Statistical Data Mining and Machine Learning Hilary Term 2016

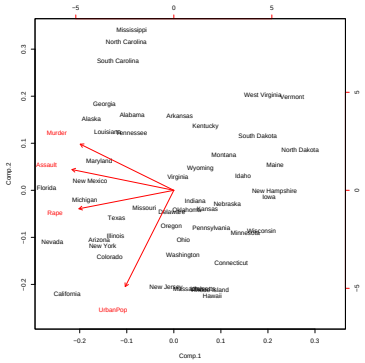
Dino Sejdinovic
Department of Statistics
Oxford

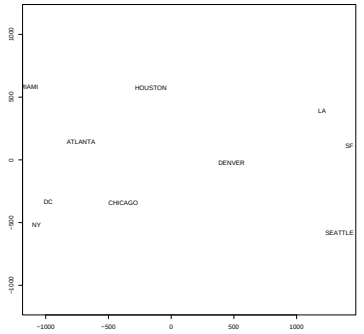
Slides and other materials available at:
http://www.stats.ox.ac.uk/~sejdinov/sdmml

Last time: **Biplots**, **MDS**

Biplots: a way to visualise relationships between principal components and the original variables. In its scaled version, projected points are uncorrelated and equi-variant and angles between variables correspond to correlation.

Multidimensional Scaling: a class of dimensionality reduction techniques which find a representation of the data items $x_1, \ldots, x_n \in \mathbb{R}^p$ in a lower-dimensional space $z_1, \ldots, z_n \in \mathbb{R}^k$ which approximately preserves the inter-point (dis)similarities.





Varieties of MDS

- Choices of (dis)similarities and stress functions $S(\mathbf{Z})$ lead to different algorithms.
  - **Classical/Torgerson**: preserves inner products instead - **strain function** (`cmdscale`)
$$S(\mathbf{Z}) = \sum_{i \neq j} (b_{ij} - \langle z_i - \bar{z}, z_j - \bar{z} \rangle)^2$$
  - **Metric Shephard-Kruskal**: preserves distances w.r.t. squared stress
$$S(\mathbf{Z}) = \sum_{i \neq j} (d_{ij} - \|z_i - z_j\|_2)^2$$
  - **Sammon**: preserves shorter distances more (`sammon`)
$$S(\mathbf{Z}) = \sum_{i \neq j} \frac{(d_{ij} - \|z_i - z_j\|_2)^2}{d_{ij}}$$
  - **Non-Metric Shephard-Kruskal**: ignores actual distance values, only preserves ranks (`isoMDS`)
$$S(\mathbf{Z}) = \min_{g \text{ increasing}} \frac{\sum_{i \neq j} (g(d_{ij}) - \|z_i - z_j\|_2)^2}{\sum_{i \neq j} \|z_i - z_j\|_2^2}$$

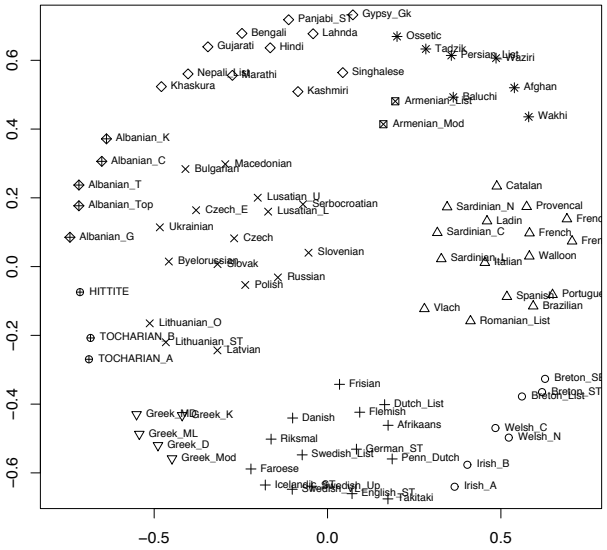
Example: Language data

Presence or absence of 2867 homologous traits in 87 Indo-European languages.

```
> X<-read.table("http://www.stats.ox.ac.uk/~sejdinov/sdmml/data/cognate.txt")
> X[1:15,1:16]
               V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16
Irish_A         0  0  0  0  1  0  0  0  0   0   0   0   0   0   0   0
Irish_B         0  0  0  0  1  0  0  0  0   0   0   0   0   0   0   0
Welsh_N         0  0  0  1  0  0  0  0  0   0   0   0   0   0   0   0
Welsh_C         0  0  0  1  0  0  0  0  0   0   0   0   0   0   0   0
Breton_List     0  0  0  0  1  0  0  0  0   0   0   0   0   0   0   0
Breton_SE       0  0  0  0  1  0  0  0  0   0   0   0   0   0   0   0
Breton_ST       0  0  0  0  1  0  0  0  0   0   0   0   0   0   0   0
Romanian_List   0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
Vlach           0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
Italian         0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
Ladin           0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
Provencal       0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
French          0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
Walloon         0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
French_Creole_C 0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
```

Example: Language data

Using MDS with non-metric (Sammon) scaling.

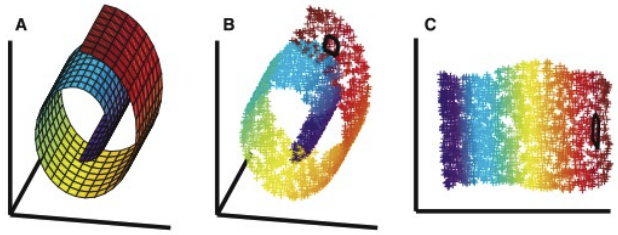


Nonlinear Dimensionality Reduction

Two aims of different varieties of MDS:

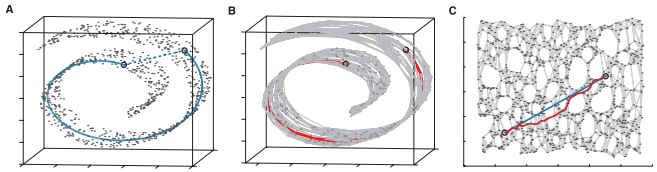
- To visualize the (dis)similarities among items in a dataset, where these (dis)disimilarities may not have Euclidean geometric interpretations.
- To perform **nonlinear** dimensionality reduction.

Many high-dimensional datasets exhibit low-dimensional structure ("live on a low-dimensional menifold").



Isomap

Isomap is a non-linear dimensional reduction technique based on classical MDS. Differs from other MDSs as it uses estimates of **geodesic distances** between the data points.

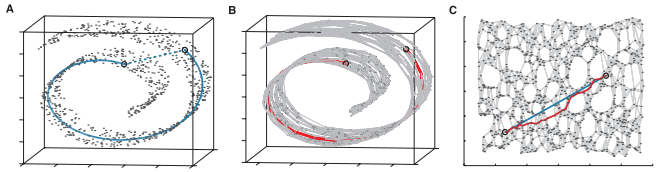


Tenenbaum et al. (2000)

Isomap

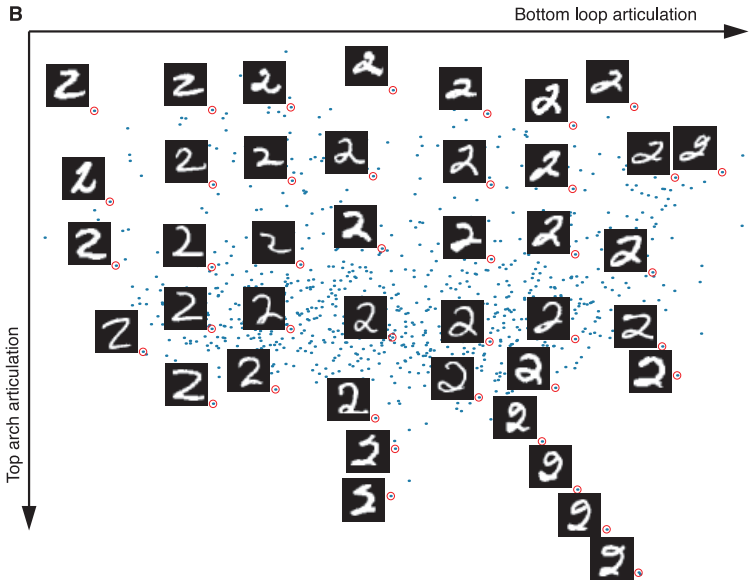
Isomap

- Calculate Euclidean distances d_{ij} for $i, j = 1, \ldots, n$ between all data points.
- Form a graph G with n samples as nodes, and edges between the respective K nearest neighbours (K-Isomap) or between i and j if $d_{ij} < \epsilon$ (ϵ-Isomap).
- For i, j linked by an edge, set $d_{ij}^G = d_{ij}$. Otherwise, set d_{ij}^G to the shortest-path distance between i and j in G.
- Run classical MDS using distances d_{ij}^G.

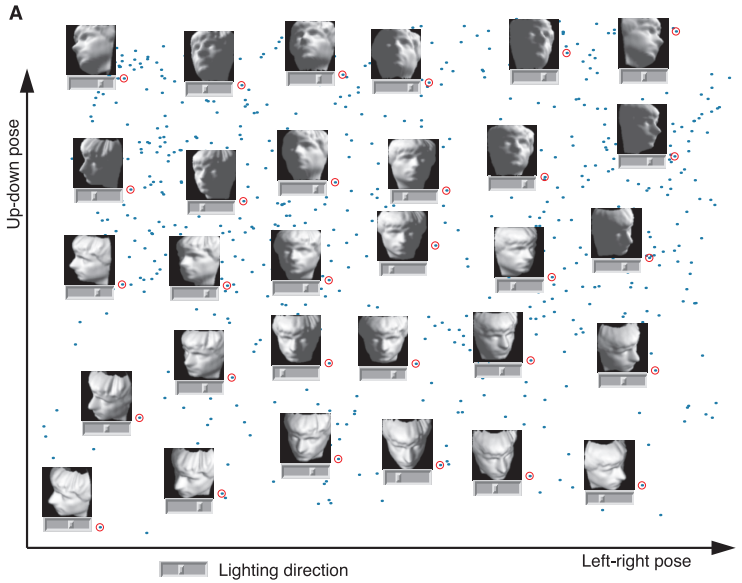


R function: `isomap{vegan}`.

Handwritten Characters



Faces



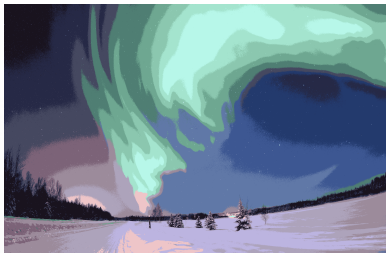
Nonlinear Dimensionality Reduction Techniques

- Kernel PCA
- Locally Linear Embedding
- Laplacian Eigenmaps
- Maximum Variance Unfolding

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 of shoppers based on browsing and purchase histories
  - different types of breast cancer based on the gene expression measurements
  - discovering communities in social networks
  - image segmentation



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.

This Lecture: Two "Model-free" Clustering Methods

- **K-means clustering**: a partition-based method into K clusters. Finds groups such that variation within each group is small. The number of clusters K is usually fixed beforehand or various values of K are investigated as a part of the analysis.
- **Hierarchical clustering**: nearby data items are joined 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. Dendrogram allows to view the clusterings for each possible number of clusters, from 1 to n (number of data items).

K-means

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

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

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:

$$c_i = \underset{k}{\operatorname{argmin}} \|x_i - \mu_k\|_2^2$$

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

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

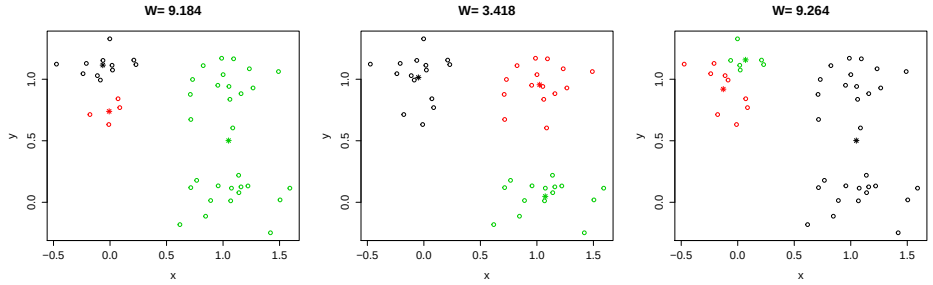
   Set $C_k := \{i : c_i = k\}$ for each k.
3. **Move centroids:** Set $\mu_1, \ldots, \mu_K$ to the averages of the new clusters:

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

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

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.



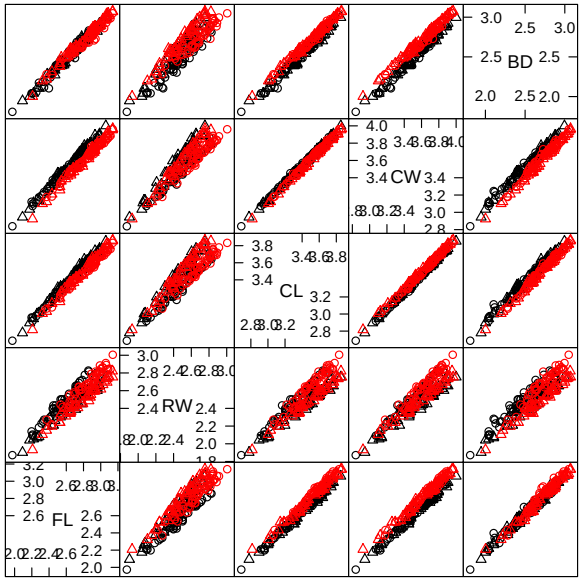
K-means on Crabs

Looking at the Crabs data again.

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

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

K-means on Crabs

circle/triangle is gender, black/red is species



Scatter Plot Matrix

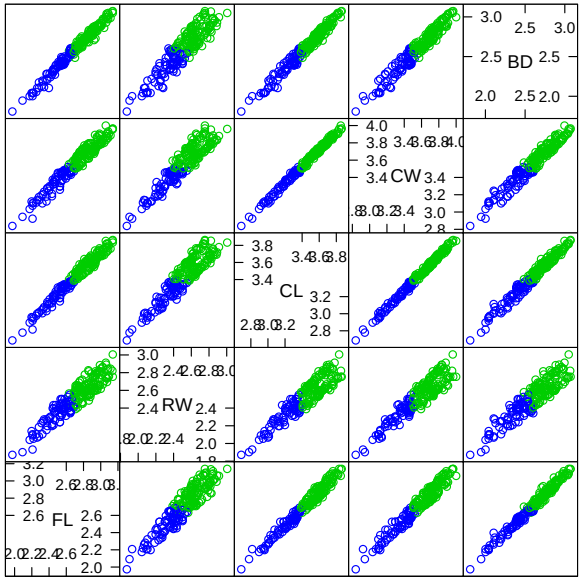
K-means on Crabs

Apply K-means with 2 clusters and plot results.

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

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


K-means on Crabs

blue/green is cluster finds big/small



Scatter Plot Matrix

K-means on Crabs

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

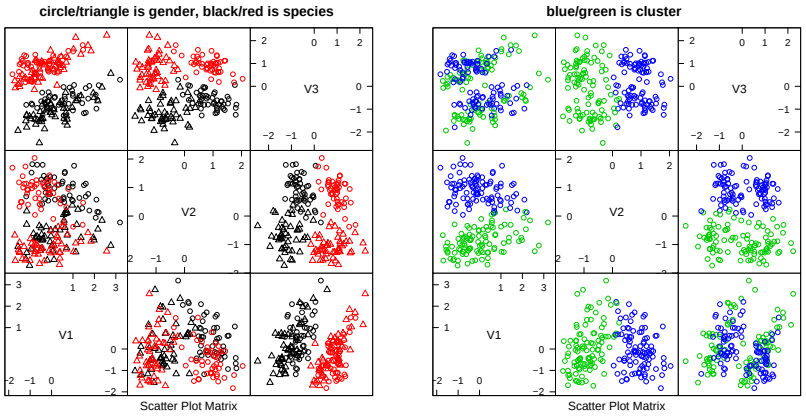
```
pcp <- princomp( log(crabs[,4:8]) )
Crabs.sphered <- pcp$scores %*% diag(1/pcp$sdev)
splom( ~Crabs.sphered[,1:3],
    col=as.numeric(crabs[,1]),
    pch=as.numeric(crabs[,2]),
    main="circle/triangle is gender, black/red is species")
```

And apply K-means again.

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

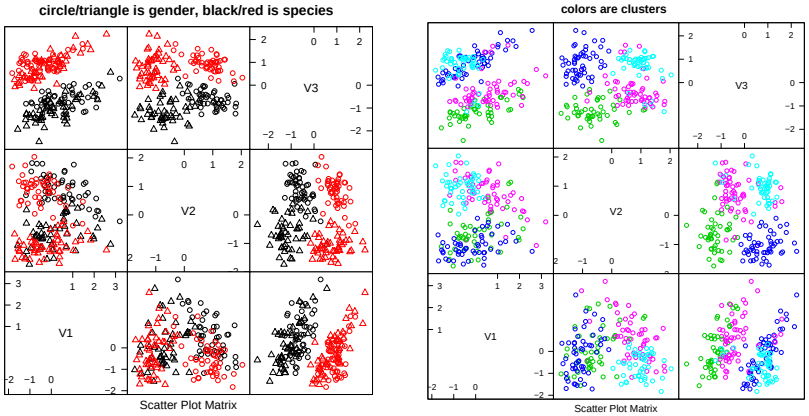
[1] Apply a linear transformation so that the covariance matrix is identity.

K-means on Crabs



Discovers gender difference...
But the result depends crucially on sphering the data first!

K-means on Crabs with $K = 4$




```
> table(Crabs.kmeans$cluster,Crabs.class)
   Crabs.class
    BF BM OF OM
1   3  0 41  0
2  39  8  6  0
3   8 42  0  0
4   0  0  3 50
```


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.

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.
- Avoids specifying how many clusters are appropriate.

`hclust, agnes{cluster}`

EU Indicators Data

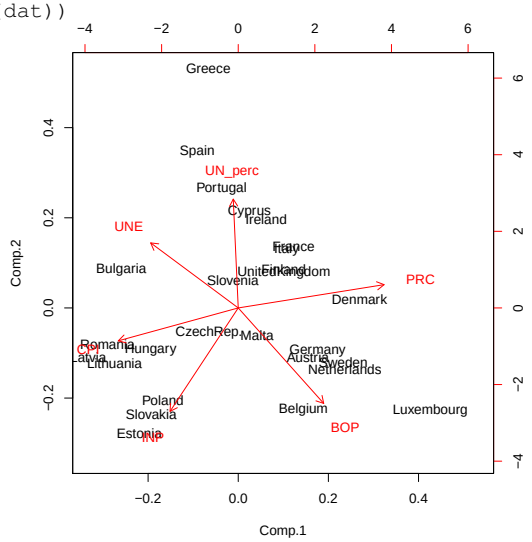
6 Economic indicators for EU countries in 2011.

```
> eu<-read.csv(
      'http://www.stats.ox.ac.uk/~sejdinov/sdmml/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)

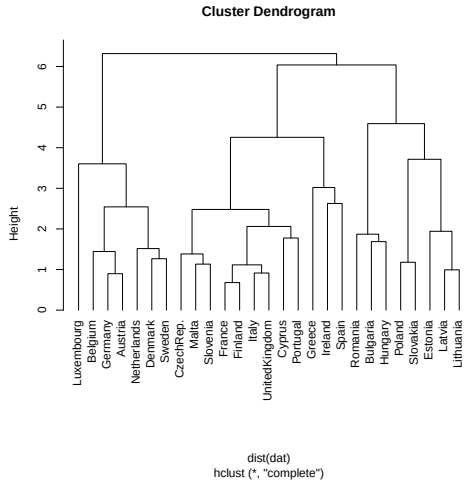
EU Indicators Data

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

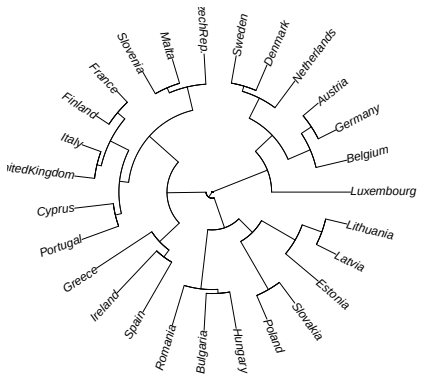


Visualising Hierarchical Clustering

```
> 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.

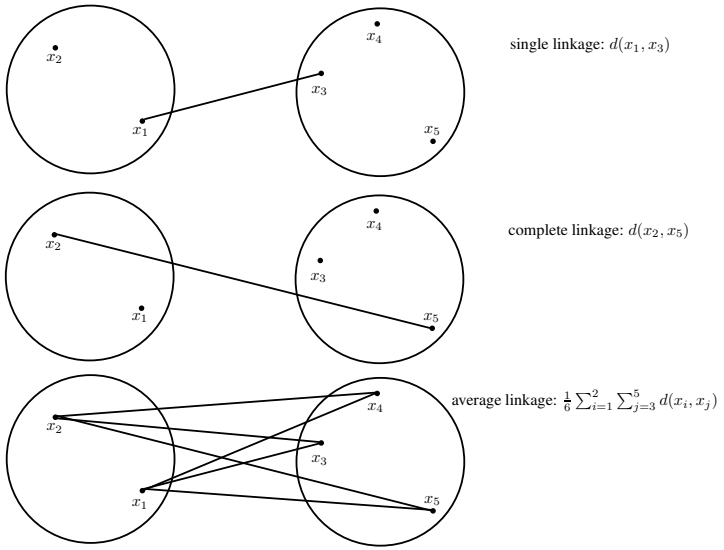
- Tree dissimilarity d_{ij}^T = minimum height in the tree at which examples i and j belong to the same cluster.
- **ultrametric** (stronger than triangle) inequality:

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

- 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.



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.

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

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

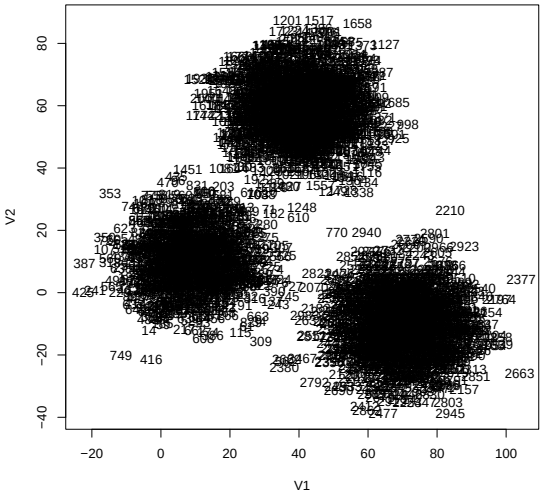
(b) **Complete Linkage**: compact clusters, relatively similar objects can remain separated at high levels

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

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

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

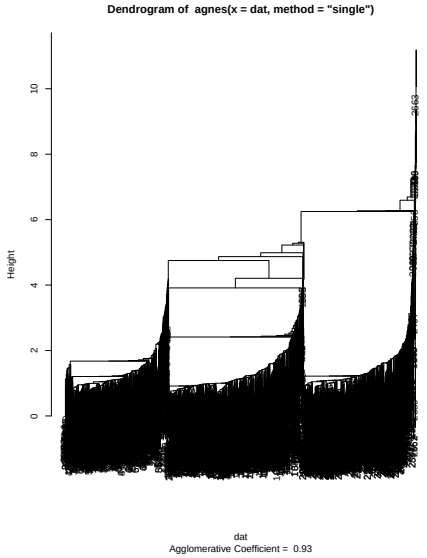
Hierarchical Clustering on Artificial Dataset



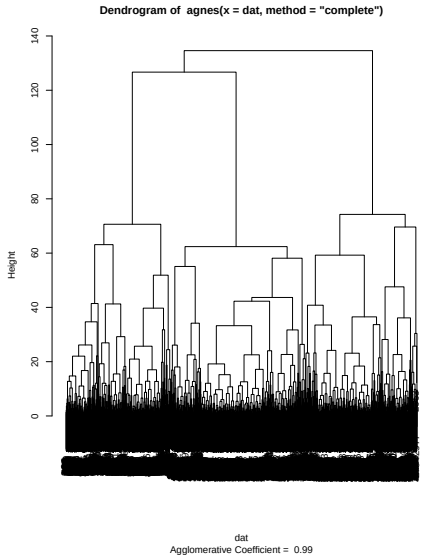
Hierarchical Clustering on Artificial Dataset

```
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"))
```

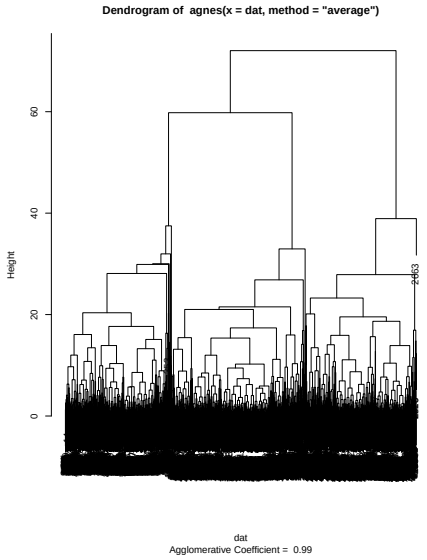

Hierarchical Clustering on Artificial Dataset



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.