Statistical Methods for High Dimensional Biology

Clustering

Keegan Korthauer

3 March 2021

with slide contributions from Sara Mostafavi, Gabriela Cohen-Freue, and Jenny Bryan

Learning objectives

- [continuing from last lecture] Explain the purpose of **unsupervised learning**
  - (and in future lectures) understand how it differs from **supervised learning**
- Explain the goals of **clustering** with respect to high-dimensional biological data
- Understand the *objective functions* of two popular clustering algorithms:
  - **K-means** clustering
  - **Hierarchical** clustering
- Apply K-means and Hierarchical clustering to a high-dimensional genomic dataset

Recall: Unsupervised learning

A procedure or algorithm which aims to find patterns or structure in the data - *without using any outside knowledge (e.g. response variable, metadata, 'labels', etc)*

The distinction between unsupervised and supervised learning will become more clear in future lectures

Recall: Two general frameworks:

1. **Dimension reduction** (last lecture)
2. **Clustering** (today)

Clustering analysis in Microarrays

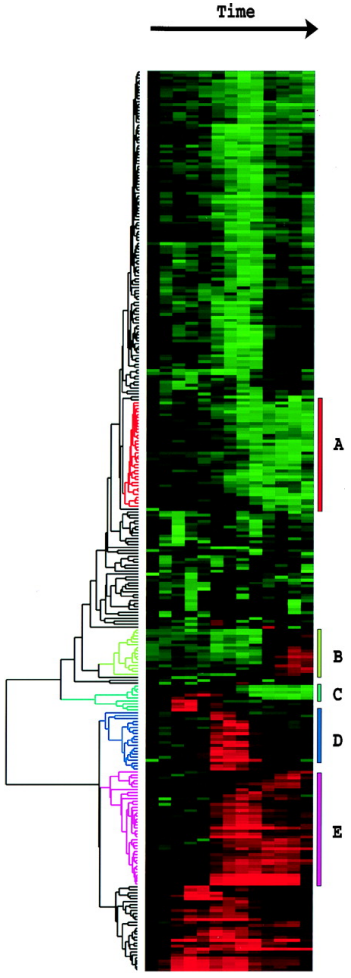


Figure 1 from Eisen, et al. 1998, "Cluster analysis and display of genome-wide expression patterns": time course of serum stimulation of primary human fibroblasts

- Introduced cluster analysis on the microarray community
  - Clustering was used to "organize" genes into groups (clusters) and create dendrogram
- Almost 20K citations as of early 2021
- This precedent + explosion of array data + ease of application = widespread use of clustering
- Currently, clustering analysis is used similarly in many other omics studies

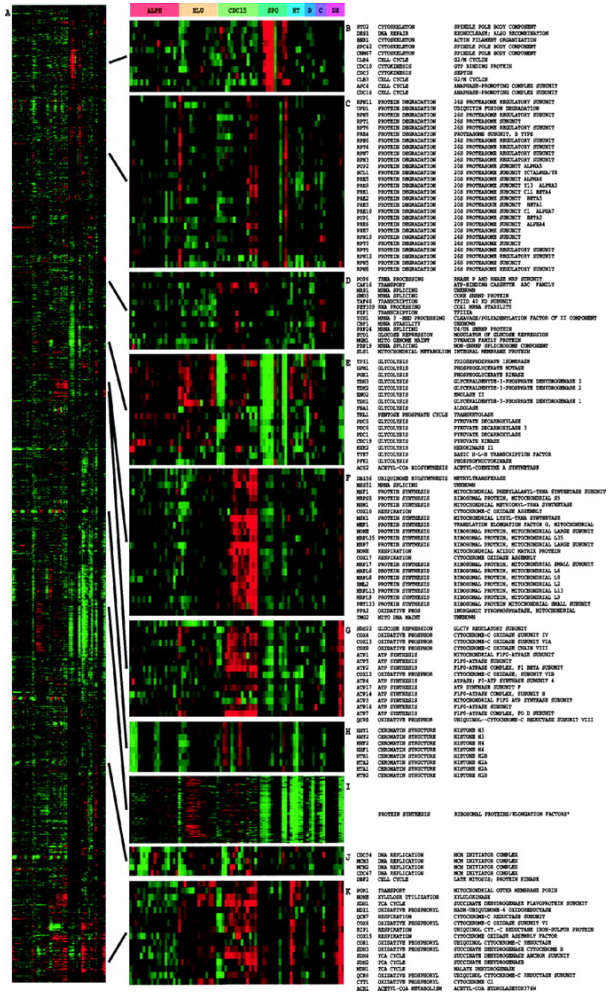
Clustering analysis in Microarrays

Key finding of Eisen, et al. (1998):

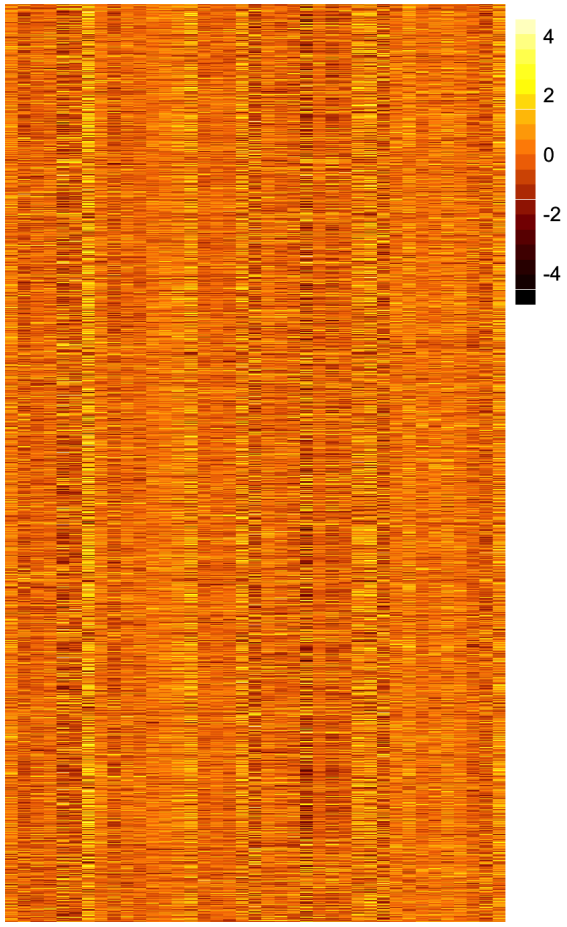
Clustering genes by co-expression patterns results in groups of genes that have commonalities in function

i.e. involved in similar cellular/biological processes

Right - Figure 2: Various time courses of gene expression in the yeast were combined; representative clusters containing functionally related genes are highlighted

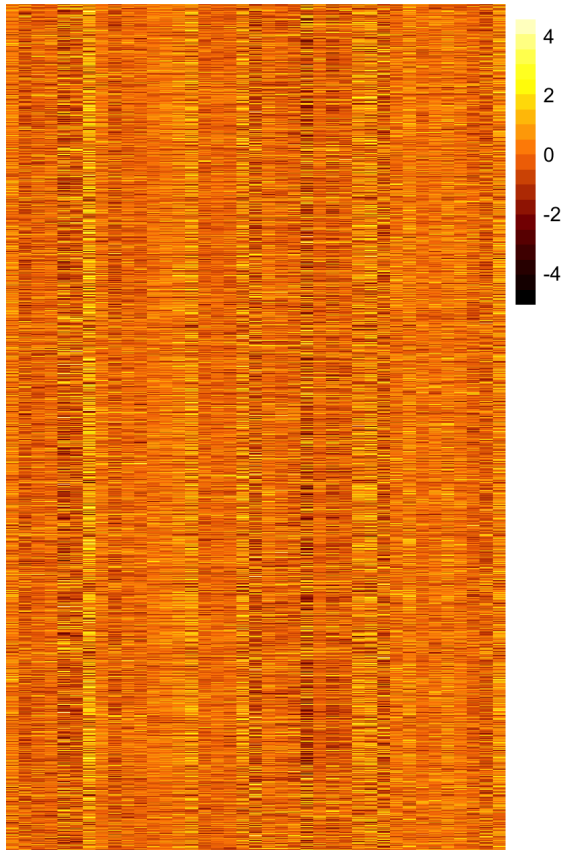


Visualizing raw expression data is not informative

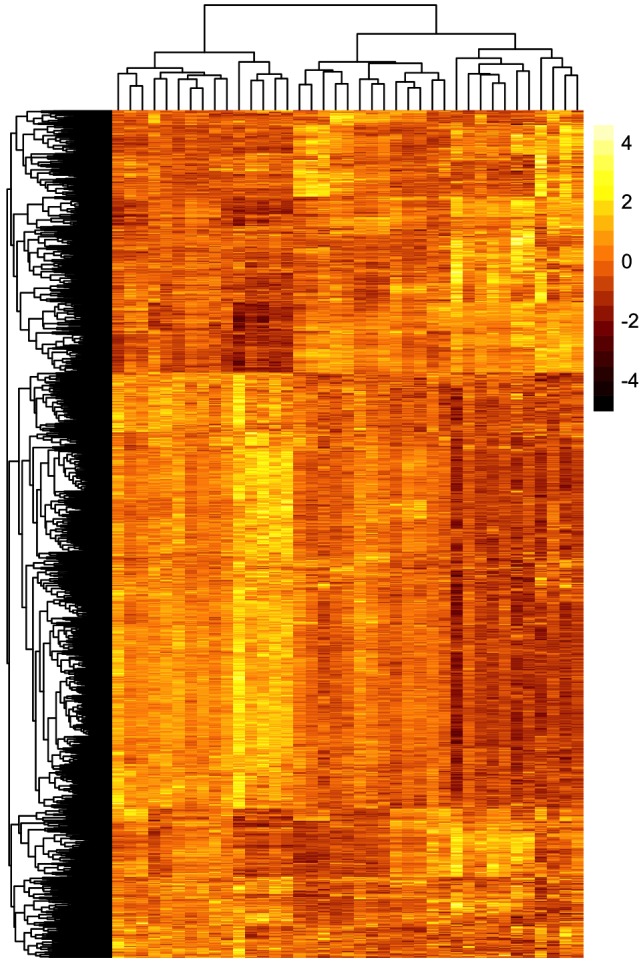


Photoreceptor dataset: ~45K genes (rows), 39 samples (columns); centered/scaled

After clustering rows and columns



Photoreceptor dataset: centered/scaled



Same heatmap, but clustered rows and columns

Two predominant applications of clustering

1. Identify groups of samples that have similar feature measurements
    - column clustering in previous slide
    - e.g. identifying disease subtypes
2. Identify groups of features (e.g. genes) that have correlated measurements
    - row clustering in previous slide
    - e.g. co-expressed genes that have similar functionality

What is clustering?

- Clustering colloquially means placing objects into groups such that they are more *similar* to one another than to objects in other groups
- Clustering is a formal problem in computer science and statistics, with formal definitions and "solutions"
- Clustering in bioinformatics is often used as a tool for visualization, hypothesis generation, selection of features for further analysis
- Keep in mind, with typical use of clustering in bioinformatics: there is no measure of *"strength of clustering structure/evidence"* available
- Rigorous application of clustering is very powerful but also hard to do (computational complexity, suitable definition of clustering objective, determining the number of clusters

Clustering problem: definition

Goal: place a set of objects into groups such that they are more *similar* to one another **within the group** than to objects in **other groups**

Rocks were clustered according to their colour and texture

- What other criteria might you cluster on?

Similar how?

Goal: place a set of objects into groups such that they are more *similar* to one another **within the group** than to objects in **other groups**

- How to define **similar**?
  - Need to gather a set of **attributes** (or features) for each object
  - Compare similarity of objects to one another based on these attributes
- Clustering **objective function**: maximize within cluster similarity
  - For a precise definition of good/optimal clustering, we need to translate this criteria into an equation

Defining attribute/feature vector for each object

- Numerically, we need to define a vector for each object that contains the feature/attribute information
- For a set of n objects, we can represent each object i $(i = 1, \ldots, n)$ as a numeric vector $\boldsymbol{x}_i$ of length p (containing p attributes)
- For example, for rock i we have $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$
  - x_{i1} : numerical value for colour/shade
  - x_{i2} : numerical value representing texture
  - x_{ip} : numerical value for attribute p

Similarity measures

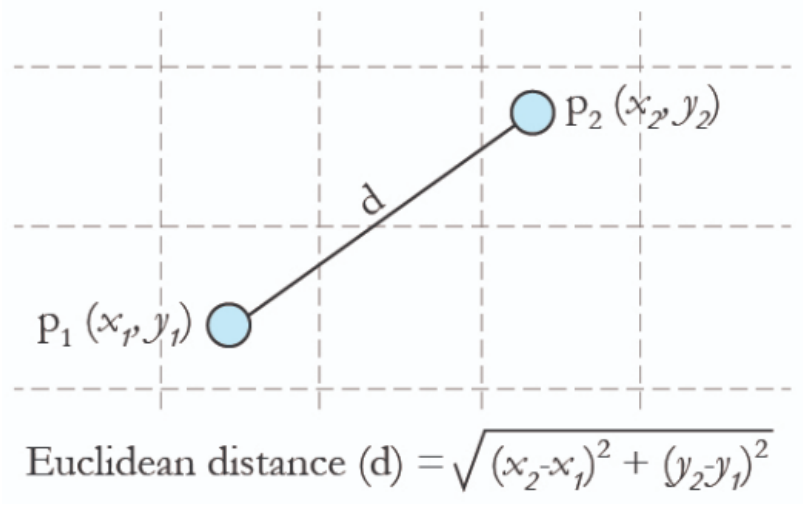
- Many clustering methods rely on a measure of similarity (or **distance**, which is just one minus similarity)
- From the feature vectors $\boldsymbol{x}_i$, we need to compute a measure of similarity or distance between all pairs of objects
  - This gives us a *matrix* of similarity or distance values
- Some commonly used measures
  - **Distance**: Euclidean, Manhattan
  - **Similarity**: Correlation (Spearman rank, Pearson)

Euclidean distance

Euclidean distance between two feature vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$:

$$D_{euc}(\boldsymbol{x}_1, \boldsymbol{x}_2) = ||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2 = \sqrt{\sum_{j=1}^{p}(x_{1j} - x_{2j})^2}$$

In two dimensions:



Pearson correlation

Pearson correlation (similarity) between two feature vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$:

$$r_{\boldsymbol{x}_1,\boldsymbol{x}_2} = Cor(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{Cov(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\sqrt{Var(\boldsymbol{x}_1)Var(\boldsymbol{x}_2)}}$$

Convert to a distance by taking $1 - r_{\boldsymbol{x}_1,\boldsymbol{x}_2}$

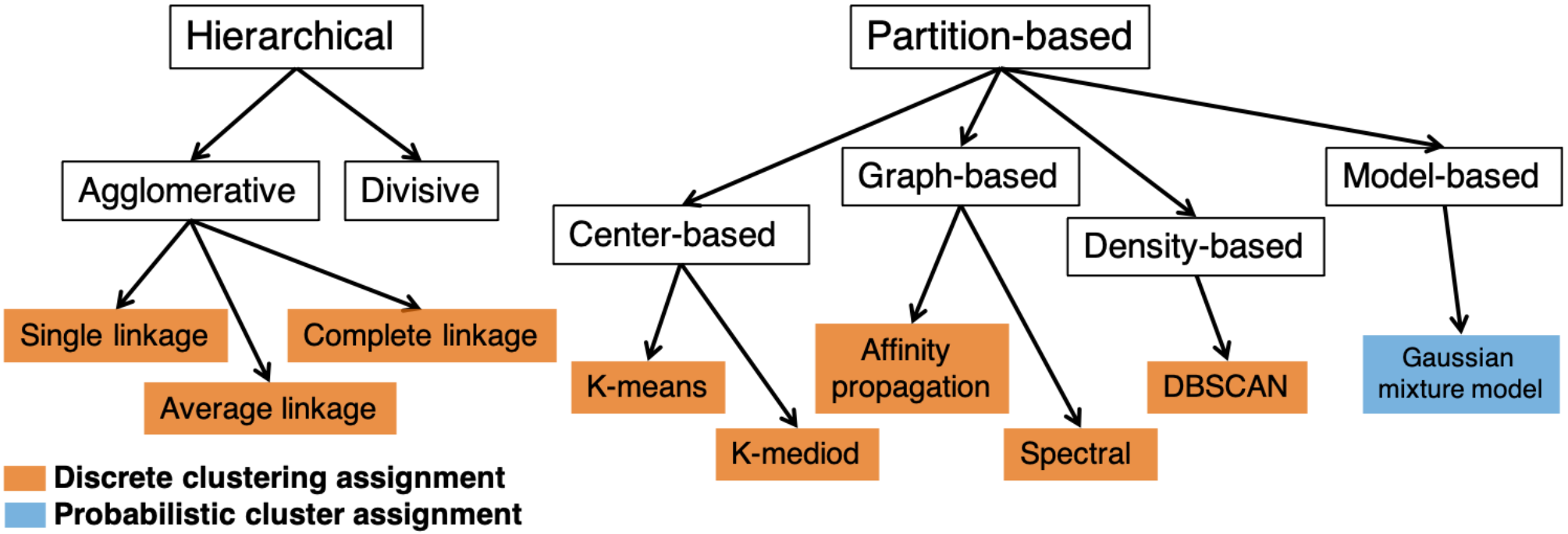
If features are **standardized** (centered and scaled), then:

$$r_{\boldsymbol{x}_1,\boldsymbol{x}_2} = \sum_{j=1}^{p} x_{1j}x_{2j}$$

Note the connection with Euclidean distance for standardized features:

$$D^2_{euc}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{j=1}^{p}(x_{1j} - x_{2j})^2 = \sum_{j=1}^{p} x_{1j}^2 + \sum_{j=1}^{p} x_{2j}^2 + 2\sum_{j=1}^{p} x_{1j}x_{2j} = 2(1 - r)$$

Example clustering algorithms



How many clusters?

Almost all clustering algorithms that partition the objects require the user to specify the number of clusters

There are ways of 'automatically' determining the number of clusters, but they require us to specify additional criteria or objective function

What is an algorithm?

An **algorithm** is a self-contained step-by-step set of operations to be performed in order to achieve a given task

Through a set of steps, an algorithm transforms a given **input** data into the desired **output**

Clustering algorithms: inputs and outputs

- **Input**:
  - Data matrix, e.g. $X_{p \times n}$ (features in rows, objects in columns)
  - Number of clusters K
- **Output**:
  - Discrete: Assignment of cluster membership for each object C_i, where $C_i = k$ if object i is assigned to cluster k
  - Probabilistic: K-length vector of probabilities for each object $\boldsymbol{C}_i$, were $C_{ik} \in [0, 1]$ is the probability that object i belongs to cluster k, and $\sum_{k=1}^{K} C_{ik} = 1$

K-means clustering

- One of the most widely used partition-based clustering approaches
- Partition-based (flat), and discrete
- **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
- Objective function formula to minimize for *n* objects, each with p attributes $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$:

$$\sum_{k=1}^{K} \sum_{i=1, i \in k}^{n} ||\boldsymbol{x}_i - \mu_k||^2 = \sum_{k=1}^{K} \sum_{i=1, i \in k}^{n} \sum_{j=1}^{p} (x_{ij} - \mu_k)^2$$

where μ_k is the cluster center for cluster k

More on the K-means objective function

Note that the **total** distance (T) between pair all pairs of objects i and j can be broken down into the sum of **within** cluster distance (W) and **between** cluster distance (B)

$$
\begin{aligned}
\sum_{i=1}^{n}\sum_{j=1}^{n} d(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \sum_{k=1}^{K}\sum_{i=C_k}\sum_{j=1}^{n} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&= \sum_{k=1}^{K}\sum_{i=C_k}\left[\sum_{j\in C_k} d(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{j\notin C_k} d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right] \\
&= \sum_{k=1}^{K}\sum_{i,j\in C_k} d(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{k=1}^{K}\sum_{i\in C_k}\sum_{j\notin C_k} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
T &= W + B
\end{aligned}
$$

More on the K-means objective function

When $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2$,

$$W = \sum_{k=1}^{K} \sum_{i,j \in C_k} ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 = \sum_{k=1}^{K} \sum_{i \in C_k} ||\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k||^2$$

where $\bar{\boldsymbol{x}}_k = \{\bar{x}_{1k}, \bar{x}_{2k}, \ldots, \bar{x}_{pk}\}$ are the cluster vectors of feature means
$\bar{x}_{jk} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$

Given cluster means $\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \ldots, \bar{\boldsymbol{x}}_K$, the minimum of W is obtained by assigning $\boldsymbol{x}_i$ to the cluster C_k with the closest mean $\bar{\boldsymbol{x}}_k$

K-means algorithm

Initialize: Pick K random points as initial cluster centers

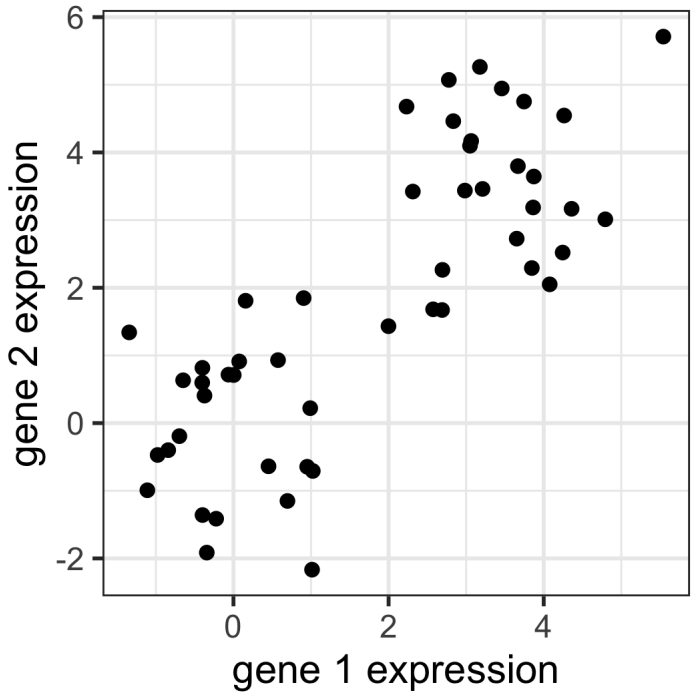
Iterate: steps 1-3 until there is no change from previous assignment

1. Measure distance (squared Euclidean) between all points and the cluster centers

2. Assign points to nearest cluster

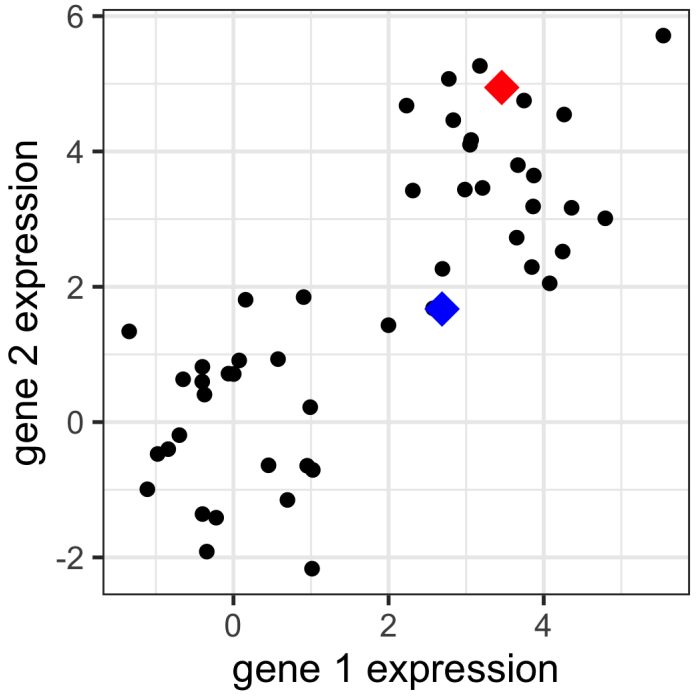
3. Update cluster means

Example: K-means algorithm



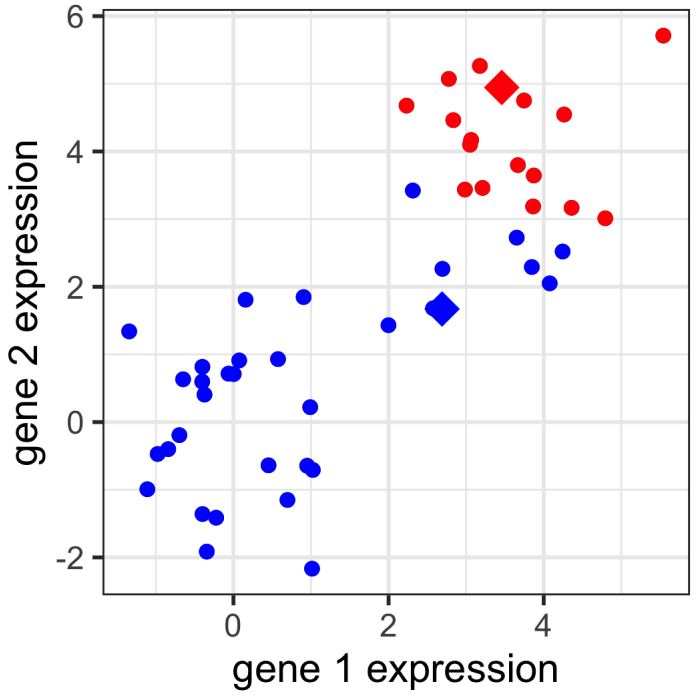
- Suppose we measure expression levels for 2 genes in 50 individuals
- Let's go through the K-means clustering algorithm step by step

Example: K-means algorithm



Initialize: Pick K random points as initial cluster centers (We'll use K = 2)

Example: K-means algorithm

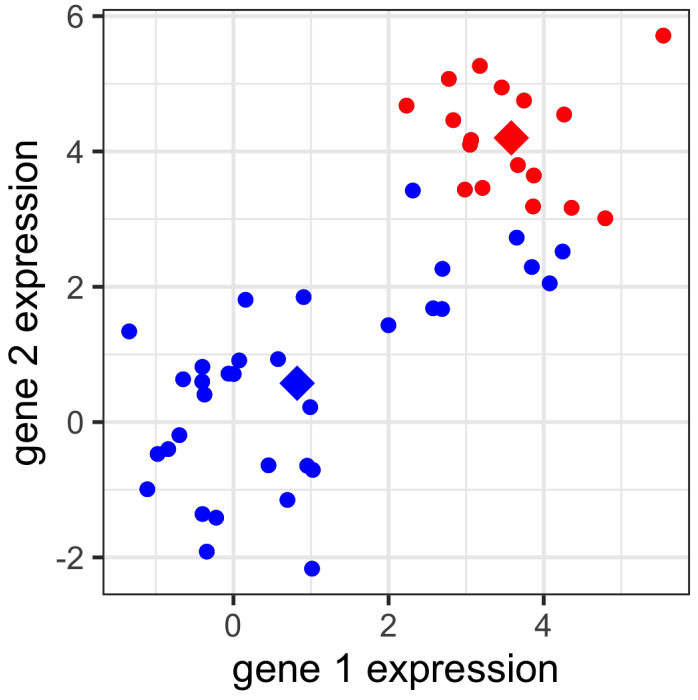


Initialize: Pick K random points as initial cluster centers

1. Measure distance (squared Euclidean) between all points and the cluster centers

2. Assign points to nearest cluster

Example: K-means algorithm



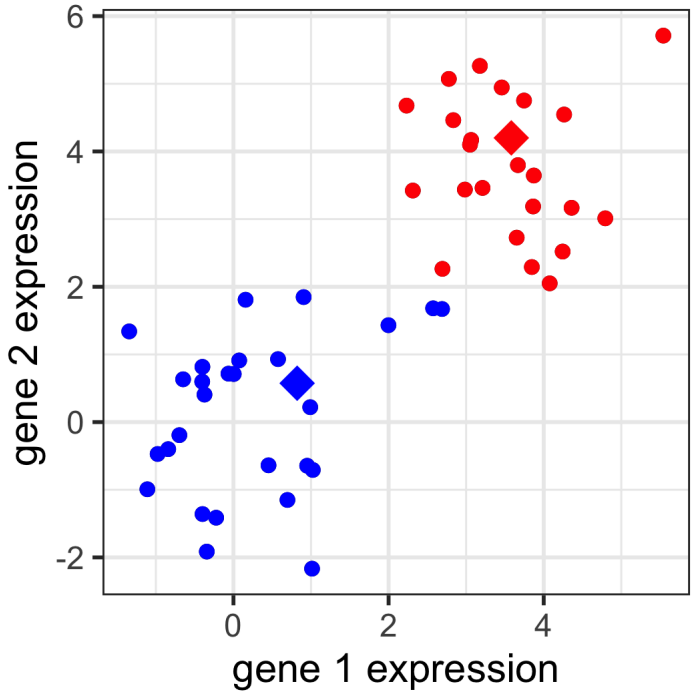
Initialize: Pick K random points as initial cluster centers

1. Measure distance (squared Euclidean) between all points and the cluster centers

2. Assign points to nearest cluster

3. Update cluster means

Example: K-means algorithm



Initialize: Pick K random points as initial cluster centers

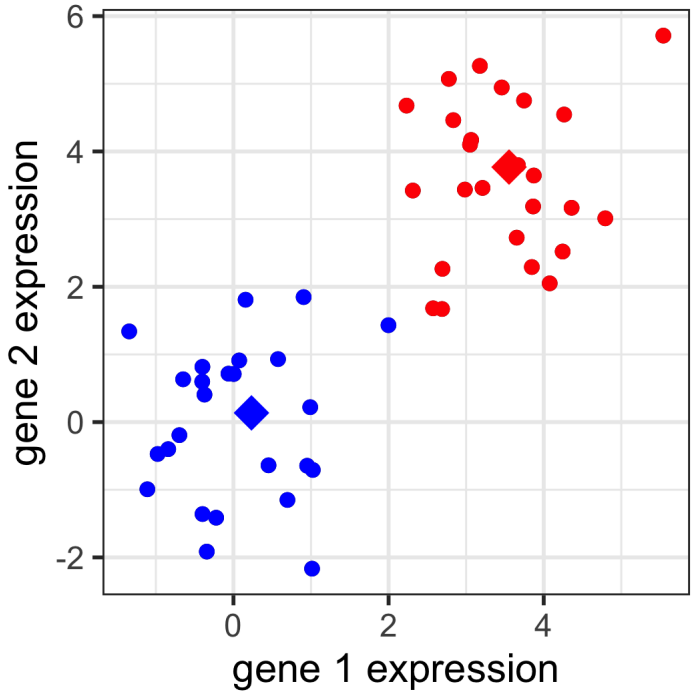
1. Measure distance (squared Euclidean) between all points and the cluster centers

2. Assign points to nearest cluster

3. Update cluster means

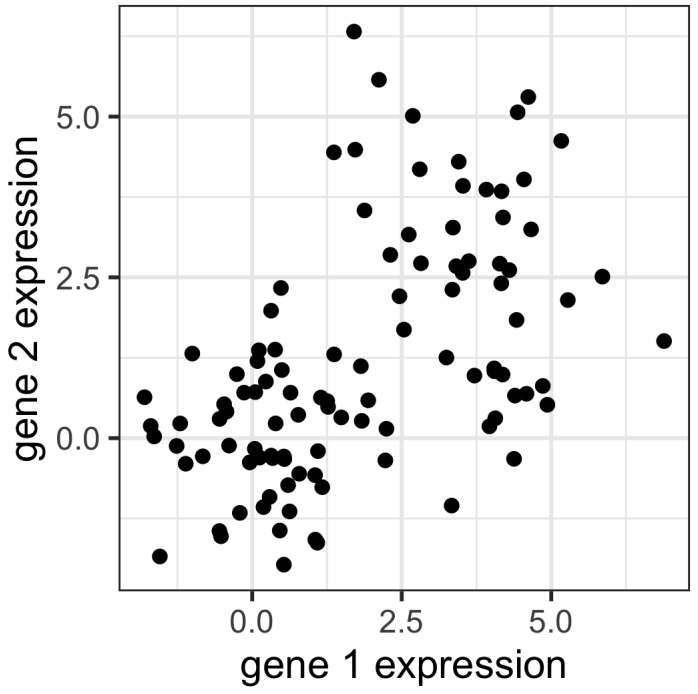
4. Go back to steps 1-2 (measure distance and reassign points)

Example: K-means algorithm



Continue iterating until no change in point assignment

Toy examples are easy



How to decide howmany clusters are there?

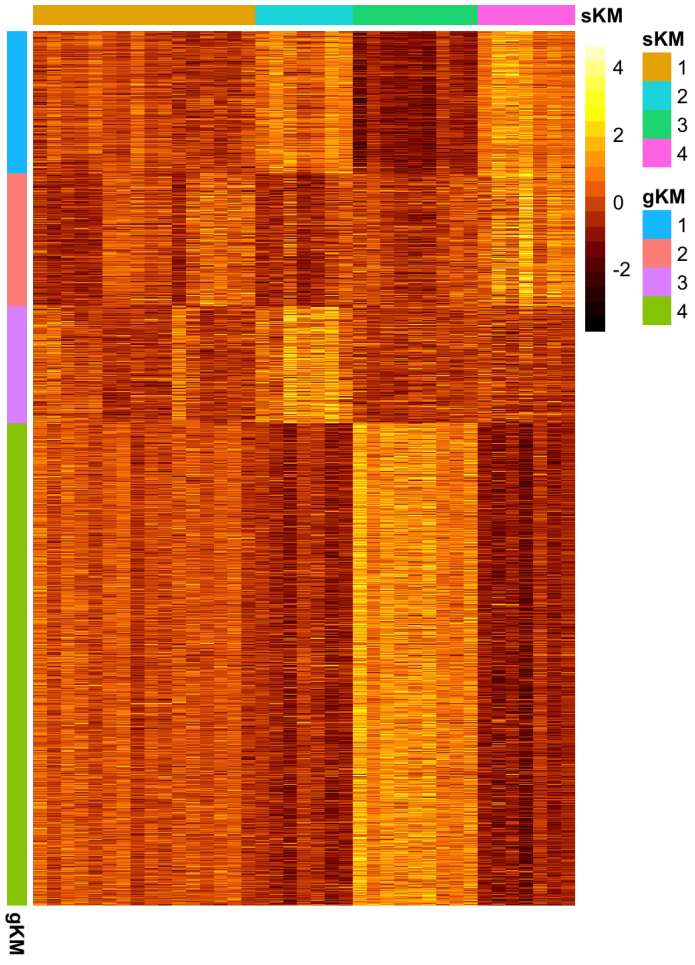
- Your own judgement and prior knowledge
- Reliance on a *model selection* procedure

K-means in practice

Application of K-means to photoreceptor data (2K random genes) with $K = 4$. We specify `nstarts` so that the algorithm is initialized with many different initial clusters.

```
gene.km <- kmeans(x, centers = 4, nstart = 50)
sample.km <- kmeans(t(x), centers = 4, nstart = 50)
str(sample.km)
```

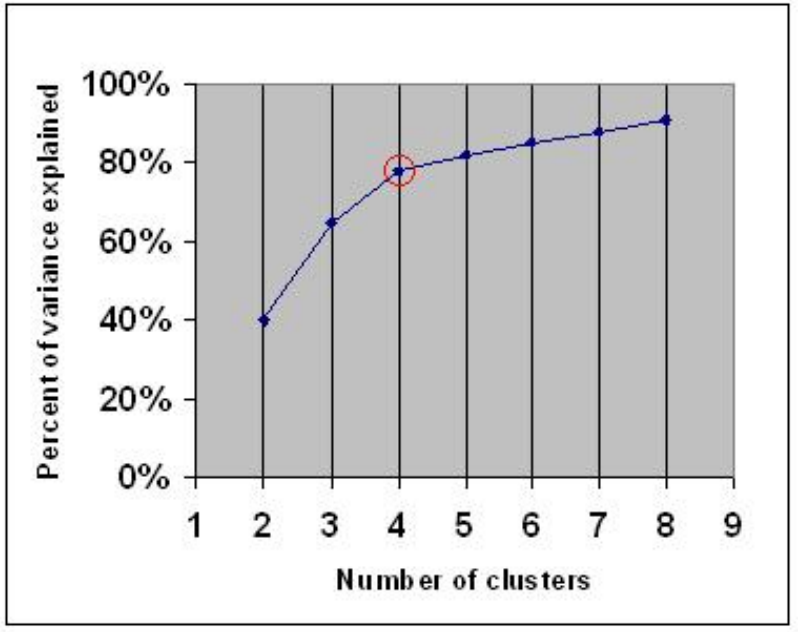
```
## List of 9
## $ cluster : Named int [1:39] 1 4 2 1 4 4 3 1 2 1 ...
## ..- attr(*, "names")= chr [1:39] "GSM92610" "GSM92611"
"GSM92612" "GSM92613" ...
## $ centers : num [1:4, 1:2000] 0.0787 -0.7049 1.1647
-0.9725 0.2186 ...
## ..- attr(*, "dimnames")=List of 2
## .. ..$ : chr [1:4] "1" "2" "3" "4"
## .. ..$ : chr [1:2000] "1433701_at" "1458680_at"
"1421814_at" "1456626_a_at" ...
## $ totss : num 76000
## $ withinss : num [1:4] 16227 7828 9624 7640
## $ tot.withinss: num 41319
## $ betweenss : num 34681
```



Choosing K

Note: maximizing the clustering objective will **not** be informative, as it leads to the solution that each object should be in its own cluster. Therefore, need an algorithm that takes into account the "cost" of adding additional clusters

- Prior knowledge
- "Elbow"/"Knee" method: point of diminishing returns
- Information Criteria (AIC or BIC): likelihood penalized for number of parameters
- Silhouette metric: how similar an object is to its own cluster compared to other clusters
- Gap Statistics: total intracluster variation compared to that expected under the null

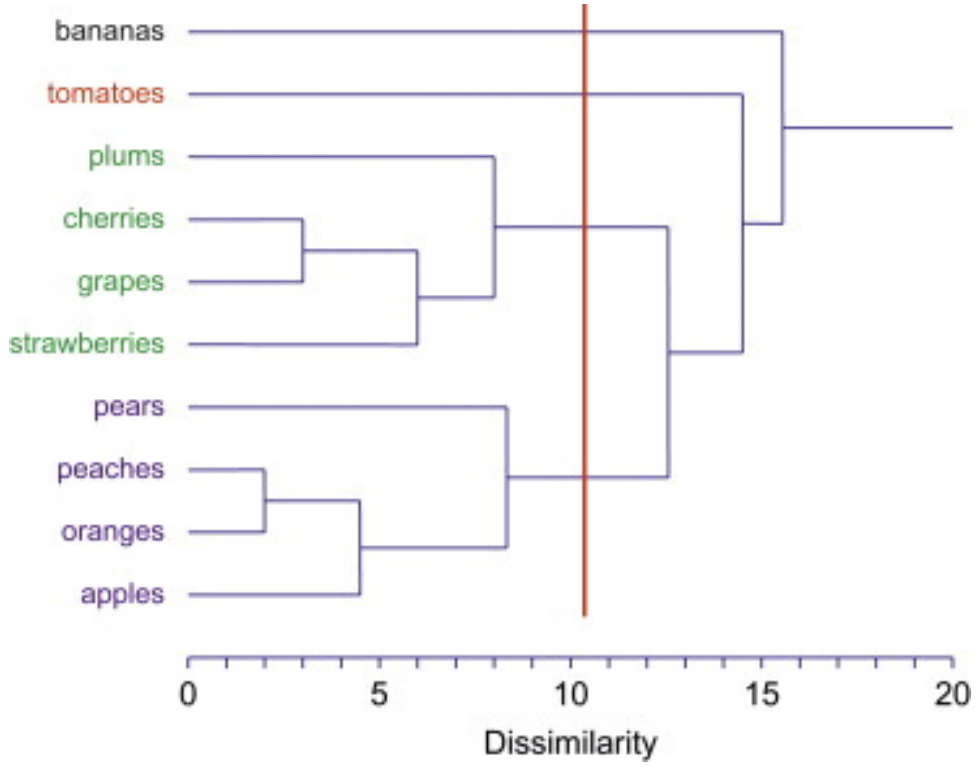


Elbow method

Hierarchical aggolomerative clustering

A clustering approach for revealing **hierarchical** relationships between objects

Agglomerative: each observation starts in its own cluster; going up in hierarchy pairs of clusters are merged



Algorithm: Hierarchical agglomerative clustering

Given n objects with p attributes, and a distance metric:

Initialize: Assign each object to its own cluster and compute pairwise distances between all clusters

Iterate: Repeat steps 1 and 2 until all objects belong to a single cluster

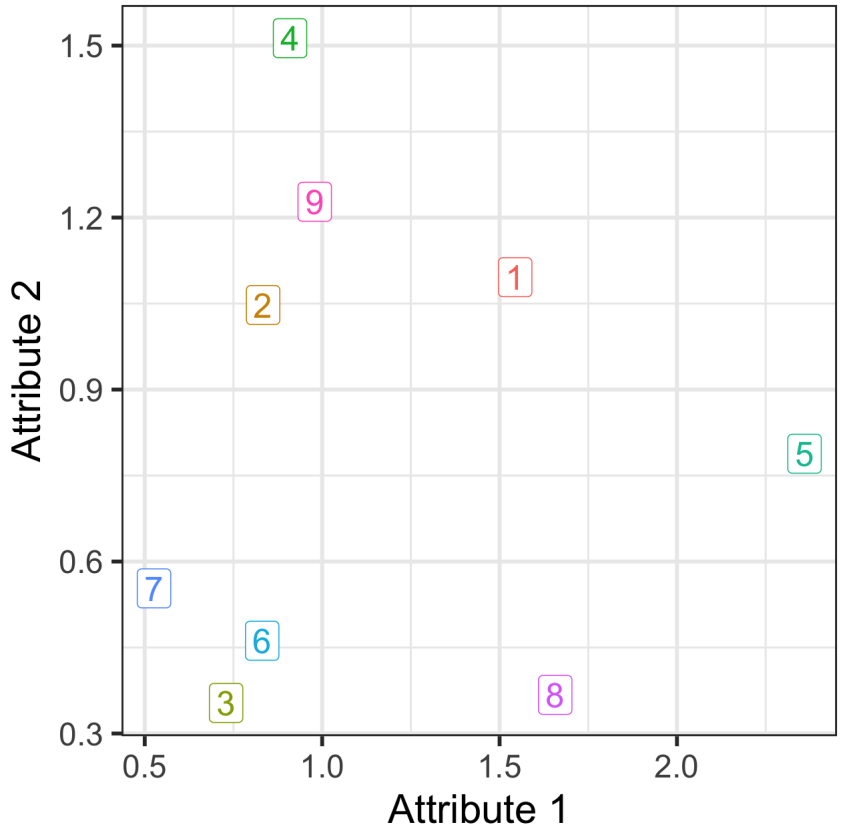
1. Find the "closest" pair of clusters, and **merge** them into a single cluster

2. Compute new distances between clusters

Cluster linkage

- Distance between clusters is computed based on cluster centers; these are common ways to define cluster centers:
    - **Single** linkage: distance between two clusters is the **minimum** distance between any pair of elements
    - **Average** linkage:distance between two clusters is the **average** distance between all pairs of elements
    - **Complete** linkage: distance between two clusters is the **maximum** distance between any pair of elements
- Default in `pheatmap` is complete Linkage using Euclidean distance

Example: Hierarchical agglomerative clustering



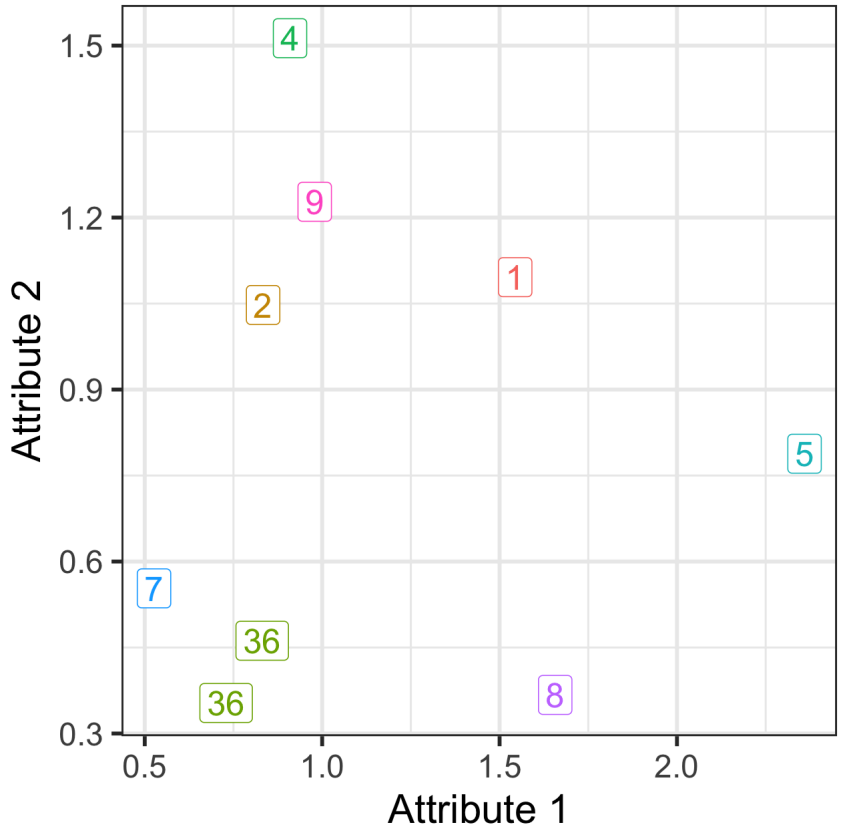

```
round(dist(x),2)
```

```
##      1    2    3    4    5    6    7    8
## 2 0.71
## 3 1.10 0.70
## 4 0.76 0.47 1.17
## 5 0.87 1.55 1.69 1.62
## 6 0.95 0.59 0.15 1.05 1.56
## 7 1.15 0.58 0.29 1.03 1.85 0.32
## 8 0.74 1.07 0.93 1.37 0.82 0.83 1.15
## 9 0.58 0.23 0.91 0.29 1.45 0.78 0.81 1.10
```

Merge cluster 3 and 6

We'll use average linkage

Example: Hierarchical agglomerative clustering

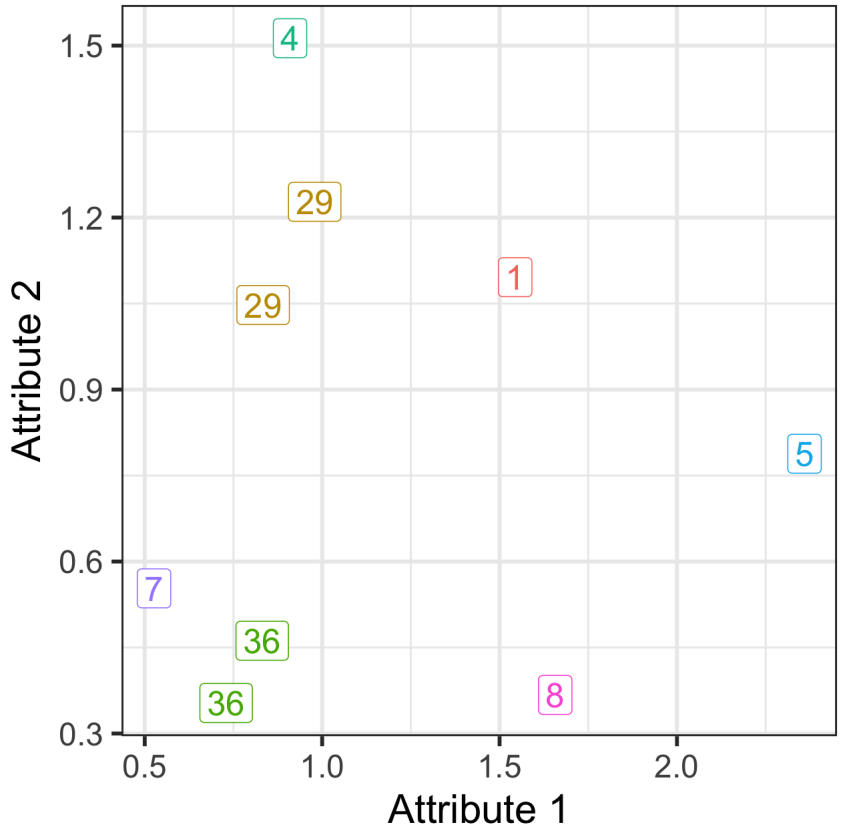



```
round(dist(x),2)
```

```
##       1    2    4    5    7    8    9
## 2  0.71
## 4  0.76 0.47
## 5  0.87 1.55 1.62
## 7  1.15 0.58 1.03 1.85
## 8  0.74 1.07 1.37 0.82 1.15
## 9  0.58 0.23 0.29 1.45 0.81 1.10
## 36 1.03 0.64 1.11 1.62 0.29 0.88 0.84
```

Merge cluster 2 and 9

Example: Hierarchical agglomerative clustering

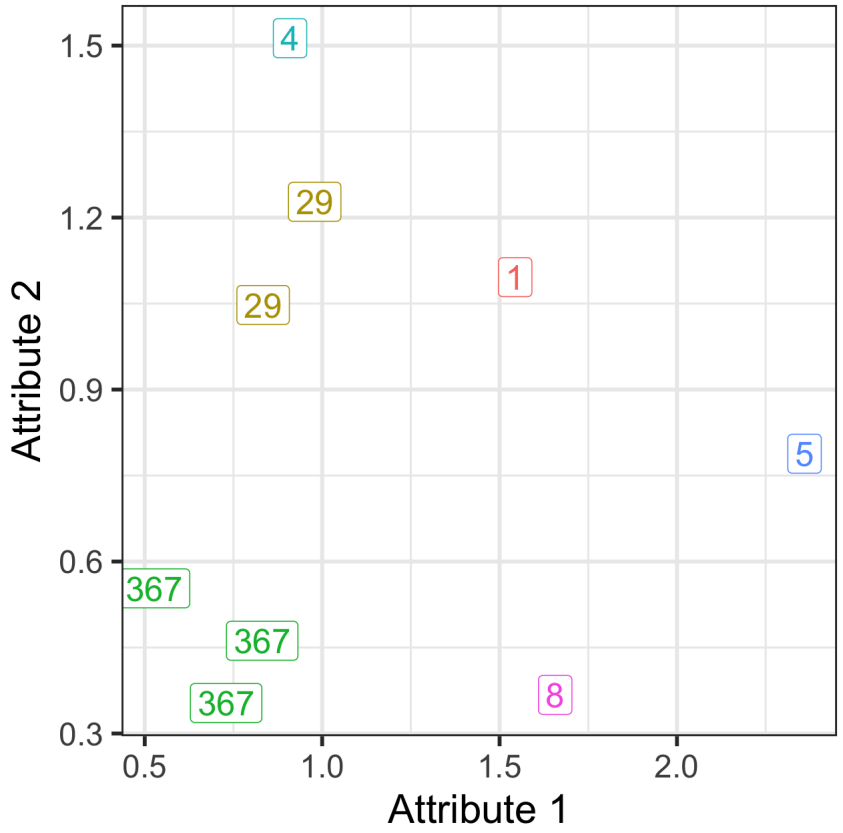



```
round(dist(x),2)
```

```
##       1    4    5    7    8   36
## 4  0.76
## 5  0.87 1.62
## 7  1.15 1.03 1.85
## 8  0.74 1.37 0.82 1.15
## 36 1.03 1.11 1.62 0.29 0.88
## 29 0.64 0.38 1.49 0.70 1.08 0.74
```

Merge cluster 36 and 7

Example: Hierarchical agglomerative clustering



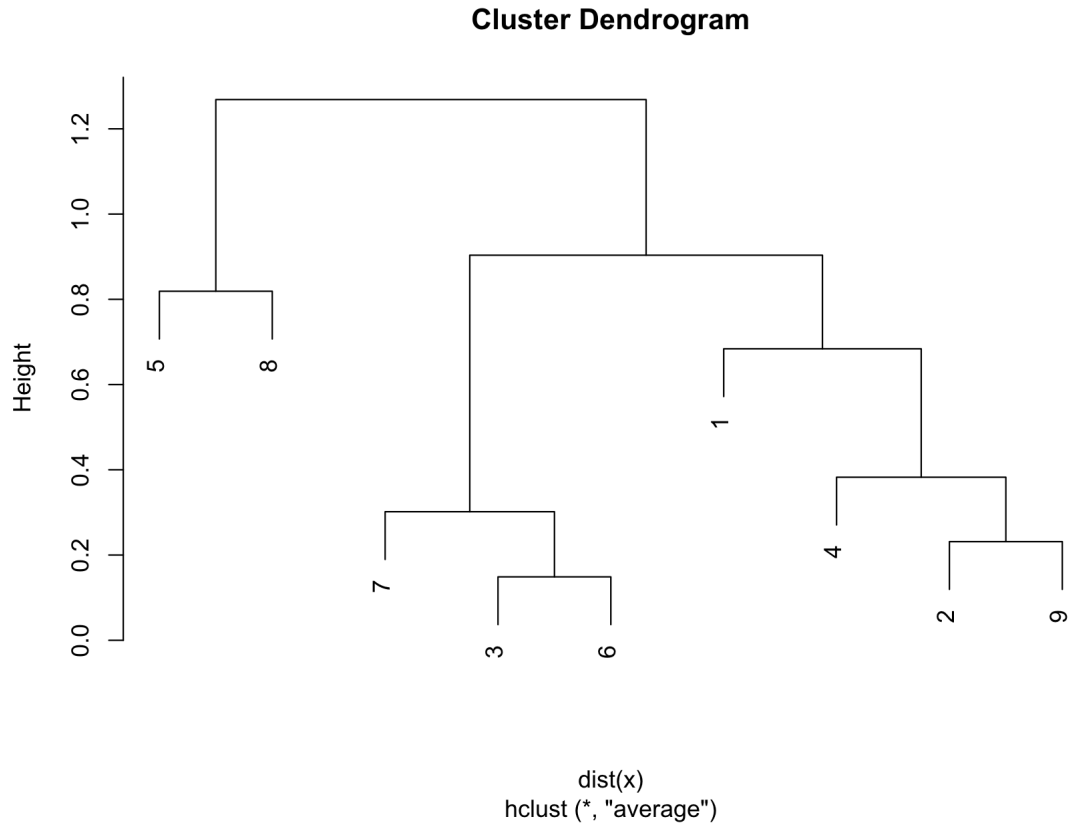

```
round(dist(x),2)
```

```
##         1    4    5    8   29
## 4    0.76
## 5    0.87 1.62
## 8    0.74 1.37 0.82
## 29   0.64 0.38 1.49 1.08
## 367 1.06 1.08 1.70 0.97 0.71
```

Merge cluster 29 and 4...

Dendrogram of example

```
plot(hclust(dist(x), method = "average"))
```



Assessing uncertainty

Clustering **always gives you an answer**, even if there aren't really any underlying clusters

There are many ways to address this

- e.g. `pvclust`, an R package for assessing the uncertainty in hierarchical clustering: uses bootstrap sampling to put uncertainty estimates on dendrogram partitions
- Explore in Seminar 7

Other types of clustering

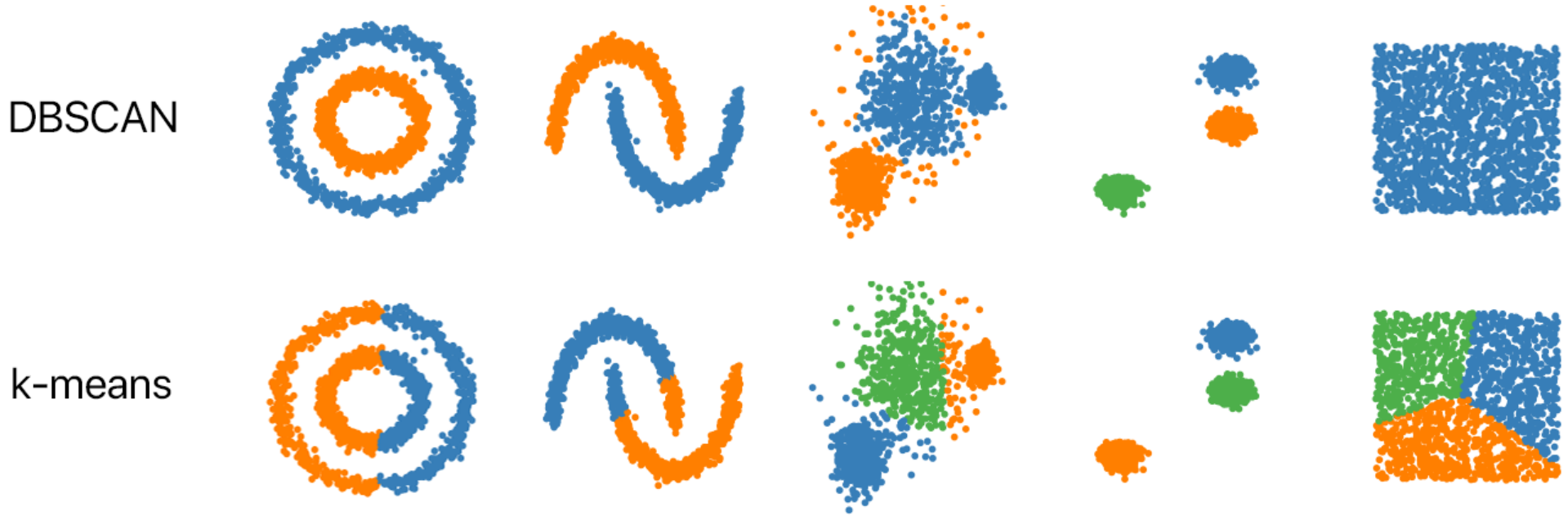


image source

Summary

- Many choices to make when you want to cluster a set of objects:
  - Objective, algorithm, attributes/features, distance metric, number of clusters
- Not possible to say which method is the best - it all depends on data and goal
- Clustering is very powerful, but reckless application leads to misguided conclusions