BioE 594 Computational Functional Genomics

Lecture 14

Genomic data-mining method 2 - clustering

Yang Dai
UIC

Clustering algorithms

- Methods
  - Sequential
  - Partition methods
    - k-mean method
    - Partitioning around medoids (PAM)
    - Self-organizing maps (SOM)
  - Hierarchical
  - Cost function based
  - Others
- Important tasks
  - Estimate the number of clusters
  - Assign each point to a cluster
  - Assess the strength/confidence of cluster assignments for individual observations
  - Assess cluster homogeneity

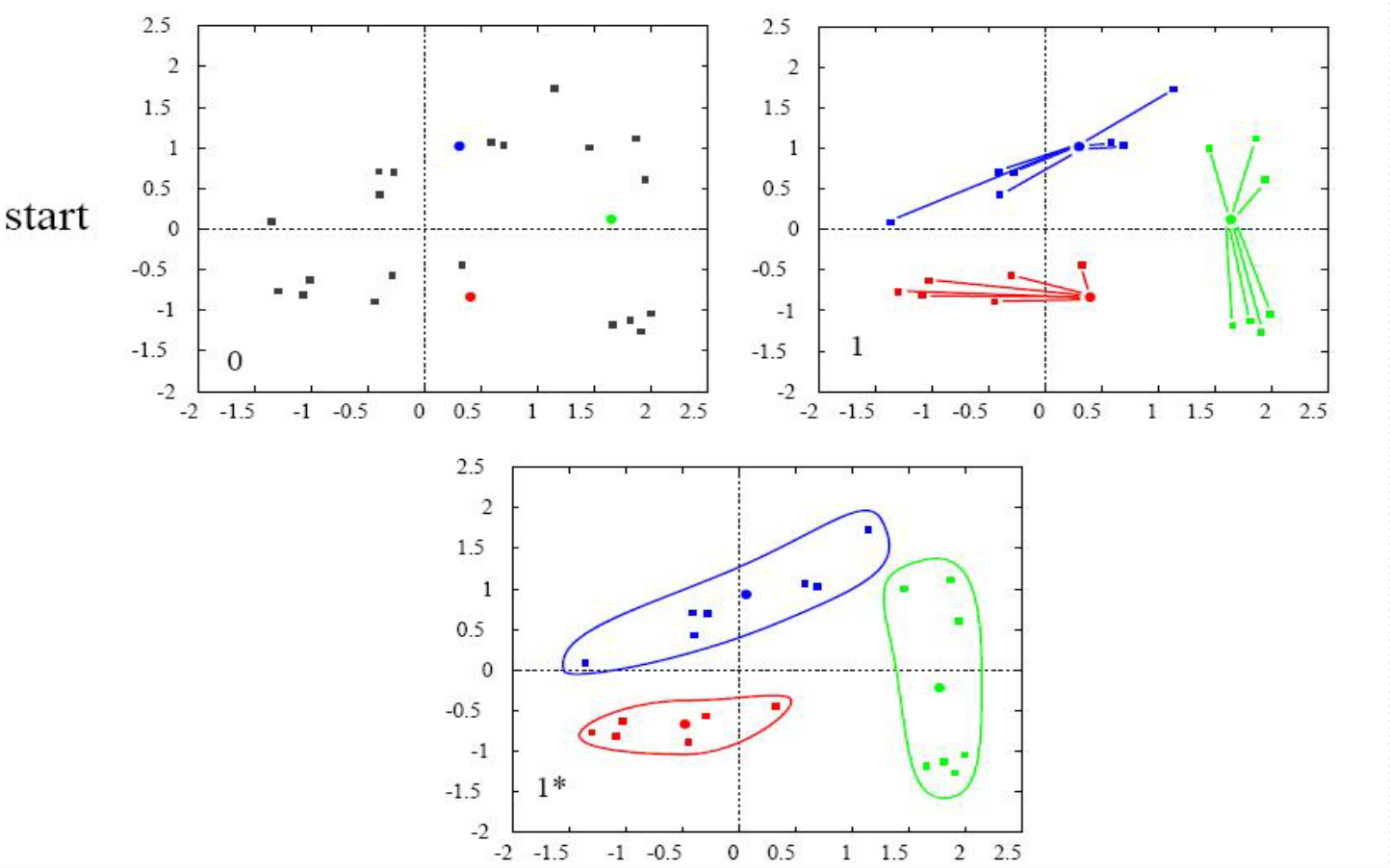
Cluster analysis packages in R

- **mva**
  - hierarchical clustering (**hclust**)
  - k-means (**kmeans**)
- **cluster**:
  - Partitioning Around Medoids (**PAM**)
- **som**
  - Self-Organizing Maps (**SOM**)
- **mclust**
  - model-based clustering (**mclust**)
- There are specialized summary, plot, and print methods for clustering results

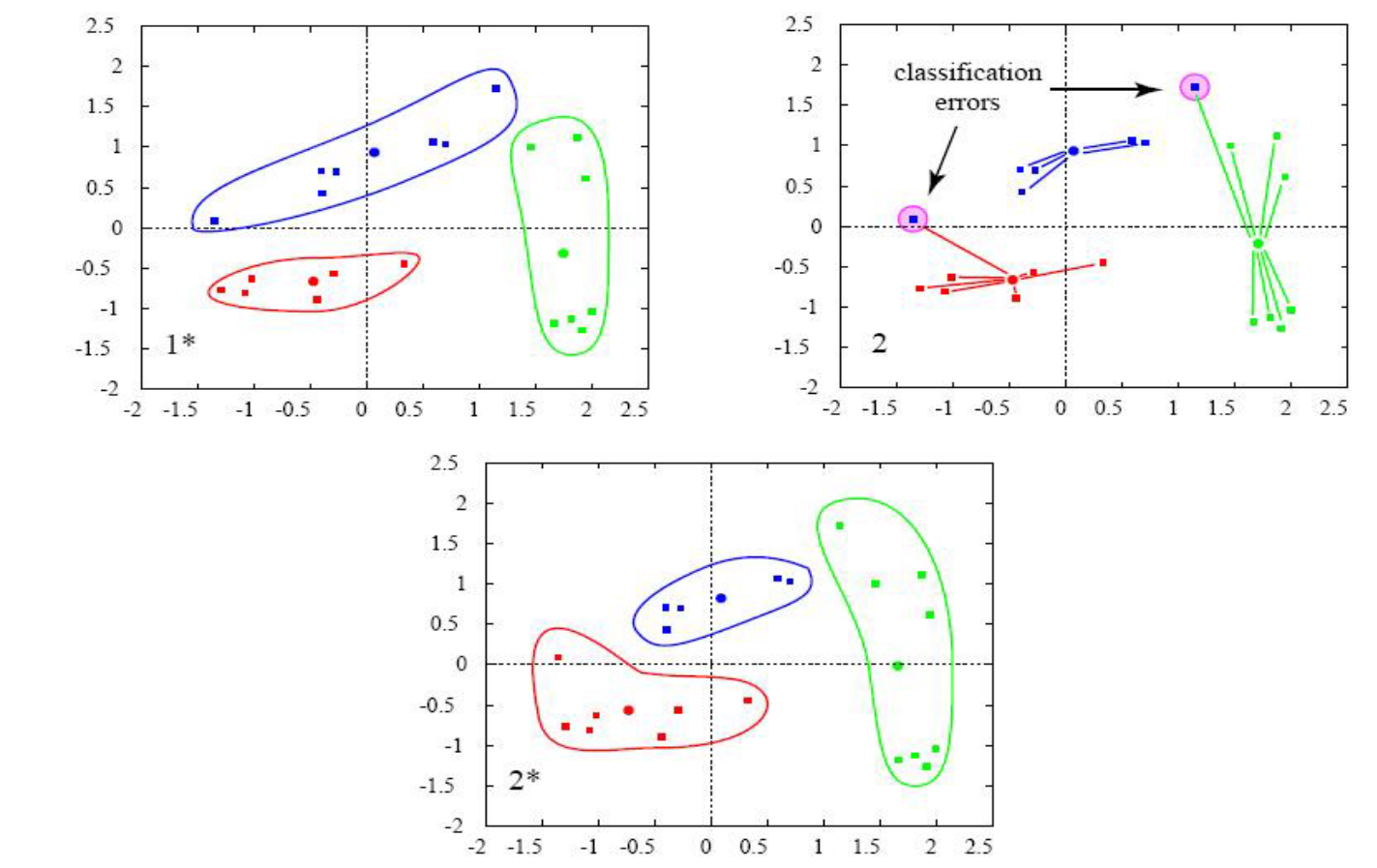
Partition method - K-means clustering

- A partitioning algorithm with a prefixed number *k* of clusters, that tries to minimize the sum of within-cluster-variances
- Algorithm
  - Randomly choose K points as the center of the K clusters
  - Visit each point and calculate the distances from this point to the center of every cluster
  - Assign each point to its closest cluster
  - Update the center of each newly formed cluster
  - Repeat steps (2)-(4) until there is no change to the centers (or reach the maximum cycles)
- Pros: Relatively efficient; Often terminates at a local optimum; The global optimum may be found using techniques such as: deterministic annealing and genetic algorithms
- Cons: Applicable only when mean is defined; Need to specify K in advance; Unable to handle noisy data and outliers; Not suitable to discover clusters with non-convex shapes.

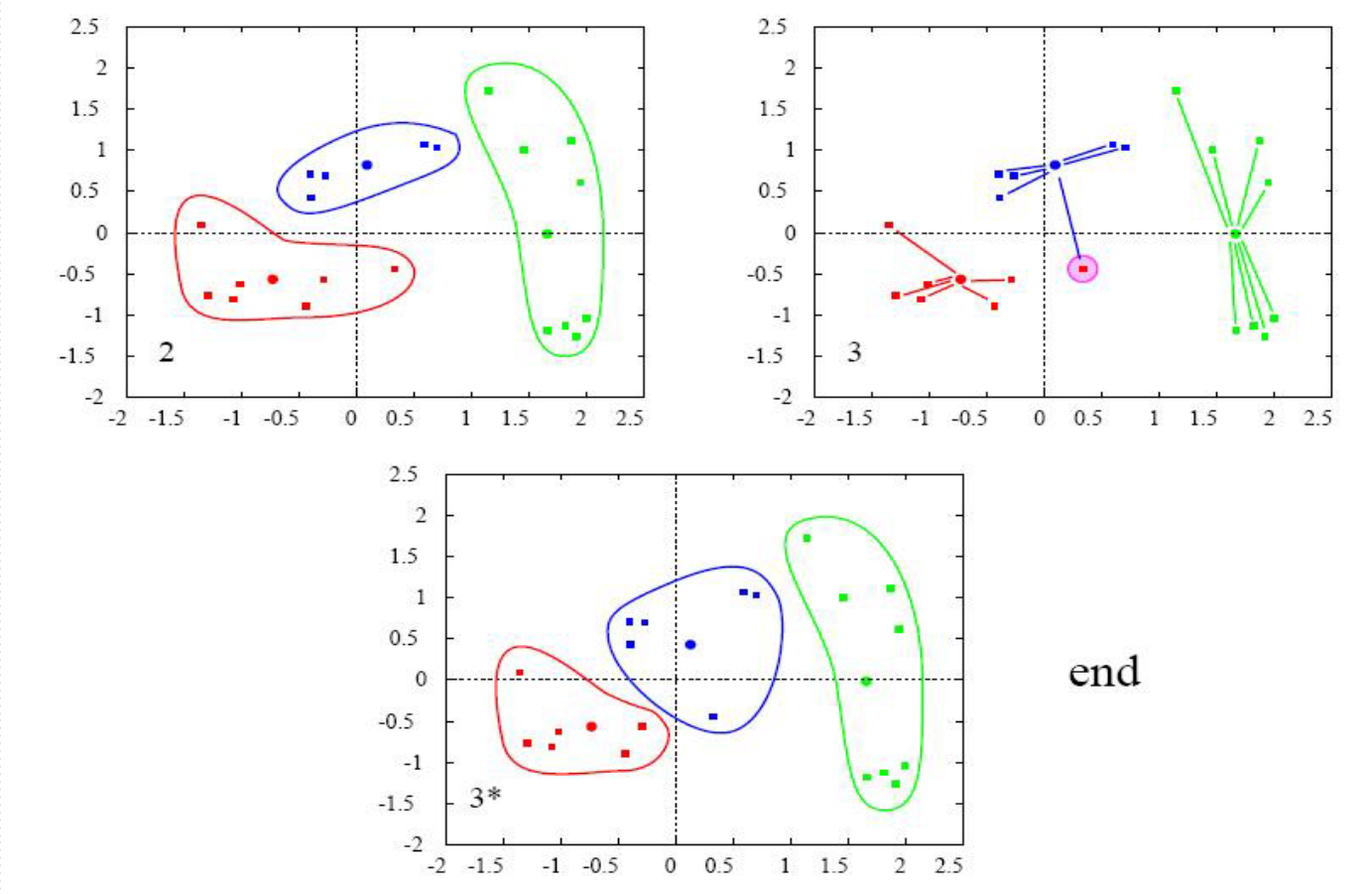
K-means clustering, K=3



K-means clustering, K=3



K-means clustering, K=3



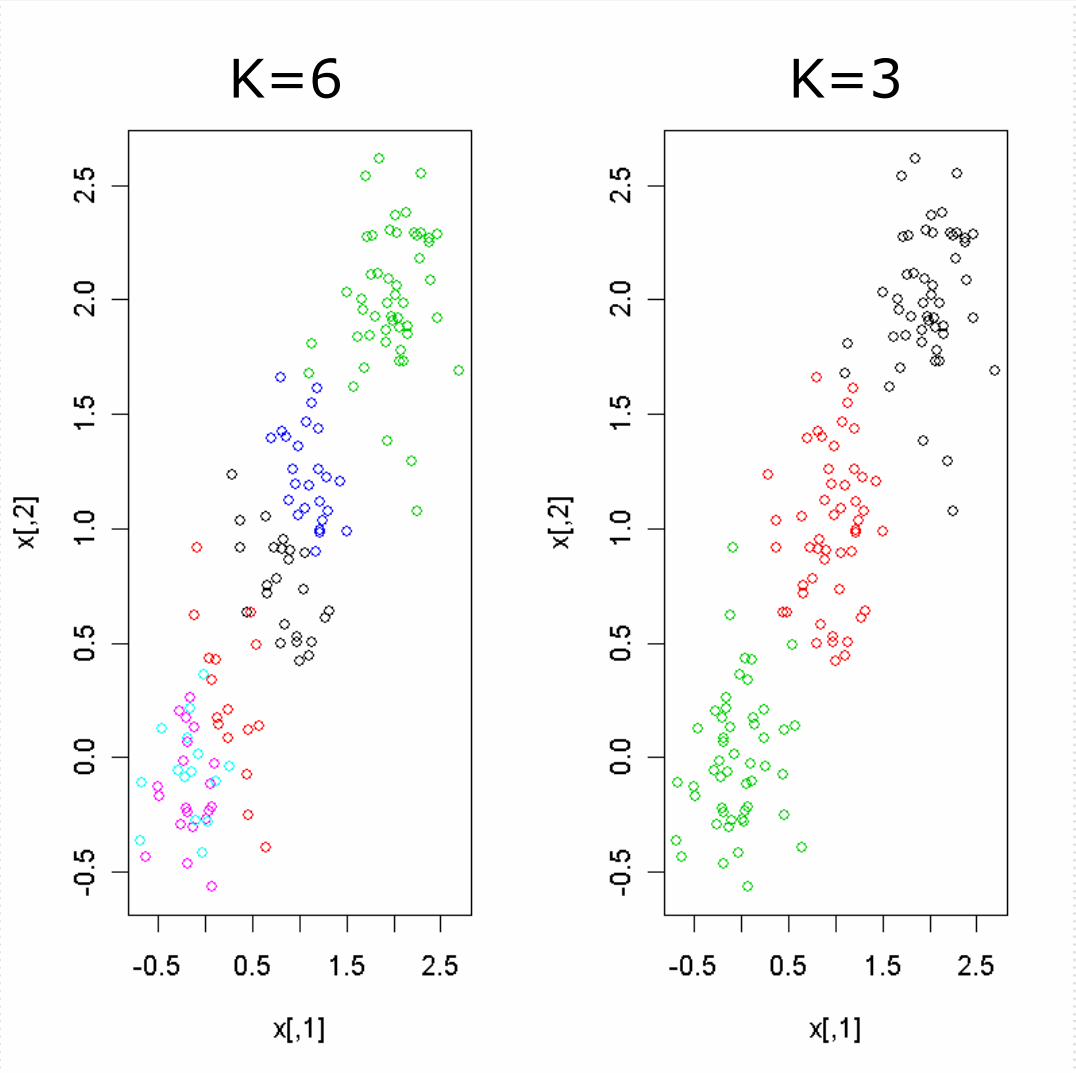
K-means clustering in R

- Use package

  cclust

```
##load cluster package
>library(cclust)
## randomly generate 150 objects
> x <- rbind(matrix(rnorm(150,sd=0.3),ncol=3),
+          matrix(rnorm(150,mean=1,sd=0.3),ncol=3),
+          matrix(rnorm(150,mean=2,sd=0.3),ncol=3))
##initial values for cluster centers are 6
>c1 <- cclust(x,6,20,verbose=TRUE,method="kmeans")
Iteration:  1   Changes:        138
Iteration:  2   Changes:         8
Iteration:  3   Changes:         5
Iteration:  4   Changes:         7
Iteration:  5   Changes:         4
Iteration:  6   Changes:         4
Iteration:  7   Changes:         1
Iteration:  8   Changes:         0
##initial values for cluster centers are 3
>c2 <- cclust(x,3,20,verbose=TRUE,method="kmeans")
Iteration:  1   Changes:        102
Iteration:  2   Changes:         3
Iteration:  3   Changes:         0
##plot the above results
>opar <- par(mfrow = c(1,2))
>plot(x, col=c1$cluster)
>plot(x, col=c2$cluster)
```

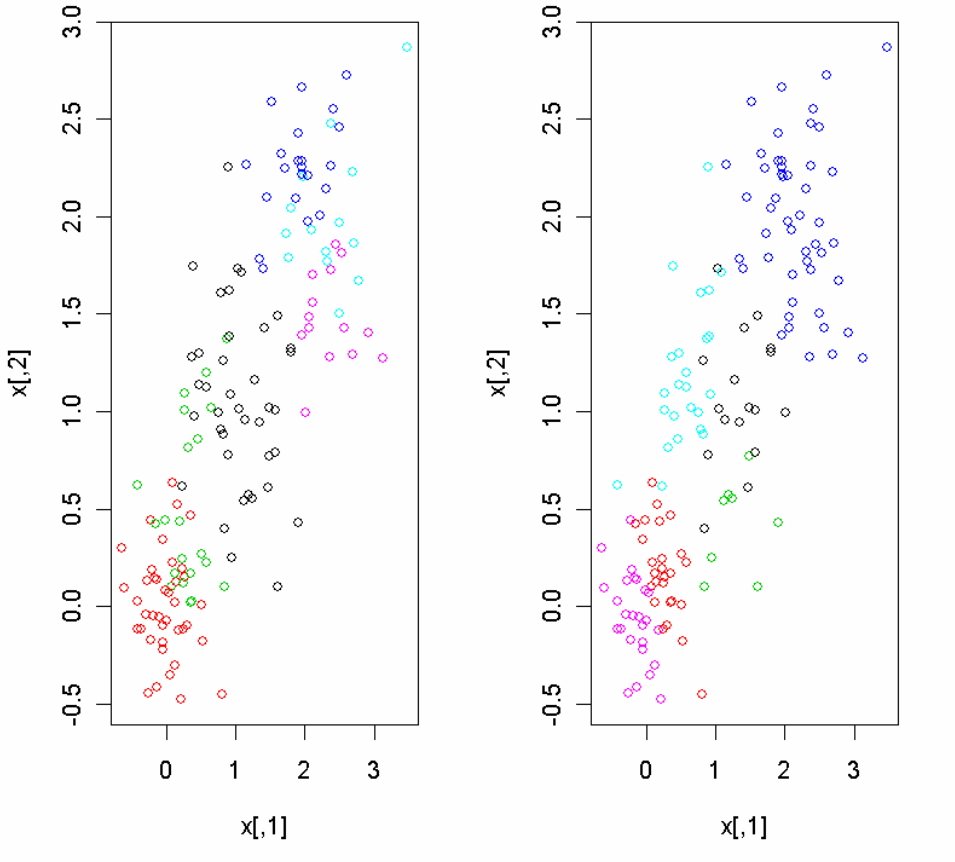
Inappropriate K leads to bad clustering



Different starts result different clusterings

```
> x <- rbind(matrix(rnorm(150,sd=0.3),ncol=3),
     matrix(rnorm(150,mean=1,sd=0.5),ncol=3),
     matrix(rnorm(150,mean=2,sd=0.5),ncol=3))
```

K=6



Partitioning around medoids (PAM)

- PAM is a partitioning method which operates on a distance matrix, e.g., Euclidean distance matrix.
- Algorithm
  - For a prespecified number of clusters *K*, the PAM procedure is based on the search for *K* representative objects, or **medoids**, **M** = ($\mathbf{m}_1, \ldots, \mathbf{m}_k$), among the observations to be clustered.
  - The medoids minimize the sum of the distances of the observations to their closest medoid:

$$M^* = argmin_M \sum_i \min_k d(x_i, m_k)$$

  - After finding a set of *K* medoids, *K* clusters are constructed by assigning each observation to the nearest medoid.
- PAM can be applied to general data types and tends to be more robust than k-means

Partitioning around medoids (PAM)

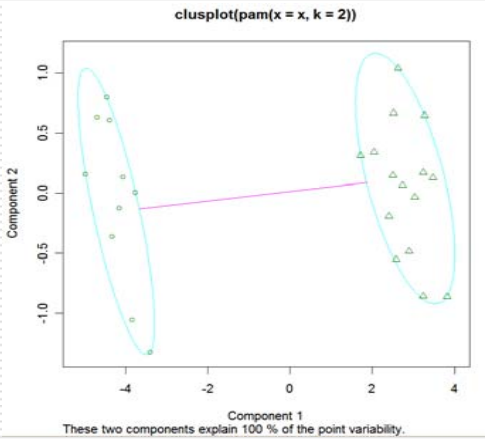
- Silhouette plots are graphical displays, which can be used to
  - select the number of clusters
  - assess how well individual observations are clustered
- Silhouette width of observation *i* is defined as
  - $sil_i = (b_i - a_i) / max(a_i, b_i)$, *where* a_i denotes the average distance between *i* and all other observations in the cluster to which *i* belongs and b_i denotes the minimum average distance of *i* to objects in other clusters
- Objects with large silhouette width are well-clustered, others tend to lie between clusters
- Average silhouette width for clusters *k*: $\overline{sil} = \sum_i sil_i / n$

  i is the observation
- The number of clusters *k* can be estimated by that which gives the largest average silhouette width

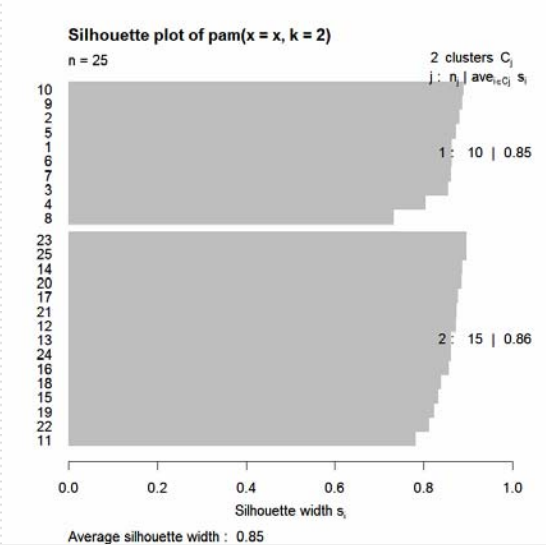
Partitioning around medoids (PAM) – an example in R

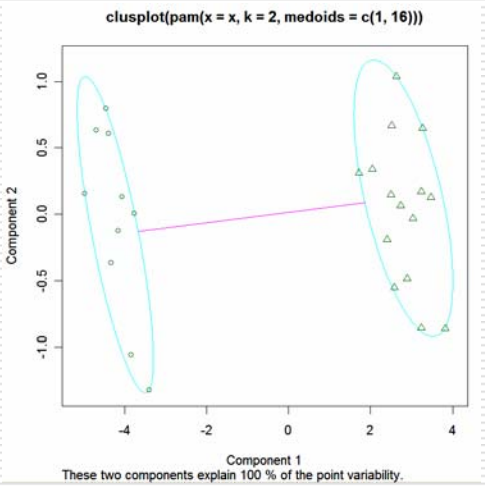
```
## first load cluster package
>library(cluster)
## randomly generate 25 objects, divided into two clusters.
>x <- rbind(cbind(rnorm(10,0,0.5), rnorm(10,0,0.5)),
          cbind(rnorm(15,5,0.5), rnorm(15,5,0.5)))
>pamx <- pam(x, 2)
>pamx
>summary(pamx)
>plot(pamx)
## use obs. 1 & 16 as starting medoids – same result (typically)
>p2m <- pam(x, 2, medoids = c(1,16))
>p2m
>summary(p2m)
>plot(p2m)
## use obs. 1, 13 & 22 as starting medoids – three clusters
>p3m <- pam(x, 3, medoids = c(1,13,22))
>p3m
>summary(p3m)
>plot(p3m)
```

Partitioning around medoids (PAM) – an example in R

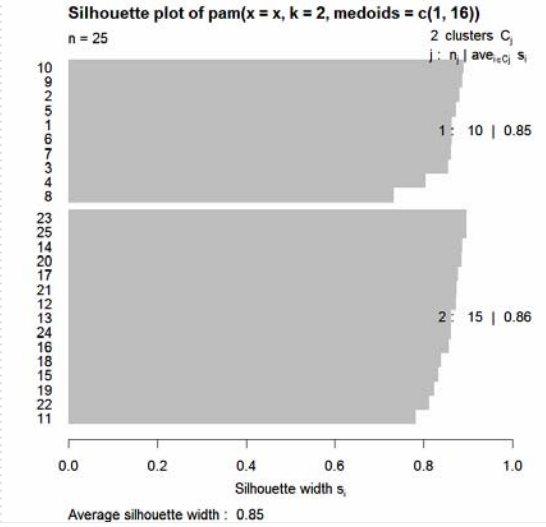


Use randomly selected observations as starting medoids



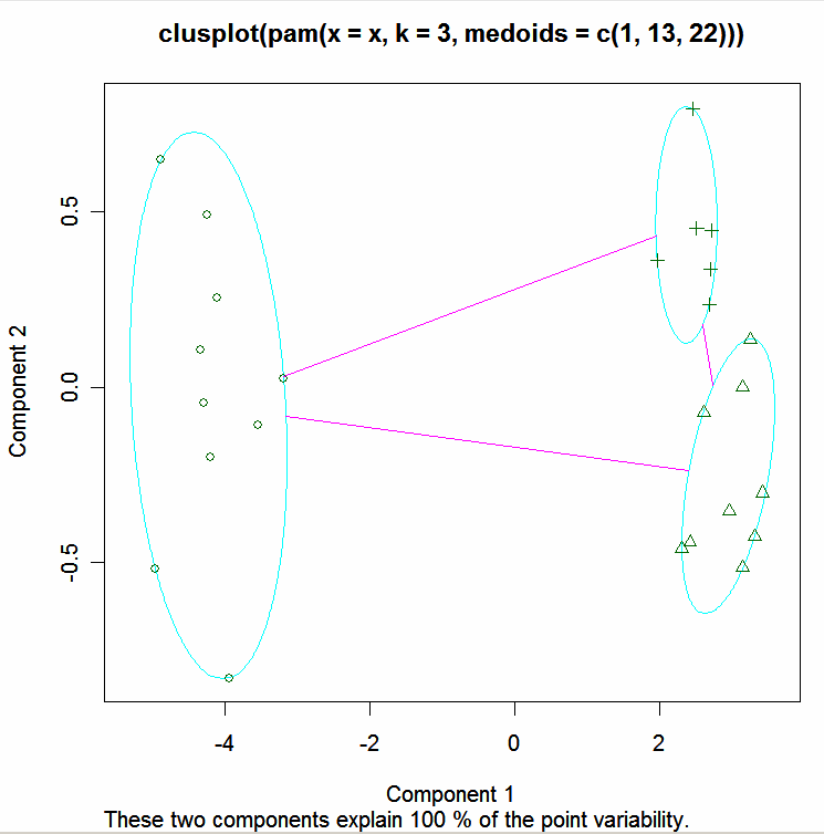


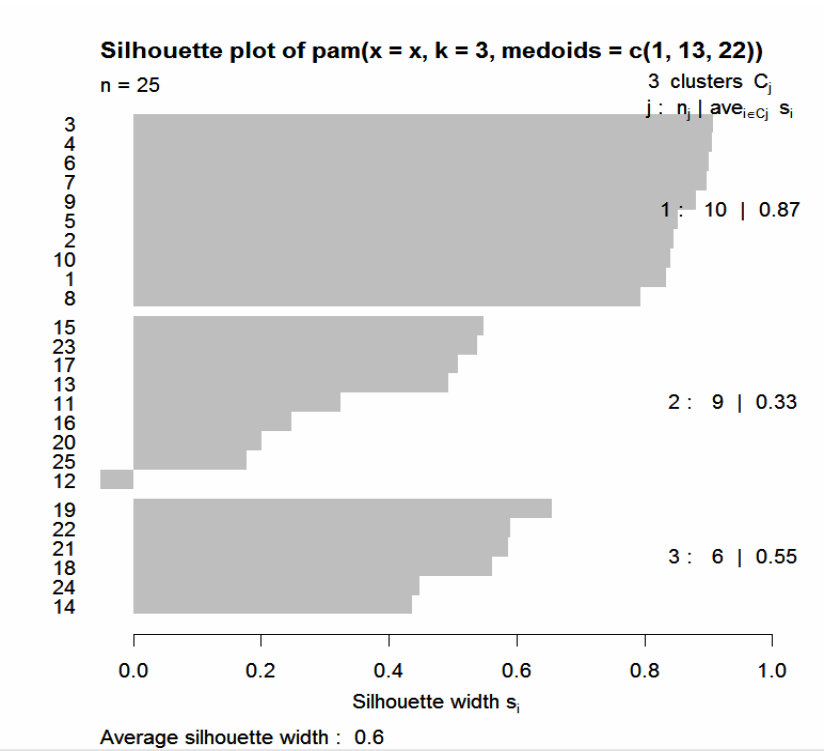
Use observation 1 and 16 as starting medoids



Partitioning around medoids (PAM) – an example in R

Use observation 1, 13, 22 as starting medoids





Hierarchical methods

- Hierarchical clustering methods produce a **tree** or **dendrogram**
- They avoid specifying how many clusters are appropriate by providing a partition for each *K* obtained from cutting the tree at some level.
- The tree can be built in two distinct ways
  - bottom-up: **agglomerative** clustering
  - top-down: **divisive** clustering

Agglomerative methods

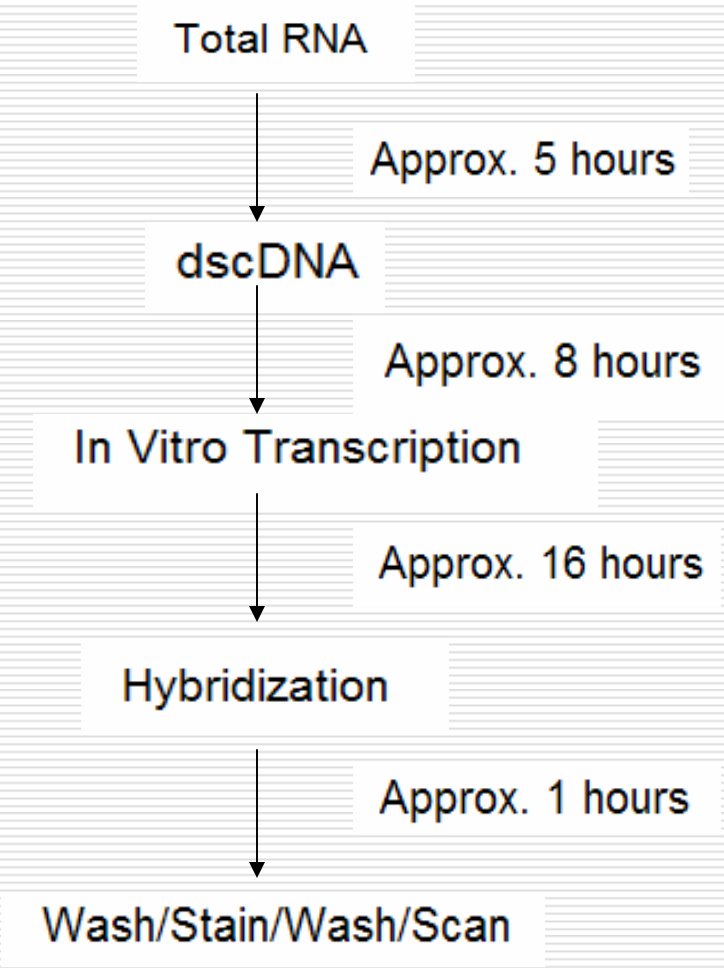
- Algorithm
  - Start with N clusters
  - At each step, merge the two closest clusters using a measure of between-cluster distance which reflects the shape of the clusters. Until every point is in one cluster.
- Between cluster distances (between subsets)
  - Single linkage — minimum distance
  - Complete linkage — maximum distance
  - Average linkage — average distance
  - Centroid linkage — the distance between their centroids
  - Ward — weight squared Euclidean distance between means

Agglomerative methods in R – an example of microarray data clustering

- Purpose: Discriminate between samples from ALL and AML patients based on differentially expressed genes
- Data
  - 27 samples from patients with acute lymphoblastic leukemia (ALL) and 11 from patients with acute myeloid leukemia (AML).
  - The samples were assayed using Affymetrix Hgu6800 chips and data on the expression of 7129 genes (affymetrix probes) are available
  - *t*-test filter is used to select genes that are good differentiators between two groups. 609 out of 7129 gene expressions are selected as features for clustering

Agglomerative methods in R – an example of microarray data clustering

Experimental protocol

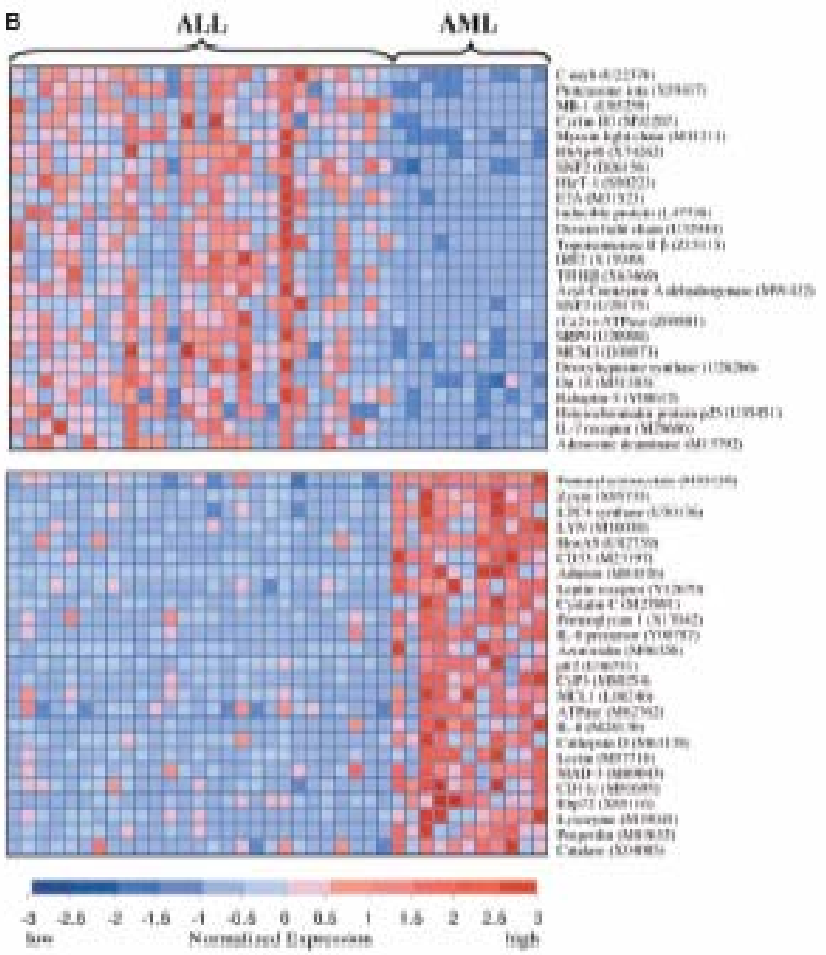


Screen shot of dataset in Excel file

Gene Desc	Gene Accessio	1	call	2	call	3	call	4	ca
AFFX-Biot	AFFX-BioB-5_	-214	A	-139	A	-76	A	-135	A
AFFX-Biot	AFFX-BioB-M_	-153	A	-73	A	-49	A	-114	A
AFFX-Biot	AFFX-BioB-3_	-58	A	-1	A	-307	A	265	A
AFFX-Bio(	AFFX-BioC-5_	88	A	283	A	309	A	12	A
AFFX-Bio(	AFFX-BioC-3_	-295	A	-264	A	-376	A	-419	A
AFFX-Biot	AFFX-BioDn-5	-558	A	-400	A	-650	A	-585	A
AFFX-Biot	AFFX-BioDn-3	199	A	-330	A	33	A	158	A
AFFX-Cre	AFFX-CreX-5_	-176	A	-168	A	-367	A	-253	A
AFFX-Cre	AFFX-CreX-3_	252	A	101	A	206	A	49	A
AFFX-Biot	AFFX-BioB-5_	206	A	74	A	-215	A	31	A
AFFX-Biot	AFFX-BioB-M_	-41	A	19	A	19	A	363	A
AFFX-Biot	AFFX-BioB-3_	-831	A	-743	A	-1135	A	-934	A
AFFX-Bio(	AFFX-BioC-5_	-653	A	-239	A	-962	A	-577	A
AFFX-Bio(	AFFX-BioC-3_	-462	A	-83	A	-232	A	-214	A
AFFX-Biot	AFFX-BioDn-5	75	A	182	A	208	A	142	A
AFFX-Biot	AFFX-BioDn-3	381	A	164	A	432	A	271	A
AFFX-Cre	AFFX-CreX-5_	-118	A	-141	A	84	A	-107	A
AFFX-Cre	AFFX-CreX-3_	565	A	423	A	501	A	101	A

Agglomerative methods in R – an example of microarray data clustering

- Genes distinguishing ALL from AML. The 50 genes most highly correlated with the ALL-AML class distinction are shown. Each row corresponds to a gene, with the columns corresponding to expression levels in different samples. Expression levels for each gene are normalized across the samples such that the mean is 0 and the SD is 1. Expression levels greater than the mean are shaded in red, and those below the mean are shaded in blue.



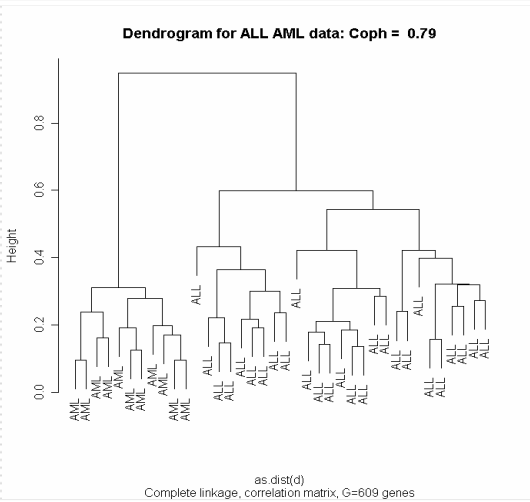
- Reference: Golub *et al*. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science. 1999 Oct 15;286(5439):531-7. (http://www-genome.wi.mit.edu/mpr/data_set_ALL_AML.html)

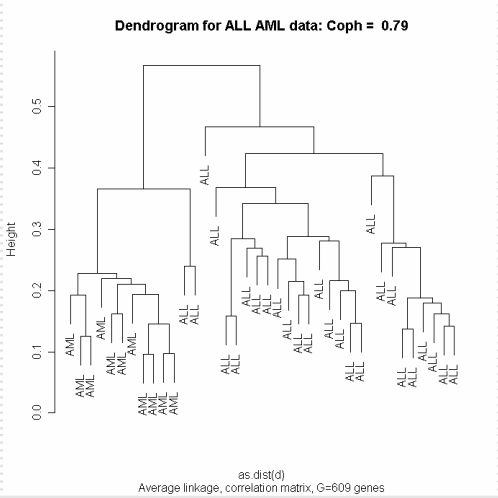
R code for agglomerative clustering analysis

```
> library(Biobase)
> library(golubEsets)
> library(genefilter)
> library(cluster)
> library(stats)
>  X <- exprs(golubTrain)
>  X[X < 100] <- 100
>  X[X > 16000] <- 16000
> mmfilt <- function(r = 5, d = 500, na.rm = TRUE) {
+    function(x) {
+       minval <- min(x, na.rm = na.rm)
+       maxval <- max(x, na.rm = na.rm)
+       (maxval/minval > r) && (maxval - minval > d)
+    }
+ }
> mmfun <- mmfilt()
> ffun <- filterfun(mmfun)
> sub <- genefilter(X, ffun)
> sum(sub)
> X <- X[sub, ]
> X <- log2(X)
> golubTrainSub <- golubTrain[sub, ]
> golubTrainSub@exprs <- X
> gtt <- ttest(golubTrainSub$ALL, p = 0.01)
> gf1 <- filterfun(gtt)
> whT <- genefilter(golubTrainSub, gf1)
> gTrT <- golubTrainSub[whT, ]
> gTrT
> r <- cor(exprs(gTrT))
> dimnames(r) <- list(gTrT$ALL, gTrT$ALL)
> d <- 1 - r
> hc1 <- hclust(as.dist(d), method = "complete")
> coph1 <- cor(cophenetic(hc1), as.dist(d))
> plot(hc1, main = paste("Dendrogram for ALL AML data: Coph = ",round(coph1, 2)), sub = "Complete linkage, correlation matrix, G=609 genes")
> hc2 <- hclust(as.dist(d), method = "average")
> coph2 <- cor(cophenetic(hc2), as.dist(d))
> plot(hc2, main = paste("Dendrogram for ALL AML data: Coph = ",round(coph2, 2)), sub = "Average linkage, correlation matrix, G=609 genes")
> hc3 <- hclust(as.dist(d), method = "single")
> coph3 <- cor(cophenetic(hc3), as.dist(d))
> plot(hc3, main = paste("Dendrogram for ALL AML data: Coph = ",round(coph3, 2)), sub = "Single linkage, correlation matrix, G=609 genes")
> hc4 <- hclust(as.dist(d), method = "single")
> coph4 <- cor(cophenetic(hc4), as.dist(d))
> plot(hc4, main = paste("Dendrogram for ALL AML data: Coph = ",round(coph4, 2)), sub = “Centroid linkage, correlation matrix, G=609 genes")
```

Dendrogram for ALL AML data (different linkages)

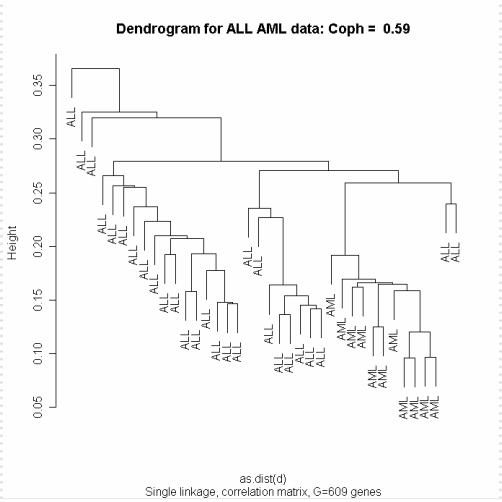
Complete

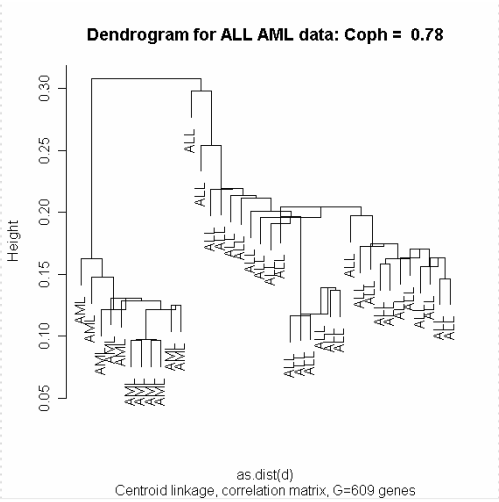




Average

Single





Centroid

Comparison of hierarchical and partitioning clustering

- Hierarchical
  - Advantages: Fast computation, at least for agglomerative clustering
  - Disadvantages: Rigid, cannot correct later for erroneous decisions made earlier
- Partitioning
  - Advantages: Provide clusters that (approximately) satisfy an optimality criterion
  - Disadvantages: Need initial *k*, long computation time

Recommendations from Jörg Rahnenführer

- Interest in specific genes:
  - If you search for genes that are co-regulated with a specific gene of you choice, **DO SO!** Don't use clustering, but generate a list of genes close to your gene with respect to some distance.
- Clustering after feature selection?
  - **NO!** Do not first select genes based on the outcome of some covariable (*e.g.* tumor type) for your clustering. You will ALWAYS find differences w.r.t. this covariable, since this is how you selected the genes!
- Number of clusters
  - No general rule how to select the 'correct' number of clusters. *Ad hoc* approach is to **try different numbers** and choose cutoff, for which performance of the clustering algorithm breaks down. The quality of a clustering result depends on the concept of a cluster