

Machine Learning in R

Lecture 8 – Unsupervised Learning: Clustering and
Data Reduction Algorithms

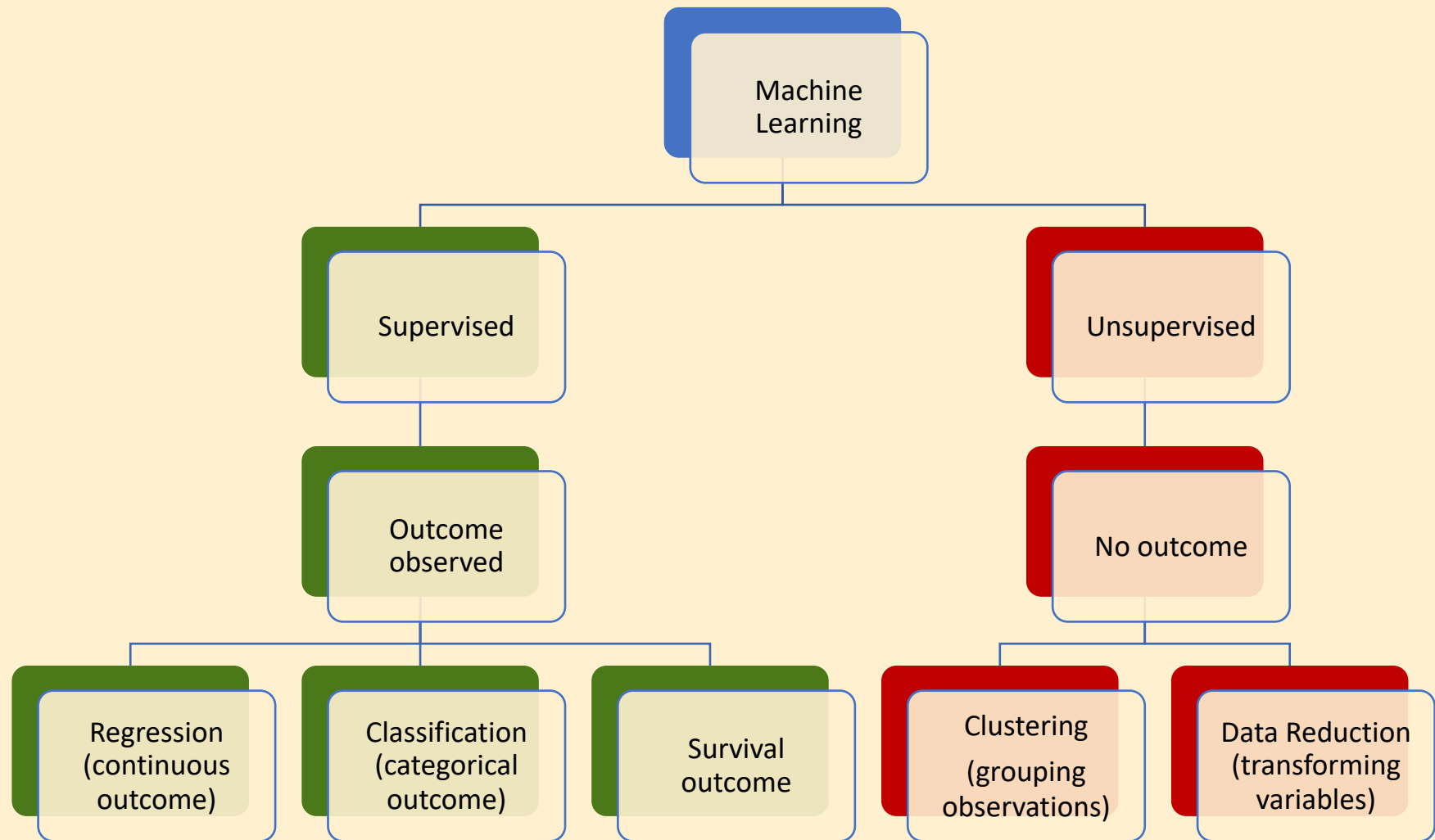
John Kornak – Epidemiology and Biostatistics

john.kornak@ucsf.edu

In this lecture:

- Unsupervised Learning
 - Cluster analysis
 - K-means clustering
 - Hierarchical clustering
 - Dimension reduction
 - Principal components analysis

Supervised vs. unsupervised



Unsupervised learning

- Most of this course focuses on *supervised learning* methods such as regression and classification
- In the *supervised learning* setting we observe both a set of features $X_1, X_2, \dots, X_p$ for each object, as well as a response or outcome variable Y . The goal is then to predict Y using $X_1, X_2, \dots, X_p$
- Now look at *unsupervised learning*: No responses, just a set of variables/predictors/features $X_1, X_2, \dots, X_p$ measured on a set of observations (no Y)
- Objective can be:
 - Clustering: find groups for which predictors follow a more similar pattern (correlated) than between groups
 - Dimension reduction: find ways to reduce the information in a large set of predictors to something more manageable – ideally, such that you get new features that are interpretable

Unsupervised learning

- The goal is to discover “interesting” things about the measurements: is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations?
- We will look at 2 approaches:
 - *clustering*, a broad class of methods for discovering unknown subgroups in data
 - *principal components analysis*, a tool used for data visualization or data pre-processing before supervised techniques are applied

The Challenge of Unsupervised Learning

- Unsupervised learning is more subjective than supervised learning, as there is no simple goal for the analysis, such as prediction of a response
- A common scenario in biomedical research where we need unsupervised learning is in finding subgroups of a disease population:
 - subgroups of breast cancer patients grouped by their gene expression measurements
 - subgroups of dementia types based on neuropsychological or brain tissue changes

An advantage of unsupervised methods

- It is often easier to obtain *unlabeled data* — from a lab instrument or a computer — than *labeled data*, which can require human intervention
- For example, it can be easy to collect a massive dataset of medical images, but much harder to get a diagnosis or score for each of them, which would require human intervention

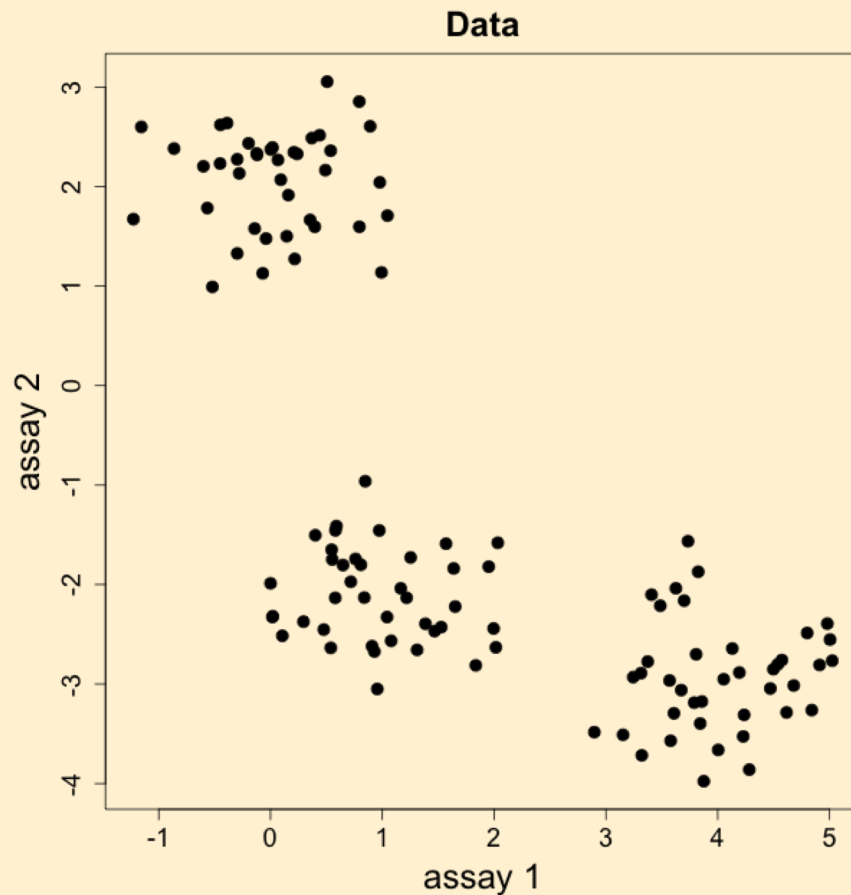
Clustering

- *Clustering* refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set
- We want to find a *partition* of the data into distinct groups so that the observations within each group are more similar to each other than observations in differing groups
- To make this concrete, we must define what it means for two or more observations to be *similar* or *different*
- This *should be* a domain-specific consideration that is made based on knowledge of the data being studied

Clustering to find different subgroups among different types of dementia

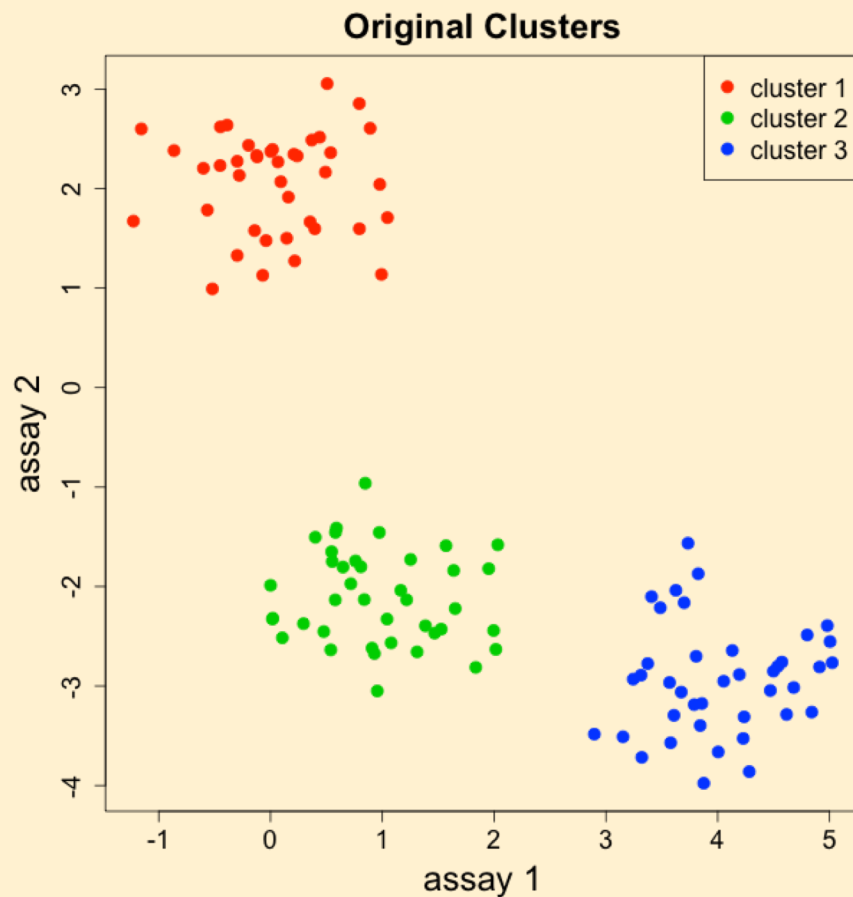
- We have imaging data on different types of dementia
- Within a particular type of dementia (e.g. Alzheimer's disease) can we find distinct subgroups that correspond to distinct patterns of the disease
- The task of finding such subgroups is to perform cluster analysis within Alzheimer's patients

Clustering example



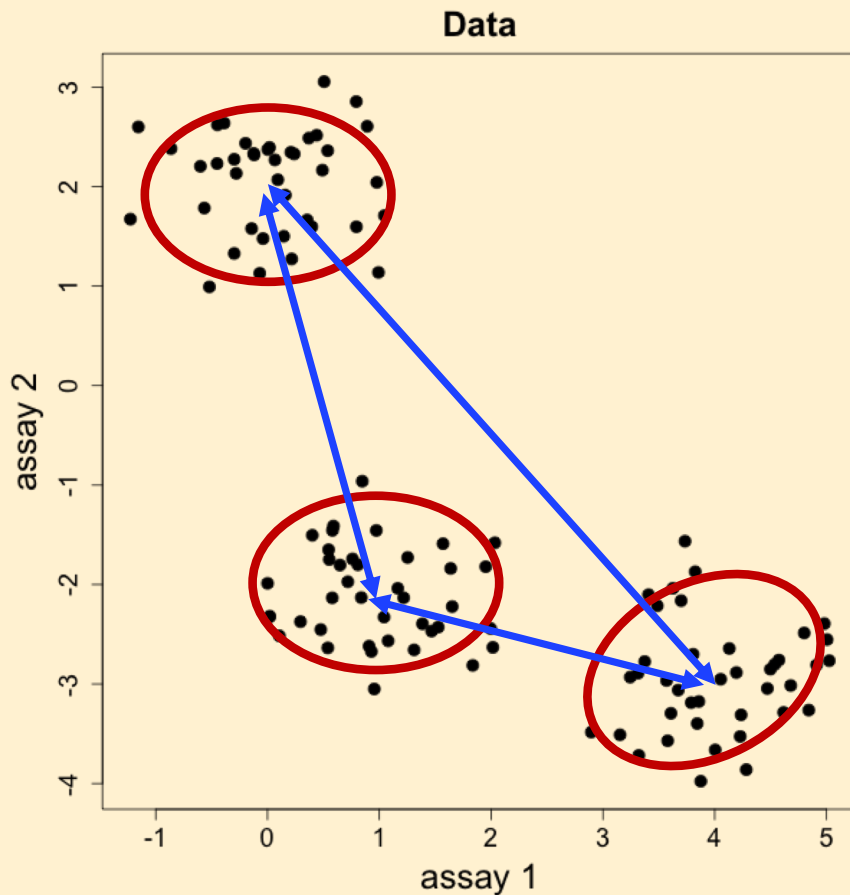
- How many clusters?
- How do we come up with a rule to assign each point to an appropriate cluster?

You guessed it...



- How many clusters?
- How do we come up with a rule to assign each point to an appropriate cluster?

Properties of good cluster separation (or segmentation)



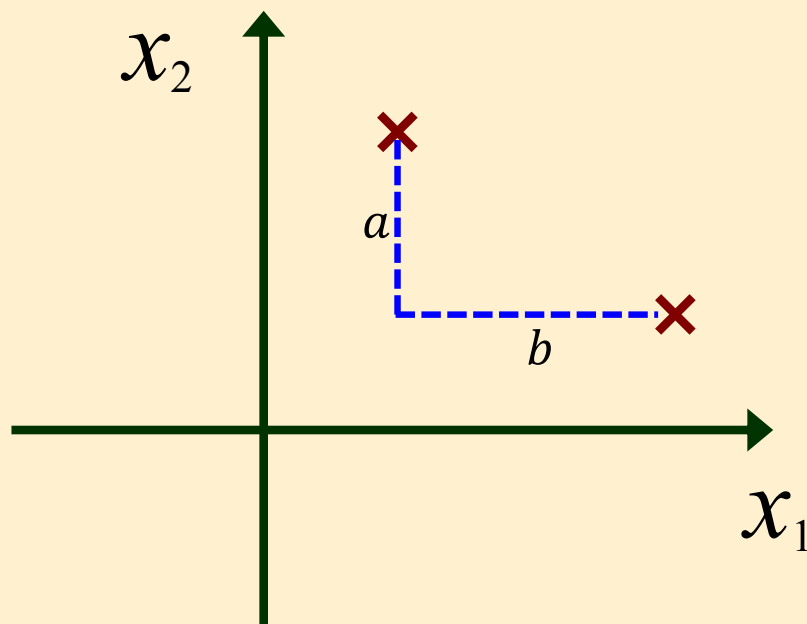
- How many clusters?
- How do we come up with a rule to assign each point to an appropriate cluster?
- Roughly speaking:
maximize distance (*dissimilarity*) between clusters while minimizing area (variance) within clusters (maximize *similarity*) within clusters

Similarity and dissimilarity

- Need to define *similarity* (or *dissimilarity*) and we need to be able to measure it
- Want to *maximize similarity* (or *minimize dissimilarity*) within clusters and *minimize similarity* (*maximize dissimilarity*) between clusters

Measures of similarity/dissimilarity

Most common choice for dissimilarity between 2 instances is the squared difference summed over all attributes (higher values = more dissimilar)



Consider 2 observations with only 2 variables x_1 and x_2 – dissimilarities could be

- Square diff = $a^2 + b^2$
- Absolute diff = $|a| + |b|$
- Can also differentially weight attributes e.g. $ka^2 + mb^2$

Extension to 3D would be e.g. $a^2 + b^2 + c^2$, and so on for higher dimensions

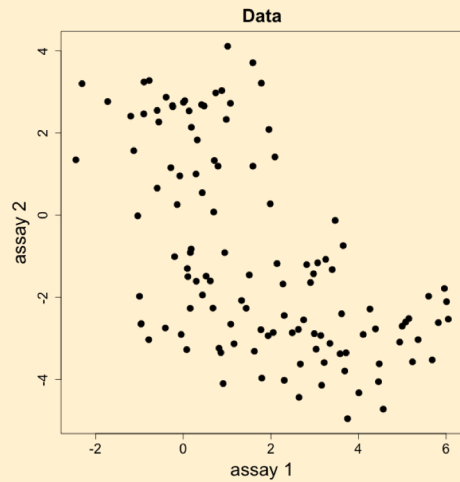
Measures of similarity/dissimilarity

- Generally need to “standardize variables”
- Ordinal variables are treated the same as continuous variables
- Nominal variable differences between groups need to be explicitly given. A common choice is difference 1 if the groups are the same and 0 if they are different. (This then needs to be standardized if any non-categorical variables are also considered.)
- Remember, we want to maximize similarity (or minimize dissimilarity) within clusters and minimize similarity (maximize dissimilarity) between clusters

Two clustering methods

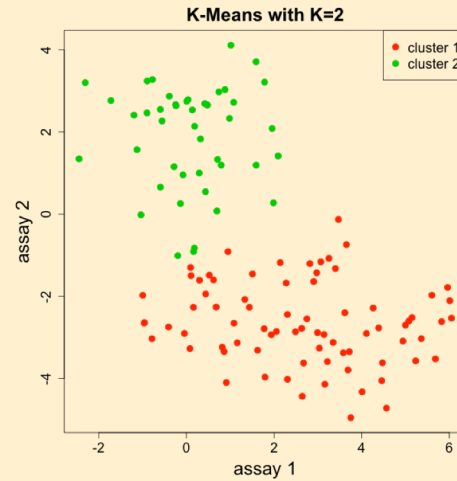
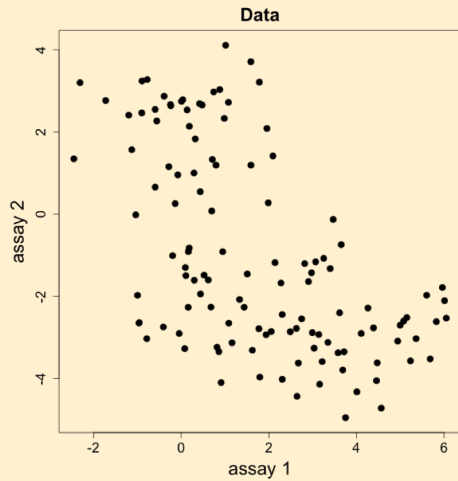
- In *K-means clustering*, we seek to partition the observations into a pre-specified number of clusters
- In *hierarchical clustering*, we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a *dendrogram*, that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to n

K-means example



```
> head(dat)
      assay1 assay2 group
1 -0.89691455 3.2411998    1
2  0.18484918 2.1388584    1
3  1.58784533 3.7106316    1
4 -1.13037567 1.5693590    1
5 -0.08025176 0.9557704    1
6  0.13242028 2.5375795    1
>
```

K-means example



```
> set.seed(4)
> km.out2 <- kmeans(x, centers=2)
> km.out2
```

K-means clustering with 2 clusters of sizes 76, 44

Cluster means:

	[,1]	[,2]
1	2.6196967	-2.644889
2	0.1077512	1.866740

Clustering vector:

```
[1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
[34] 2 2 2 2 2 2 2 1 1 1 1 1 2 1 1 2 2 1 1 1 1 1 1 1 1 1 2  
[67] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
[100] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Within cluster sum of squares by cluster:

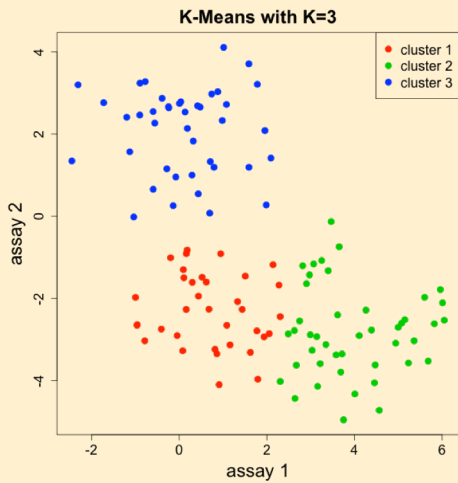
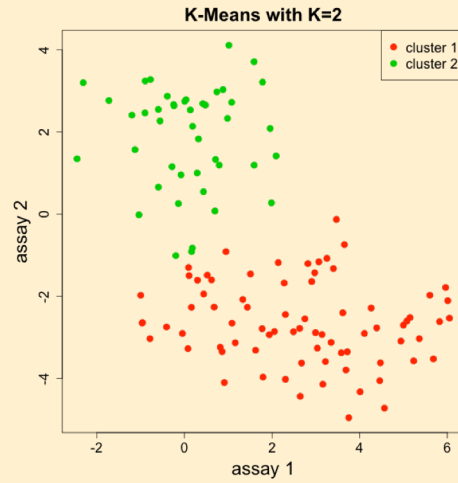
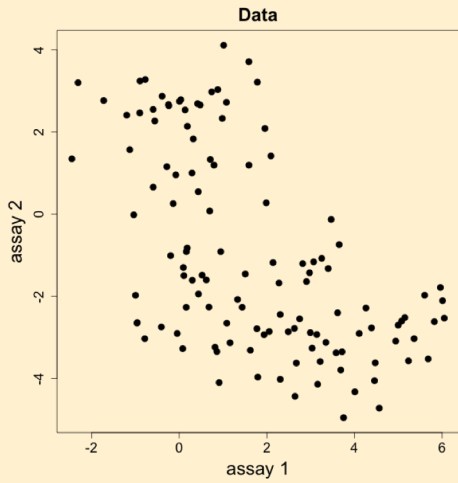
```
[1] 345.0718 118.0078  
(between_SS / total_SS =  61.6 %)
```

Available components:

```
[1] "cluster"      "centers"      "totss"       "withinss"  
[5] "tot.withinss" "betweenss"    "size"        "iter"  
[9] "ifault"
```

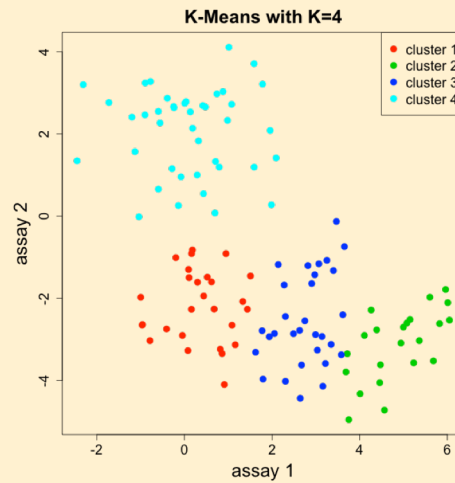
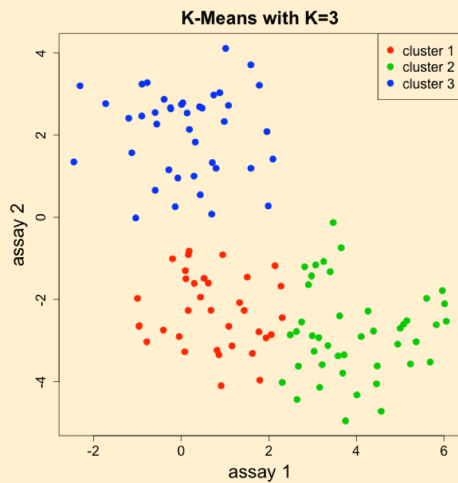
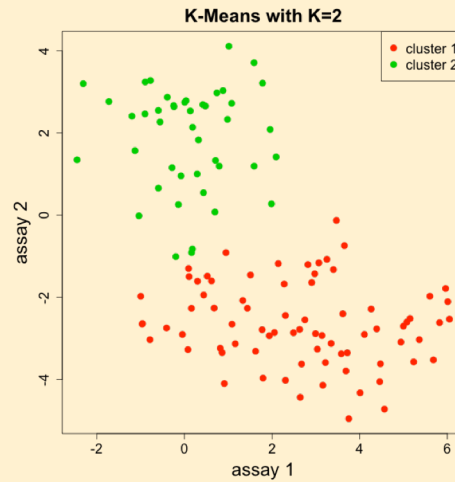
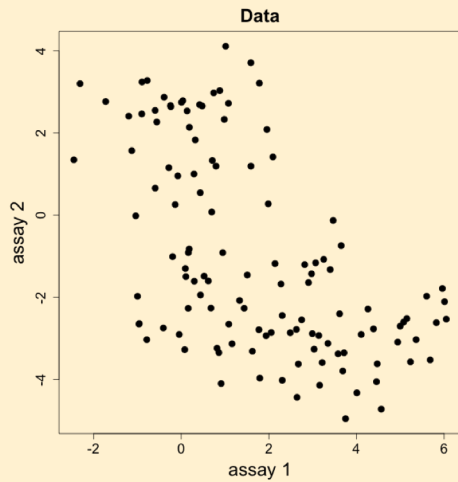
```
>
```

K-means example



```
> set.seed(3)
> km.out3 <- kmeans(x, centers=3)
> km.out3$cluster
 [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[34] 3 3 3 3 3 3 3 1 1 1 2 1 2 3 1 1 1 1 2 1 1 1 1 1 1 2 1 2 1 1 1
[67] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2
[100] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
>
```

K-means example



```
> set.seed(6)
> km.out4 <- kmeans(x,4)
> km.out4
```

K-means clustering with 4 clusters of sizes 31, 38, 26, 25

Cluster means:

```
      [,1]      [,2]
1 0.45629186 -1.855648
2 0.07786265  2.224842
3 2.86555229 -3.506412
4 4.48919252 -2.189089
```

Clustering vector:

```
[1] 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[34] 2 2 2 2 2 2 2 1 1 1 4 3 3 1 3 1 1 1 4 1 1 1 1 3 1 3 1 3 1 1 1 1
[67] 1 1 1 1 1 1 1 1 1 1 3 3 3 1 4 3 4 3 3 4 3 3 4 4 4 4 3 3 4 3 3
[100] 4 4 4 3 4 3 4 4 4 3 4 4 3 4 4 3 4 4 3 3 4 4 3
```

Within cluster sum of squares by cluster:

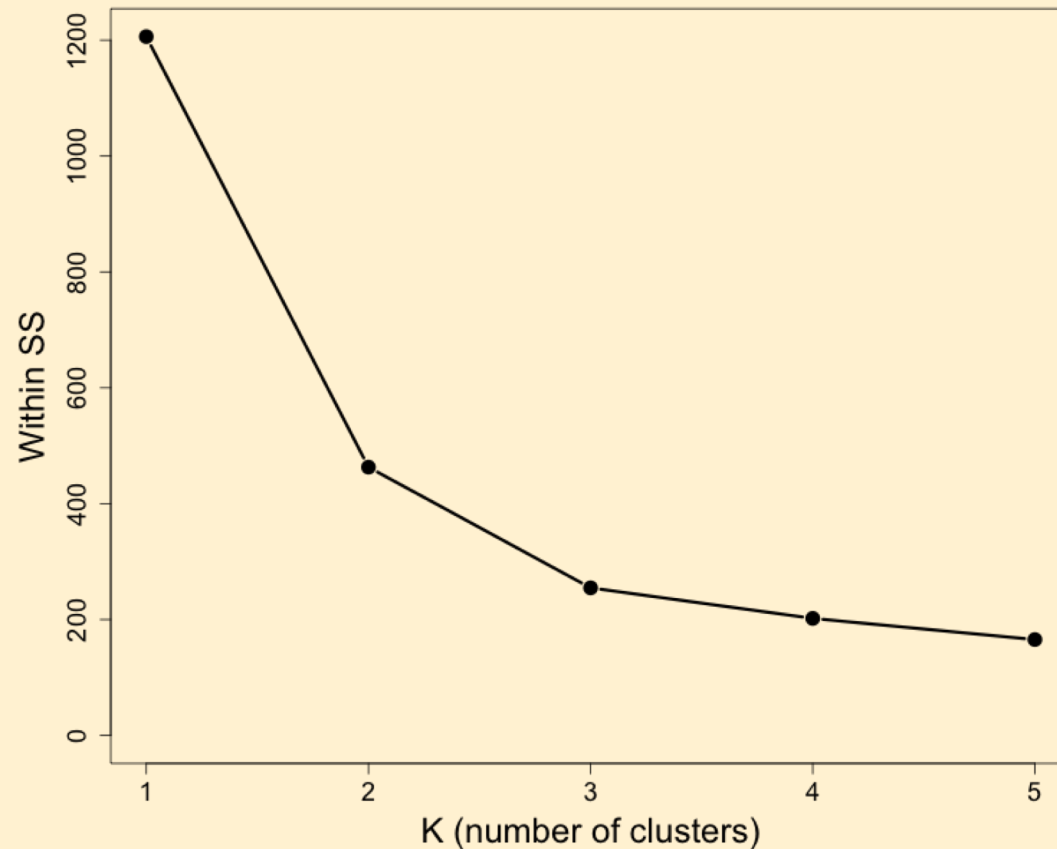
```
[1] 54.97082 75.30927 32.13507 49.41380
(between_SS / total_SS = 82.4 %)
```

Available components:

```
[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

```
>
```

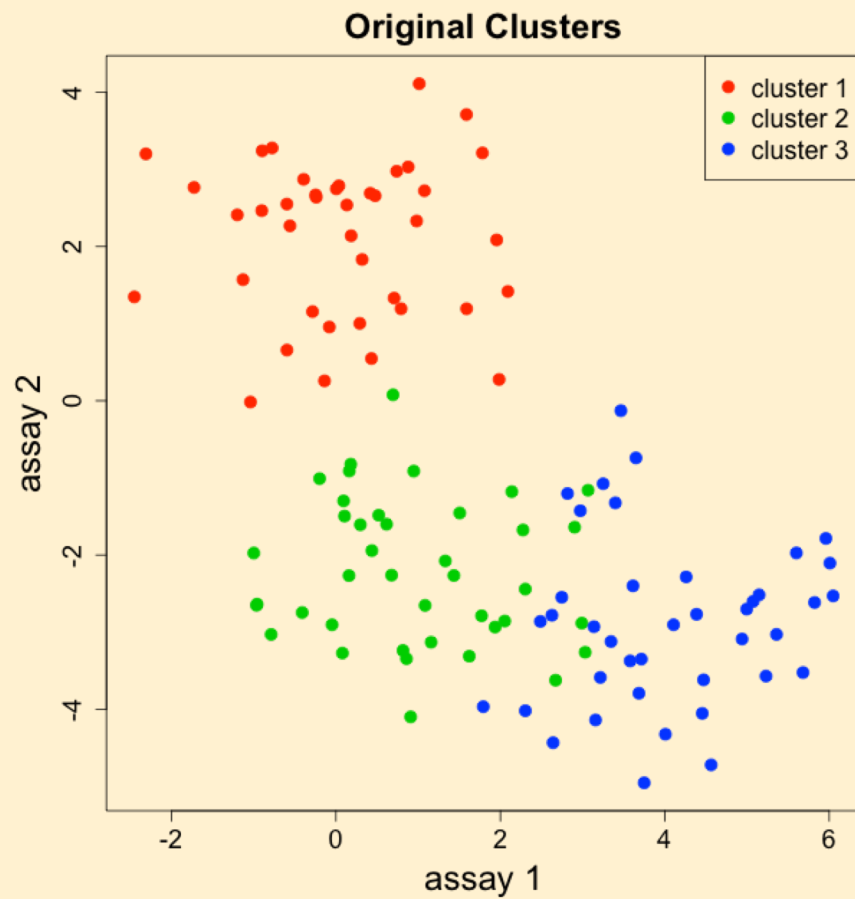

Elbow method



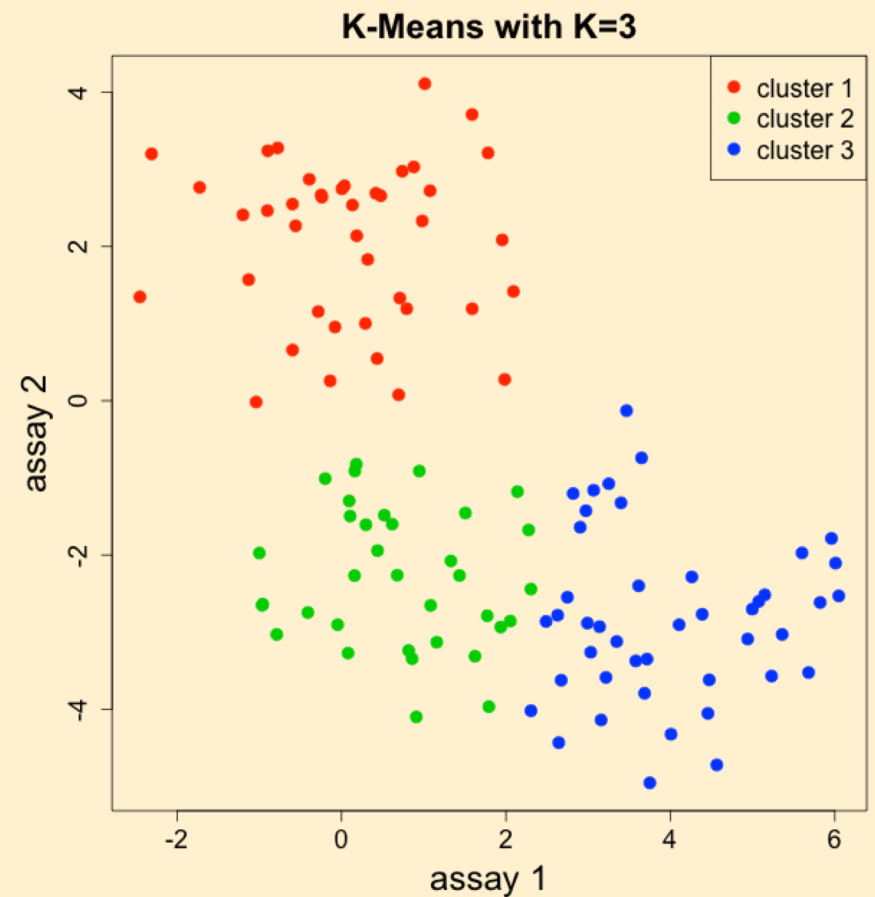
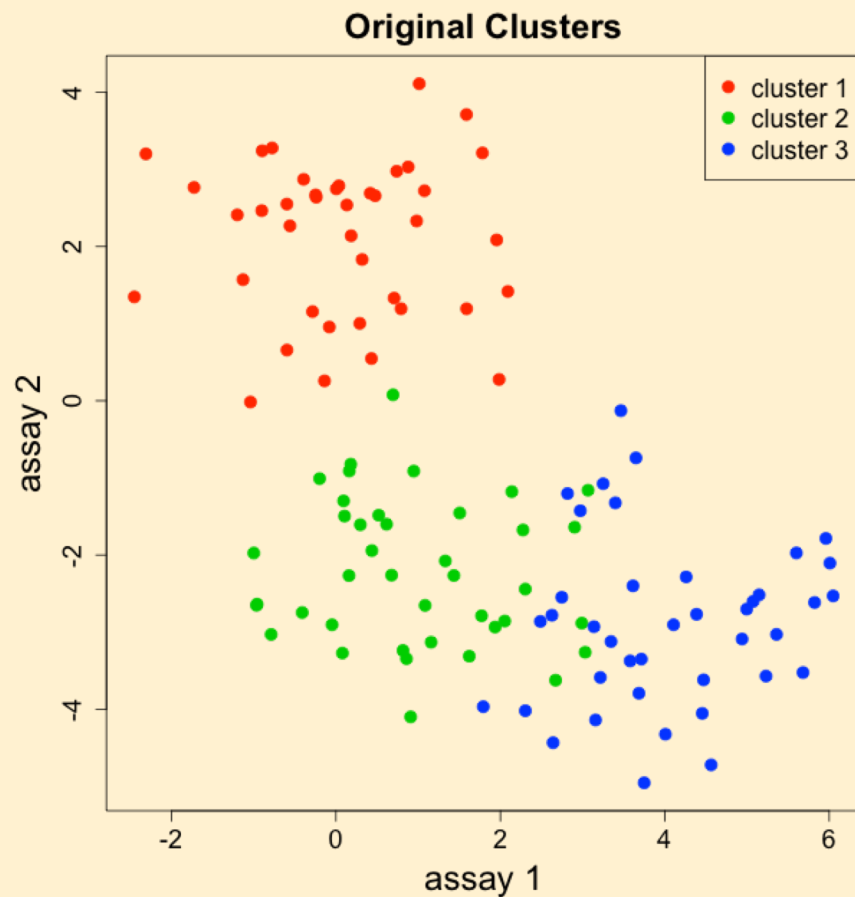
Where does the Within SS stop changing much?

Other methods exist (notably “silhouette method” which tries to quantify quality of clusters) but still somewhat *ad hoc*

Truth is $K = 3$



Truth is $K = 3$



Labels are arbitrary so can permute at will

K-means clustering overview

- Algorithm to split set of observations into K clusters, where K can equal 2, 3, 4, ...
- K is pre-chosen (though often choose a set of different K 's to try)
- Uses square difference dissimilarity measure $a^2 + b^2 + c^2 + \dots$
- Procedure: starts from a random cluster assignment of observations, and then iterates toward increased similarity (reduced square difference dissimilarity) – estimating cluster centers on each iteration
- Can try a bunch of different values for K and try to determine the “best K ” after the fact.

Details of K -means clustering

Let $C_1, C_2, \dots, C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup \dots \cup C_K = \{1, 2, \dots, n\}$, i.e., each observation belongs to at least one of the K clusters.
2. $C_k \cap C_l = \emptyset$ for all $k \neq l$, i.e., the clusters are non-overlapping: no observation belongs to more than one cluster.

If the i^{th} observation is in the k^{th} cluster, then $i \in C_k$

Details continued

- The idea behind K-means clustering is that a **good** clustering is one for which the **within-cluster variation** is as small as possible.
- The within-cluster variation for cluster C_k is a measure $WCV(C_k)$ of the amount by which the observations within a cluster differ from each other.
- Hence we want to solve the problem

$$\text{minimize}_{C_1, C_2, \dots, C_K} \left\{ \sum_{k=1}^K WCV(C_k) \right\}$$

- In words, this formula says that we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.

Defining within-cluster variation

- Use Euclidean distance: squared difference dissimilarity

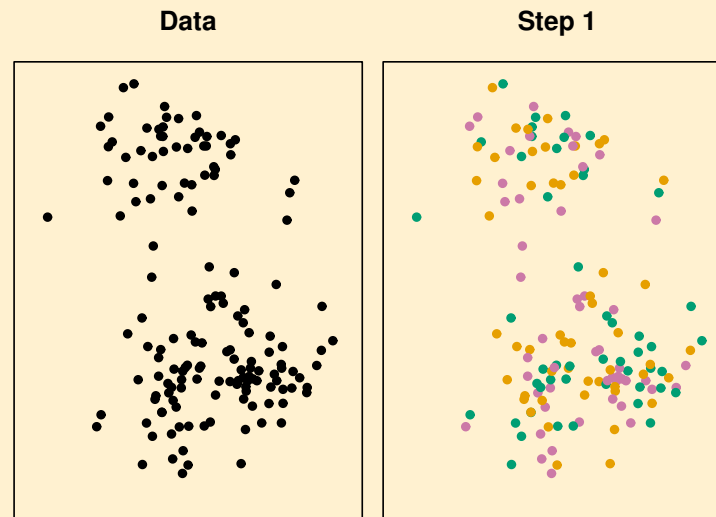
$$\text{WCV}(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2$$

- So minimizing this summed over all clusters, per the previous slide, we get K -means clustering via the optimization

$$\underset{C_1, C_2, \dots, C_K}{\text{minimize}} \left\{ \sum_{k=1}^K \text{WCV}(C_k) \right\} = \underset{C_1, C_2, \dots, C_K}{\text{minimize}} \left\{ \sum_{k=1}^K \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \right\}$$

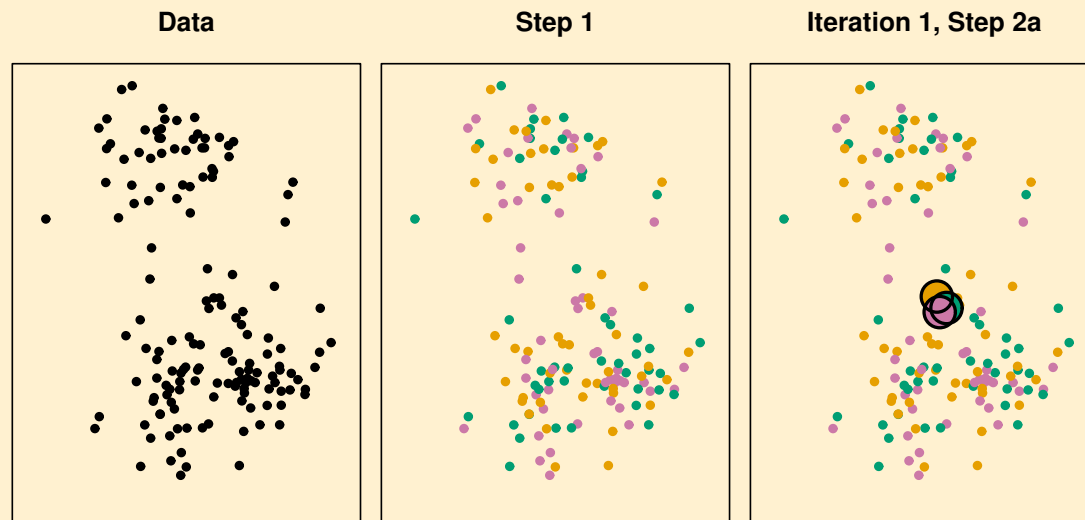
K-means algorithm

- Algorithm proceeds as follows:
 1. Randomly assign a cluster number (between 1 and K) to each of the observations/instances in the dataset. These are the *initial* cluster assignments



K-means algorithm

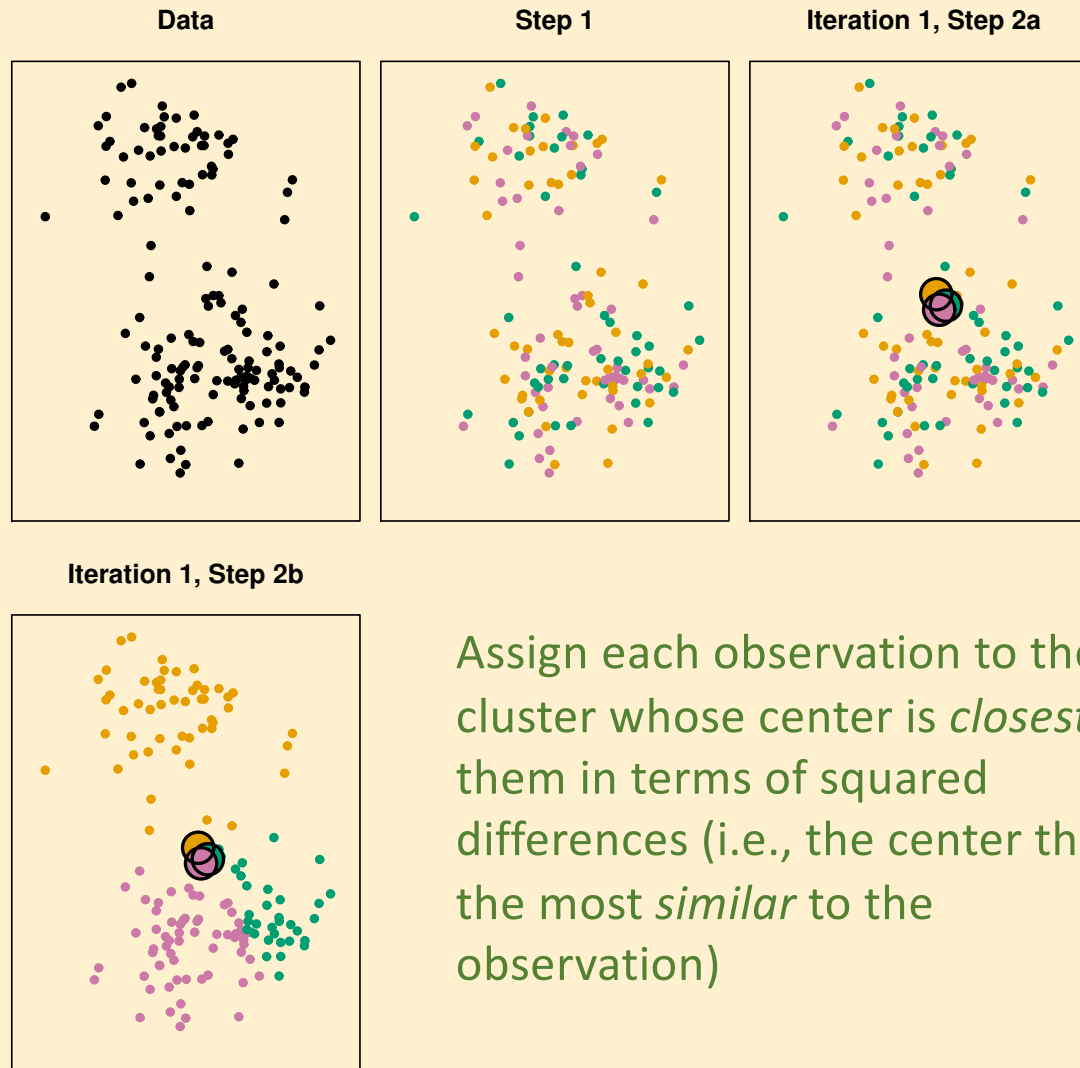
- Algorithm proceeds as follows:
 1. Randomly assign a cluster number (between 1 and K) to each of the observations/instances in the dataset. These are the *initial* cluster assignments
 2. Iterate between the following until the cluster assignments no longer change:
 - a. For each cluster determine the center (centroid) of the cluster



K-means algorithm

- Algorithm proceeds as follows:
 1. Randomly assign a cluster number (between 1 and K) to each of the observations/instances in the dataset. These are the *initial* cluster assignments
 2. Iterate between the following until the cluster assignments no longer change:
 - a. For each cluster determine the center (centroid) of the cluster
 - b. Assign each observation to the cluster whose center is *closest* to them in terms of squared differences (i.e., the center that is the most *similar* to the observation)

Illustration

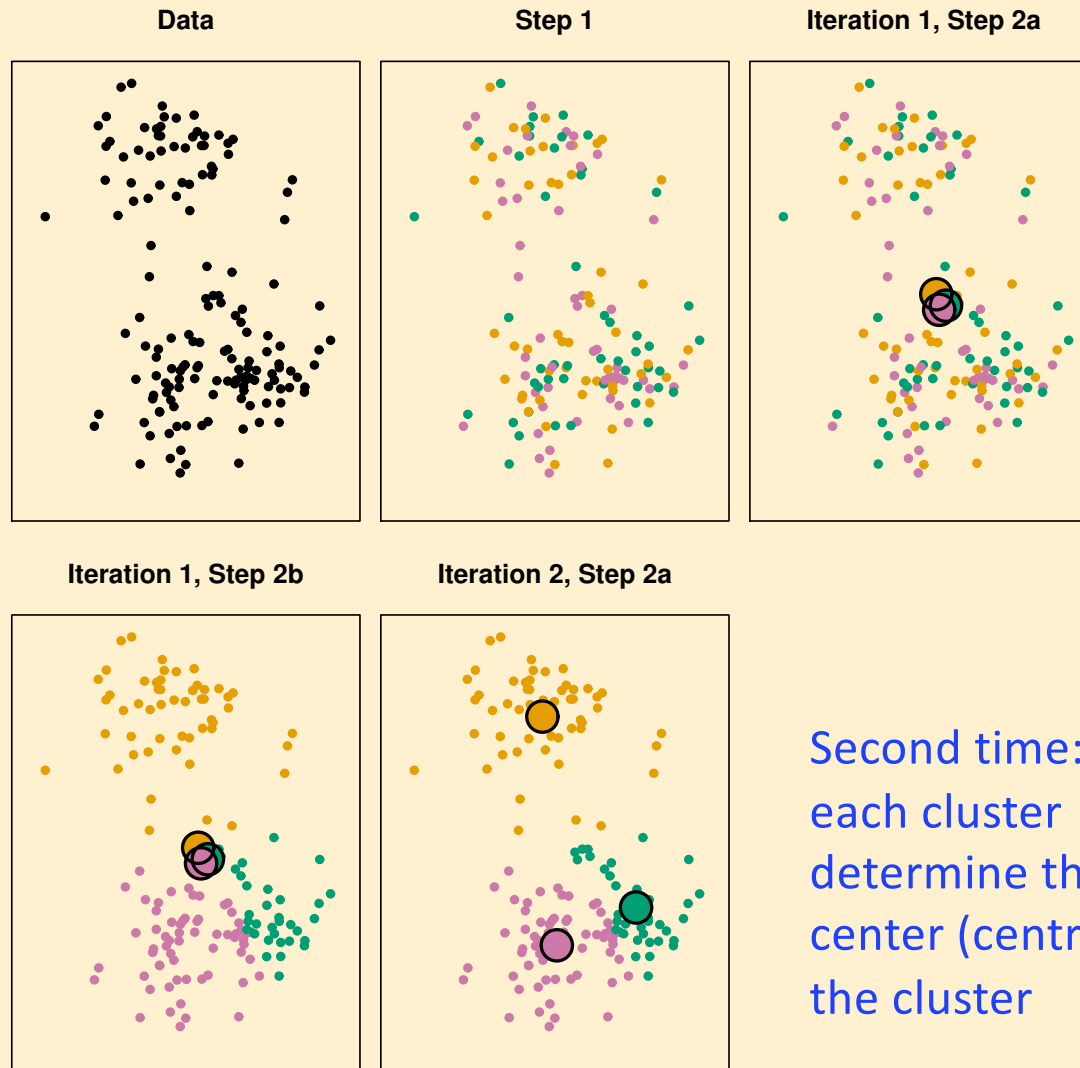


Assign each observation to the cluster whose center is *closest* to them in terms of squared differences (i.e., the center that is the most *similar* to the observation)

K-means algorithm

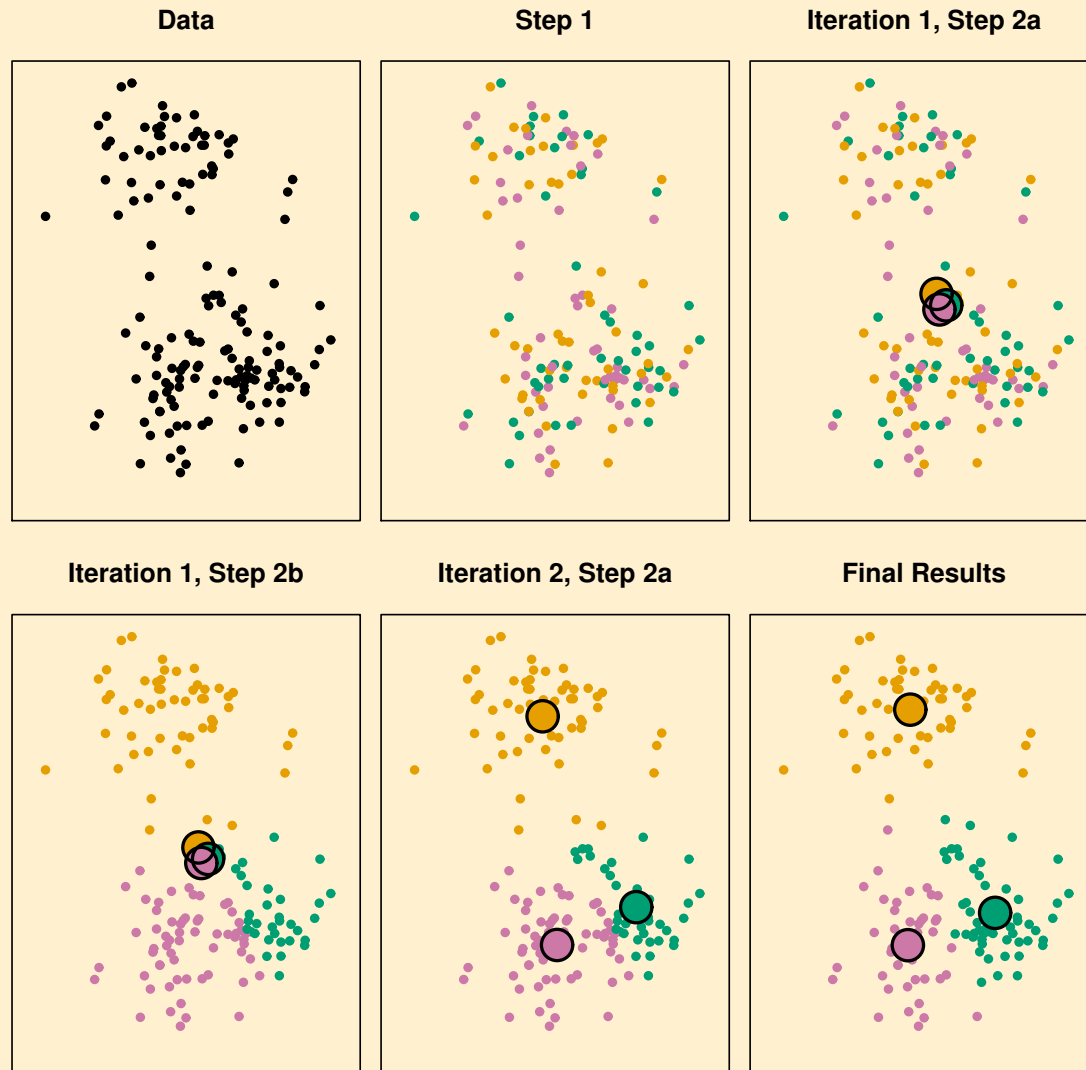
- Algorithm proceeds as follows:
 1. Randomly assign a cluster number (between 1 and K) to each of the observations/instances in the dataset. These are the *initial* cluster assignments
 2. Iterate between the following until the cluster assignments no longer change:
 - a. For each cluster determine the center (centroid) of the cluster
 - b. Assign each observation to the cluster whose center is *closest* to them in terms of squared differences (i.e., the center that is the most *similar* to the observation)

Illustration



Second time: For each cluster determine the center (centroid) of the cluster

Illustration

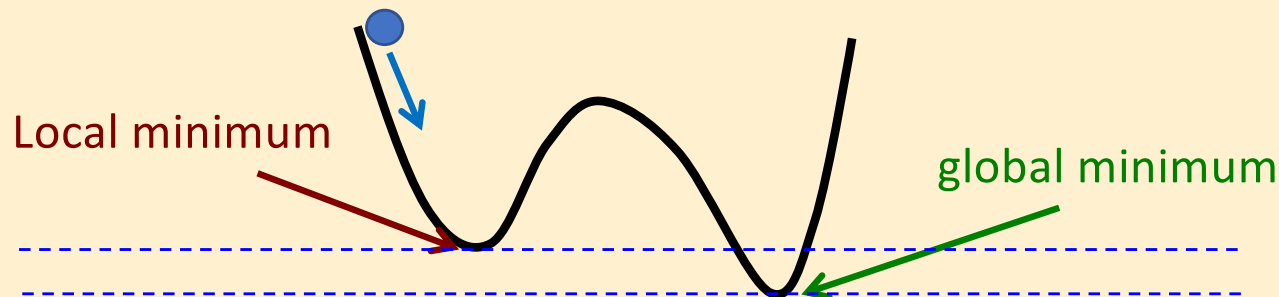


K-means algorithm

- Algorithm proceeds as follows:
 1. Randomly assign a cluster number (between 1 and K) to each of the observations/instances in the dataset. These are the *initial* cluster assignments
 2. Iterate between the following until the cluster assignments no longer change:
 - a. For each cluster determine the center (centroid) of the cluster
 - b. Assign each observation to the cluster whose center is *closest* to them in terms of squared differences (i.e., the center that is the most *similar* to the observation)
- The algorithm increases the within-cluster similarity at every step

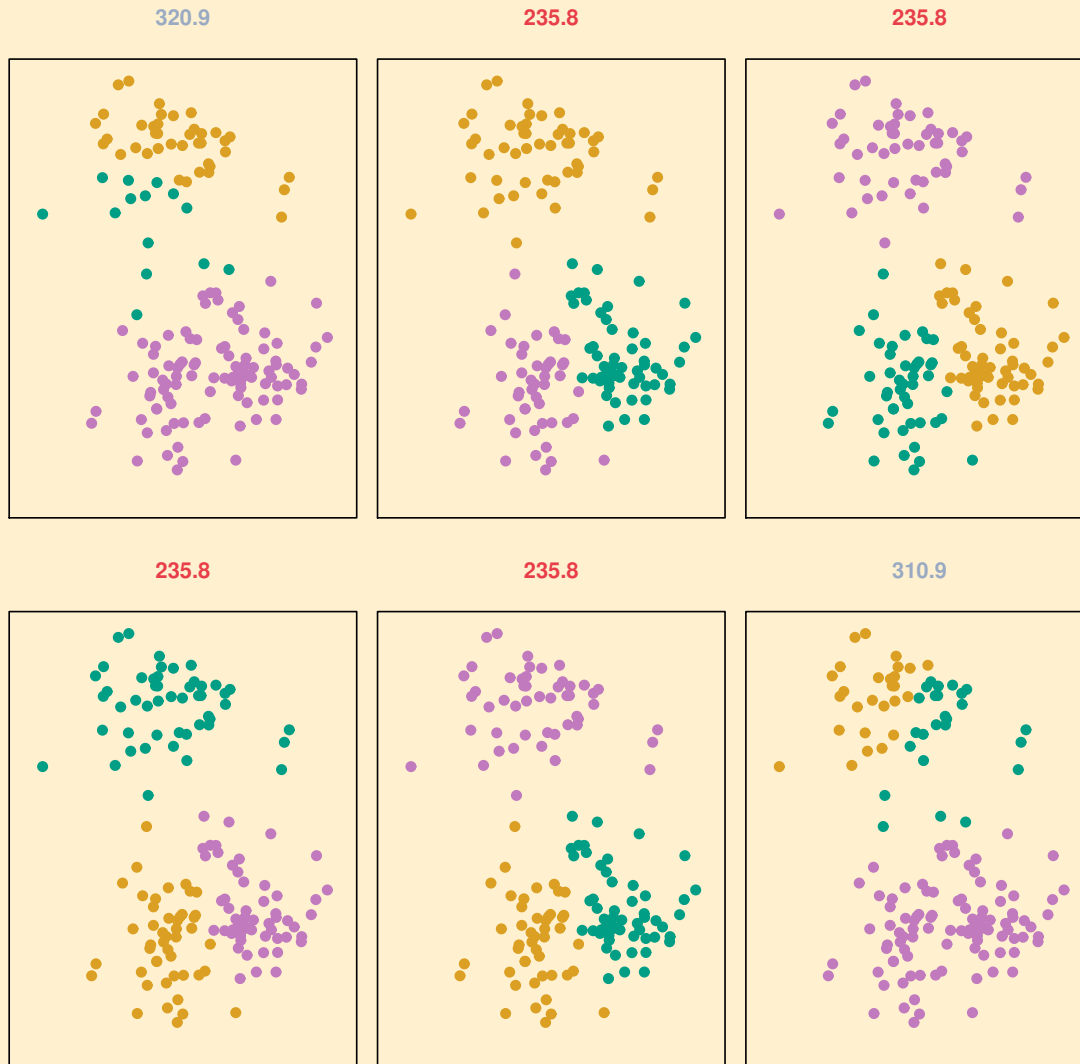
K-means algorithm

- The center of a cluster is simply the point associated with the mean of each of the attributes for the observations assigned to that cluster – e.g. in the Age direction the center would be located at the mean Age for all subjects in the cluster.
- Note that for different initial random assignments of observations to clusters, K-means may lead to different final clustering (it is a *local* not *global* minimizer of dissimilarity)



Upshot - may not converge to best answer

Different starting values



This problem of finding only a local minimum can be somewhat mitigated by using multiple starting random configurations and taking the best one

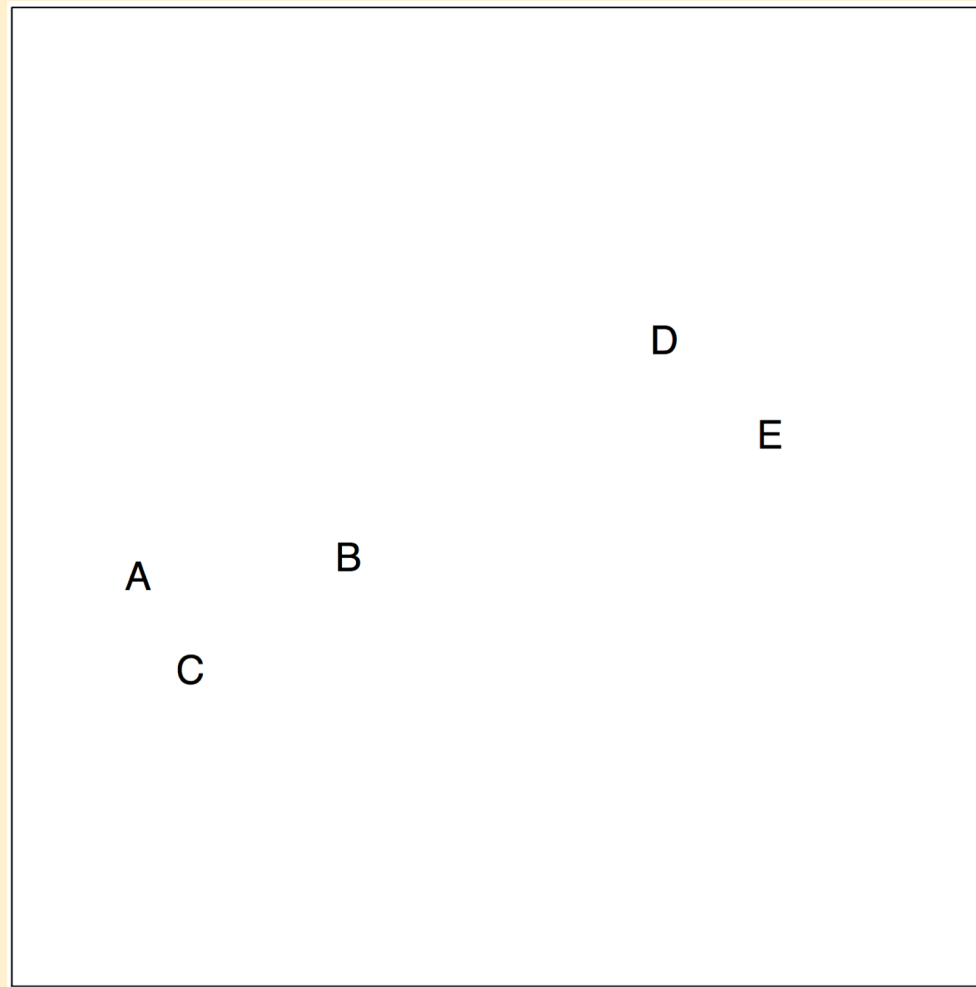
This is easy in R:

```
> km.out2 <- kmeans(x, centers=2, nstart=20)  
>
```

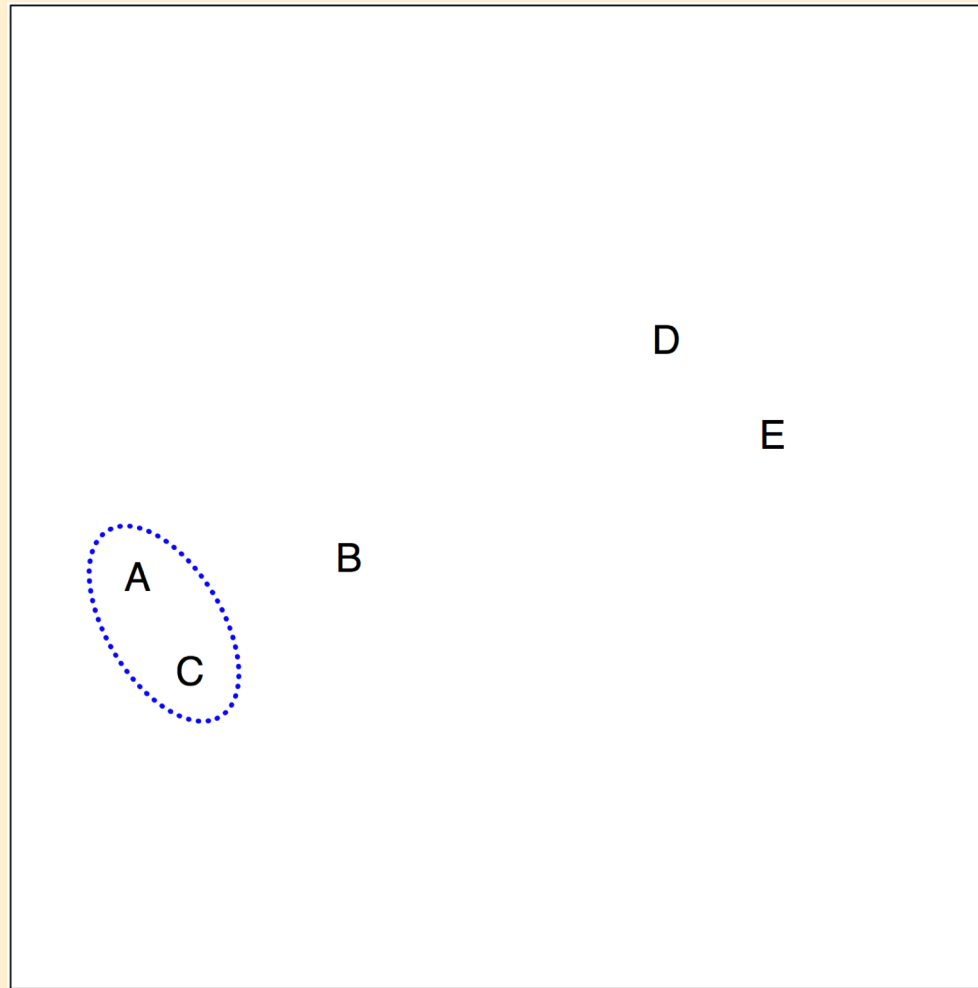
Hierarchical clustering

- K -means clustering requires us to pre-specify the number of clusters K . This can be a disadvantage (later we discuss strategies for choosing K)
- *Hierarchical clustering* is an alternative approach which does not require that we commit to a particular choice of K .
- We will consider *bottom-up* or *agglomerative* clustering. This is the most common type of hierarchical clustering, and refers to the fact that a *dendrogram* is built starting from the leaves and combining clusters up to the trunk.

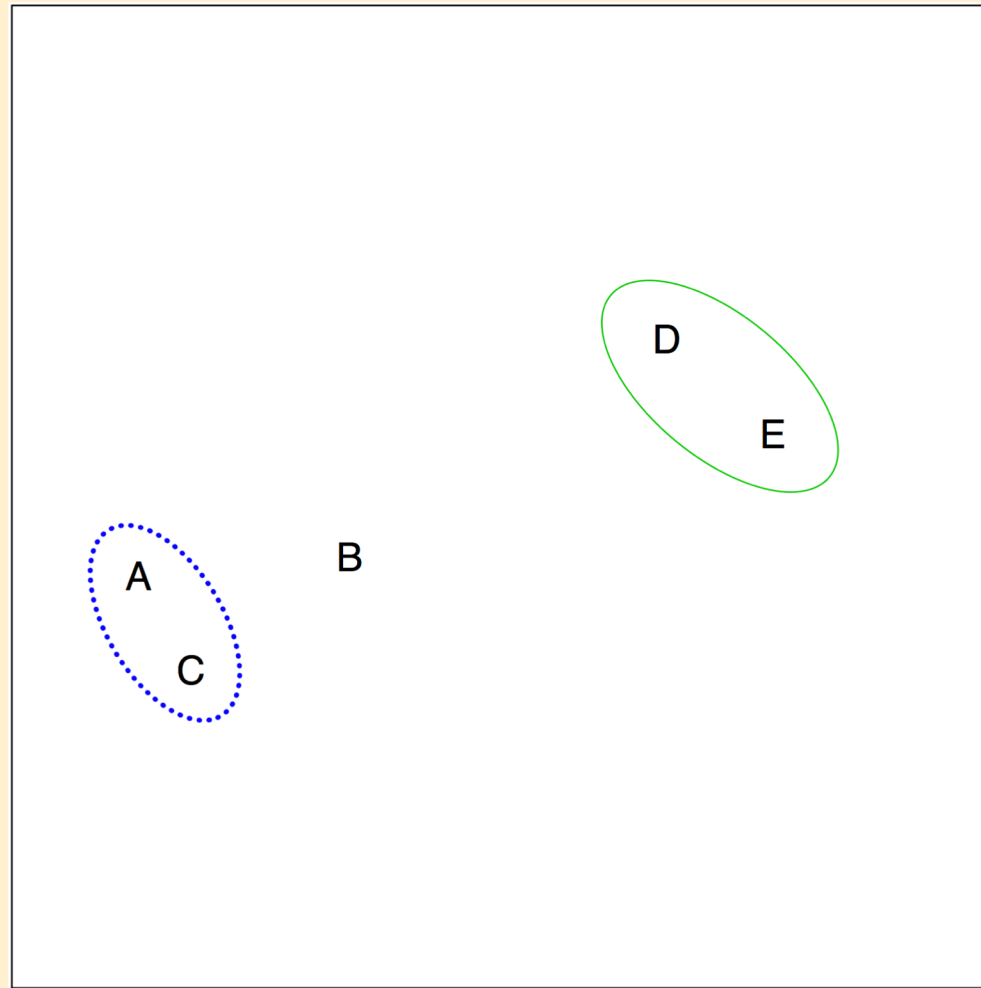
Hierarchical clustering – bottom up



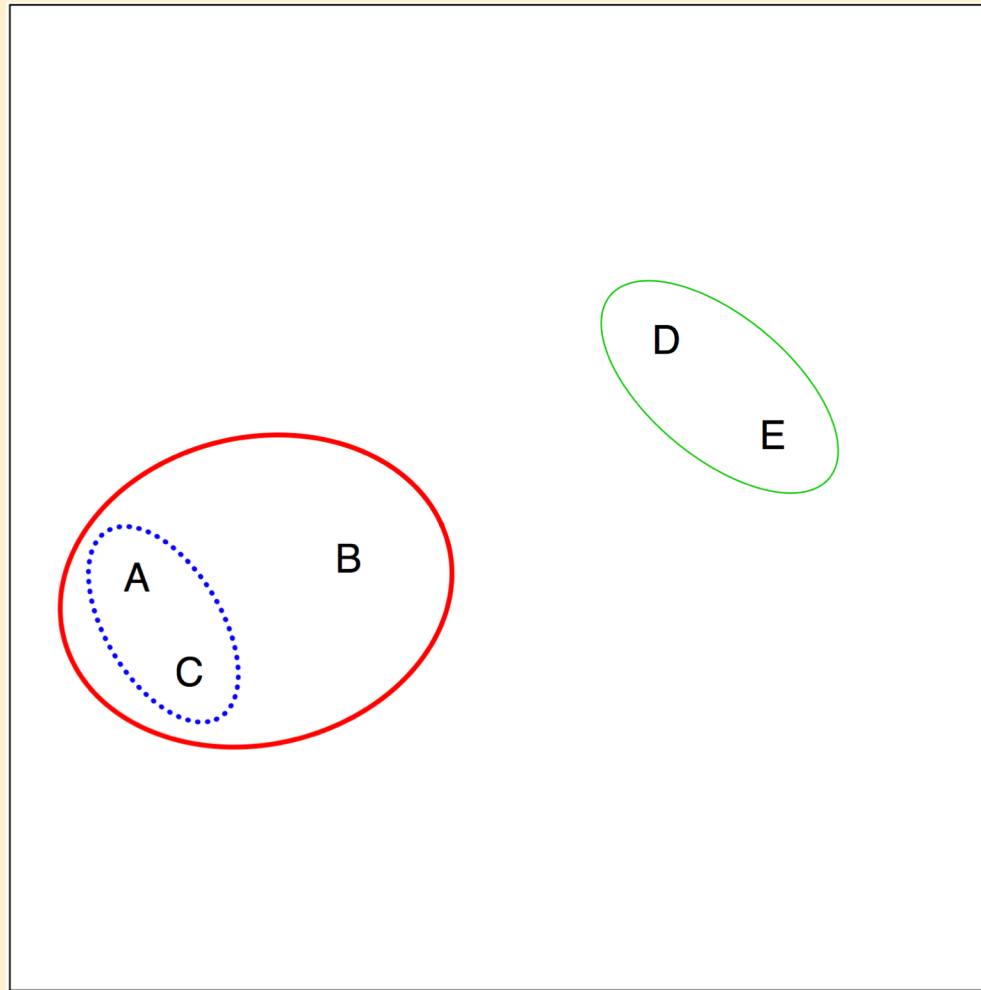
Hierarchical clustering – bottom up



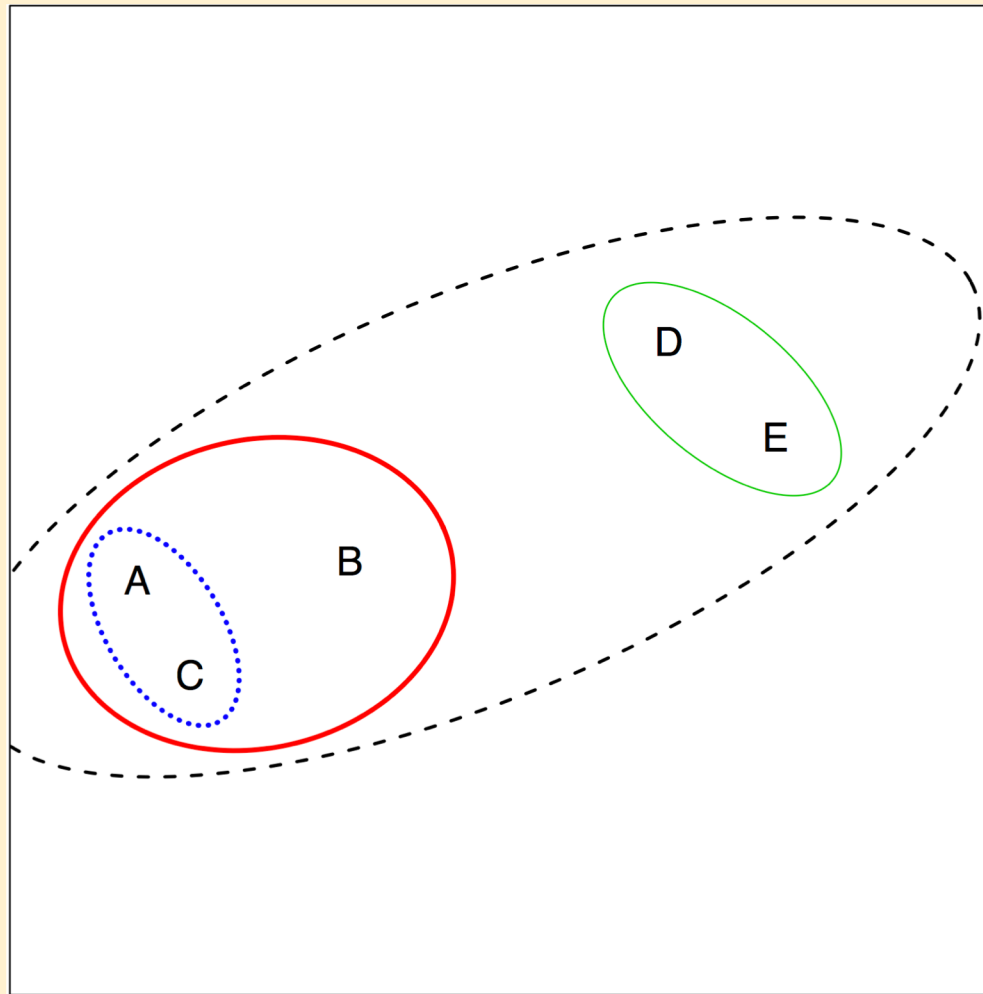
Hierarchical clustering – bottom up



Hierarchical clustering – bottom up



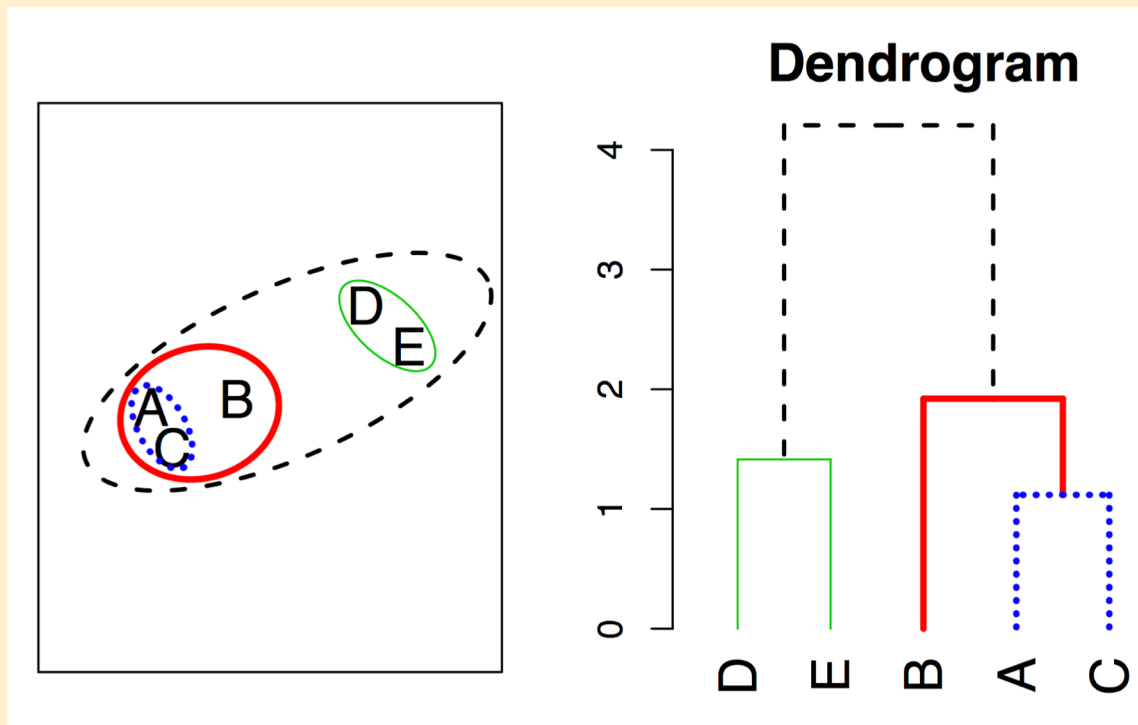
Hierarchical clustering – bottom up



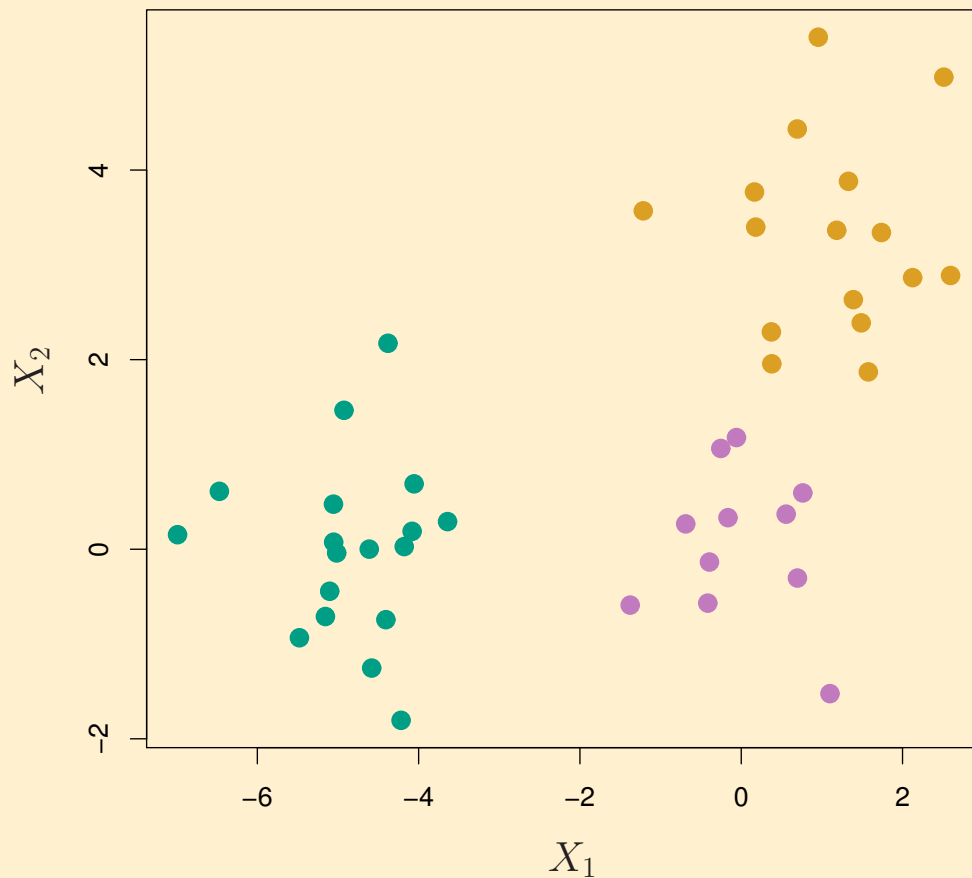
Hierarchical Clustering Algorithm

The approach in words:

- Start with each point in its own cluster
- Identify the closest two clusters and merge them
- Repeat.
- Ends when all points are in a single cluster.



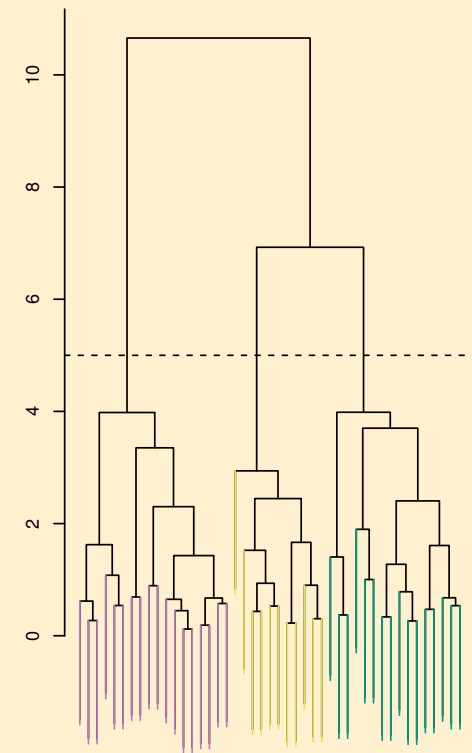
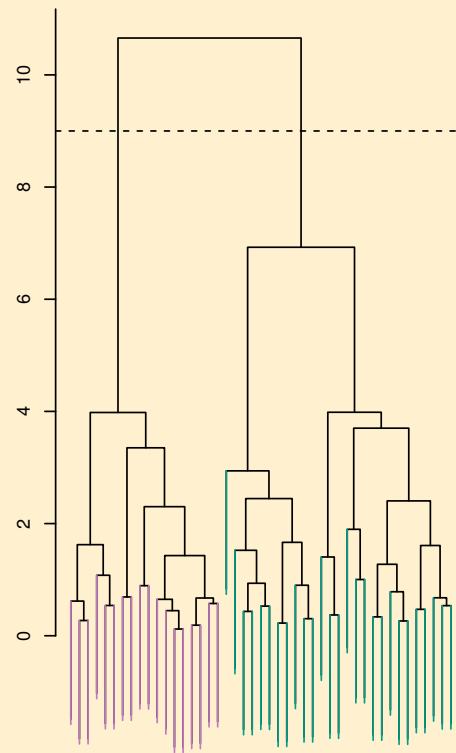
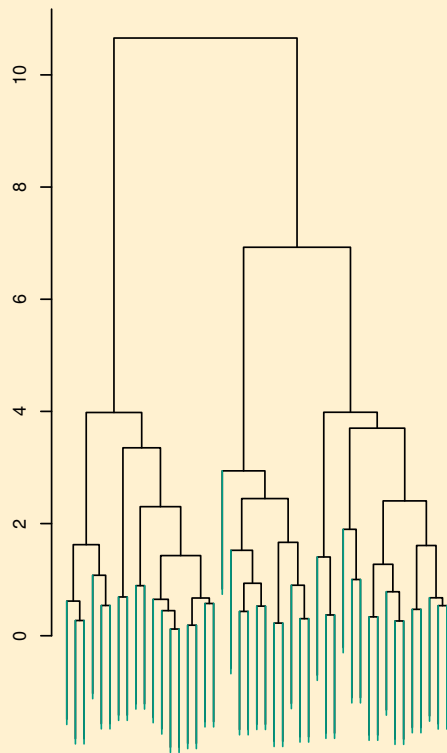
Simulated example



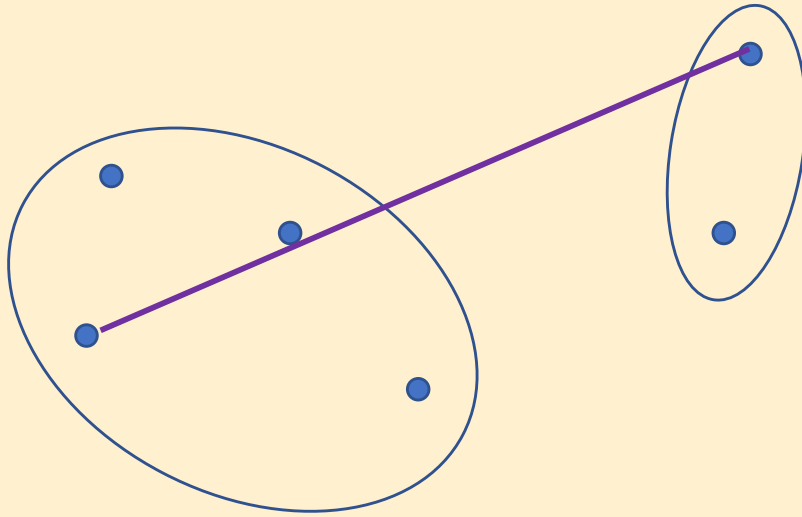
- 45 observations generated in 2D space
- 3 distinct classes (separate colors)
- Treat class labels as unknown and try to cluster the observations in order to discover the classes from the data.

Application of hierarchical clustering

- Left: Dendrogram from hierarchical clustering (complete linkage and Euclidean distance)
- Center: cut at 9 = 2 clusters
- Right: cut at 5 = 3 clusters

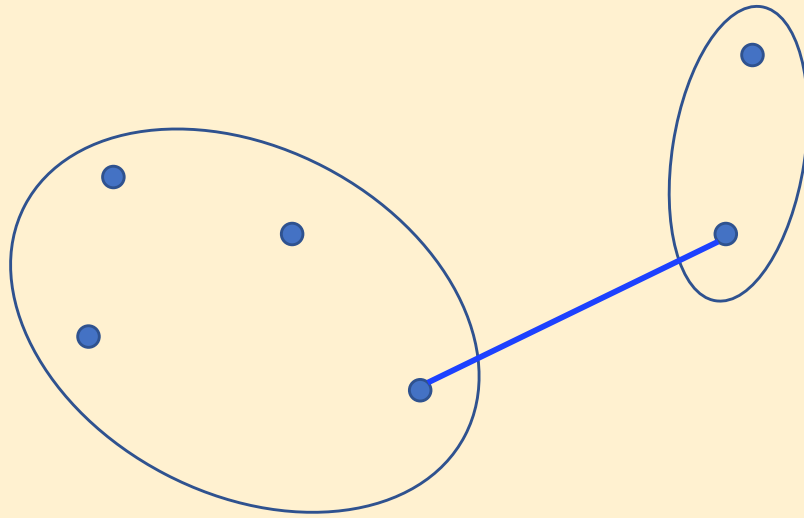


Types of linkage – complete



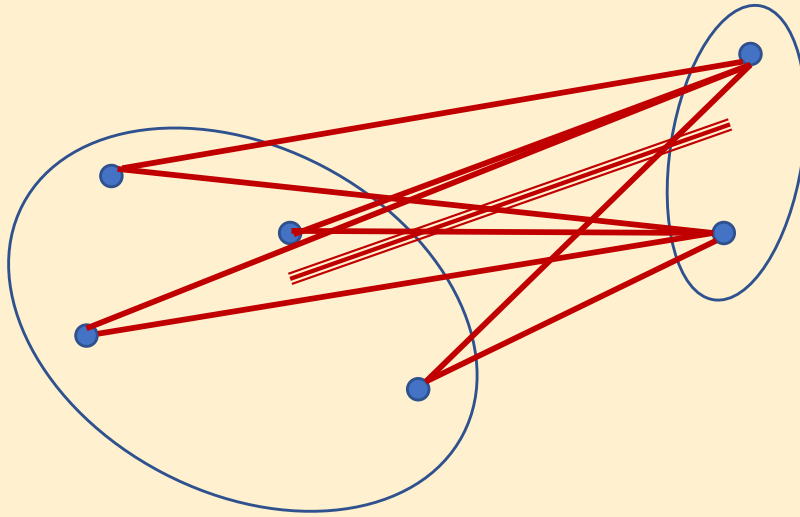
Complete – largest dissimilarity between 2 points

Types of linkage – single



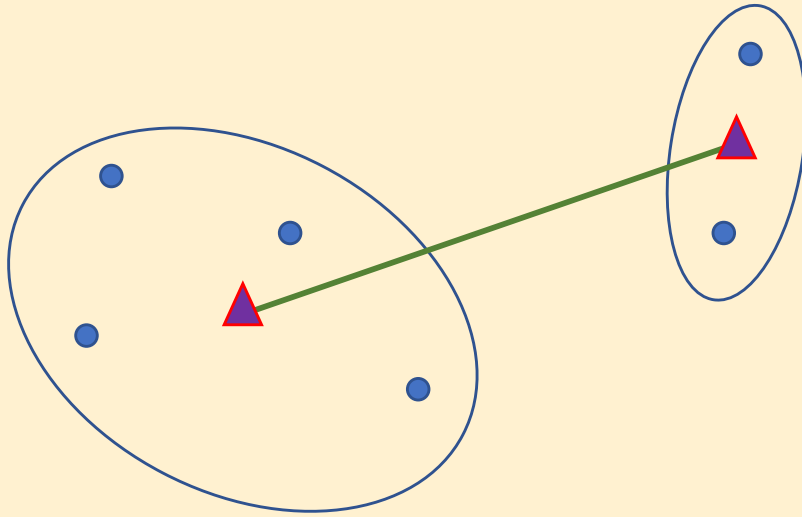
Single – smallest dissimilarity between 2 points

Types of linkage – average



Average – average dissimilarity between 2 points

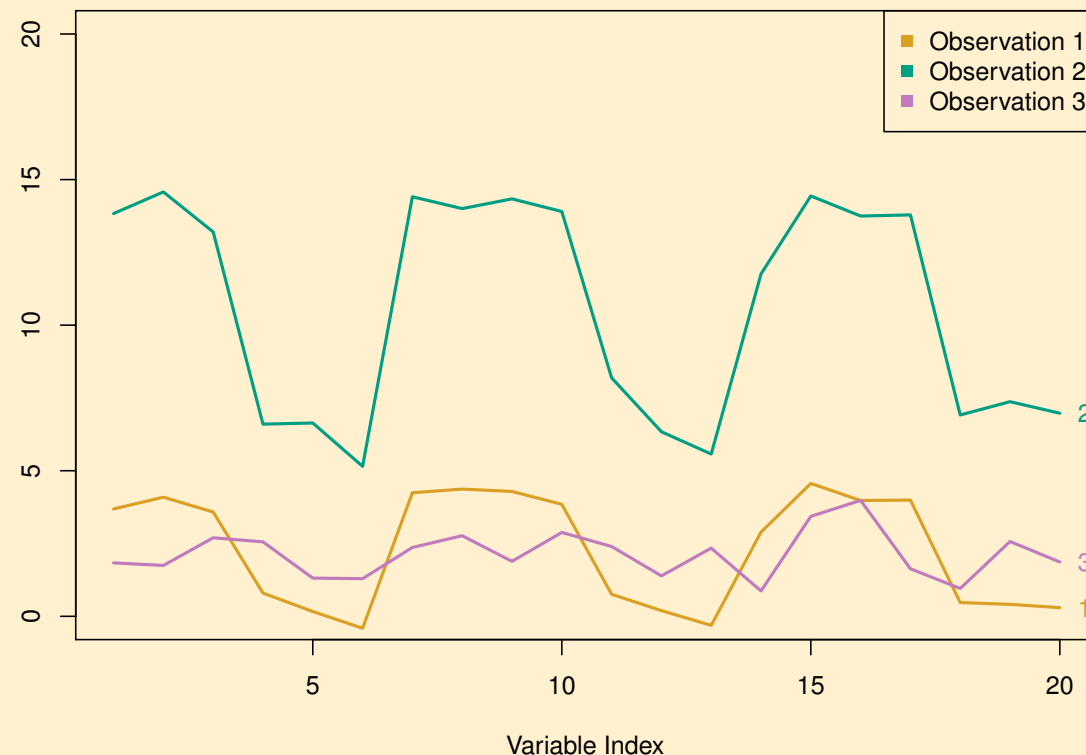
Types of linkage – single



Centroid – dissimilarity between 2 centroids

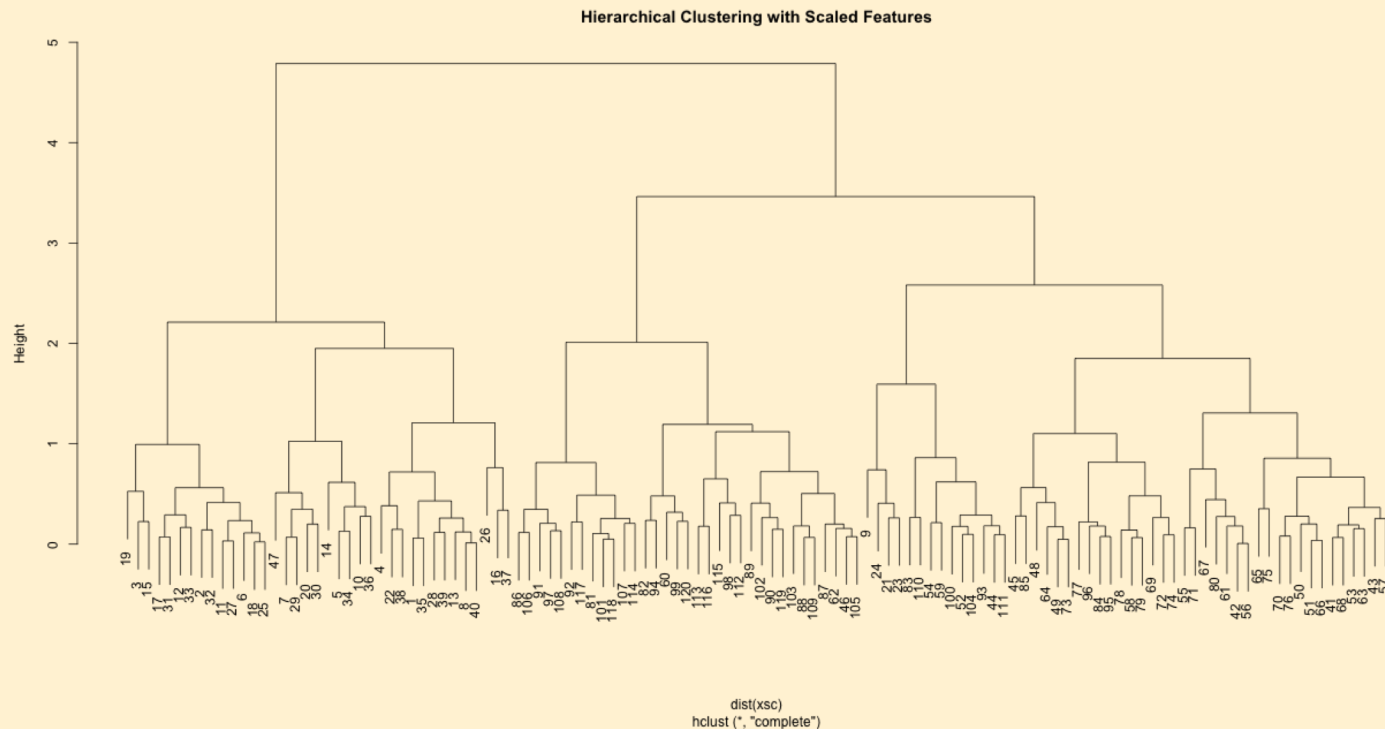
Choice of dissimilarity measure

- So far have used Euclidean distance.
- An alternative is *correlation-based distance* which considers two observations to be similar if their features are highly correlated.
- This is an unusual use of correlation, which is normally computed between variables; here it is computed between the observation profiles for each pair of observations.



Hierarchical clustering in R

```
> htree <- hclust(dist(x), method="complete")
> cutree(htree, 2)
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[34] 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[67] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[100] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
> xsc <- scale(x)
> plot(hclust(dist(xsc),method="complete"), main="Hierarchical Clustering
with Scaled Features")
> |
```



Principal components analysis

- PCA produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated.
- Apart from producing derived variables for use in supervised learning problems, PCA also serves as a tool for data visualization.

Clustering vs. PCA

- Clustering looks for homogeneous subgroups among the observations
- PCA looks for a low-dimensional representation of the observations that explains a good fraction of the variance

PCA details

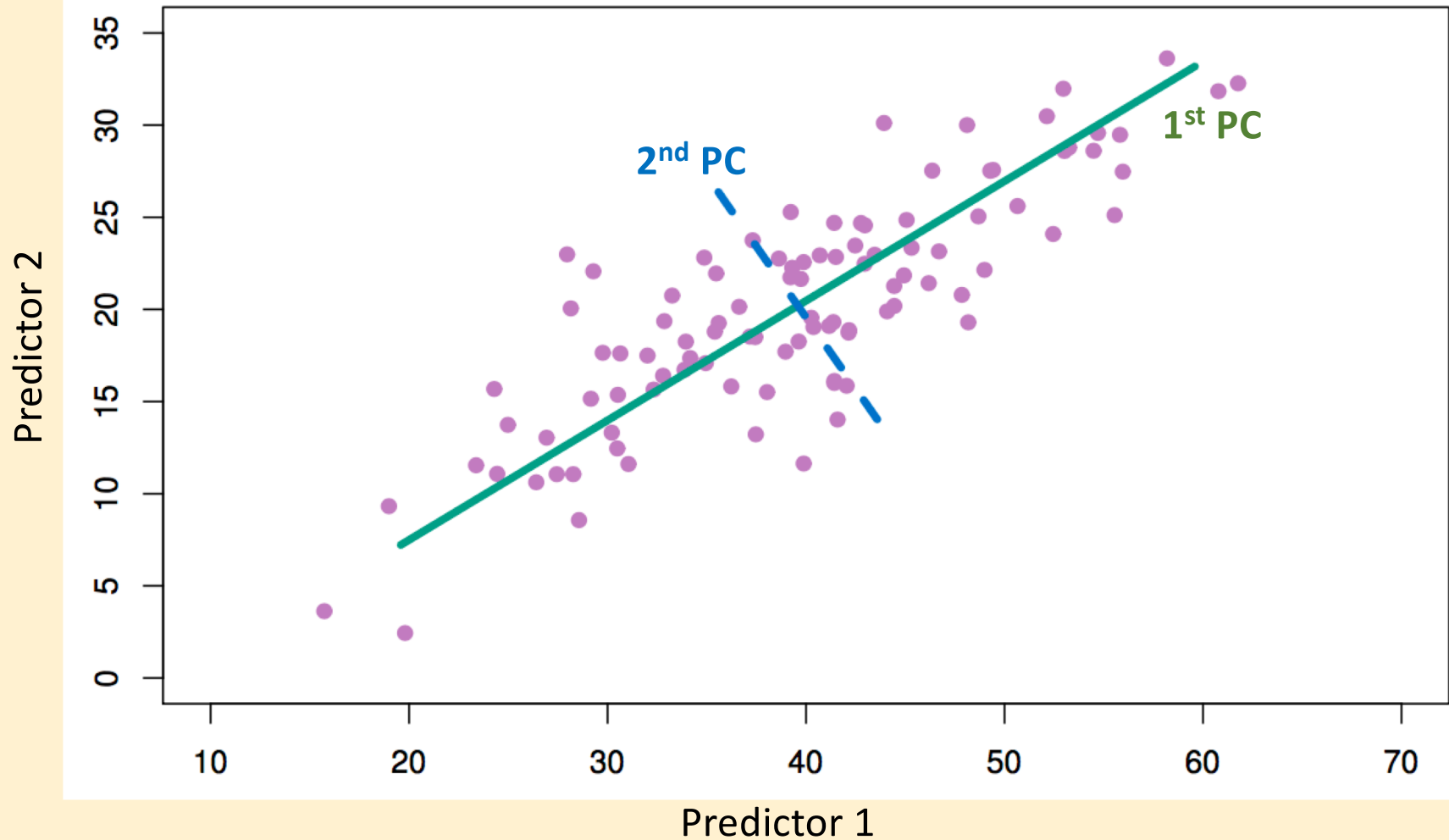
- The *first principal component* of a set of features $X_1, X_2, \dots, X_p$ is the normalized linear combination of the features

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \dots + \phi_{p1}X_p$$

that has the largest variance. By *normalized*, we mean that $\sum_{j=1}^p \phi_{j1}^2 = 1$.

- We refer to the elements $\phi_{11}, \dots, \phi_{p1}$ as the loadings of the first principal component; together, the loadings make up the principal component loading vector,
 $\phi_1 = (\phi_{11} \ \phi_{21} \ \dots \ \phi_{p1})^T$.
- We constrain the loadings so that their sum of squares is equal to one, since otherwise setting these elements to be arbitrarily large in absolute value could result in an arbitrarily large variance.

PCA illustration



Computation of PCA

- Suppose we have a $n \times p$ data set $\mathbf{X}$. Since we are only interested in variance, we assume that each of the variables in $\mathbf{X}$ has been centered to have mean zero (that is, the column means of $\mathbf{X}$ are zero).
- We then look for the linear combination of the sample feature values of the form

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \dots + \phi_{p1}x_{ip} \quad (1)$$

for $i = 1, \dots, n$ that has largest sample variance, subject to the constraint that $\sum_{j=1}^p \phi_{j1}^2 = 1$.

- Since each of the x_{ij} has mean zero, then so does z_{i1} (for any values of ϕ_{j1}). Hence the sample variance of the z_{i1} can be written as $\frac{1}{n} \sum_{i=1}^n z_{i1}^2$.

Computation of PCA continued

- Plugging in (1) the first principal component loading vector solves the optimization problem

$$\underset{\phi_{11}, \dots, \phi_{p1}}{\text{maximize}} \frac{1}{n} \sum_{i=1}^n \left(\sum_{j=1}^p \phi_{j1} x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^p \phi_{j1}^2 = 1.$$

- This problem can be solved via a singular-value decomposition of the matrix $\mathbf{X}$, a standard technique in linear algebra.
- We refer to Z_1 as the first principal component, with realized values $z_{11}, \dots, z_{n1}$

Geometry of PCA

- The loading vector ϕ_1 with elements $\phi_{11}, \phi_{21}, \dots, \phi_{p1}$ defines a direction in feature space along which the data vary the most.
- If we project the n data points $x_1, \dots, x_n$ onto this direction, the projected values are the principal component scores $z_{11}, \dots, z_{n1}$ themselves.

Additional PC's

- The second principal component is the linear combination of $X_1, \dots, X_p$ that has maximal variance among all linear combinations that are *uncorrelated* with Z_1 .
- The second principal component scores $z_{12}, z_{22}, \dots, z_{n2}$ take the form

$$z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + \dots + \phi_{p2}x_{ip},$$

where ϕ_2 is the second principal component loading vector, with elements $\phi_{12}, \phi_{22}, \dots, \phi_{p2}$.

Additional PC's continued

- It turns out that constraining Z_2 to be uncorrelated with Z_1 is equivalent to constraining the direction ϕ_2 to be orthogonal (perpendicular) to the direction ϕ_1 . And so on.
- The principal component directions $\phi_1, \phi_2, \phi_3, \dots$ are the ordered sequence of right singular vectors of the matrix $\mathbf{X}$, and the variances of the components are $\frac{1}{n}$ times the squares of the singular values. There are at most $\min(n - 1, p)$ principal components.

Proportion variance explained

- To understand the strength of each component, we are interested in knowing the proportion of variance explained (PVE) by each one.
- The *total variance* present in a data set (assuming that the variables have been centered to have mean zero) is defined as

$$\sum_{j=1}^p \text{Var}(X_j) = \sum_{j=1}^p \frac{1}{n} \sum_{i=1}^n x_{ij}^2,$$

and the variance explained by the m th principal component is

$$\text{Var}(Z_m) = \frac{1}{n} \sum_{i=1}^n z_{im}^2.$$

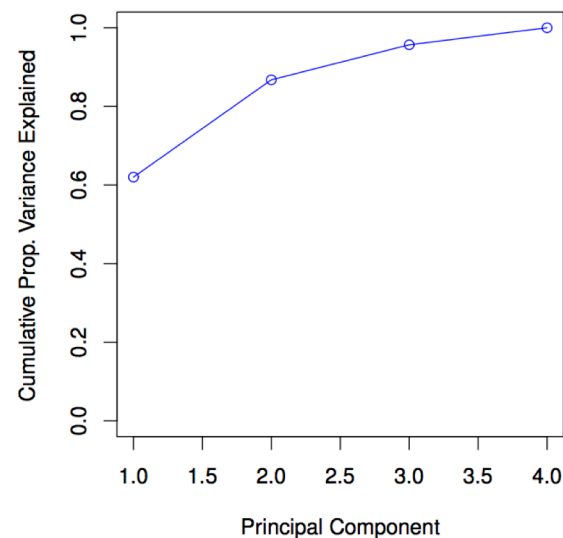
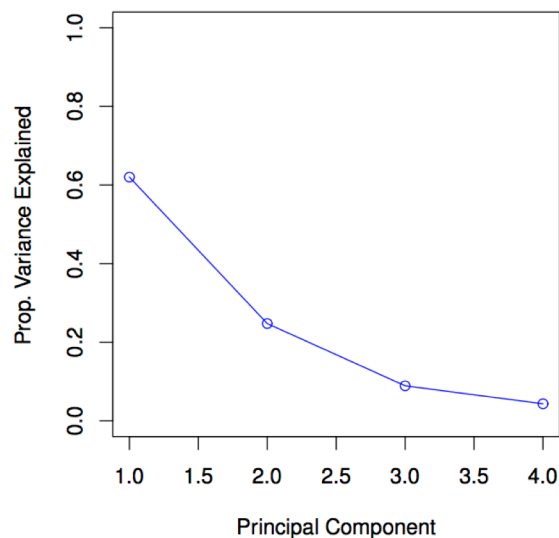
- It can be shown that $\sum_{j=1}^p \text{Var}(X_j) = \sum_{m=1}^M \text{Var}(Z_m)$, with $M = \min(n - 1, p)$.

Prop var explained continued

- Therefore, the PVE of the m th principal component is given by the positive quantity between 0 and 1

$$\frac{\sum_{i=1}^n z_{im}^2}{\sum_{j=1}^p \sum_{i=1}^n x_{ij}^2}.$$

- The PVEs sum to one. We sometimes display the cumulative PVEs.



How many PC's?

If we use principal components as a summary of our data, how many components are sufficient?

- No simple answer to this question, as cross-validation is not available for this purpose.
 - *Why not?*
 - When could we use cross-validation to select the number of components?
- the “scree plot” on the previous slide can be used as a guide: we look for an “elbow”.

PCA in R

- We will work through in lab
- `prcomp` command
- But many others also exist

```
prcomp() (stats)  
princomp() (stats)  
PCA() (FactoMineR)  
dudi.pca() (ade4)  
acp() (amap)
```

Summary

- We looked at unsupervised methods
- Clustering: K-means and hierarchical
- Dimension reduction: principal components analysis
- Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani

Next Lecture

- Applications lecture: clustering and classification of high-dimensional data in a genomic context – Dr. Adam Olshen