

# Statistical Machine Learning

**Pier Francesco Palamara**

Department of Statistics

University of Oxford

Slide credits and other course material can be found at:

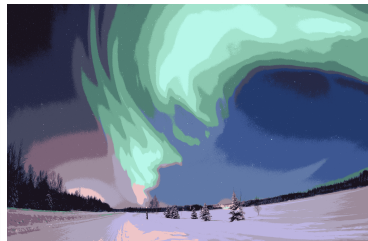
[http://www.stats.ox.ac.uk/~palamara/SML\\_BDI.html](http://www.stats.ox.ac.uk/~palamara/SML_BDI.html)

# Clustering

---

# Clustering

- Many datasets consist of multiple heterogeneous subsets.
  - **Cluster analysis:** Given 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



# Axiomatic approach

Clustering method is a map  $\mathcal{F} : (\mathcal{D} = \{x_i\}_{i=1}^n, \rho) \mapsto \{C_1, \dots, C_K\}$  which takes as an input dataset  $\mathcal{D}$  and a dissimilarity function  $\rho$  and returns a partition of  $\mathcal{D}$ . Three basic properties required

- **Scale invariance.** For any  $\alpha > 0$ ,  $\mathcal{F}(\mathcal{D}, \alpha\rho) = \mathcal{F}(\mathcal{D}, \rho)$ .
- **Richness.** For any partition  $C = \{C_1, \dots, C_K\}$  of  $\mathcal{D}$ , there exists dissimilarity  $\rho$ , such that  $\mathcal{F}(\mathcal{D}, \rho) = C$ .
- **Consistency.** If  $\rho$  and  $\rho'$  are two dissimilarities such that for all  $x_i, x_j \in \mathcal{D}$  the following holds:

$$x_i, x_j \text{ belong to the same cluster in } \mathcal{F}(\mathcal{D}, \rho) \implies \rho'(x_i, x_j) \leq \rho(x_i, x_j)$$

$$x_i, x_j \text{ belong to different clusters in } \mathcal{F}(\mathcal{D}, \rho) \implies \rho'(x_i, x_j) \geq \rho(x_i, x_j),$$

then  $\mathcal{F}(\mathcal{D}, \rho') = \mathcal{F}(\mathcal{D}, \rho)$ .

Kleinberg (2003) proves that there exists no clustering method that satisfies all three properties!

# Types of Clustering

- **Model-free** clustering:
  - Defined by **similarity/dissimilarity** among instances within clusters.
- **Model-based** clustering:
  - Each cluster is described using a probability model. (We will not look at this type of algorithms in this course).

# Examples of Model-free Clustering

- **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.
- **Spectral clustering:** Similarity/dissimilarity between data items defines a graph. Find a partition of vertices which does not “cut” many edges. Can be interpreted as nonlinear dimensionality reduction followed by  $K$ -means. We will not study this approach in this course.
- **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

Goal: divide data items into a **pre-assigned number  $K$  of clusters**

$C_1, \dots, C_K$  where for all  $k, k' \in \{1, \dots, K\}$ ,

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

Each cluster is represented using a **prototype** or **cluster centroid**  $\mu_k$ .

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:

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

$W$  is the overall objective function used to select both the cluster centroids and the assignment of points to clusters.

# 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  $\mu_k$ :

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

- 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,

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

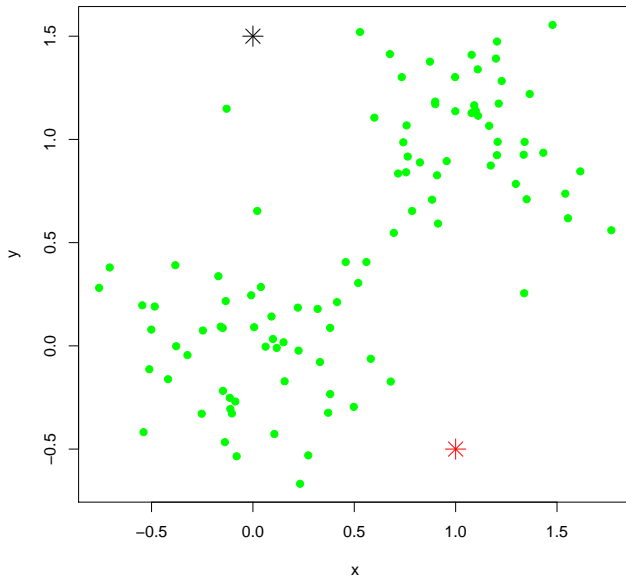
Set  $C_k := \{i : c_i = k\}$  for each  $k$ .

- 3 **Move centroids:** Set  $\mu_1, \dots, \mu_K$  to the averages of the new clusters:

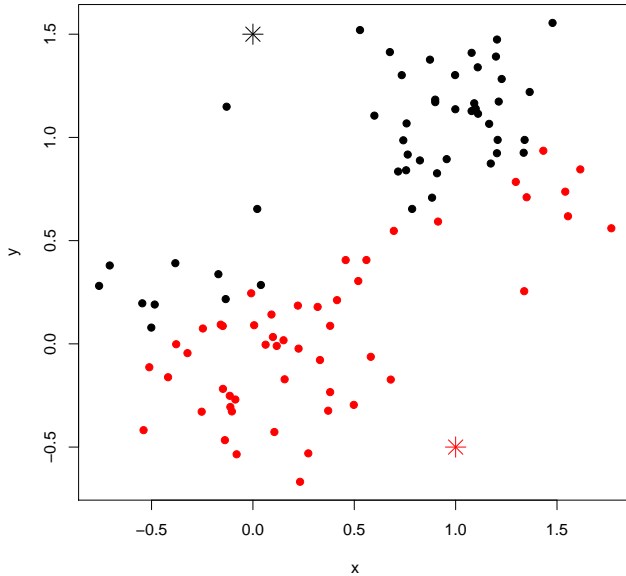
$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

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

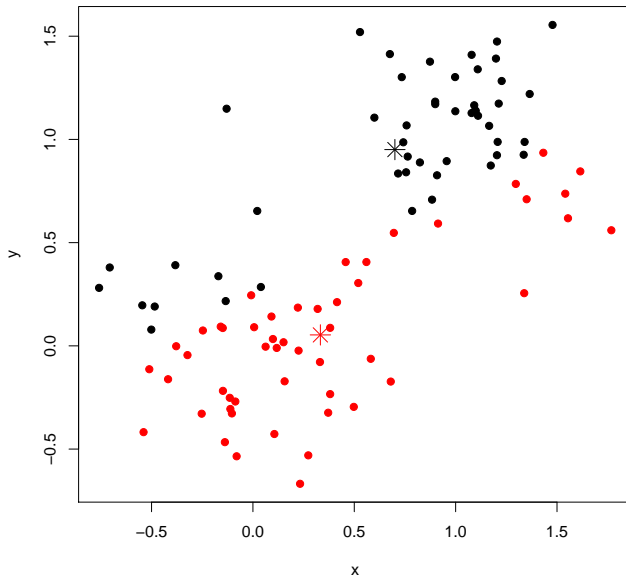
## K-means illustration



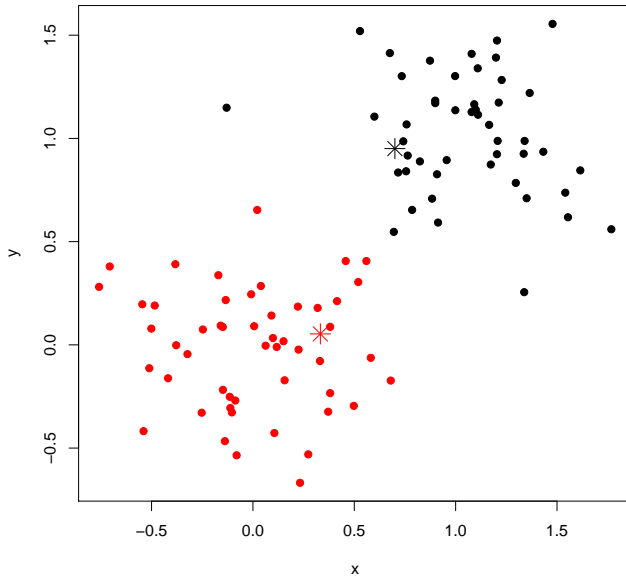
Assign points.  $W = 128.1$



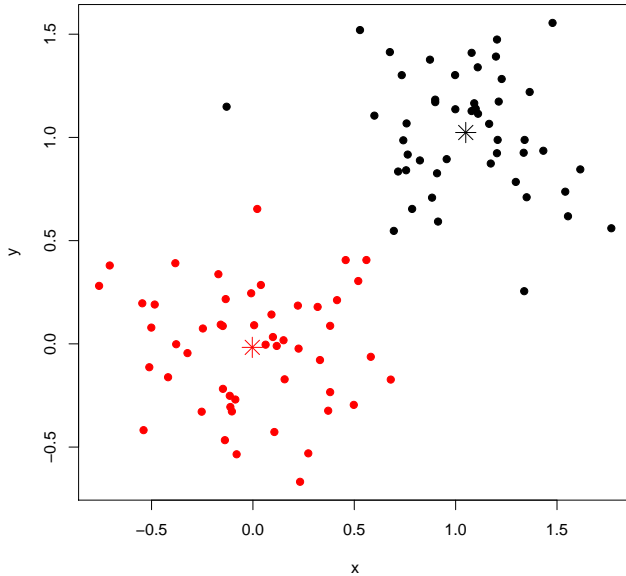
Move centroids.  $W = 50.979$



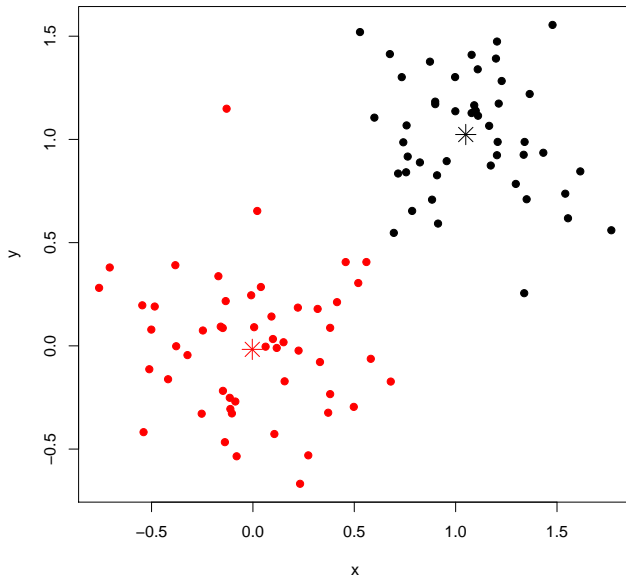
Assign points.  $W = 31.969$

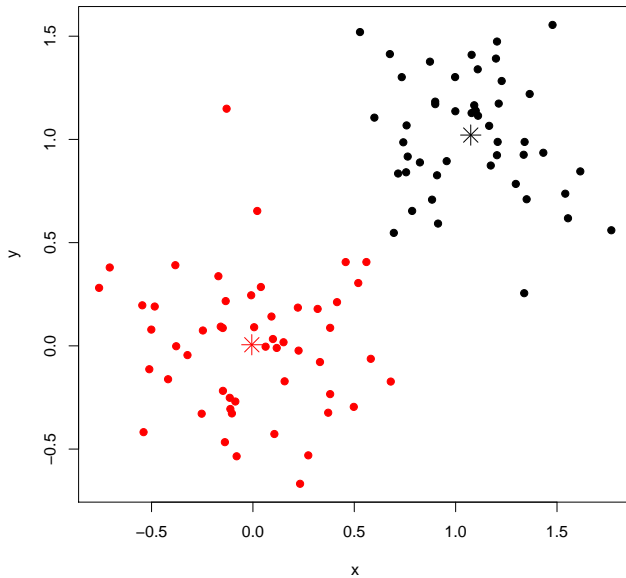


Move centroids.  $W = 19.72$



Assign points.  $W = 19.688$



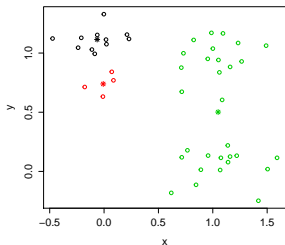
Move centroids.  $W = 19.632$ 



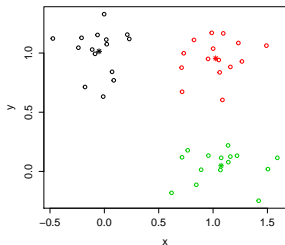
# K-means

- **The algorithm stops in a finite number of iterations.** Between steps 2 and 3,  $W$  either stays constant (same partitions) 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  $W$ .

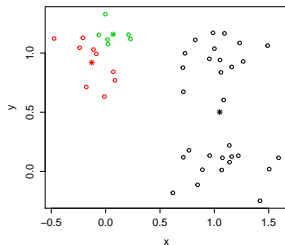
W= 9.184



W= 3.418



W= 9.264



# K-means on Crabs

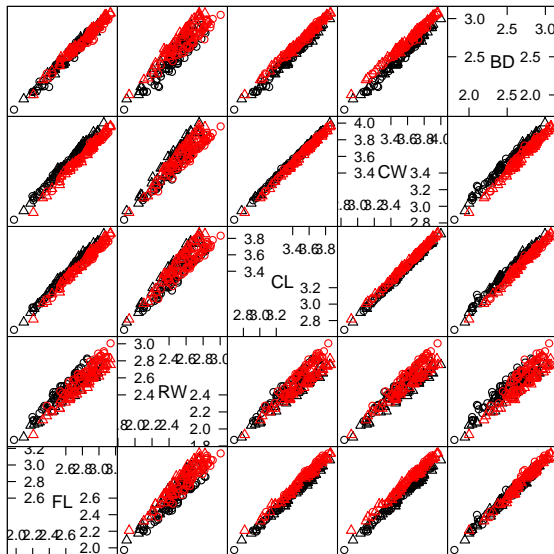
Looking at the Crabs data again.

```
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

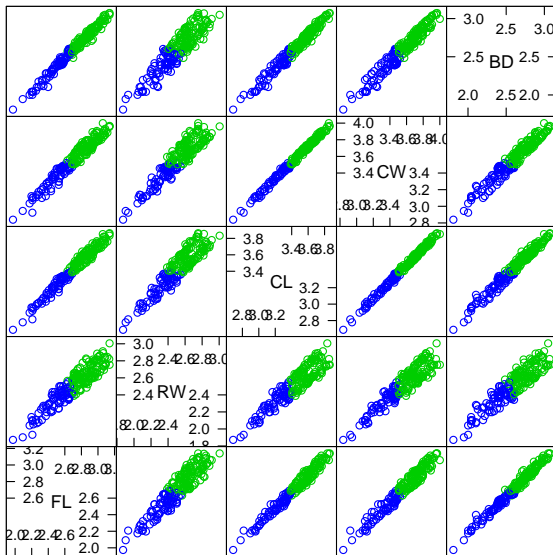
Apply K-means with 2 clusters and plot results.

```
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’<sup>1</sup> 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")
```

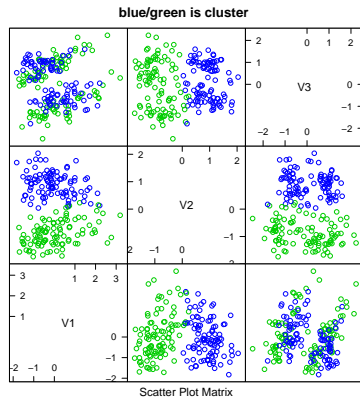
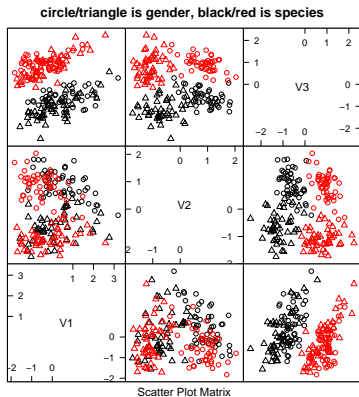
And apply K-means again.

```
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")
```

---

<sup>1</sup>Apply a linear transformation so that the covariance matrix is identity.

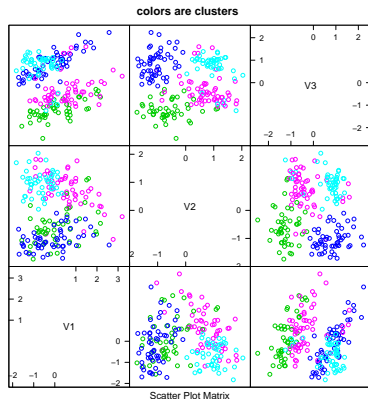
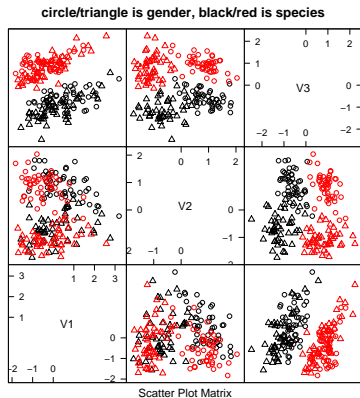
# K-means on Crabs



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  $\mu_1, \mu_2, \dots, \mu_K$  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 (more on this in later lectures). E.g., in DP-means<sup>2</sup>, use

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

---

<sup>2</sup>DP-means paper.

# 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**  $\mu_1, \dots, \mu_K$  ( $K \times p$  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  $3 \times 3$  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  $3 \times 3$  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  $3 \times 3$  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  $3 \times 3$  pixels as a single observation  $x_i \in \mathbb{R}^9$ .

# Hierarchical Clustering

- Hierarchically structured data is ubiquitous (genus, species, subspecies, individuals...)
- Here, only a brief overview of some of the key concepts in hierarchical clustering.
- 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.
- Some useful and R libraries: `hclust`, `agnes{cluster}`

# EU Indicators Data

## 6 Economic indicators for EU countries in 2011.

```
> eu<-read.csv(sep=' ',
               'http://www.stats.ox.ac.uk/~palamara/teaching/SML19/eu_indicators.csv')
```

```
> eu[1:15,]
```

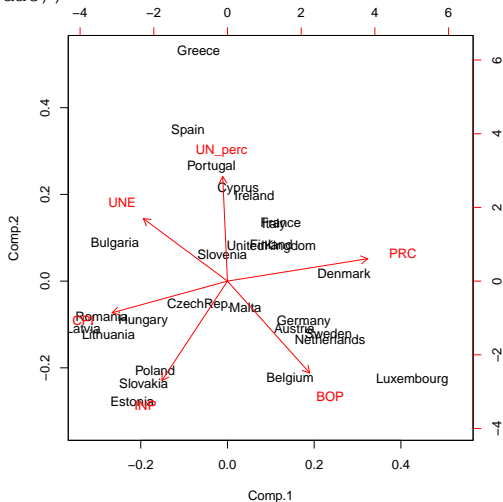
	Countries	abbr	CPI	UNE	INP	BOP	PRC	UN_perc
1	Belgium	BE	116.03	4.77	125.59	908.6	6716.5	-1.6
2	Bulgaria	BG	141.20	7.31	102.39	27.8	1094.7	3.5
3	CzechRep.	CZ	116.20	4.88	119.01	-277.9	2616.4	-0.6
4	Denmark	DK	114.20	6.03	88.20	1156.4	7992.4	0.5
5	Germany	DE	111.60	4.63	111.30	499.4	6774.6	-1.3
6	Estonia	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))
```

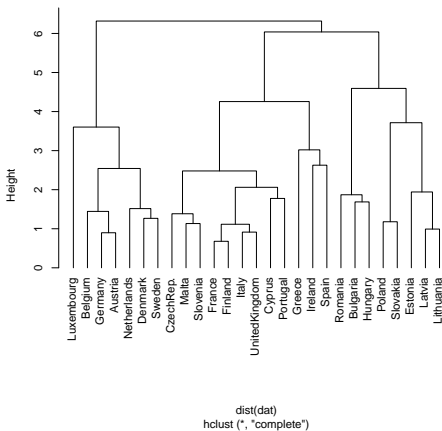


# Visualising Hierarchical Clustering

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

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

Cluster Dendrogram



# 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

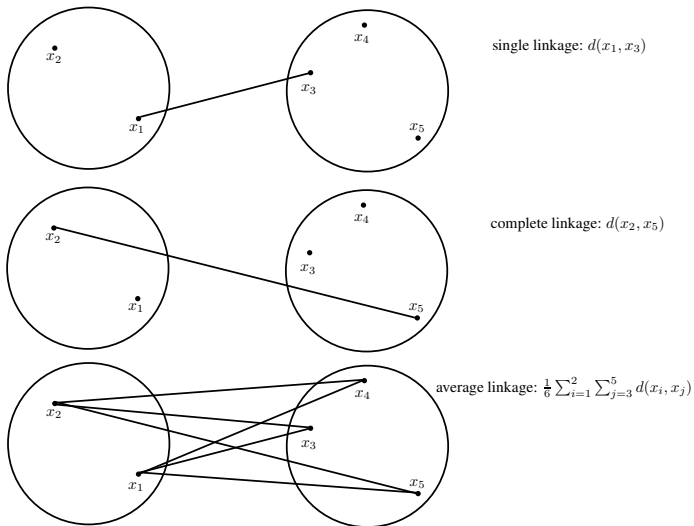
$$D(C_i, C_j) = \max_{x,y} (d(x, y) | x \in C_i, y \in C_j)$$

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

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

# 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.



# 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 depict cluster assignments with respect to increasing values of dissimilarity threshold. Cutting a dendrogram horizontally at a particular height partitions the data into disjoint clusters which are represented by the vertical lines it intersects.
- 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.