., in DP-means[2], use

$$W = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 + \lambda K$$

[2]DP-means paper.

Other partition based methods

Other partition-based methods with related ideas:

- **K-medoids**[3]: requires cluster centroids μ_i to be an observation x_j
- **K-medians**: cluster centroids represented by a median in each dimension
- **K-modes**: cluster centroids represented by a mode estimated from a cluster

[3]See also Affinity propagation.

Stochastic Optimization

- Each iteration of K-means requires a pass through whole dataset. In extremely large datasets, this can be computationally prohibitive.
- Stochastic optimization: update cluster means after assigning each data point to the closest cluster.
- Repeat for $t = 1, 2, \ldots$ until satisfactory convergence:
  1. Pick data item x_i either randomly or in order.
  2. Assign x_i to the cluster with the nearest centroid,

$$c_i := \operatorname*{argmin}_k \|x_i - \mu_k\|_2^2$$

  3. Update cluster centroid:

$$\mu_{c_i} := \mu_{c_i} + \alpha_t(x_i - \mu_{c_i})$$

  where $\alpha_t > 0$ are **step sizes**.
- Algorithm stochastically minimizes the objective function. Convergence requires slowly decreasing step sizes:

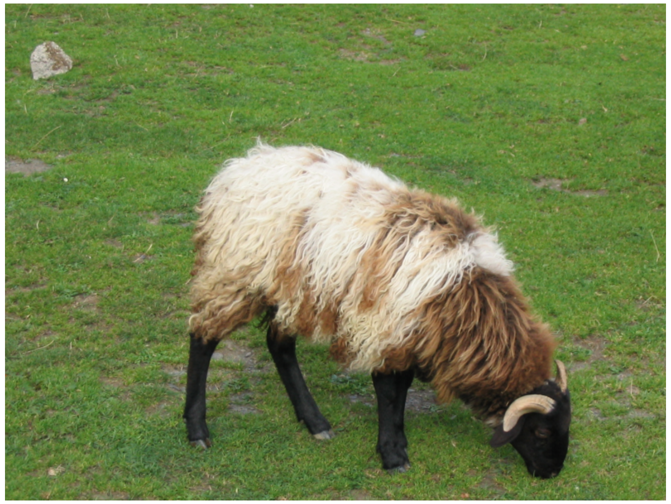
$$\sum_{t=1}^{\infty} \alpha_t = \infty \qquad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

Vector Quantization

- A related algorithm developed in the signal processing literature for **lossy data compression**.
- $\mathbf{X}$ represented by $n \times p$ real numbers
- Store instead:
  - the **codebook** of K **codewords** $\mu_1, \ldots, \mu_K$ ($K \times p$ real numbers)
  - for each vector x_i its cluster assignment c_i ($\lceil \log K \rceil \times n$ bits).
- As with K-means, K must be specified. Increasing K improves the quality of the compressed image but worsens the data compression rate, so there is a clear tradeoff.
- Some audio and video codecs use this method.
- Stochastic optimization algorithm for K-means was originally developed for VQ.

VQ Image Compression

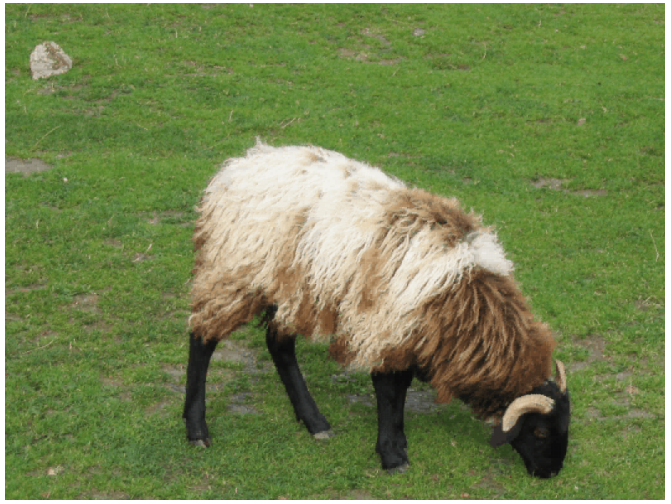
Original image (24 bits/pixel, uncompressed size 1,402 kB)



Each block of 3×3 pixels as a single observation $x_i \in \mathbb{R}^9$.

VQ Image Compression

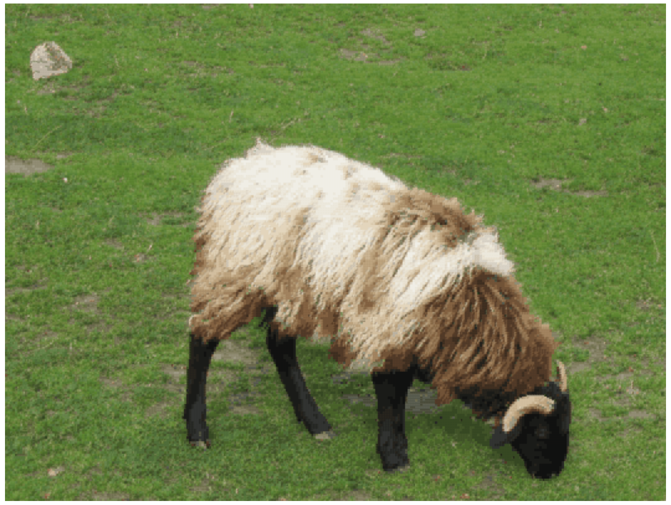
Codebook length 1024 (1.11 bits/pixel, total size 88kB)



Each block of 3×3 pixels as a single observation $x_i \in \mathbb{R}^9$.

VQ Image Compression

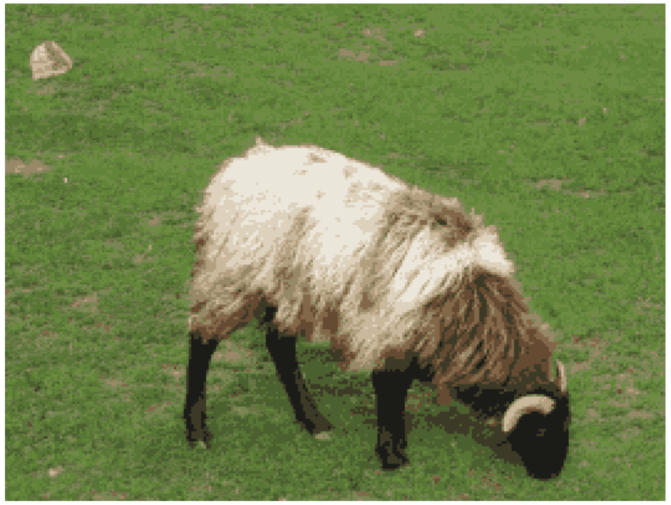
Codebook length 128 (0.78 bits/pixel, total size 50kB)



Each block of 3×3 pixels as a single observation $x_i \in \mathbb{R}^9$.

VQ Image Compression

Codebook length 16 (0.44 bits/pixel, total size 27kB)



Each block of 3×3 pixels as a single observation $x_i \in \mathbb{R}^9$.

Hierarchical Clustering

- Hierarchically structured data is ubiquitous (genus, species, subspecies, individuals...)
- There are two general strategies for generating hierarchical clusters. Both proceed by seeking to **minimize some measure of overall dissimilarity**.
  - Agglomerative / Bottom-Up / Merging
  - Divisive / Top-Down / Splitting
- Higher level clusters are created by merging clusters at lower levels. This process can easily be viewed by a tree/dendrogram.

`hclust, agnes{cluster}`

EU Indicators Data

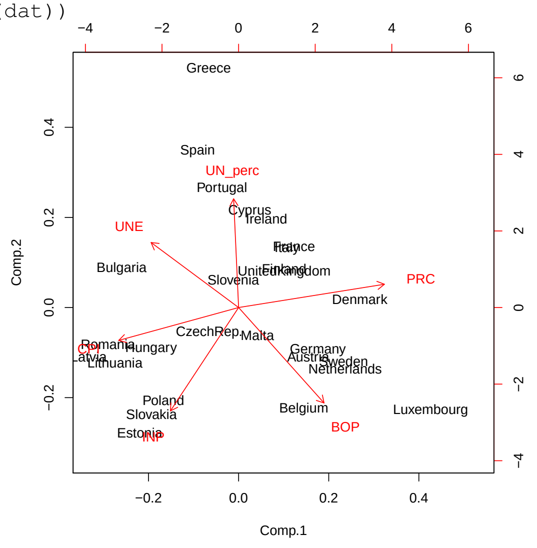
6 Economic indicators for EU countries in 2011.

```
> eu<-read.csv(
     'http://www.stats.ox.ac.uk/~sejdinov/teaching/data/eu_indicators.csv',sep=' ')
> eu[1:15,]
    Countries abbr    CPI   UNE    INP     BOP     PRC UN_perc
1     Belgium   BE 116.03  4.77 125.59   908.6  6716.5    -1.6
2    Bulgaria   BG 141.20  7.31 102.39    27.8  1094.7     3.5
3   CzechRep.   CZ 116.20  4.88 119.01  -277.9  2616.4    -0.6
4     Denmark   DK 114.20  6.03  88.20  1156.4  7992.4     0.5
5     Germany   DE 111.60  4.63 111.30   499.4  6774.6    -1.3
6     Estonia   EE 135.08  9.71 111.50   153.4  2194.1    -7.7
7     Ireland   IE 106.80 10.20 111.20  -166.5  6525.1     2.0
8      Greece   EL 122.83 11.30  78.22  -764.1  5620.1     6.4
9       Spain   ES 116.97 15.79  83.44  -280.8  4955.8     0.7
10     France   FR 111.55  6.77  92.60  -337.1  6828.5    -0.9
11      Italy   IT 115.00  5.05  87.80  -366.2  5996.6    -0.5
12     Cyprus   CY 116.44  5.14  86.91 -1090.6  5310.3    -0.4
13     Latvia   LV 144.47 12.11 110.39    42.3  1968.3    -3.6
14  Lithuania   LT 135.08 11.47 114.50   -77.4  2130.6    -4.3
15 Luxembourg   LU 118.19  3.14  85.51  2016.5 10051.6    -3.0
```

Data from Greenacre (2012)

EU Indicators Data

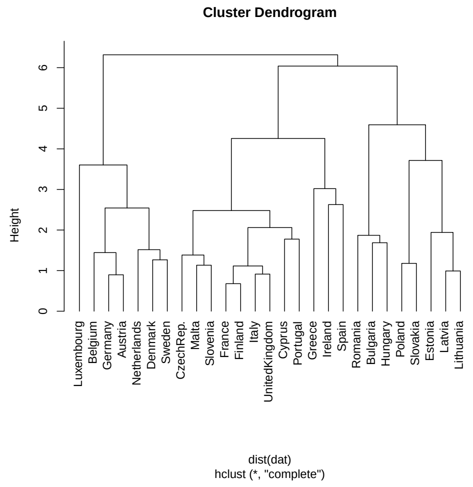
```
dat<-scale(eu[,3:8])
rownames(dat)<-eu$Countries
biplot(princomp(dat))
```

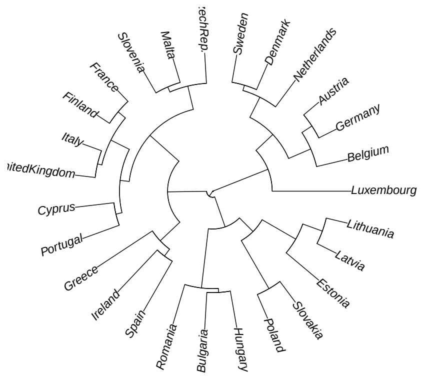


Visualising Hierarchical Clustering

```
> hc<-hclust(dist(dat))
> plot(hc,hang=-1)
```

```
> library(ape)
> plot(as.phylo(hc), type = "fan")
```





Visualising Hierarchical Clustering

Levels in the dendrogram represent a dissimilarity between examples.

- Tree dissimilarity d_{ij}^T = minimum height in the tree at which examples i and j belong to the same cluster.
- **ultrametric** (stronger than triangle) inequality:

$$d_{ij}^T \leq \max\{d_{ik}^T, d_{kj}^T\}.$$

- Hierarchical clustering can be interpreted as an approximation of a given dissimilarity d_{ij} by an ultrametric dissimilarity.

Measuring Dissimilarity Between Clusters

To join clusters C_i and C_j into super-clusters, we need a way to measure the dissimilarity $D(C_i, C_j)$ between them.
Many such proposals though no concensus as to which is best.

(a) **Single Linkage**: elongated, loosely connected clusters

$$D(C_i, C_j) = \min_{x,y} (d(x,y)|x \in C_i, y \in C_j)$$

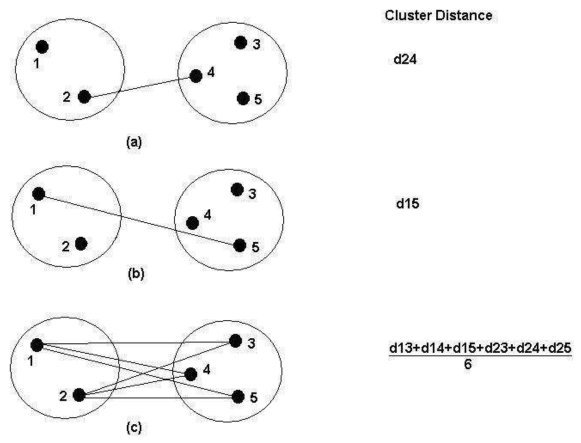
(b) **Complete Linkage**: compact clusters, relatively similar objects can remain separated at high levels

$$D(C_i, C_j) = \max_{x,y} (d(x,y)|x \in C_i, y \in C_j)$$

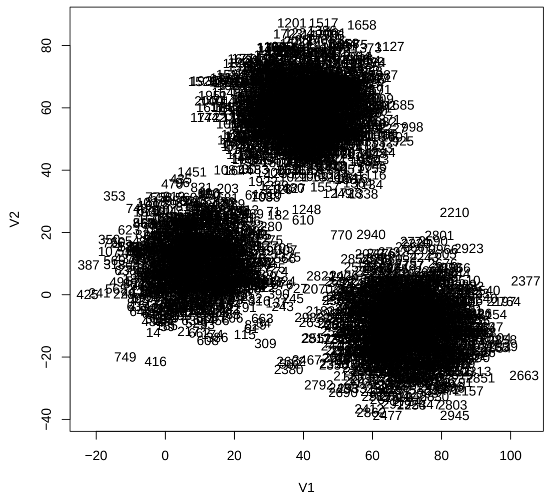
(c) **Average Linkage**: tries to balance the two above, but affected by the scale of dissimilarities

$$D(C_i, C_j) = avg_{x,y} (d(x,y)|x \in C_i, y \in C_j)$$

Measuring Dissimilarity Between Clusters



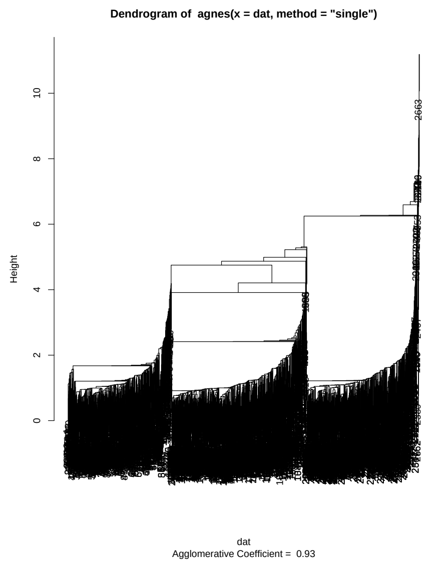
Hierarchical Clustering on Artificial Dataset



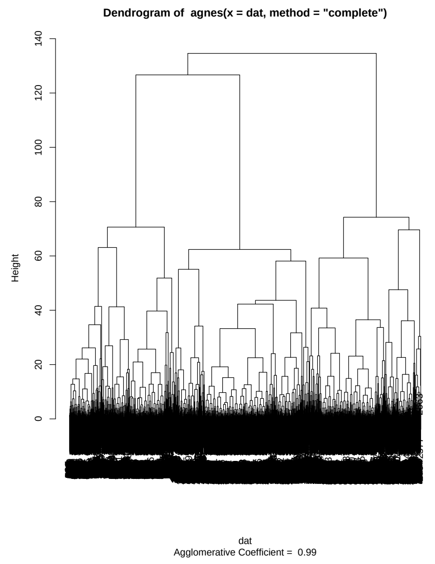
Hierarchical Clustering on Artificial Dataset

```
dat=xclara #3000 x 2
library(cluster)

#plot the data
plot(dat,type="n")
text(dat,labels=row.names(dat))

plot(agnes(dat,method="single"))
plot(agnes(dat,method="complete"))
plot(agnes(dat,method="average"))
```

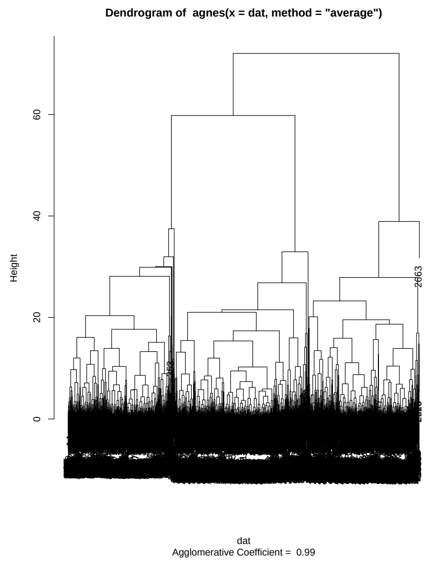
Hierarchical Clustering on Artificial Dataset



Hierarchical Clustering on Artificial Dataset



Hierarchical Clustering on Artificial Dataset



Using Dendrograms

- Different ways of measuring dissimilarity result in different trees.
- Dendrograms are useful for getting a feel for the structure of high-dimensional data though they don't represent distances between observations well.
- Dendrograms show hierarchical clusters with respect to increasing values of dissimilarity between clusters, cutting a dendrogram horizontally at a particular height partitions the data into disjoint clusters which are represented by the vertical lines it intersects. Cutting horizontally effectively reveals the state of the clustering algorithm when the dissimilarity value between clusters is no more than the value cut at.
- Despite the simplicity of this idea and the above drawbacks, hierarchical clustering methods provide users with interpretable dendrograms that allow clusters in high-dimensional data to be better understood.