Statistical Methods for High Dimensional Biology

STAT/BIOF/GSAT 540

Lecture 12 – Unsupervised learning: clustering and PCA

Sara Mostafavi

Feb 14 2018

**Slide credits: Drs. Paul Pavlidis; Gaby Freue-Cohen*

Origins of clustering: Machine learning (sub-field of CS) & Statistics

Computer Science:

- Machine Learning
  - Unsupervised learning
    - Clustering algorithms

Special insights & emphasis:

- Analyzing computational difficulty of the problem.
- Designing **Algorithms** for solving a given clustering problem.
- Mathematical insights about the algorithm.

Statistics:

- Data Mining
  - Density estimation/Clustering

Special insights:

- How do we make "decisions" about structure in the data. (e.g., model selection)

Unsupervised learning

- “Automated” way of finding patterns/structure in the data – without specifying what those patterns should tell you about! This distinction becomes more clear in the next lecture.
  - Data summarization: “main patterns”
  - Data visualization

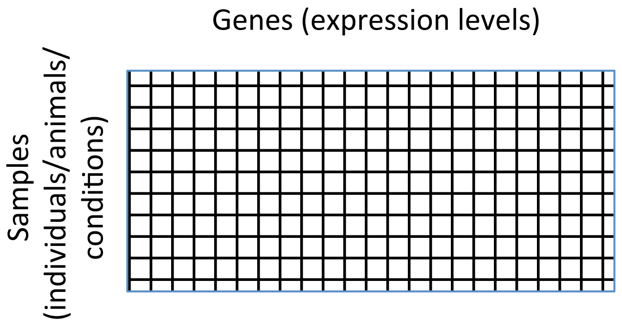
- Two general frameworks:

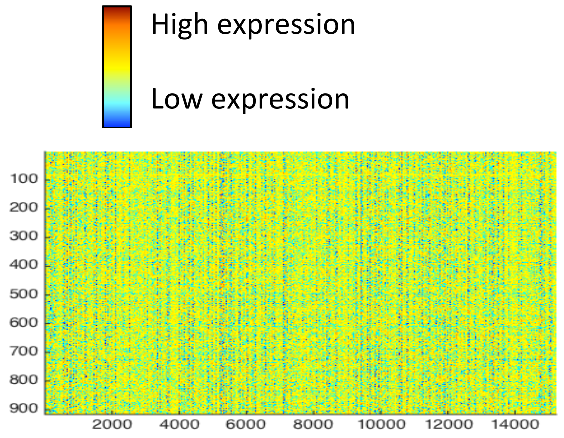
1. Dimensionality reduction (e.g., PCA)
2. Clustering

Today

- Clustering
  - Objective function vs algorithm
  - K-means
  - Hierarchical clustering
- PCA
  - Basis span and matrix rank
  - Covariance matrices
  - SVD
  - Applications

Finding structure in high-dimensional data


Genes (expression levels)
Samples (individuals/animals/ conditions)

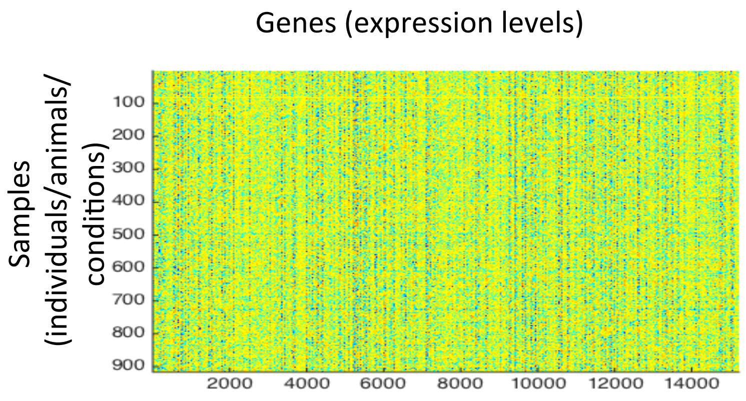


Matrix X with dimensions n by p (n rows and p columns)

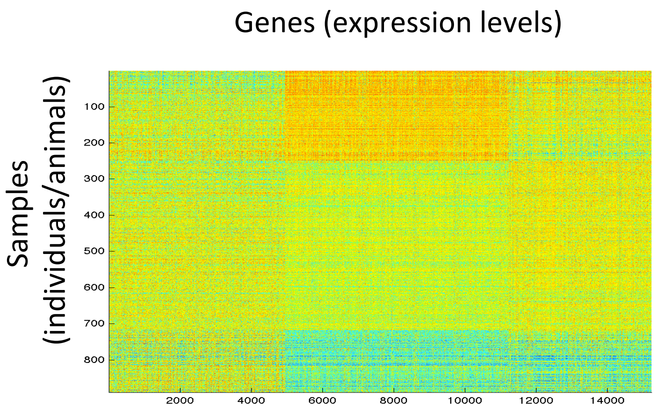
Pervasive application of clustering in analysis of gene expression data

A more familiar picture seen in "omics" papers ….

High expression
Low expression

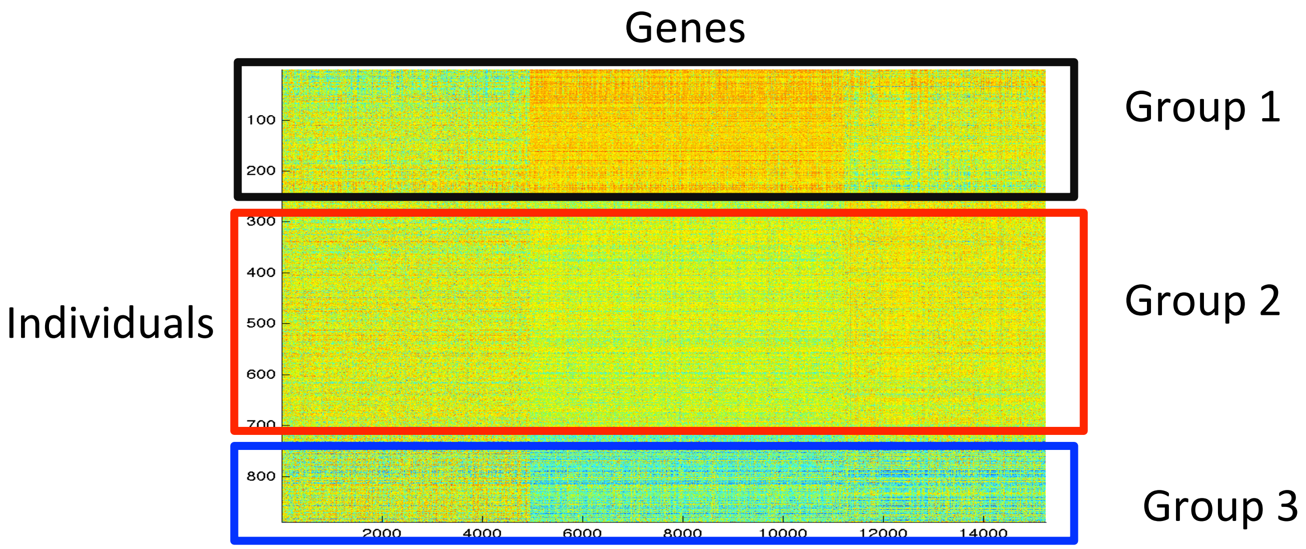


Clustering algorithm



Two predominant application of clustering in gene expression studies

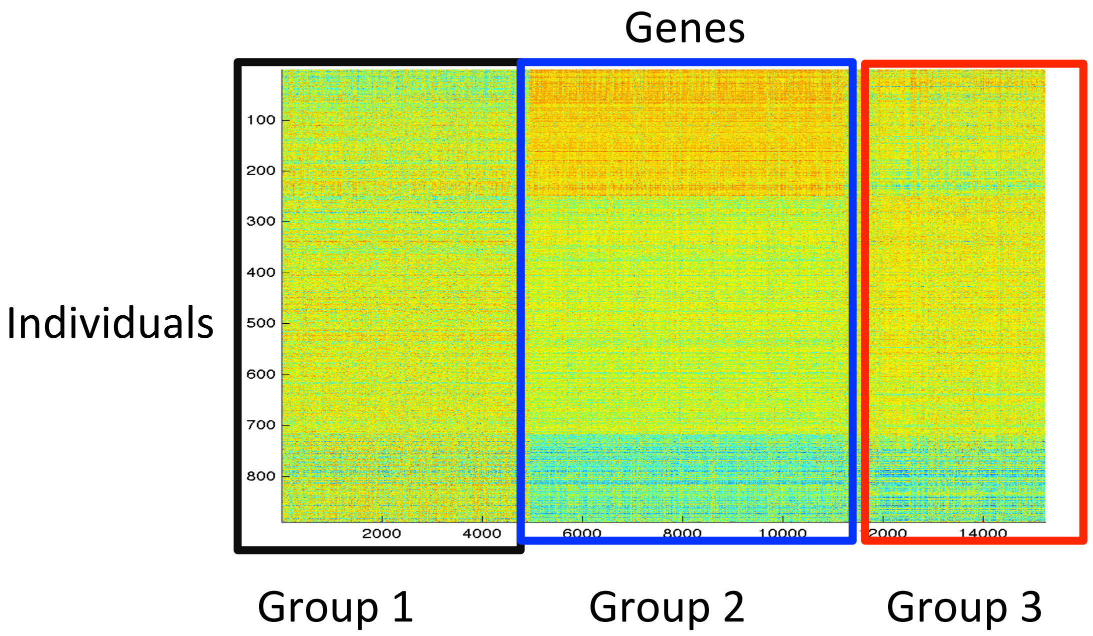
1. Identify groups of individuals that have similar expression profiles:
   – Identifying disease sub-types



Two predominant application of clustering in gene expression studies

2. Identify groups of genes that are **co-expressed**

(co-expressed gene modules)



What is Clustering?

- “Clustering” Colloquially means placing/grouping a set of objects into groups/clusters.
- Clustering is a formal **problem** in Computer Science and in Statistics, with formal definitions and “solutions”.
- Clustering in bioinformatics is often used as a tool for visualization, hypothesis generation, selection of genes for further analysis.

Three key concepts with distinct definitions:

1) The clustering problem
2) A clustering objective function (data model)
3) A clustering algorithm

Three key concepts with distinct definitions:

1) The clustering problem
2) A clustering objective function (data model)
3) A clustering algorithm

Keep in mind, with typical use of clustering in bioinformatics: there is no measure of "strength of evidence" or "strength of clustering structure" provided. **An algorithm on it's own doesn't give you a framework for statistical inference.**

Diversion: what’s an algorithm

(wiki example)

- Definition: an unambiguous specification about how to solve a class of problems.

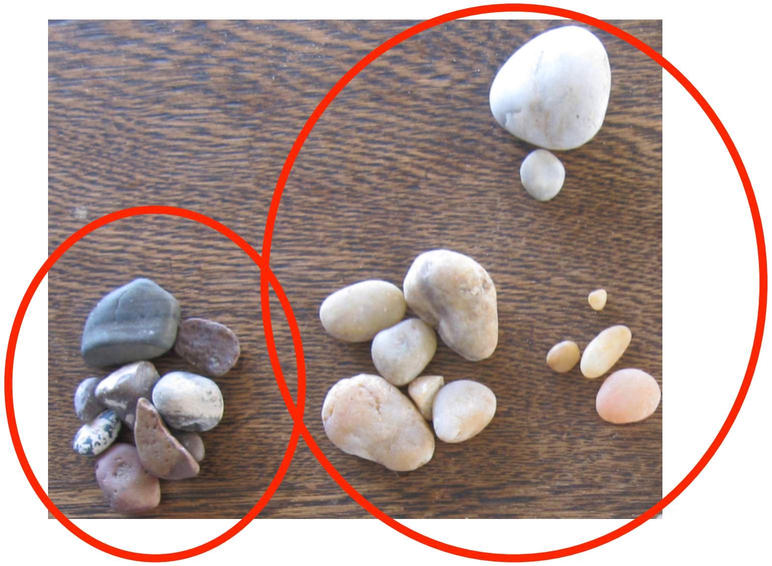
- E.g., problem: find the largest number in a list of numbers of random order.

1. What operations are available?
2. When do we fail to produce the desired result?

Clustering problem: Definition

- Goal: place a set of **objects** into groups or **clusters** in a way that **similar** objects are in the same cluster.

Cluster some rocks:



Note that you could have also considered a 2-cluster solution.

Clustering: Definitions

- Goal: place a set of objects into groups or **clusters**.
- How do we do this?
  - gather a set of attributes for each object.
  - Place objects in clusters so that objects within each cluster are more similar to each other compared to objects that outside their group/cluster.

Rocks were clustered according to their color and texture.

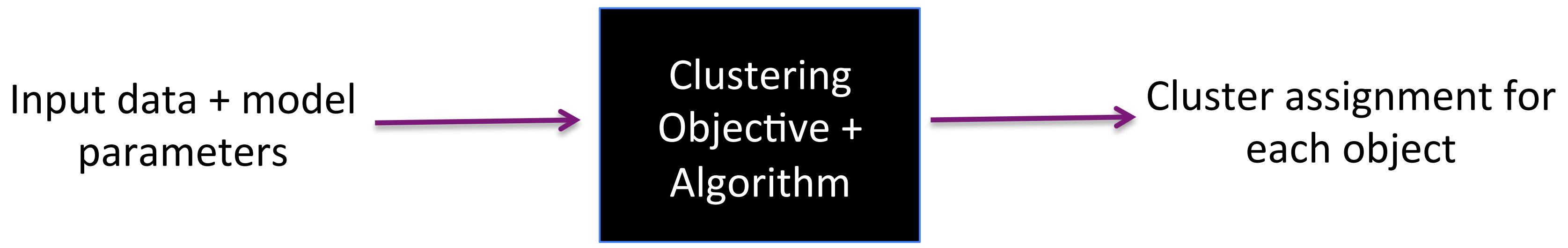
A clustering objective function

- Goal (the clustering problem):place a set of **objects** into groups or **clusters** in a way that **similar** objects are in the same cluster.
- How do we do this?
  - gather a set of attributes for each object.
  - Place objects in clusters so that objects within each cluster are more similar to each other, based on their attributes, compared to objects that outside their group/cluster.

→ Clustering **objective** function: maximize within cluster similarity

- A precise definition of "good/optimal" clustering: precise enough to be translated into an equation.

Clustering algorithm from a machine learning perspective: what are the inputs and the outputs?


Input data + model parameters
Clustering Objective + Algorithm
Cluster assignment for each object

Input: 1) data matrix X_{nxp} (rows are the objects)

2) number of clusters k

Output: an assignment of cluster membership for each object. $C_{nx1} = \{1,k\}^n$, C_i=k if object i is placed in cluster k.

(Note the output vector can also be a probabilistic assignment, we'll ignore this for now.)

Defining attribute/feature vector for each object

- We need to numerically define a attribute or feature vector that describes the relevant properties of each object

Set of objects $\{\vec{x}_1, \vec{x}_2, \vec{x}_3, \ldots, \vec{x}_n\}$

Each object is represented by a numerical vector: $\vec{x}_1 \subseteq \Re^p$

Attribute/feature p for object 1

Rock1: $\vec{x}_1 = (x_1^{(1)}, x_2^{(1)}, \ldots, x_p^{(1)})$

Numerical value representing texture

Numerical value for color/shade

Commonly Used Measures of Similarity and Distance

- Every clustering method is based on the measure of distance or similarity.
- We need to compute pairwise similarities between all objects.

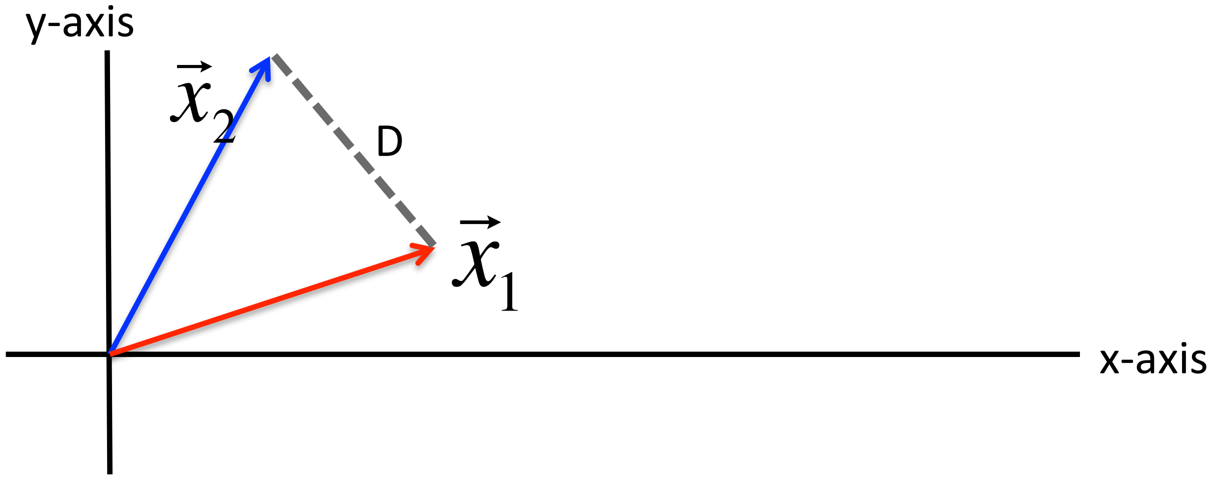
- Typical distance/similarity measures:
  - Distance:
    - Euclidean
    - Manhattan

  - Similarity: Correlation
    - Spearman
    - Pearson

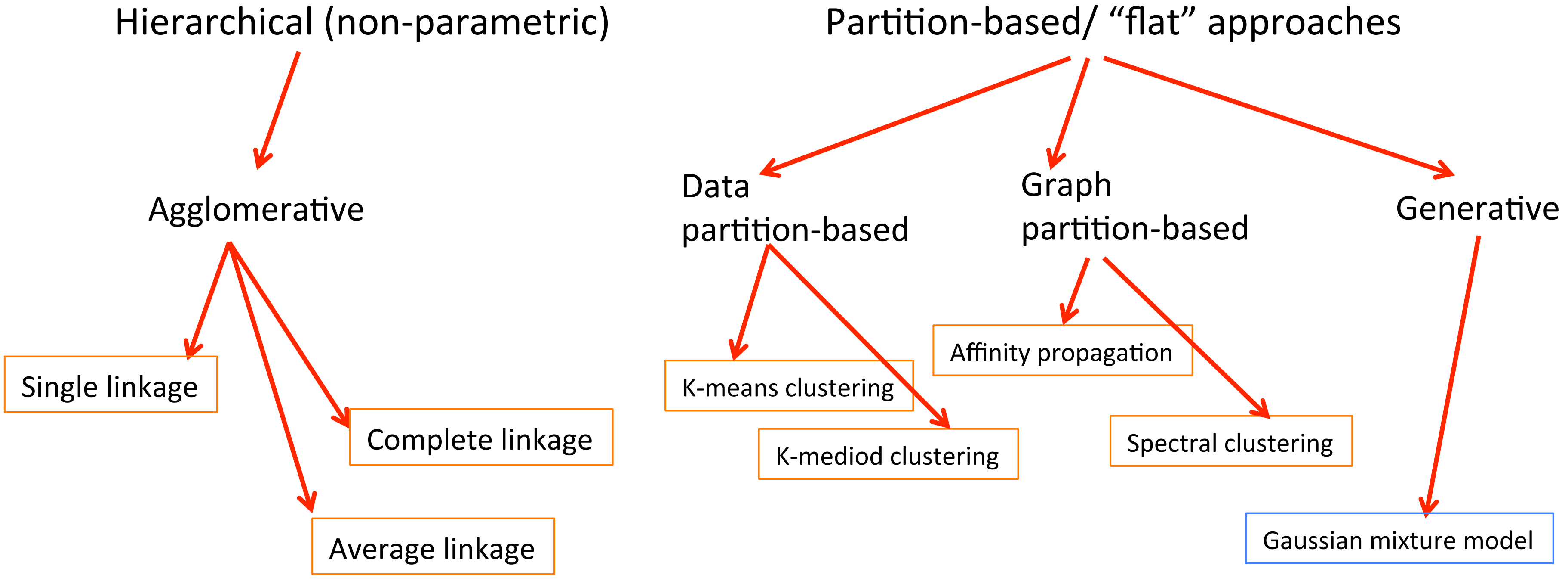
Commonly Used Measures of Similarity and Distance

- Euclidian distance between two feature vectors: $\vec{x}_1$ and $\vec{x}_2$

$$D = \| \vec{x}_1 - \vec{x}_2 \|_2 = \sqrt{\sum_{i=1}^{p} (x_i^{(1)} - x_i^{(2)})^2}$$



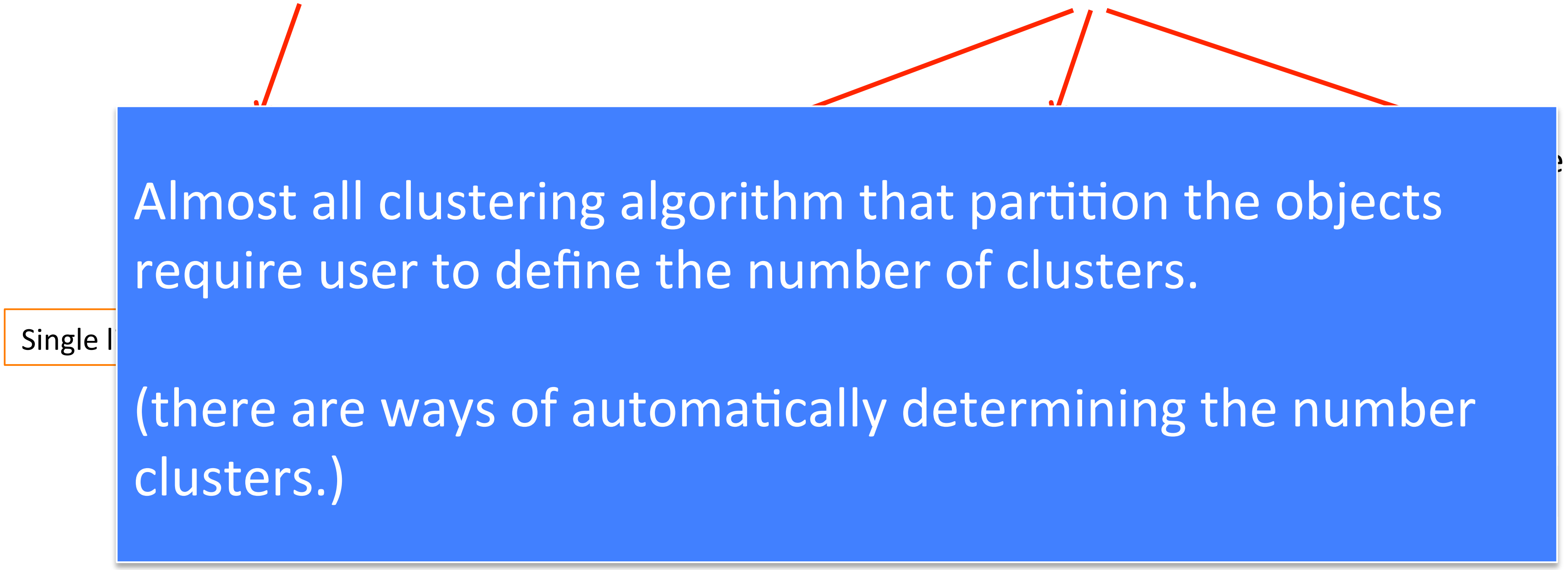
Some existing clustering algorithms



🟦 **Discrete clustering assignment**
🟧 **Probabilistic cluster assignment**

Some existing clustering algorithms

Hierarchical (non-parametric) Partition-based/ “flat” approaches



Almost all clustering algorithm that partition the objects require user to define the number of clusters.

(there are ways of automatically determining the number clusters.)

🟦 **Discrete clustering assignment**
🟧 **Probabilistic cluster assignment**

K-means clustering objective function

- One of the most widely used partition-based clustering approaches.
- **Objective function**: minimize the average squared Euclidean distance of objects from their assigned cluster centers. A cluster center (or centroid) is defined as the mean of objects in the given cluster.

K-means clustering objective function

- One of the most widely used partition-based clustering approaches.
- **Objective function**: minimize the average squared **Euclidean distance** of objects from their assigned cluster centers. A **cluster center** (or centroid) is defined as the mean of objects in the given cluster.

K-means clustering objective function

Objective function (data model)

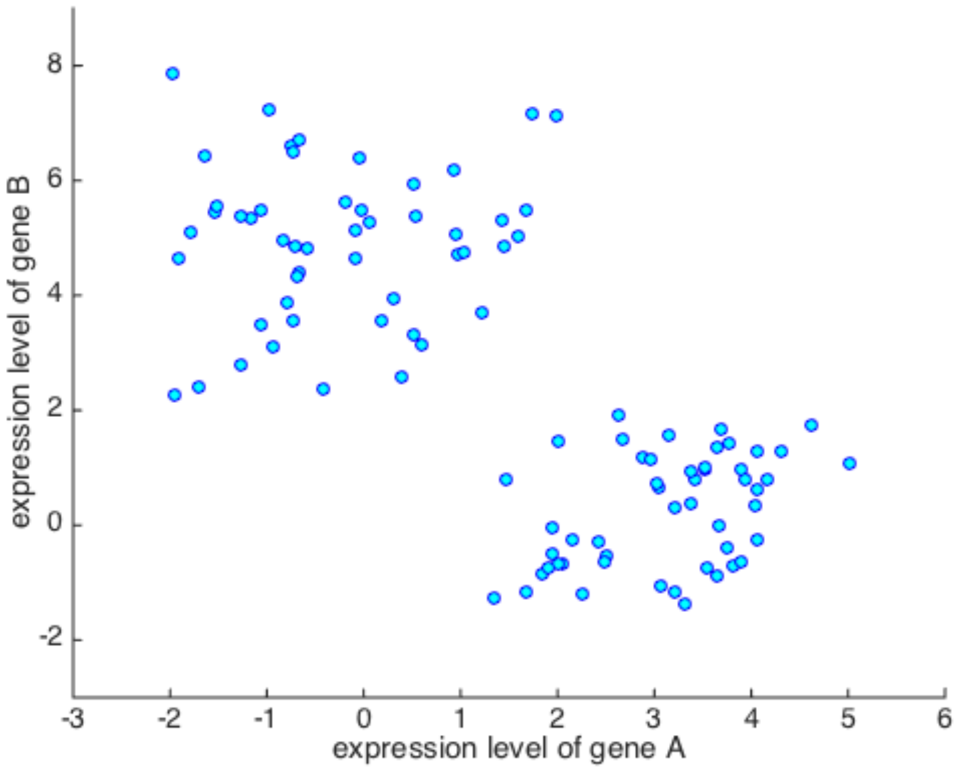
$$\arg\min_S \sum_{i=1}^{k} \sum_{x \in S_i} \| \vec{x} - \vec{\mu}_i \|^2$$

- The algorithm for solving the clustering problem needs to find the cluster means and the cluster assignment
- If we knew the cluster memberships, we could find cluster means easily:

$$\operatorname{argmin}_{S_i} \sum_{x \in S_i} \| \vec{x} - \vec{\mu}_i \|^2$$

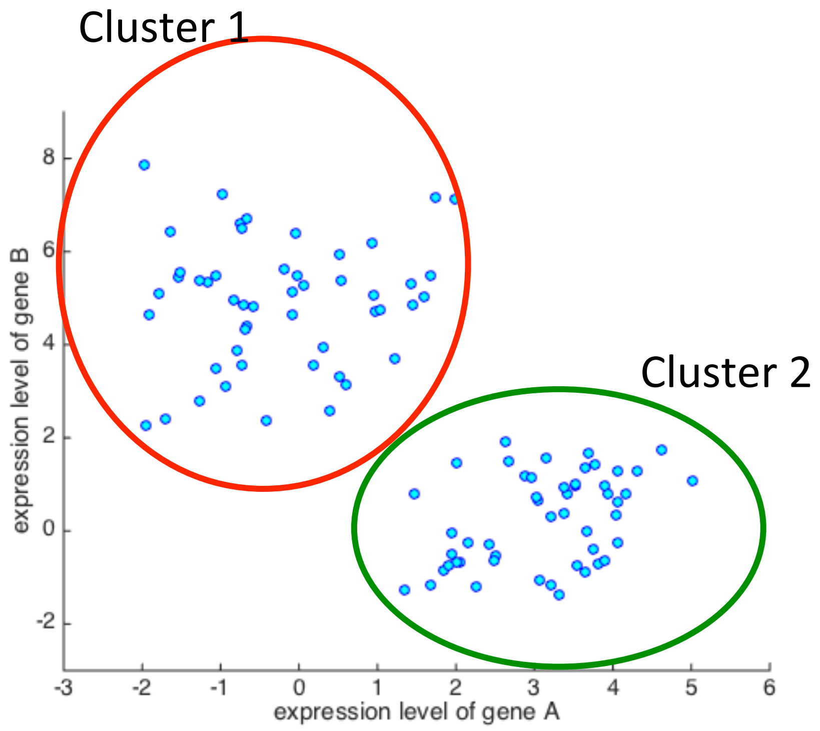
How many clusters are there?

Suppose you measured expression levels for 2 genes (gene A and gene B) for 100 individuals



How many clusters are there?

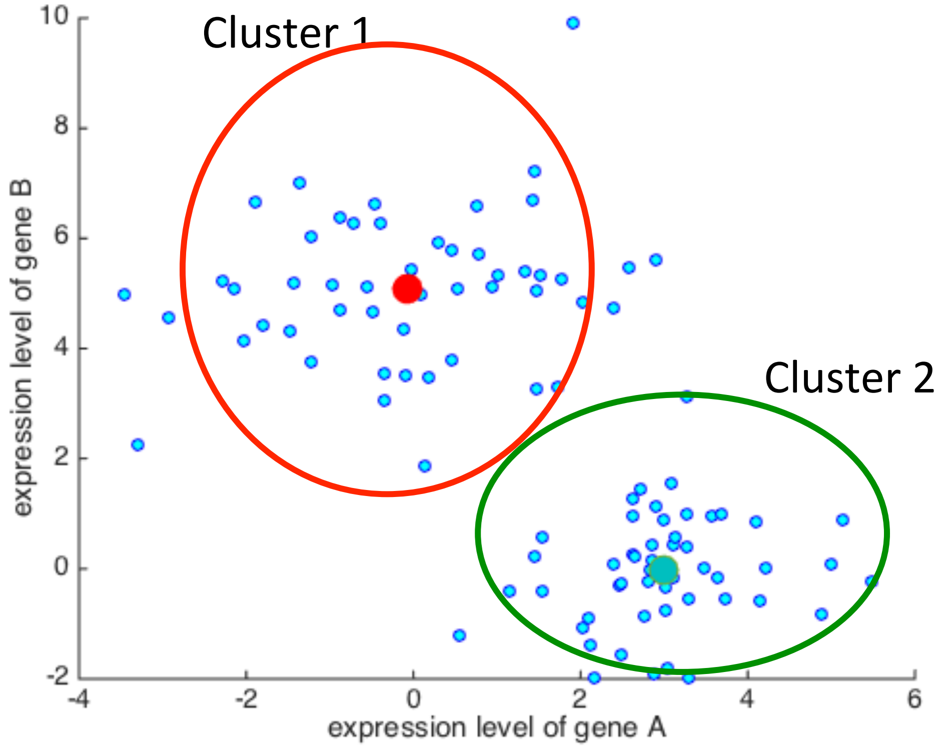
Suppose you measured expression levels for 2 genes (gene A and gene B) for 100 individuals



K-means objective function

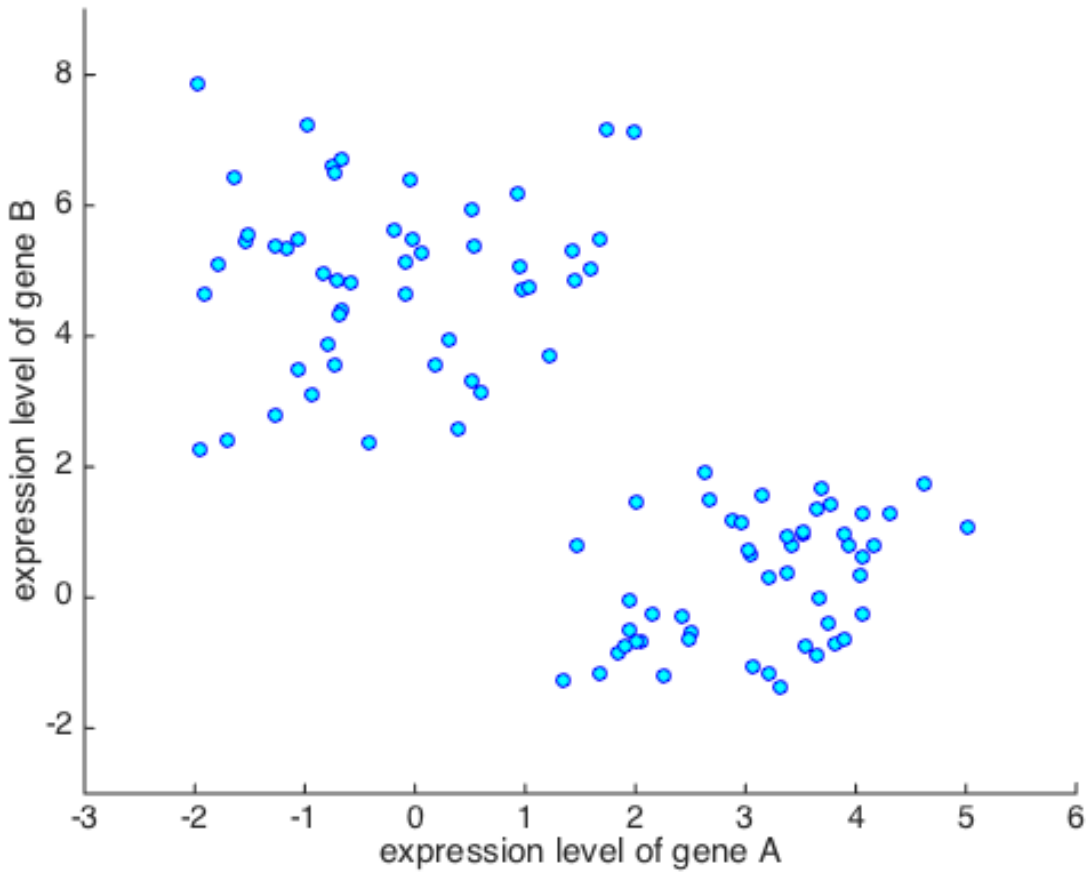
Objective function: minimize the average squared **Euclidean distance** of objects from their assigned cluster centers. A **cluster center** (or centroid) is defined as the mean of objects in the given cluster.

Computing the "mean/centroid" for each cluster:



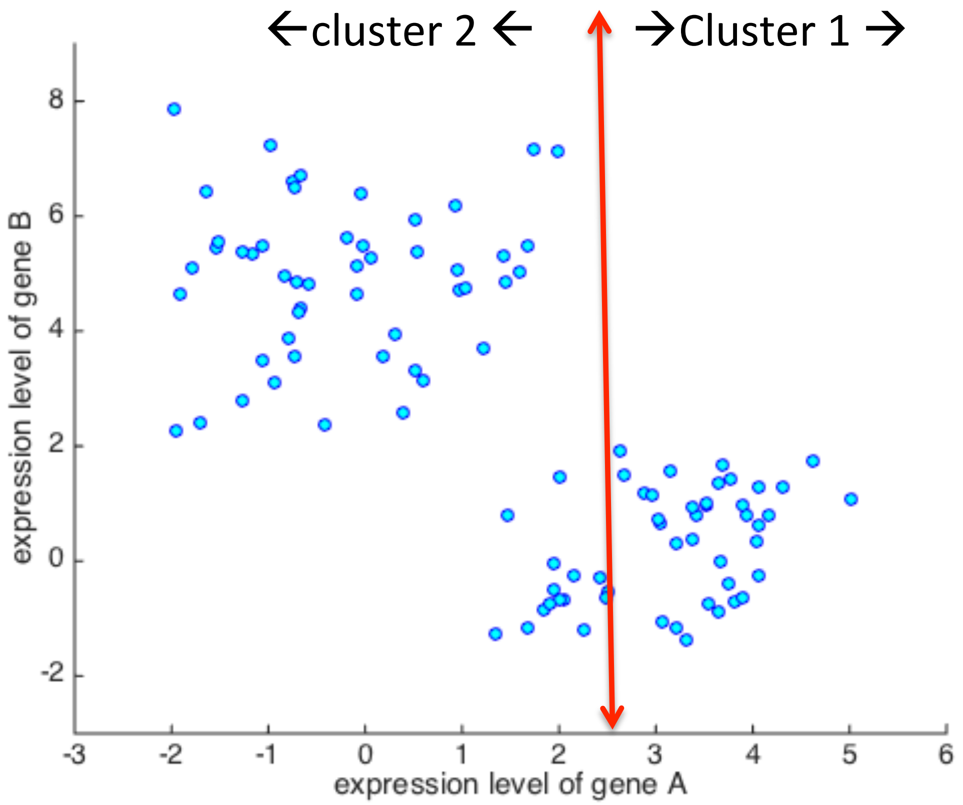
K-means algorithms in words …

1. Divide the data into K clusters
2. Initialize the “centroids” (cluster means) based on the current assignment of data to clusters
3. Compute distance between each point and all centroids. Then assign each point to the closest cluster.
4. When all points have been assigned, recalculate the centroids.
5. Repeat 2-4 until the centroids no longer change.


expression level of gene B
expression level of gene A
8
6
4
2
0
-2
-3
-2
-1
0
1
2
3
4
5
6

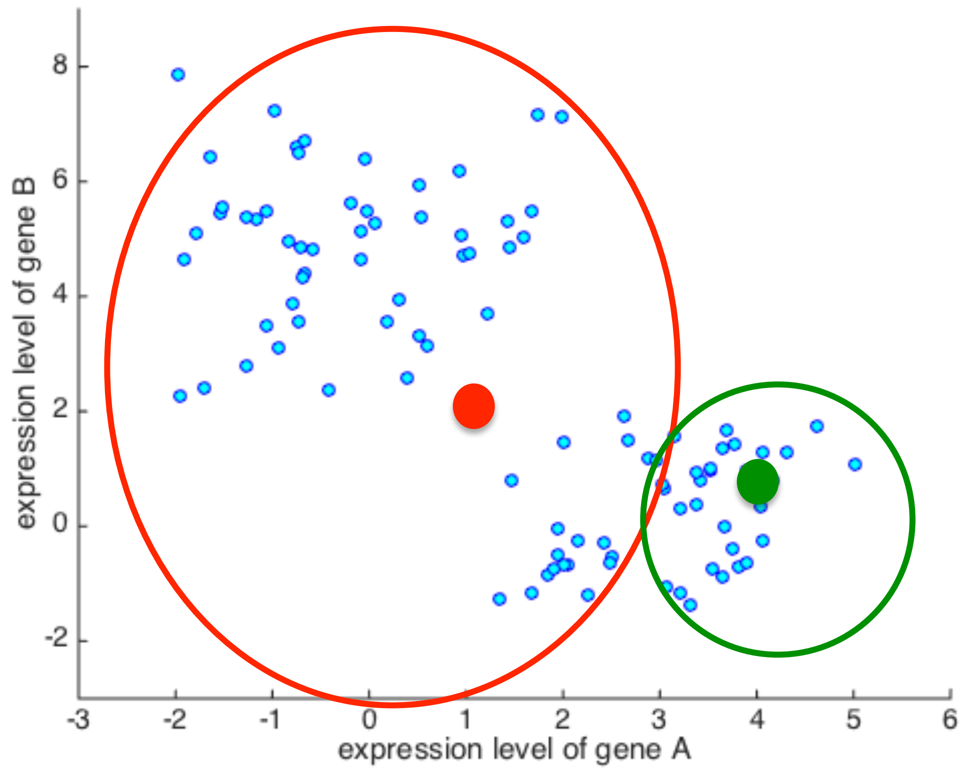
K-means objective function

Step 1: partition space in to two clusters (e.g., randomly assign objects to one of two clusters)



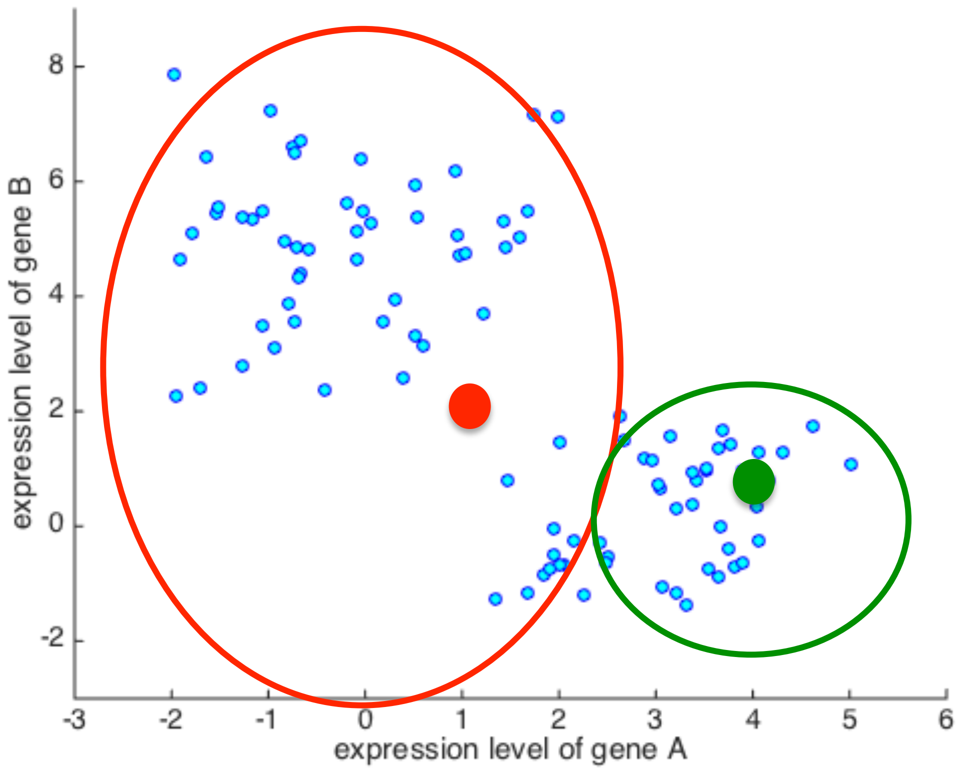
K-means objective function

Step 1: partition space in to two clusters (e.g., randomly assign objects to one of two clusters) – initialize cluster centers.



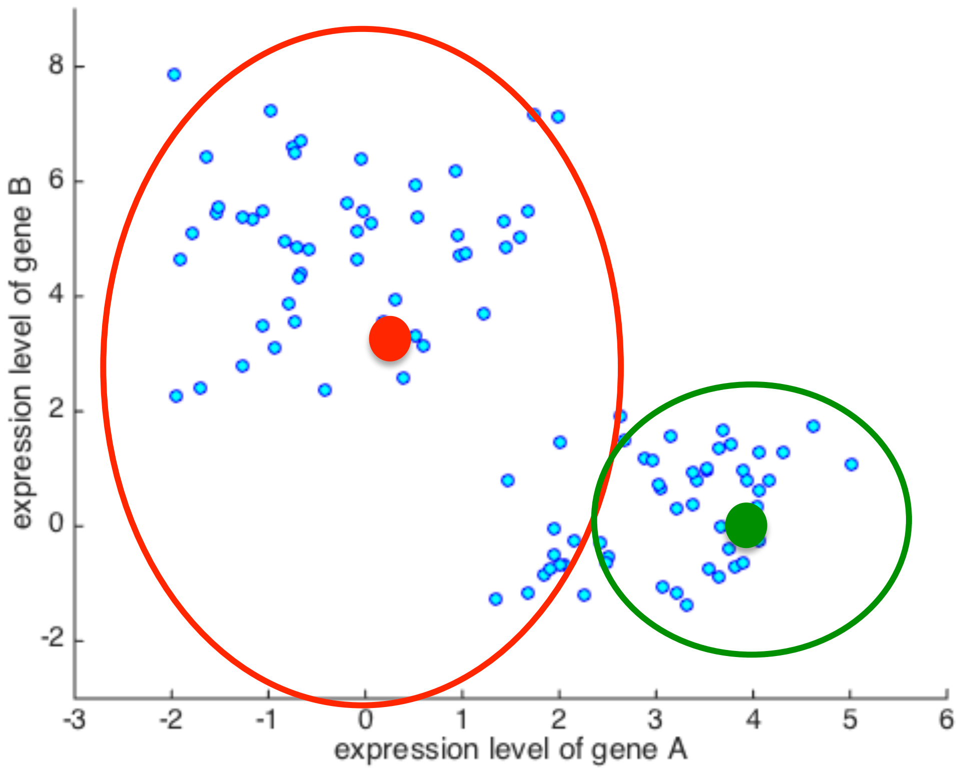
K-means objective function

Step 2: computer distances between each object and all cluster centers, then re-assign each object to its closest cluster.



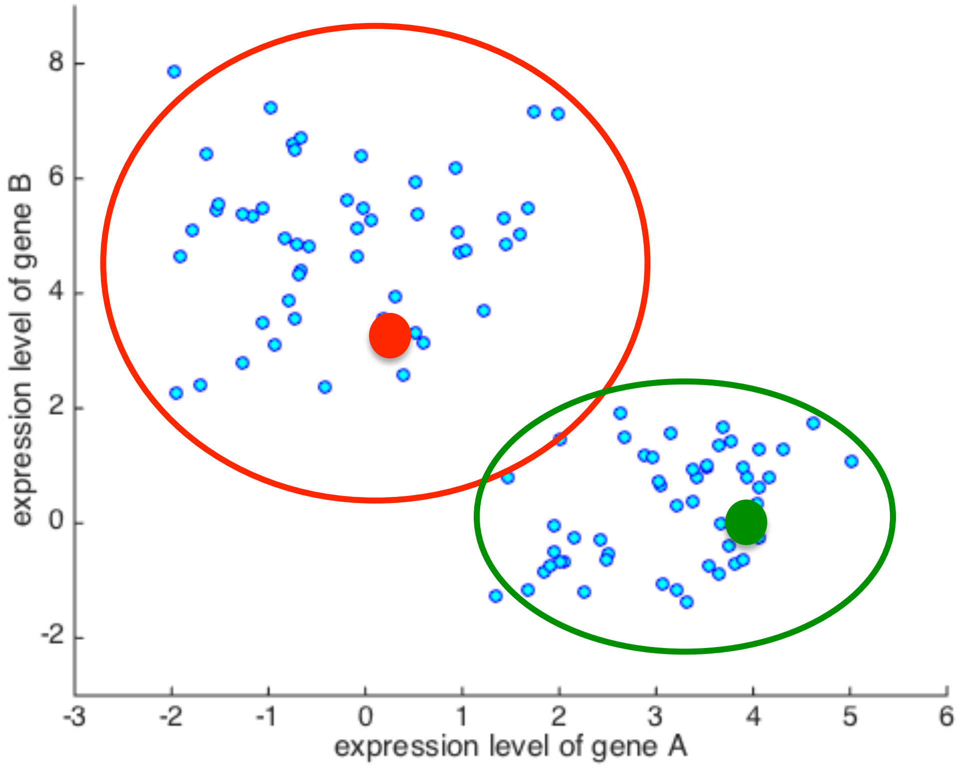
K-means objective function

Step 3: recalculate the cluster means for each cluster.

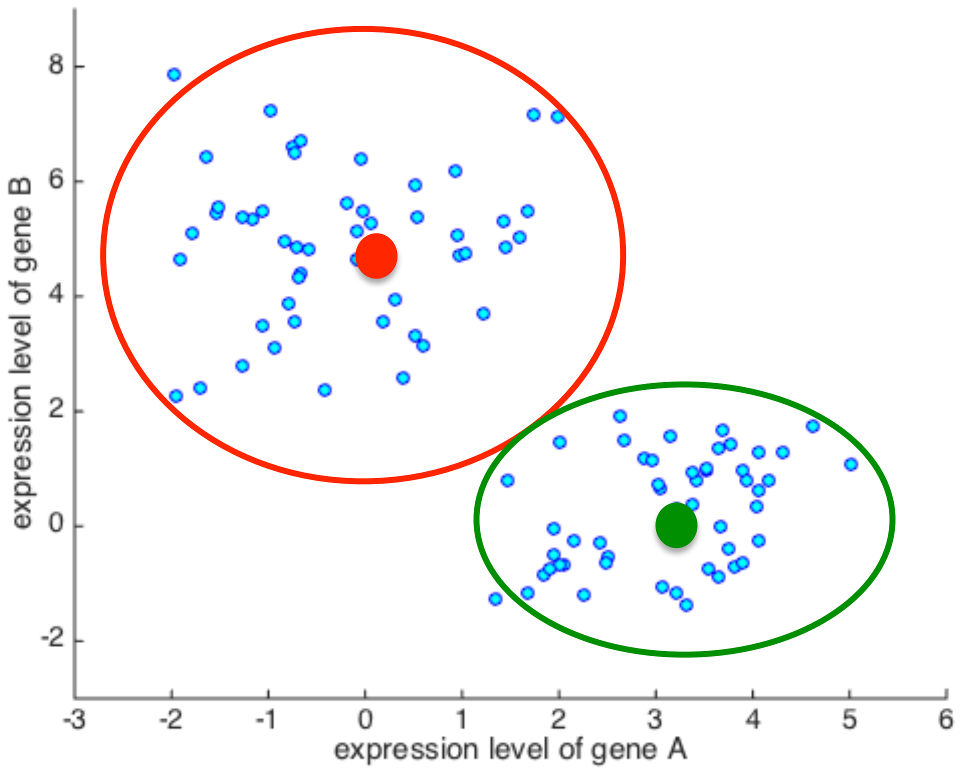


K-means objective function

Back to step 2: computer distances between each object and all cluster centers, then re-assign each object to its closest cluster.



Continue iterating until convergence:



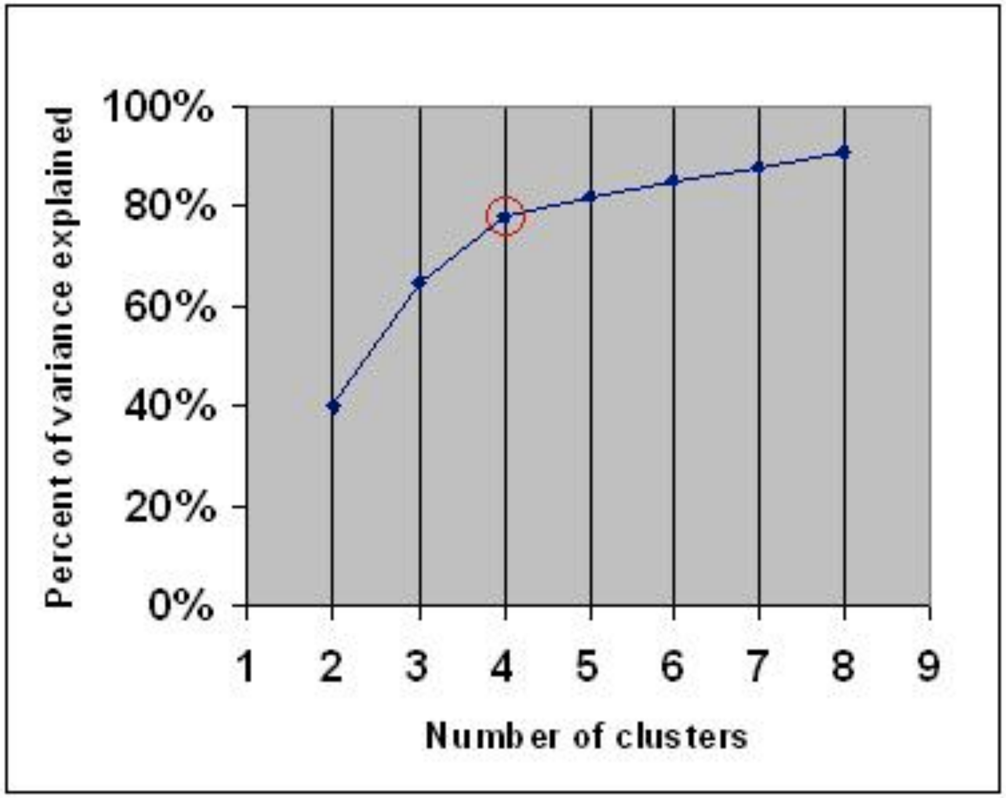
Analysis of K-means algorithm

- Do you think the k-means algorithm converges? YES
- Does it always give you the *best* solution?
  - How many partitions are there for k clusters and n data points?
- Does it always give you the *same* solution on the same data?

How do you determine *k* (number of clusters)?

Note: maximizing the clustering likelihood/objective will not be informative→ each object should be in its own cluster. Therefore, need an algorithm that takes into account the "cost" of additional clusters.

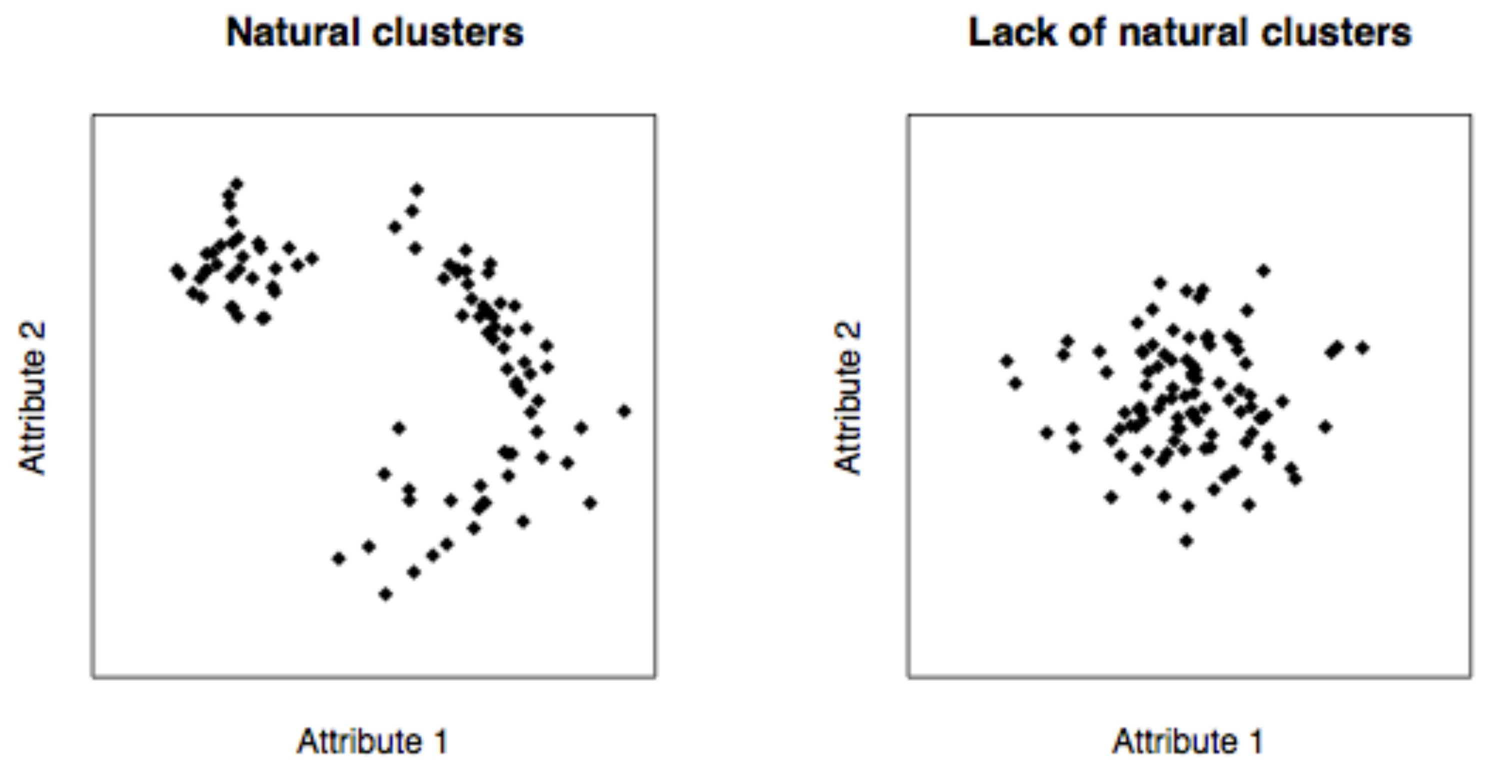
- Prior knowledge
- The "elbow method"



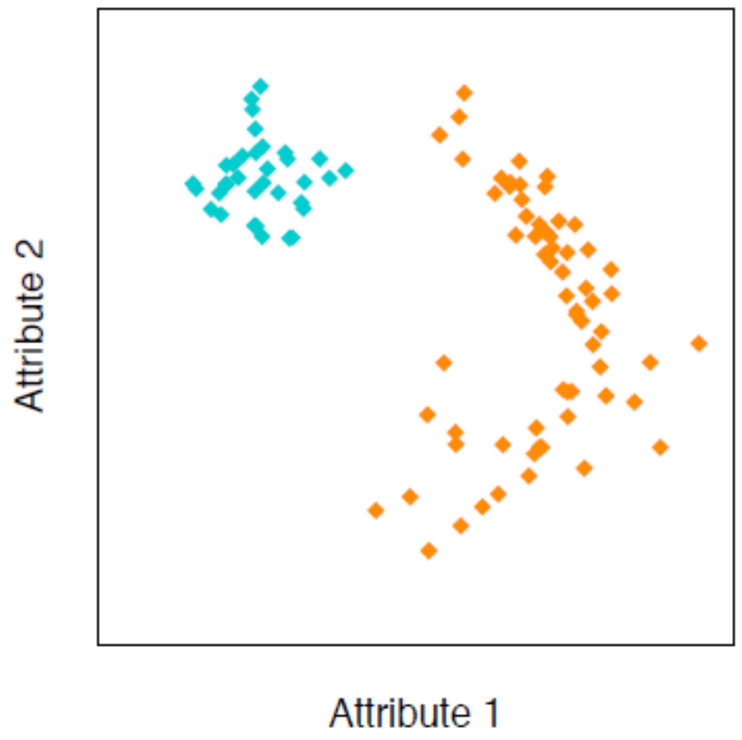
How do you determine *k* (number of clusters)?

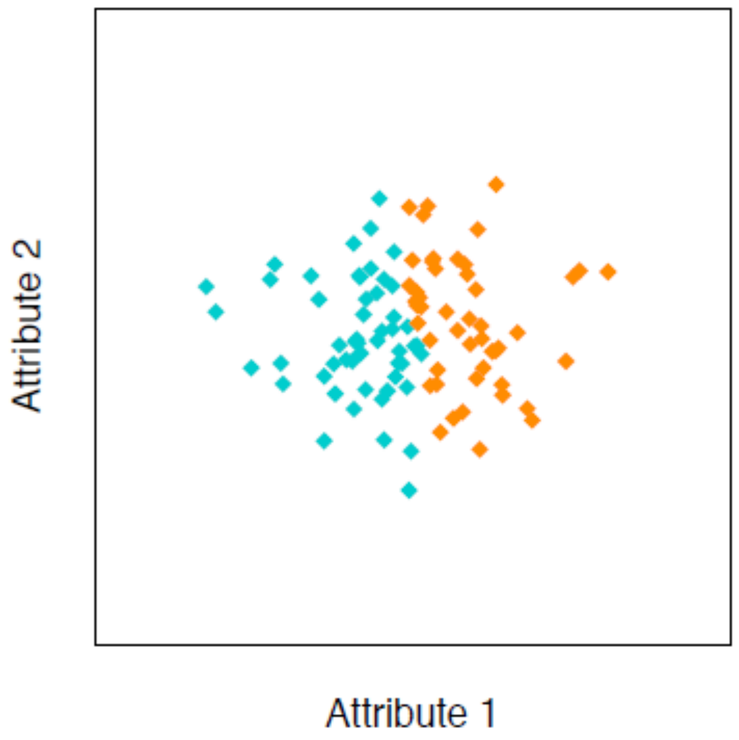
Note: maximizing the clustering likelihood/objective will not be informative→ each object should be in its own cluster. Therefore, need an algorithm that takes into account the "cost" of additional clusters.

- Prior knowledge
- The "elbow method"
- Information Criteria Approach: AIC or BIC
- Silhouette method
- The Gap Statistics
- Cross-validation



Natural clusters are regions in the attribute space that are densely populated, separated from other such regions by areas that are sparsely populated -- “internal cohesion” and “external isolation”

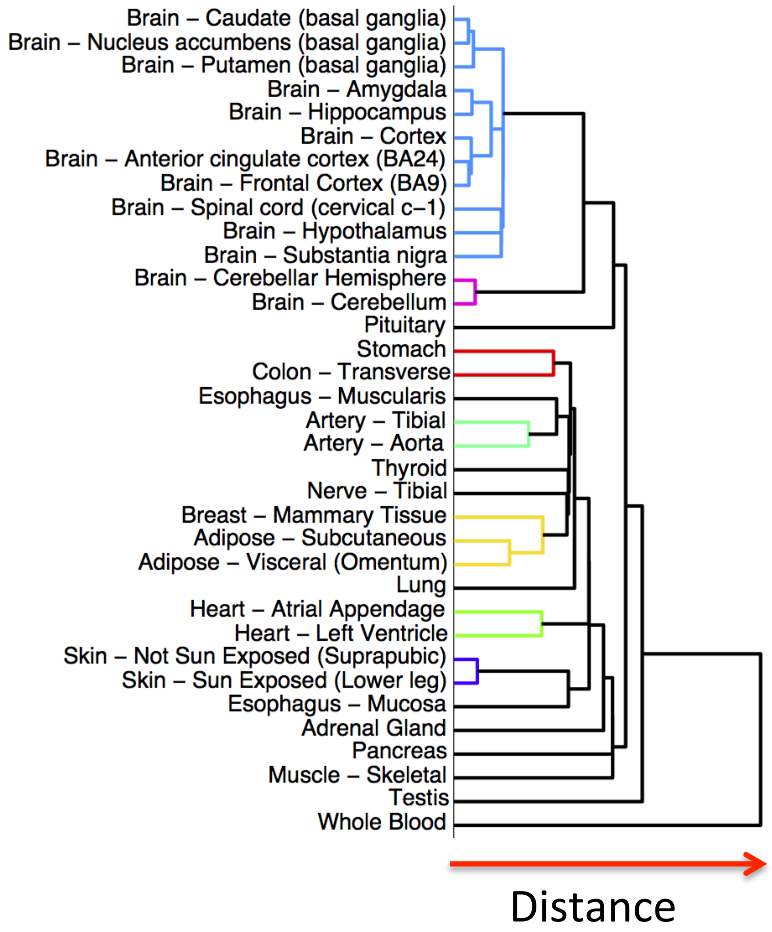




In the absence of natural clusters, grouping is called **data segmentation**. Not in the control of the analyst or the algorithm.

Hierarchical Agglomerative clustering

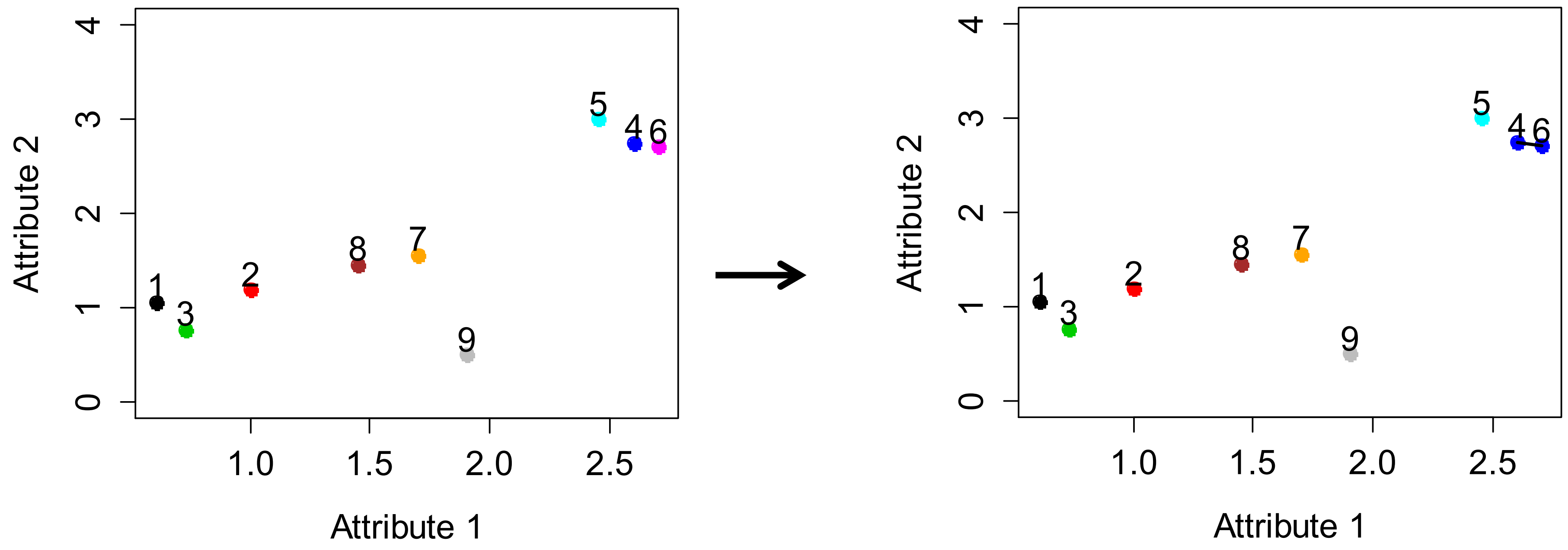
A clustering approach for revealing hierarchical relationships between objects

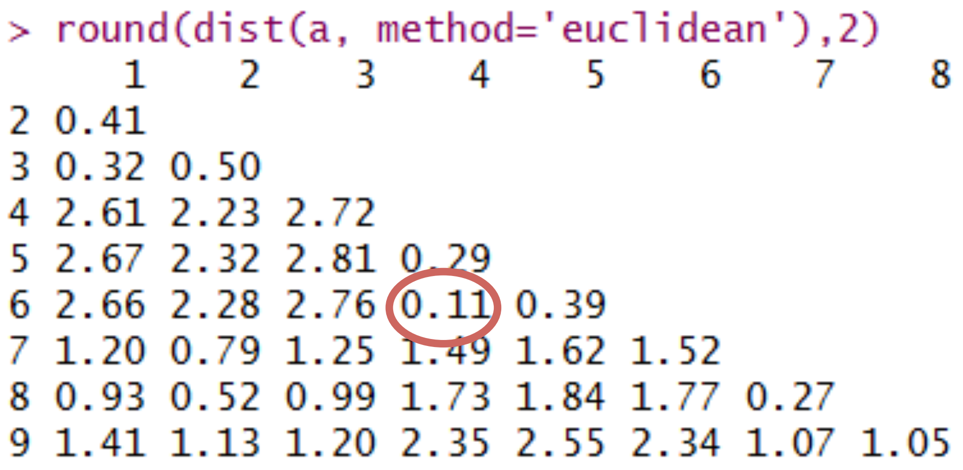


Algorithms: Hierarchical

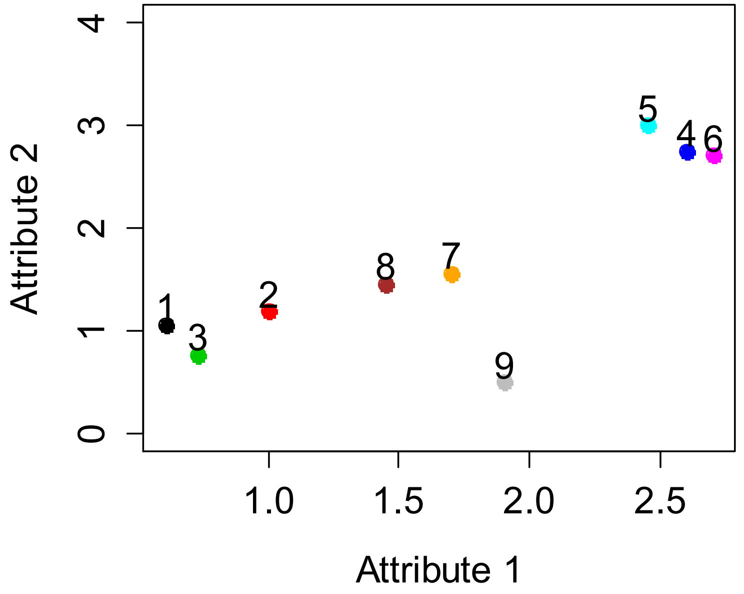
Given *N objects* with *H attributes* and a *distance metric*:

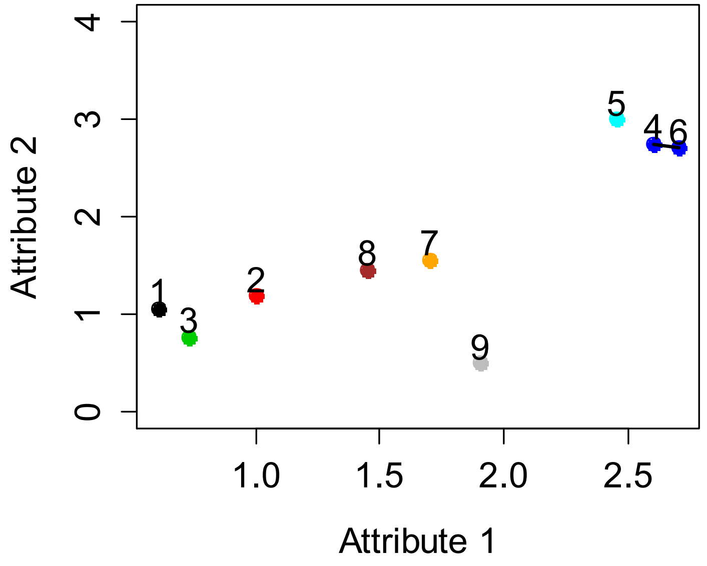
1. Assign each object to a cluster and compute the pairwise distances between all clusters
2. Find the “closest” pair of *clusters* and *merge them* into a single cluster
3. Compute new distances between clusters
4. Repeat steps 2 and 3 until all objects belong to a single cluster.

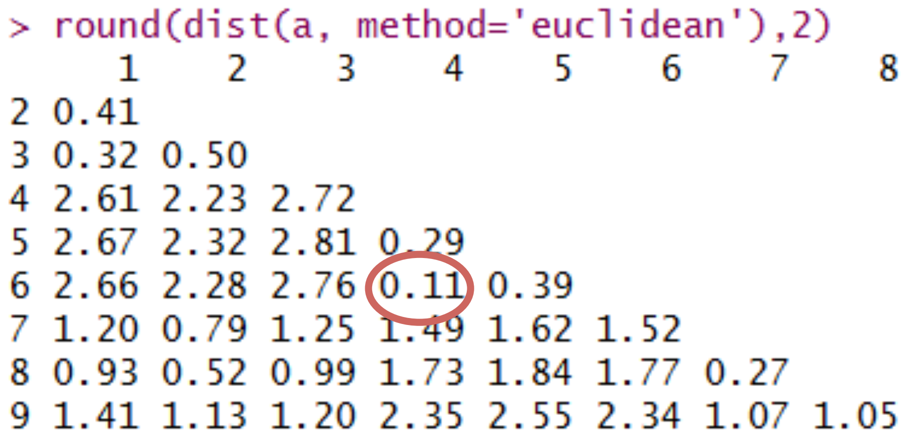




```
> round(dist(a, method='euclidean'),2)
     1    2    3    4    5    6    7    8
2 0.41
3 0.32 0.50
4 2.61 2.23 2.72
5 2.67 2.32 2.81 0.29
6 2.66 2.28 2.76 0.11 0.39
7 1.20 0.79 1.25 1.49 1.62 1.52
8 0.93 0.52 0.99 1.73 1.84 1.77 0.27
9 1.41 1.13 1.20 2.35 2.55 2.34 1.07 1.05
```



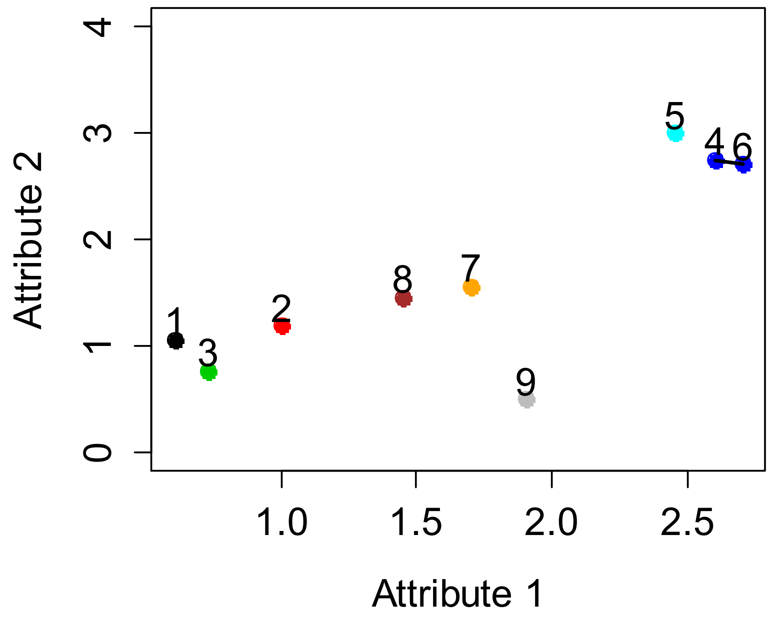




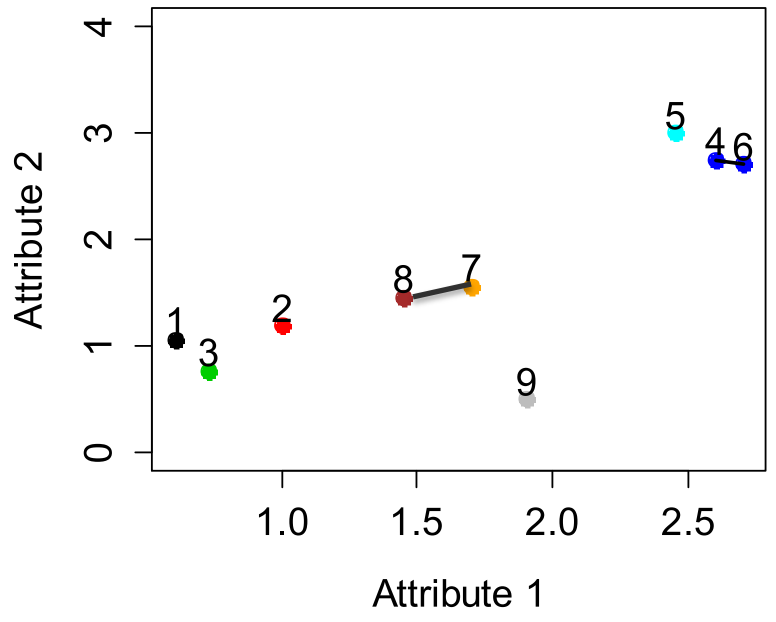
```
> round(dist(a, method='euclidean'),2)
     1    2    3    4    5    6    7    8
2 0.41
3 0.32 0.50
4 2.61 2.23 2.72
5 2.67 2.32 2.81 0.29
6 2.66 2.28 2.76 0.11 0.39
7 1.20 0.79 1.25 1.49 1.62 1.52
8 0.93 0.52 0.99 1.73 1.84 1.77 0.27
9 1.41 1.13 1.20 2.35 2.55 2.34 1.07 1.05
```

→ You can define the cluster "centroids" using:

- Single linkage
- Average linkage
- Complete linkage



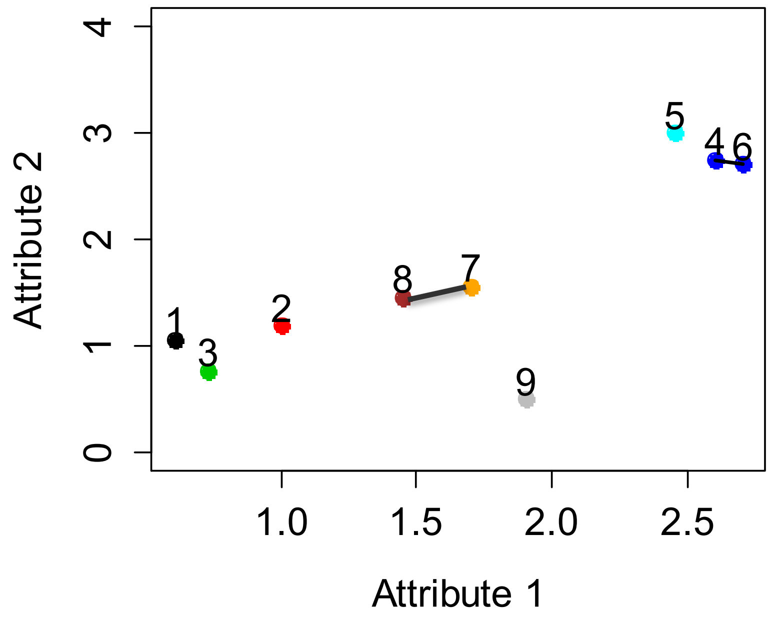
→



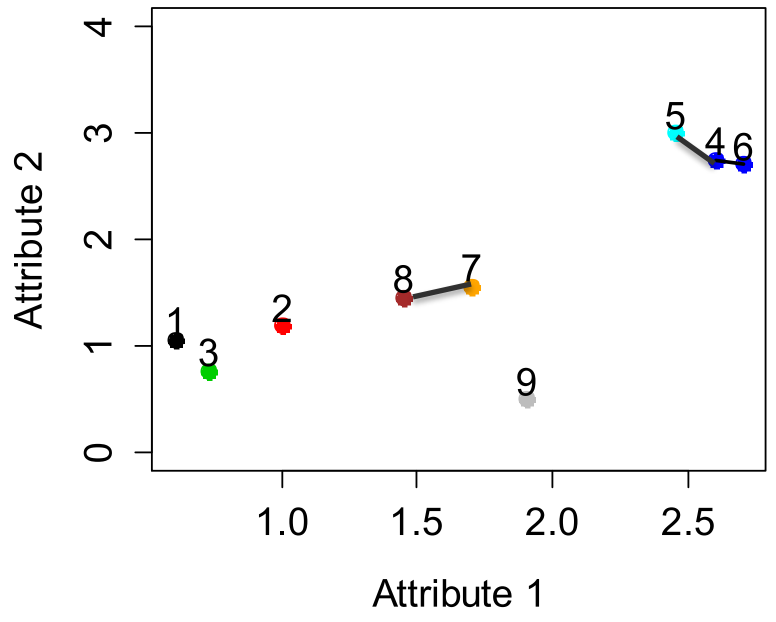
```
> round(dist(a, method='euclidean'),2)
     1    2    3    4    5    6    7    8
2 0.41
3 0.32 0.50
4 2.61 2.23 2.72
5 2.67 2.32 2.81 0.29
6 2.66 2.28 2.76 0.11 0.39
7 1.20 0.79 1.25 1.49 1.62 1.52
8 0.93 0.52 0.99 1.73 1.84 1.77 0.27
9 1.41 1.13 1.20 2.35 2.55 2.34 1.07 1.05
```

→ You can define the cluster “centroids” using:

- Single linkage
- Average linkage
- Complete linkage



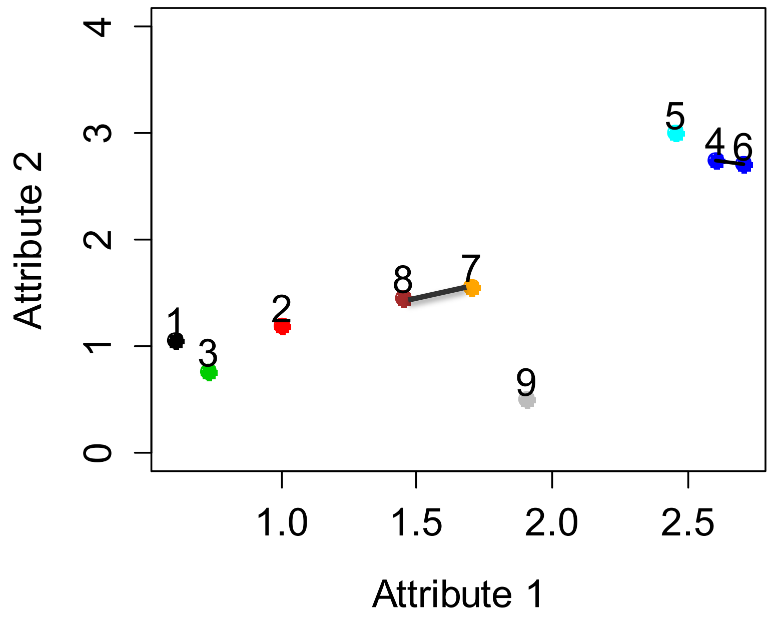
→



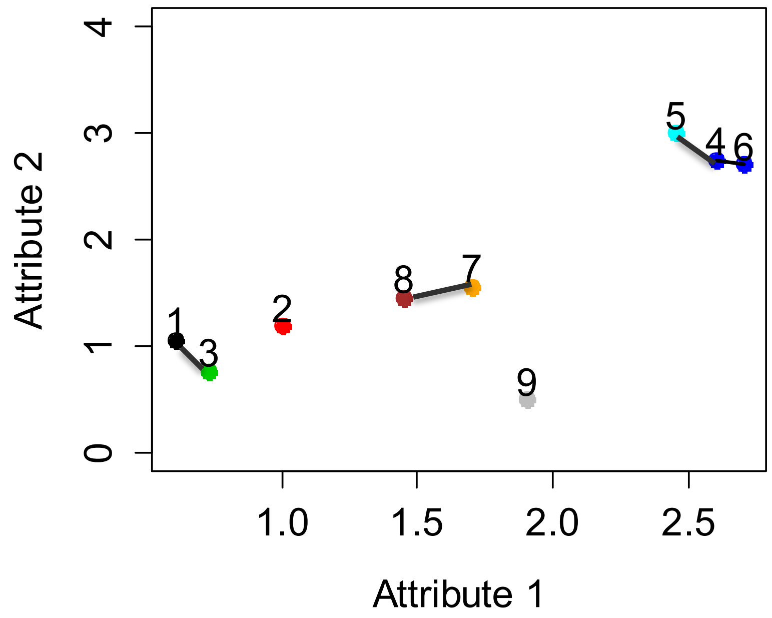
```
> round(dist(a, method='euclidean'),2)
     1    2    3    4    5    6    7    8
2 0.41
3 0.32 0.50
4 2.61 2.23 2.72
5 2.67 2.32 2.81 0.29
6 2.66 2.28 2.76 0.11 0.39
7 1.20 0.79 1.25 1.49 1.62 1.52
8 0.93 0.52 0.99 1.73 1.84 1.77 0.27
9 1.41 1.13 1.20 2.35 2.55 2.34 1.07 1.05
```

→ You can define the cluster "centroids" using:

- Single linkage
- Average linkage
- Complete linkage



→

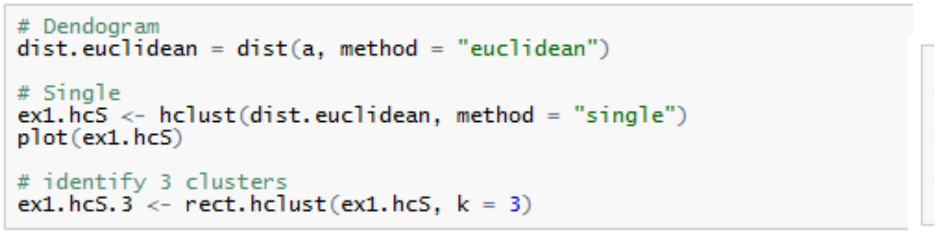


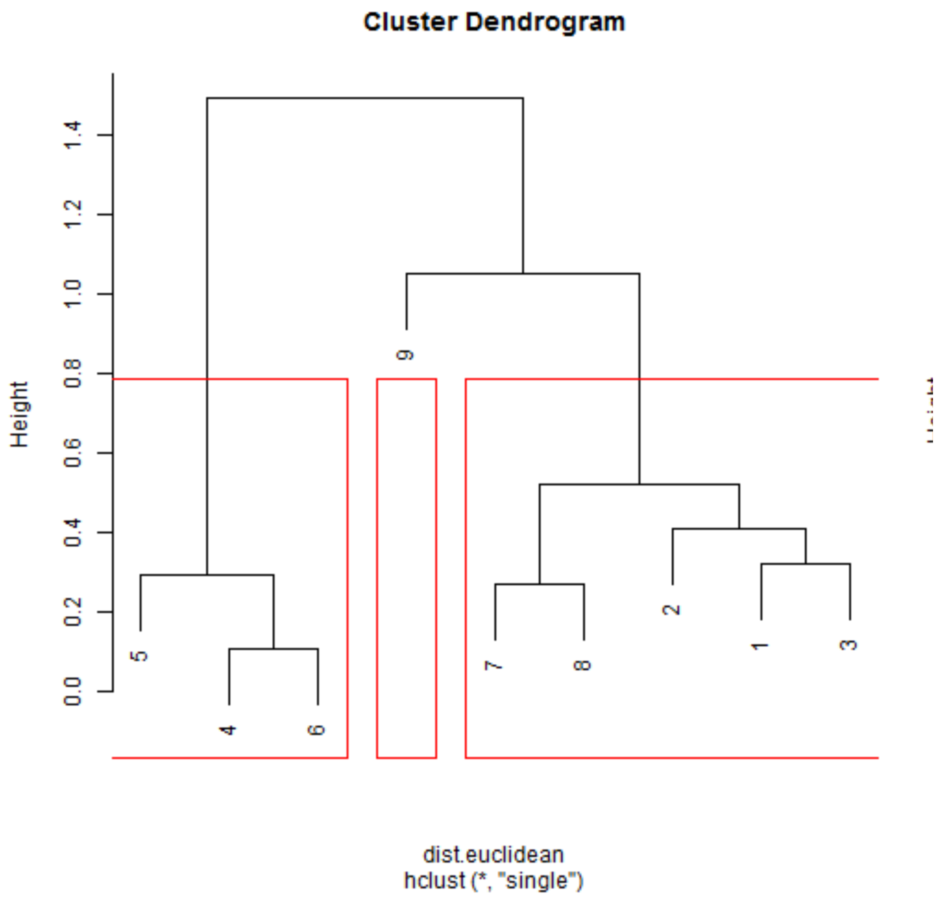
```
> round(dist(a, method='euclidean'),2)
     1    2    3    4    5    6    7    8
2 0.41
3 0.32 0.50
4 2.61 2.23 2.72
5 2.67 2.32 2.81 0.29
6 2.66 2.28 2.76 0.11 0.39
7 1.20 0.79 1.25 1.49 1.62 1.52
8 0.93 0.52 0.99 1.73 1.84 1.77 0.27
9 1.41 1.13 1.20 2.35 2.55 2.34 1.07 1.05
```

→ You can define the cluster “centroids” using:

- Single linkage
- Average linkage
- Complete linkage

Single Linkage

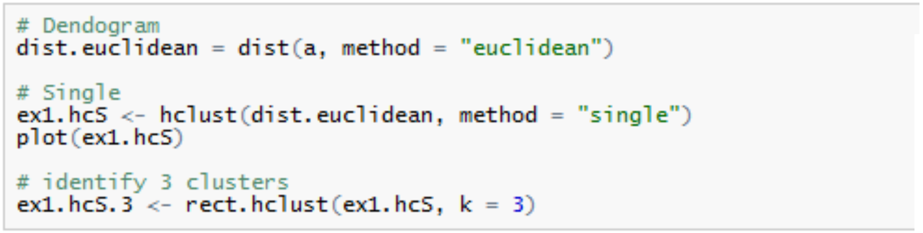

```
# Dendogram
dist.euclidean = dist(a, method = "euclidean")

# Single
ex1.hcS <- hclust(dist.euclidean, method = "single")
plot(ex1.hcS)

# identify 3 clusters
ex1.hcS.3 <- rect.hclust(ex1.hcS, k = 3)
```



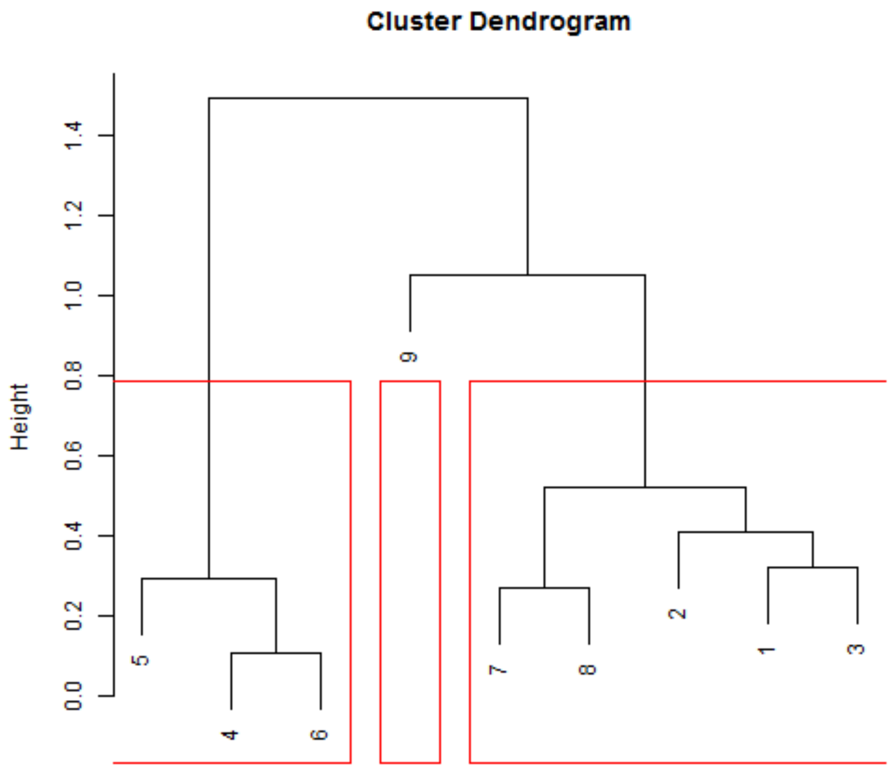
Agglomerative clustering

- **Single linkage**: The distance between two clusters is the *minimum* distance between any two elements.
- **Complete linkage**: The distance between two clusters is the *maximum* distance between any two elements.
- **Average linkage**: The distance between two clusters is the *average* of all pairwise distances between any two objects.

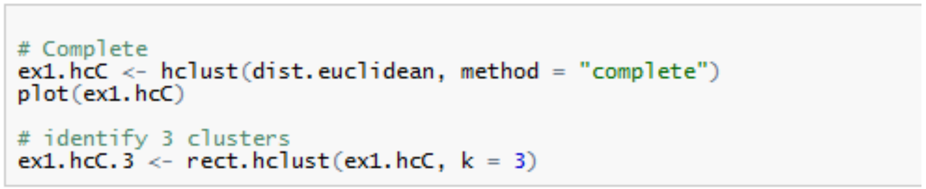
Single Linkage



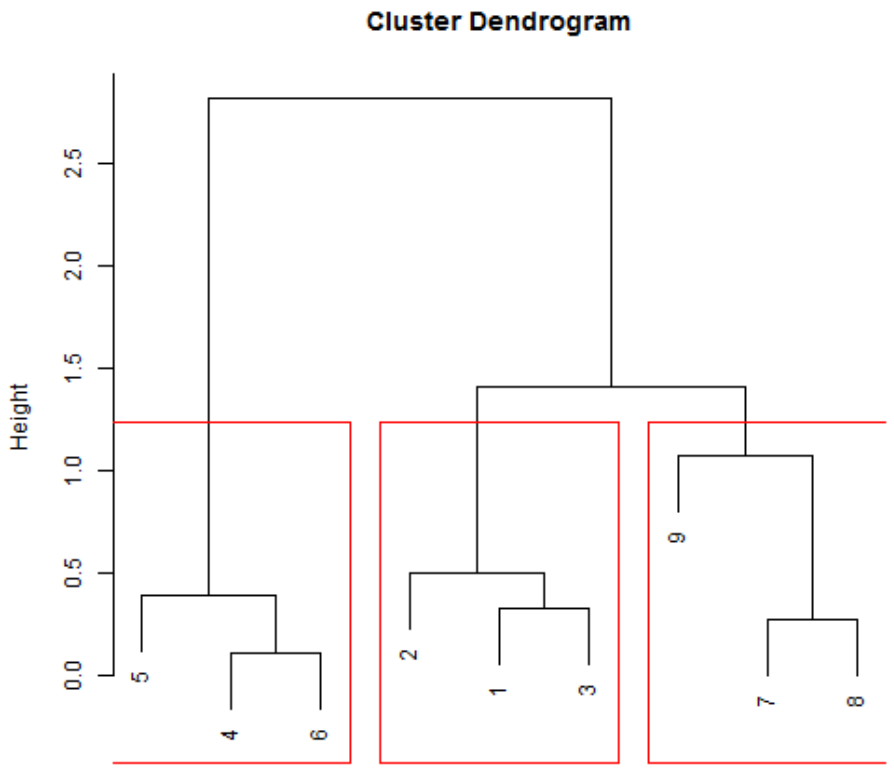
```
# Dendogram
dist.euclidean = dist(a, method = "euclidean")

# Single
ex1.hcS <- hclust(dist.euclidean, method = "single")
plot(ex1.hcS)

# identify 3 clusters
ex1.hcS.3 <- rect.hclust(ex1.hcS, k = 3)
```



Complete Linkage



```
# Complete
ex1.hcC <- hclust(dist.euclidean, method = "complete")
plot(ex1.hcC)

# identify 3 clusters
ex1.hcC.3 <- rect.hclust(ex1.hcC, k = 3)
```



Clustering

- Example of a non-convex problem (lots of local maxima)
  - Initialization dependent
  - NP hard problem (not possible to find the "optimal" solution)
- There is always a structure in high-dimensional data (even if data is random)

Part II: PCA

- Summarize the data using dimensionality reduction.

- Visualization: allows us to represent each data instance as a “point” in a lower dimensional space (in order to visualize all data points and their relative distances simultaneously)

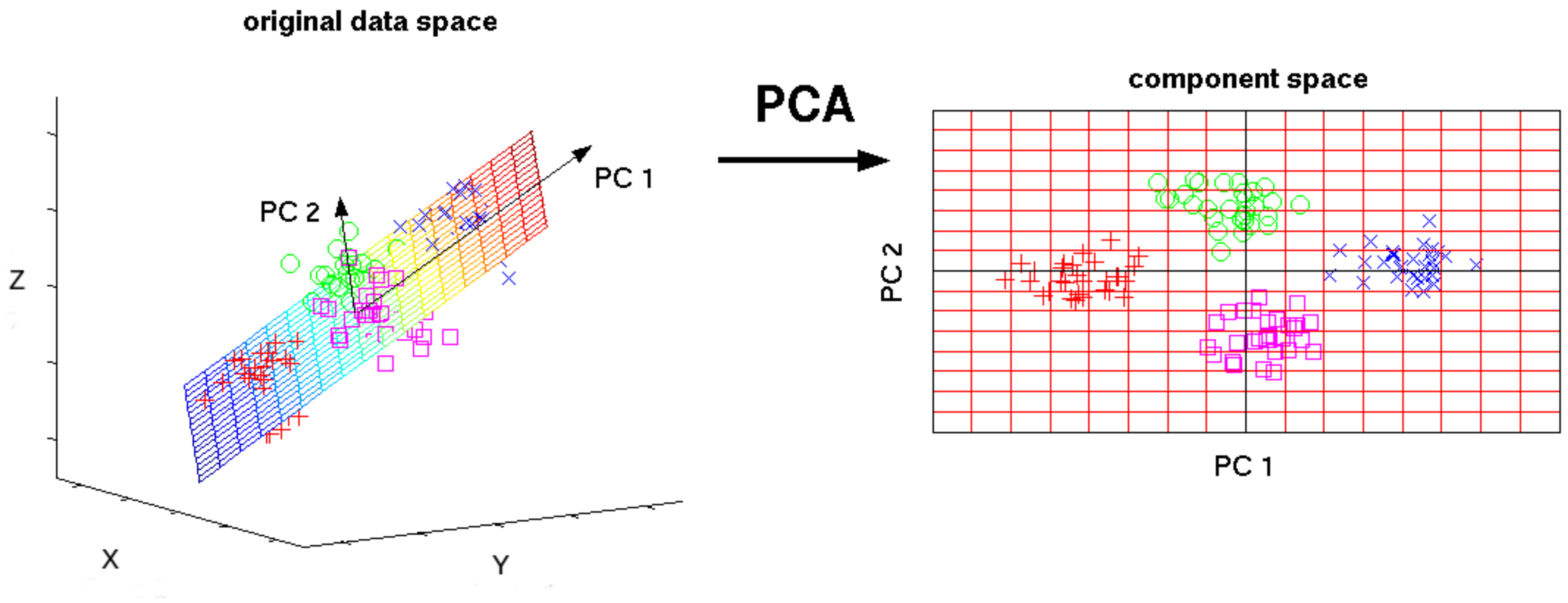
- Unsupervised (as is clustering).

What PCA does

- It chooses a “good” set of directions to which to project the data.

- The definition of “good” starts with “find the direction in the data that has the largest variance”

Projection data from 3-D to 2-D



[google images]

Basis and Span

- A basis for R^m (m-dimensional space) is (roughly) a set of *m* vectors from which you can "make" any vector in R^m

- An orthogonal basis is a nice kind of basis: the *m* vectors are orthogonal.
  - E.g., [0,2] and [2,0]
  - Any pair of orthogonal 2-vectors is a basis in R^2

- Even nice is an **orthonormal basis**, where the vectors and normalized to length 1:
  - E.g., [0,1] and [1,0]

Vector span and matrix rank

- You need at least *m* vectors space an m-dimensional space (R^m)

- With data matrices X_{nxm} that are “rectangular” the rank (max number of linearly independent column) is min(n,m)

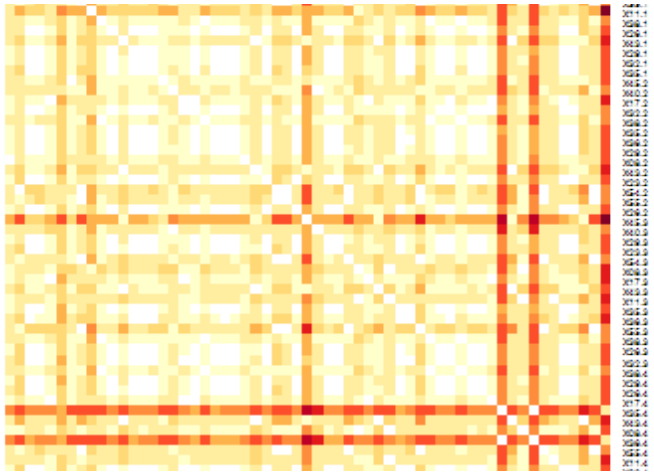
Review I: sample correlation matrix

K_{ij} = covariance between data point i and data point j
K_{ii} = Variance of data point i

$$\mathrm{Cov}(\mathbf{x},\mathbf{y}) = \frac{\sum_i^N (x_i - \overline{x})(y_i - \overline{y})}{N-1}$$

Are you "far" from your mean when I am far from my mean?

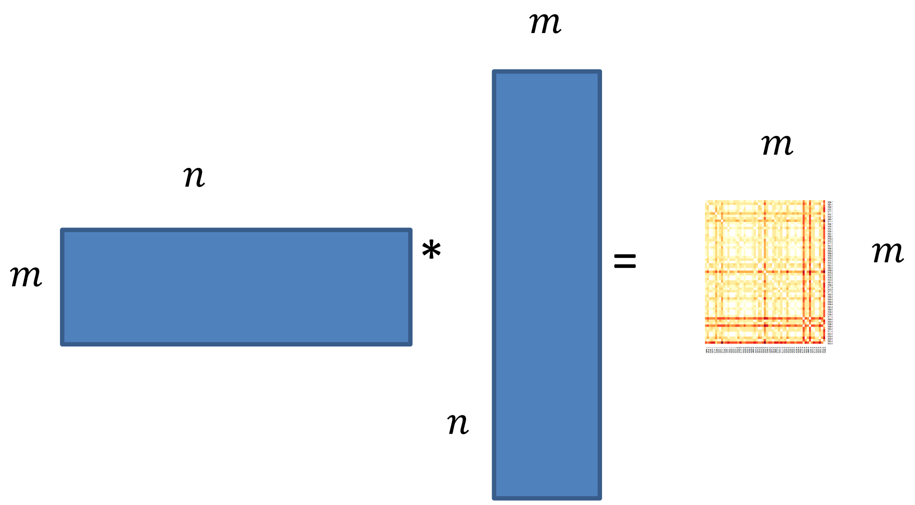
K



- X^TX on standardized data gives you the covariance matrix
- **Row vs column** covariance matrix
- Correlation vs covariance matrix – interchangeable for now (i.e., equivalent on standardized data)

Three thought experiments

- Think about the correlation structure of the data X_{mxn}, and understand that they can be described by at least *m independent* vector (where m<n)



What does the *sample* correlation matrix look like when..

1. All samples are perfectly correlated.
2. No correlation between the samples.
3. There are two groups and samples are correlated within each group.

What does the *sample* correlation matrix look like when..

1. All samples are perfectly correlated.

2. No correlation between the samples.
   – While n>>m, realize that you only need m vectors to construct such a covariance matrix.

3. There are two groups and samples are correlated within each group.

PCA: goals

- Find the m orthonormal vectors that span our data matrix X
- These orthonormal vectors should capture the “most amount of variance” when we use them to project our data.

PCA as an optimization problem

- Iteratively solve convex optimization problems:

$$\max_{v_1} v_1^T X^T X v_1,$$

$$v_1^T v_1 = 1.$$

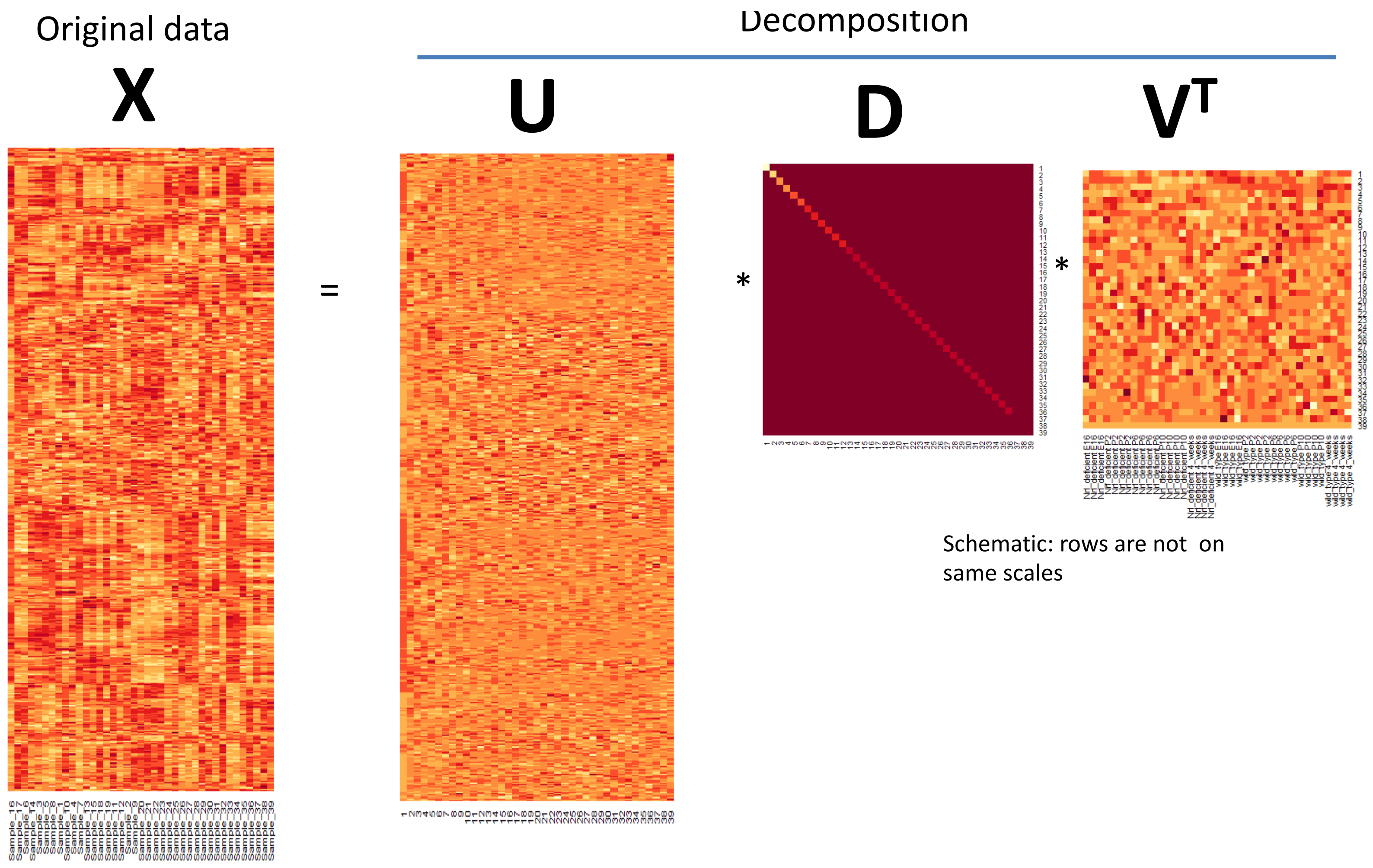
- If X is square then V_i are the eigenvectors of covariance matrix. These eigenvectors define the Principal Components.
  - Define the direction of maximal variance

SVD

- In practice for non square matrices we use SVD.

- X_{nxm}=UDV^T

- U_{nxm} left singular vectors – orthonormal basis of column space of X
- V_{nxm} right singular vectors – orthonormal basis of row space of X
- D diagonal matrix of eigenvalues



Schematic: rows are not on same scales

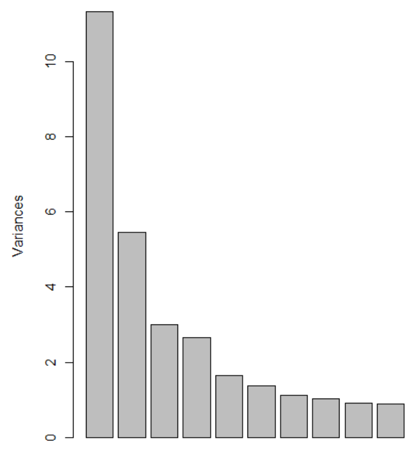
- The columns of U are "eigensamples" : samples are mixtures of them
- The columns of V are "eigengenes" (rows of V'): genes are mixtures of them
- D tells us how much of each eigensample or eigengene needs to be mixed together to reconstruct each gene or sample vector.

Practical stuff

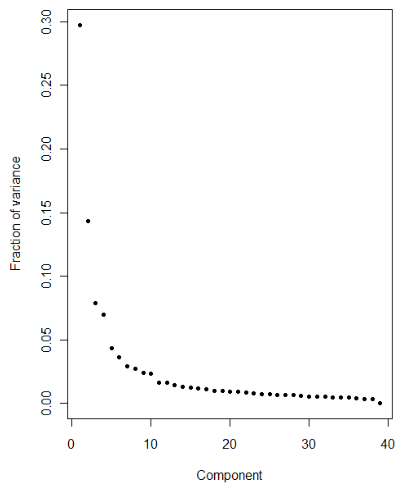
- Before applying SVD/PCA standardize your data
  - Otherwise the first PC captures the average expression/ intensity, this constraints the next direction (recall orthogonally)
  - You can standardize each feature (e.g., gene) or sample, or both.
  - The number of eigenvalues corresponds to the smaller dimension min(n,m)
- R notes:
  - svd(dat)
  - prcomp(dat): calls svd(dat) and gives you stdev(square roots of eigenvalues) and rotation(column of eigenvectors) a.k.a loadings

Scree plot

- Shows the relative magnitudes of the eigenvalues (can translate to proportion of variance explained very easily)



```
plot(pca)
```



Standard usage for PCA

- Visualize data using PCs 1-3.
- Assess correlation between top PCs and "external" variables of interest.
- Identify genes associated with different PCs (need correlation analysis; too hard to interpret the loadings)

Other applications of PCA

- SVD as a batch correction approach
- If you suspect or can show that for example PC1 correlates with batch, then you can “remove” it’s effect. (show on board)

PC1 and PC2 capture population stratification in GWAS

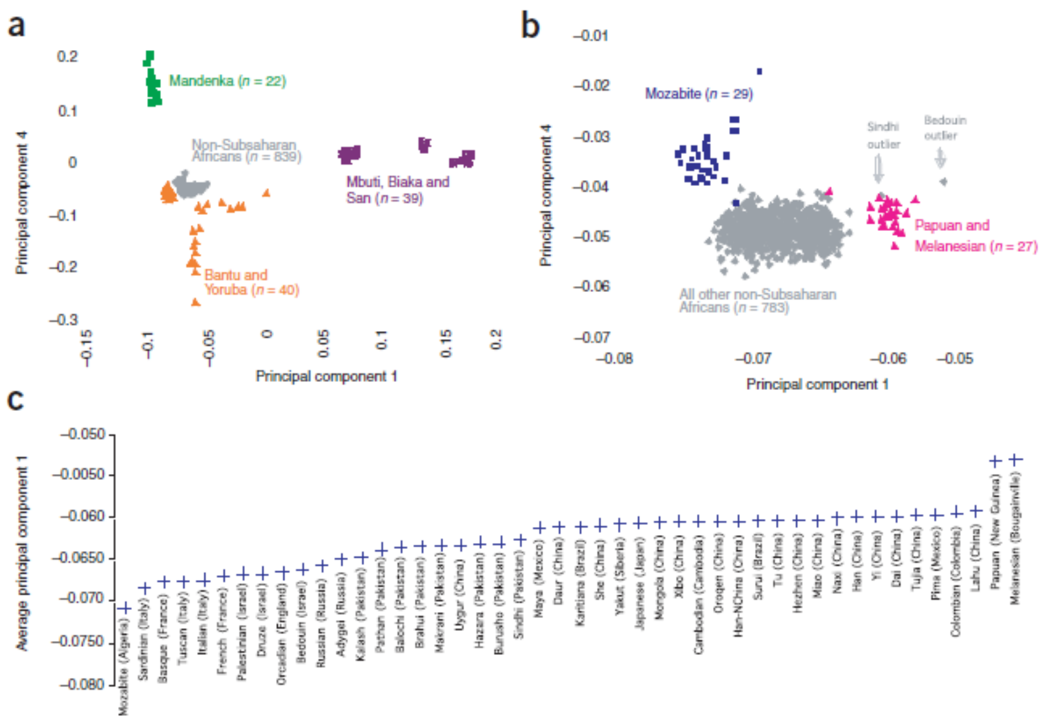


Figure 1 PCA continues to provide evidence of important migration events. (**a**) We carried out PCA on 940 individuals from the Human Genome Diversity Project that were scanned at approximately 650,000 SNPs[11] using data from 101 sub-Saharan African samples to define the PCs (Mandenka, Bantu from Kenya and South Africa, Yoruba, San, Mbuti Pygmy and Biaka Pygmy). We carried out the analysis on samples blinded to population labels (the coloring of samples was only carried out after the analysis). We plotted principal component 1 (negative values are more Bantu-related) and principal component 4 (positive values are more closely related to the Senegalese Mandenka). (**b**) Outlying populations are the Mozabite, who are more Mandenka-related, reflecting recent gene flow across the Sahara, and Papuans and Melanesians, who have inherited less Bantu-related gene flow. (**c**) To reveal the west-to-east gradient of Bantu-related ancestry across Eurasia, we averaged the first PC for each of the non-African populations and plotted the populations in rank order.

Read book chapters on PCA/clustering

- Pattern recognition in Machine Learning (Bishop)
- Elements of statistical learning (Friedman, Tibshirani, Hastie)