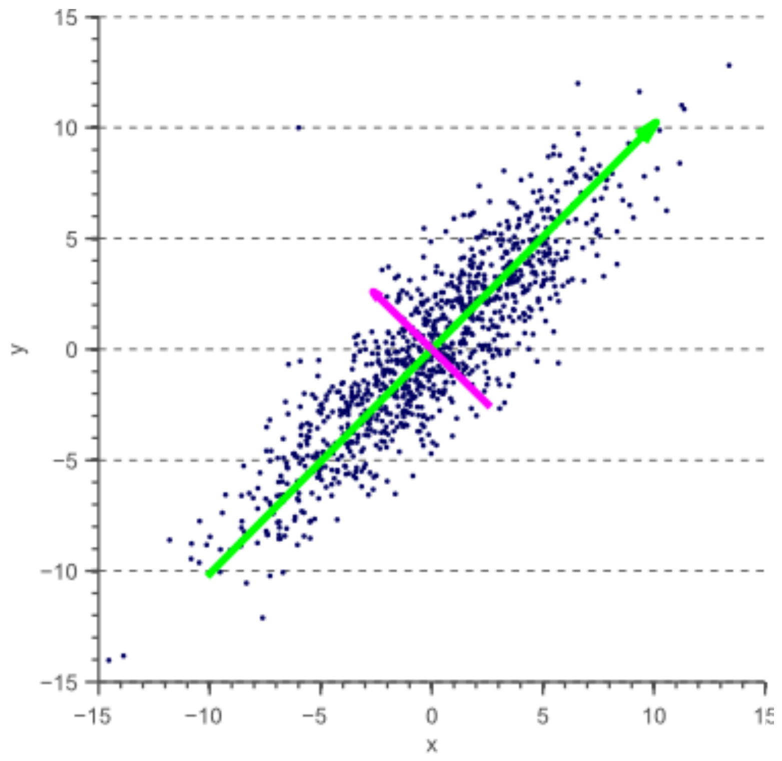
Introduction to Statistical Learning

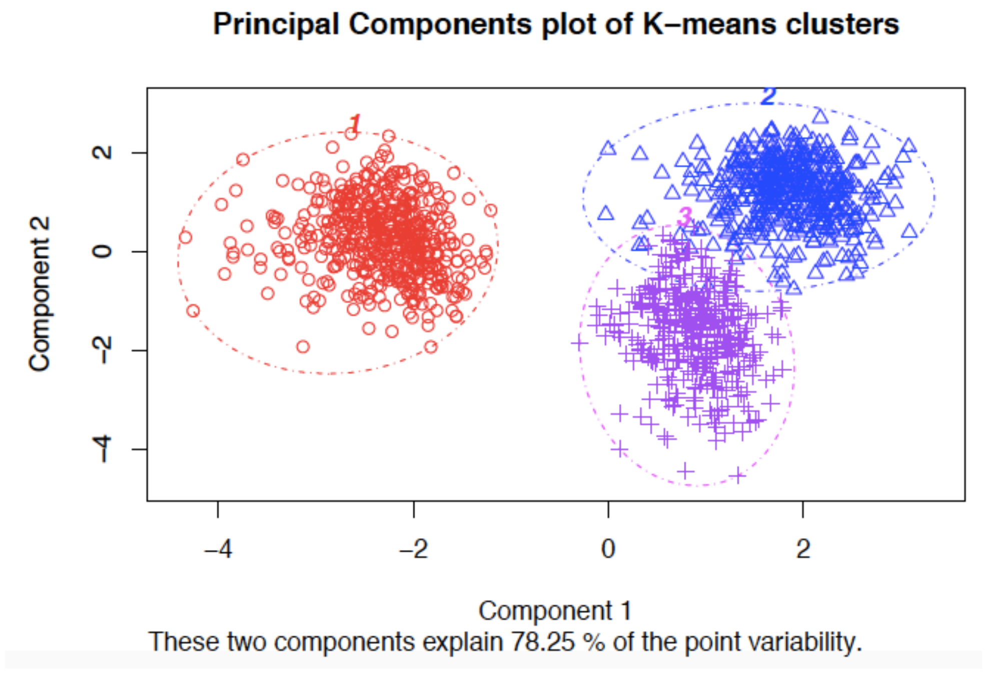
INF 552, Machine Learning for Data Informatics

University of Southern California

M. R. Rajati, PhD

Lesson 8
Unsupervised Learning





Unsupervised Learning

Unsupervised vs Supervised Learning:

- Most of this course focuses on *supervised learning* methods such as regression and classification.
- In that setting we observe both a set of features $X_1, X_2, \ldots, X_p$ for each object, as well as a response or outcome variable Y. The goal is then to predict Y using $X_1, X_2, \ldots, X_p$.

Unsupervised Learning

Unsupervised vs Supervised Learning:

- Here we instead focus on *unsupervised learning*, we where observe only the features X_1, X_2, . . . , X_p.
- We are not interested in prediction, because we do not have an associated response variable Y .

The Goals of 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 discuss two methods:
  - *principal components analysis*, a tool used for data visualization or data pre-processing before supervised techniques are applied, and
  - *clustering*, a broad class of methods for discovering unknown subgroups in data.

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.
- But techniques for unsupervised learning are of growing importance in a number of fields:
  - subgroups of breast cancer patients grouped by their gene expression measurements,
  - groups of shoppers characterized by their browsing and purchase histories,
  - movies grouped by the ratings assigned by movie viewers.

Another advantage

- 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 is difficult to automatically assess the overall sentiment of a movie review: is it favorable or not?

Clustering

- *Clustering* refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set.
- We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other.

Clustering

- We must define what it means for two or more observations to be *similar* or *different*. = dissimilarity
- Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied.

Clustering for Market Segmentation

- Suppose we have access to a large number of measurements (e.g. median household income, occupation, distance from nearest urban area, and so forth) for a large number of people.

Clustering for Market Segmentation

- Our goal is to perform *market segmentation* by identifying subgroups of people who might be more receptive to a particular form of advertising, or more likely to purchase a particular product.

Clustering for Market Segmentation

- The task of performing market segmentation amounts to clustering the people in the data set.

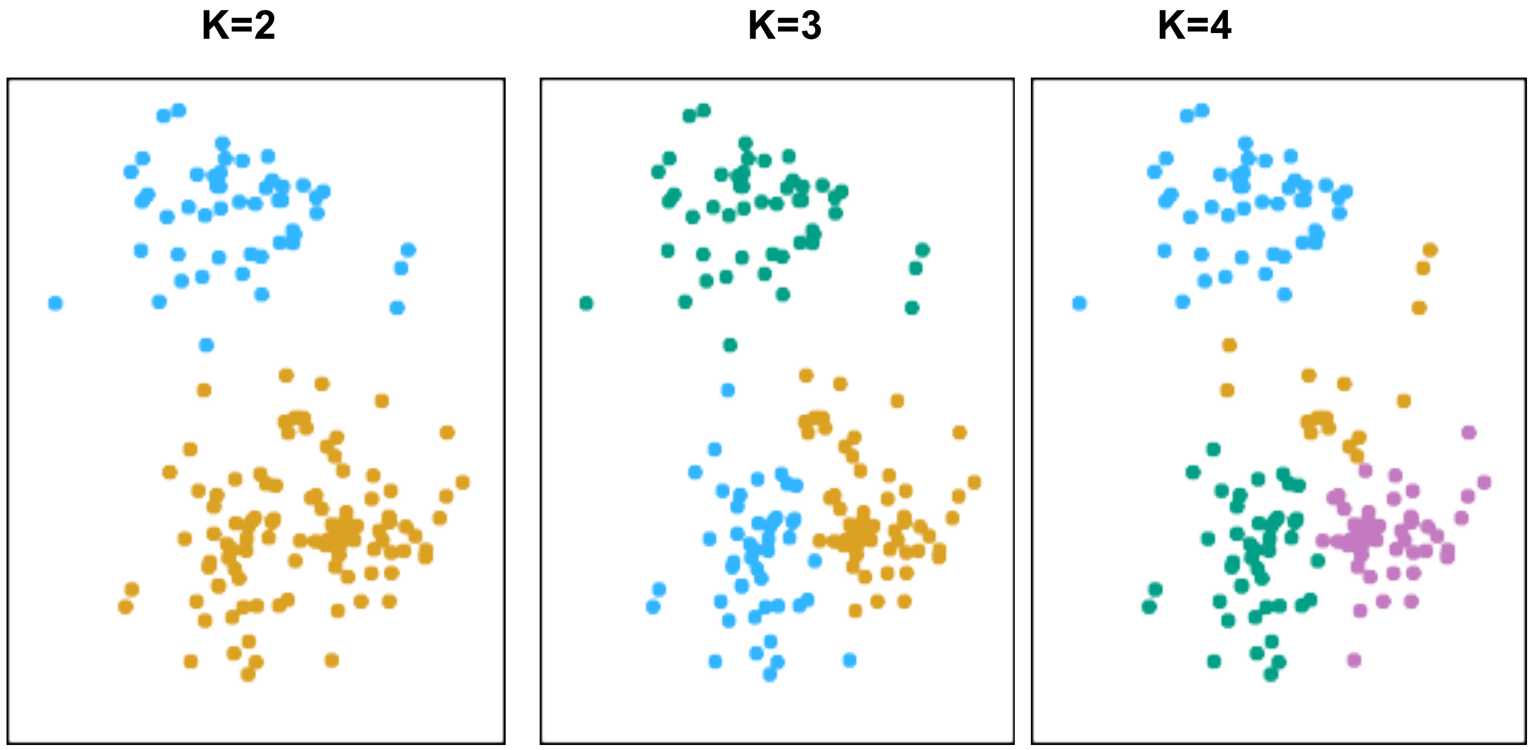
Two clustering methods

- In *K-means clustering*, we seek to partition the observations into a pre-specified number of clusters.

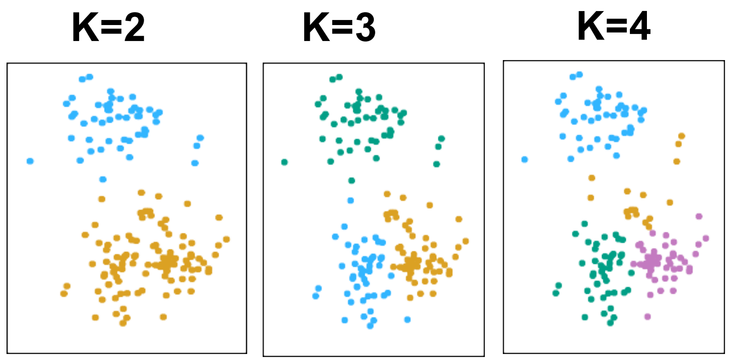
Two clustering methods

- 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 clustering



K-means clustering



A simulated data set with 150 observations in 2-dimensional space. Panels show the results of applying K-means clustering with different values of *K*, the number of clusters. The color of each observation indicates the cluster to which it was assigned using the K-means clustering algorithm. Note that there is no ordering of the clusters, so the cluster coloring is arbitrary. These cluster labels were not used in clustering; instead, they are the outputs of the clustering procedure.

Details of K-means clustering

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

1. $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.

2. $C_k \cap C_{k'} = \emptyset$ for all distinct k and k'. In other words, the clusters are non-overlapping: no observation belongs to more than one cluster.

For instance, if the ith observation is in the kth cluster, then $i \in C_k$.

Details of *K*-means clustering: 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.

Details of *K*-means clustering: continued

- The within-cluster variation for cluster C_k is a measure $\text{WCV}(C_k)$ of the amount by which the observations within a cluster differ from each other.
- Hence we want to solve the problem

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

Details of *K*-means clustering: continued

- 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.

How to define within-cluster variation?

- Typically we use Euclidean distance

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

where $|C_k|$ denotes the number of observations in the k^{th} cluster.

How to define within-cluster variation?

- Combining the above equations gives the optimization problem that defines K-means clustering.

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

K-Means Clustering Algorithm

1. Randomly assign a number, from 1 to *K*, to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
   1. For each of the *K* clusters, compute the cluster *centroid*.
      The k^{th} cluster centroid is the vector of the p feature means for the observations in the k^{th} cluster.
   2. Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).

Properties of the Algorithm

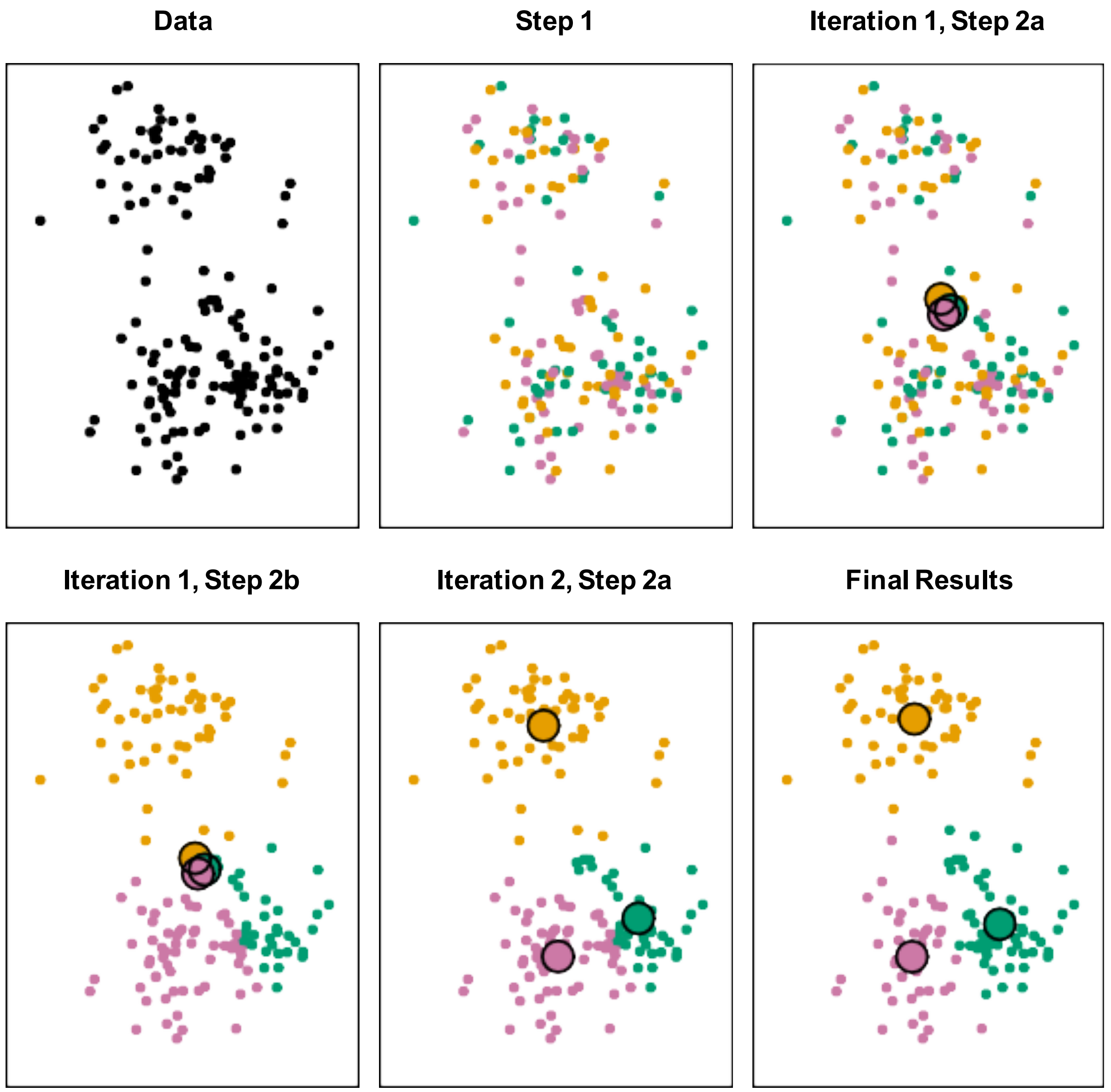
- This algorithm is guaranteed to decrease the value of the objective function at each step. *Why?* Note that

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 = 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2$$

  where $\bar{x}_{kj} = \frac{1}{|C_k|}\sum_{i\in C_k} x_{ij}$ is the mean for feature j in cluster C_k.

- however it is not guaranteed to give the global minimum. *Why not?*

Example


Data
Step 1
Iteration 1, Step 2a
Iteration 1, Step 2b
Iteration 2, Step 2a
Final Results

Details of Previous Figure

The progress of the K-means algorithm with $K = 3$.

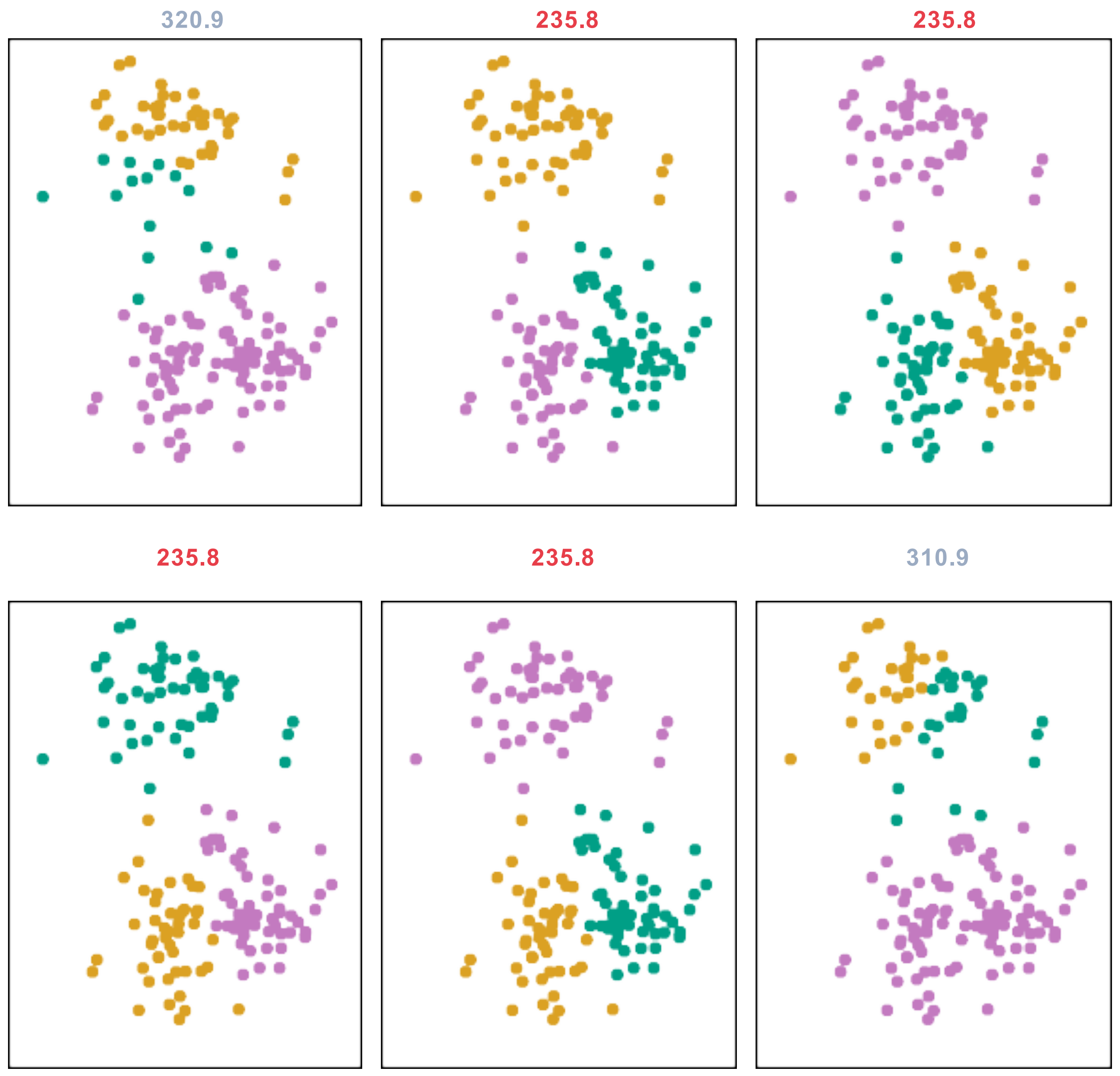
- *Top left:* The observations are shown.
- *Top center:* In Step 1 of the algorithm, each observation is randomly assigned to a cluster.
- *Top right:* In Step 2(a), the cluster centroids are computed. These are shown as large colored disks. Initially the centroids are almost completely overlapping because the initial cluster assignments were chosen at random.

Details of Previous Figure

The progress of the K-means algorithm with $K=3$.

- *Bottom left:* In Step 2(b), each observation is assigned to the nearest centroid.
- *Bottom center:* Step 2(a) is once again performed, leading to new cluster centroids.
- *Bottom right:* The results obtained after 10 iterations.

Example: different starting values



Details of Previous Figure

K-means clustering performed six times on the data from previous figure with *K* = 3, each time with a different random assignment of the observations in Step 1 of the *K*-means algorithm.

Details of Previous Figure

Above each plot is the value of the objective

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

Three different local optima were obtained, one of which resulted in a smaller value of the objective and provides better separation between the clusters.

Those labeled in red all achieved the same best solution, with an objective value of 235.8

Hierarchical Clustering

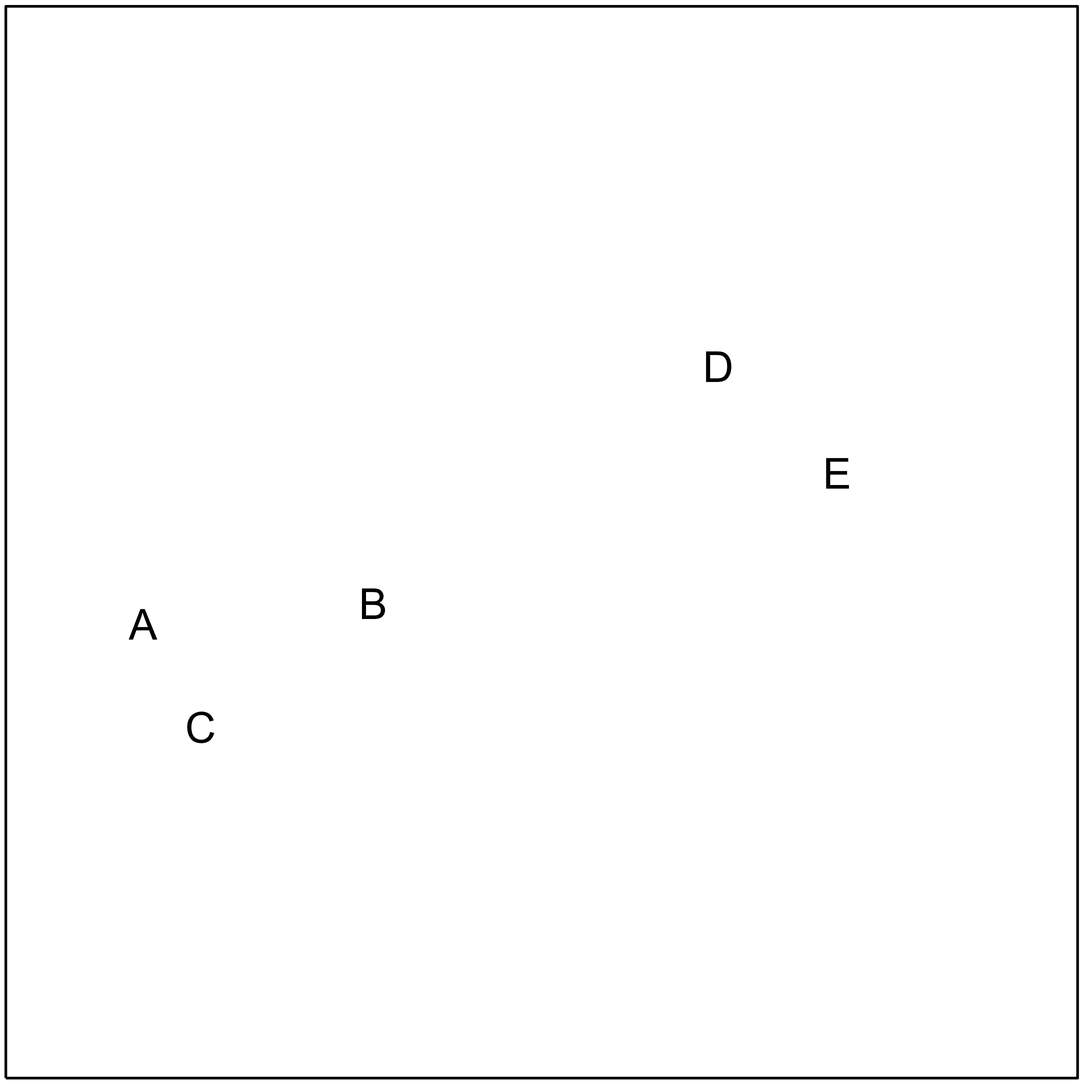
- *K*-means clustering requires us to pre-specify the number of clusters *K*. This can be a disadvantage (later we discuss strategies for choosing *K*)
- *Hierarchical clustering* is an alternative approach which does not require that we commit to a particular choice of *K*.

Hierarchical Clustering

- In this section, we describe *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.

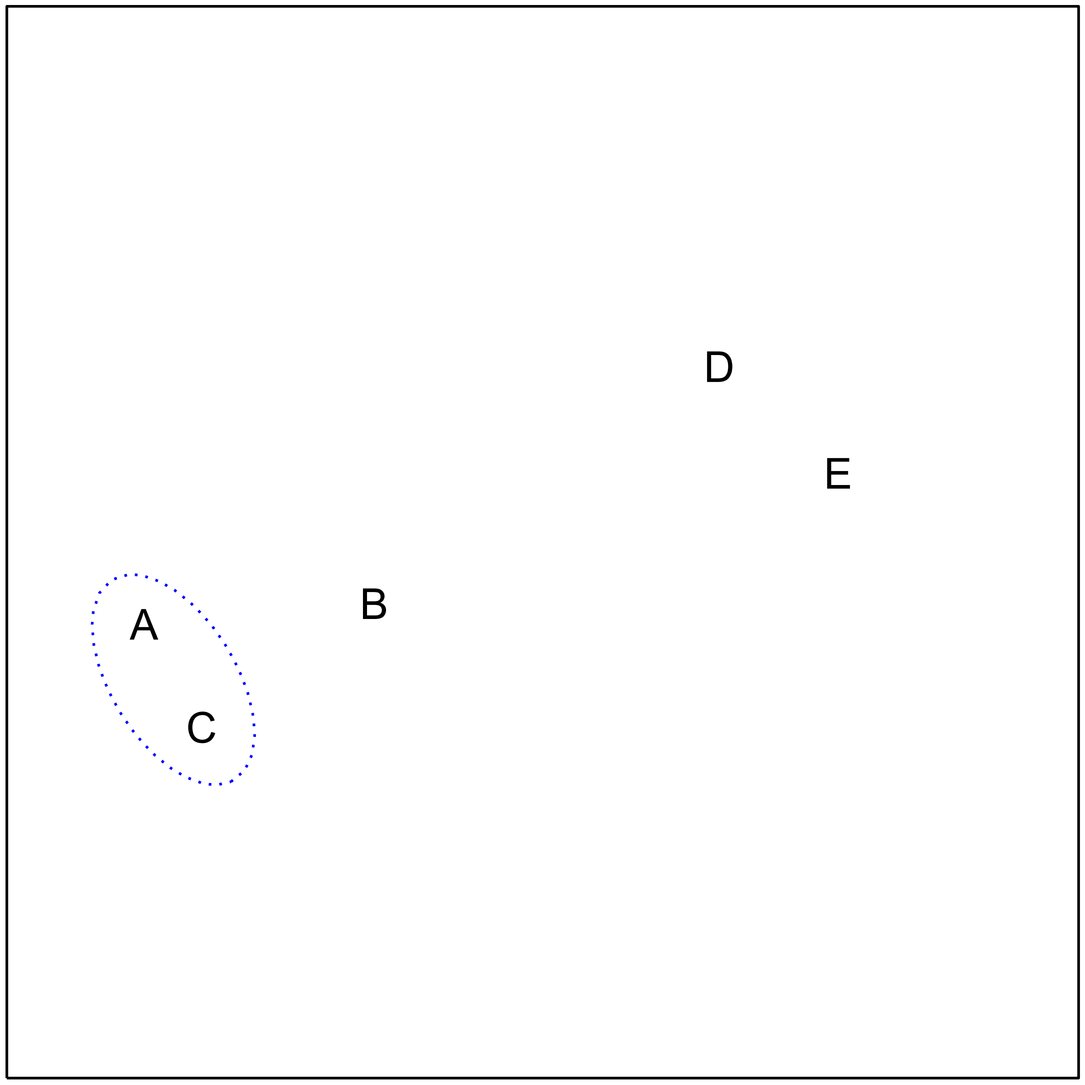
Hierarchical Clustering: the idea

Builds a hierarchy in a "bottom-up" fashion...



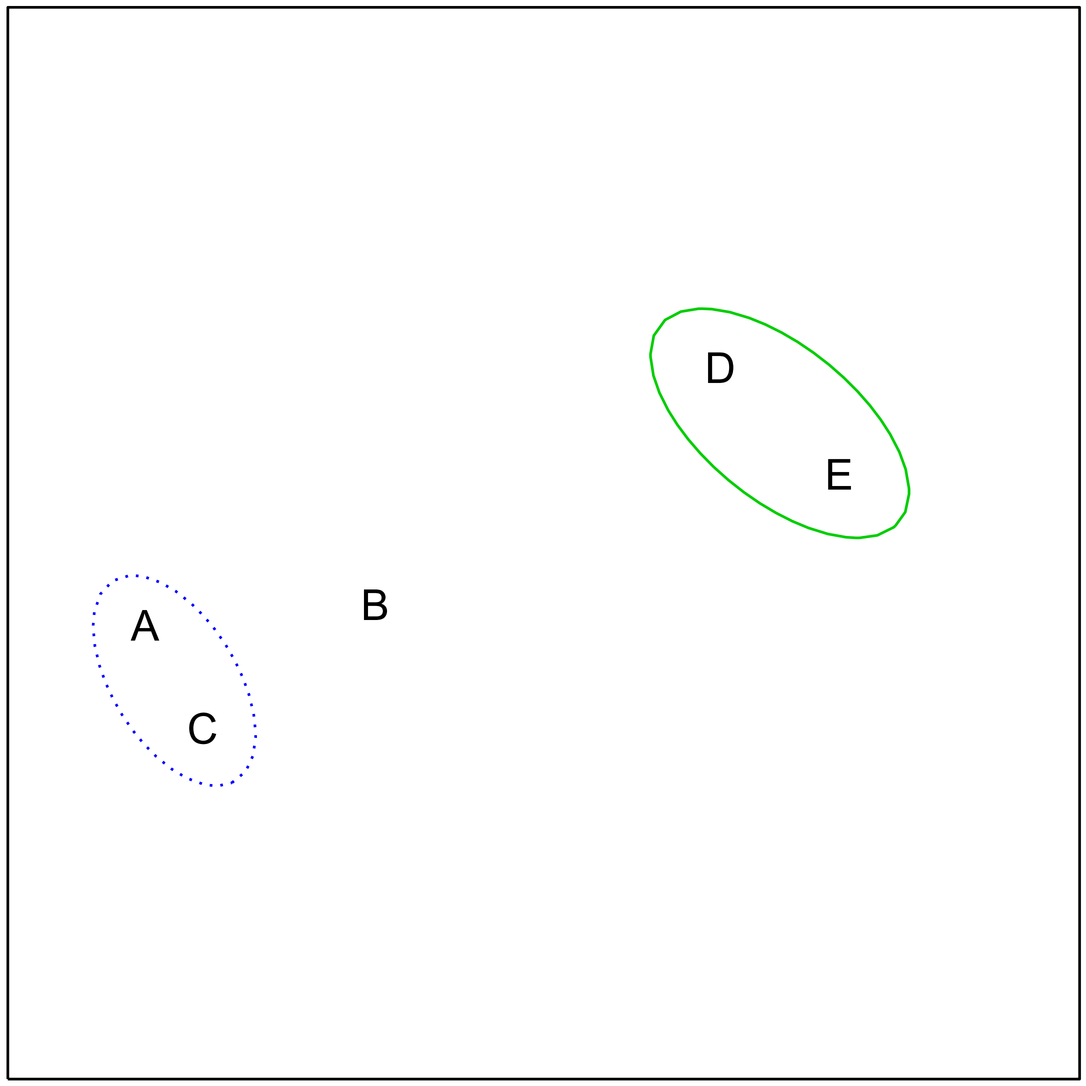
Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



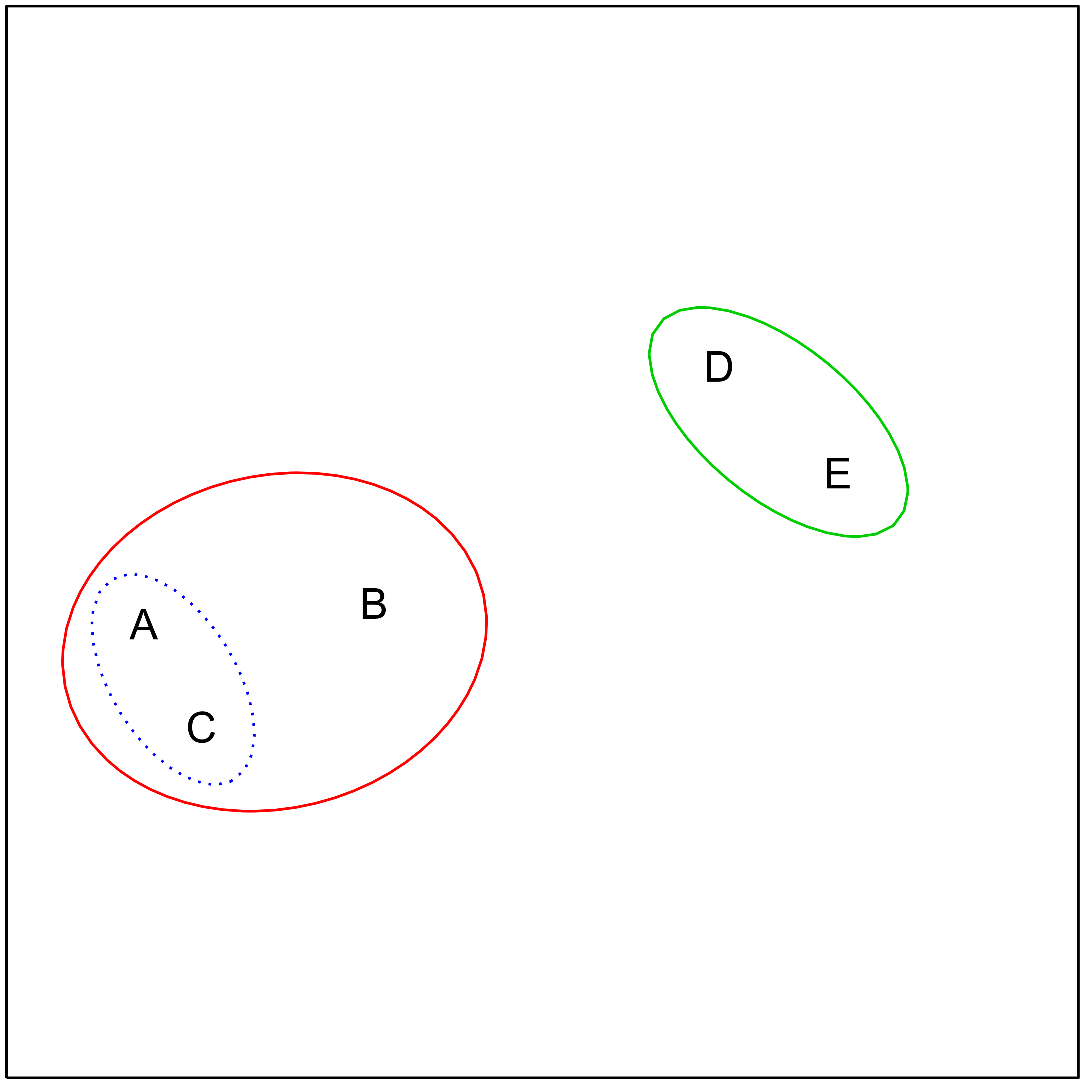
Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



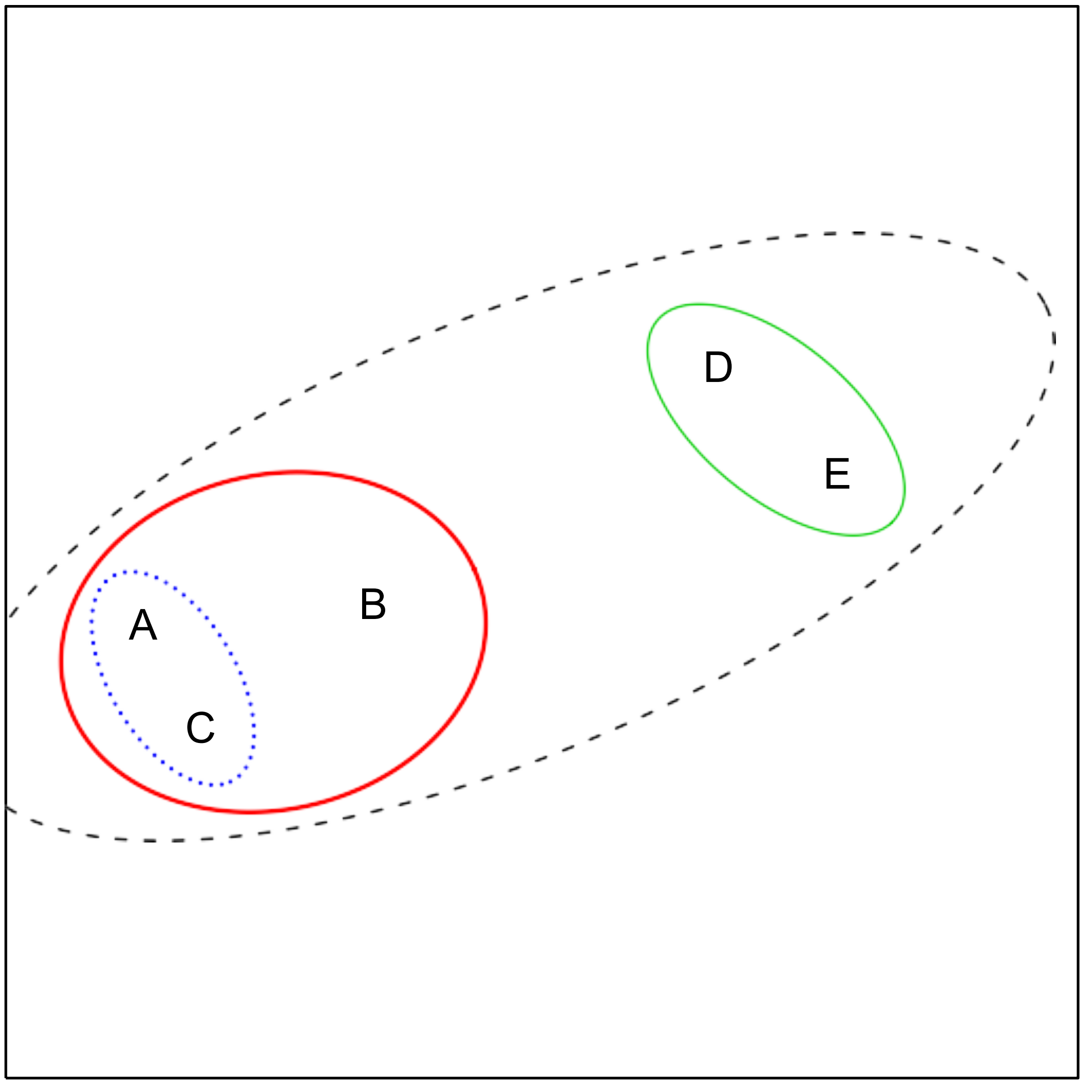
Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



Hierarchical Clustering: the idea

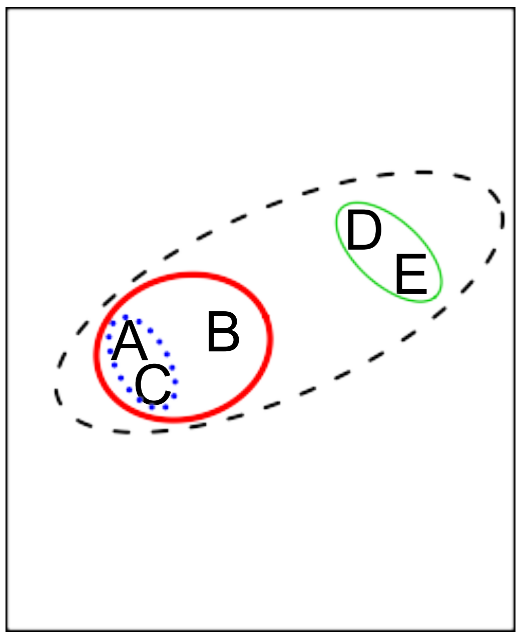
Builds a hierarchy in a “bottom-up” fashion...

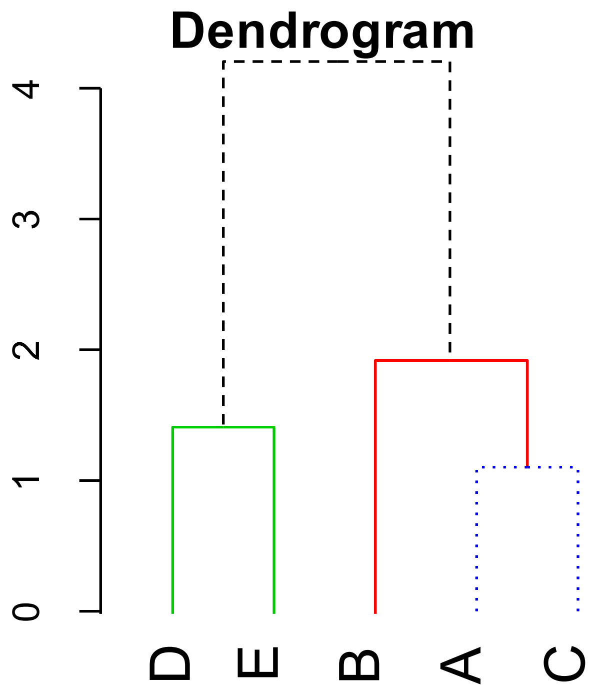


Hierarchical Clustering Algorithm

The approach in words:

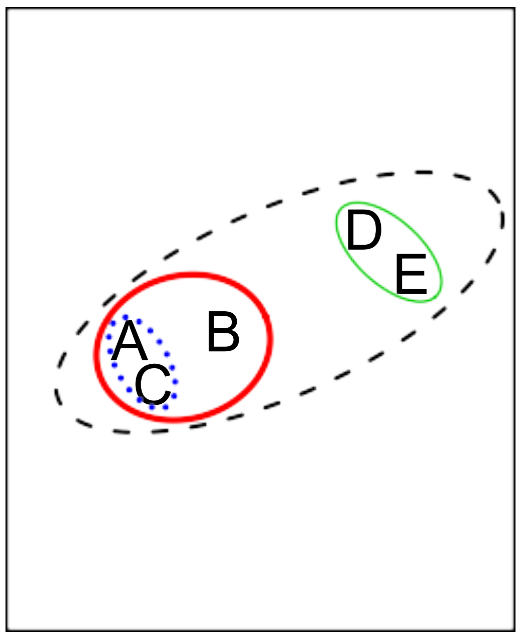
- Start with each point in its own cluster.
- Identify the closest two clusters and merge them.
- Repeat.
- Ends when all points are in a single cluster.

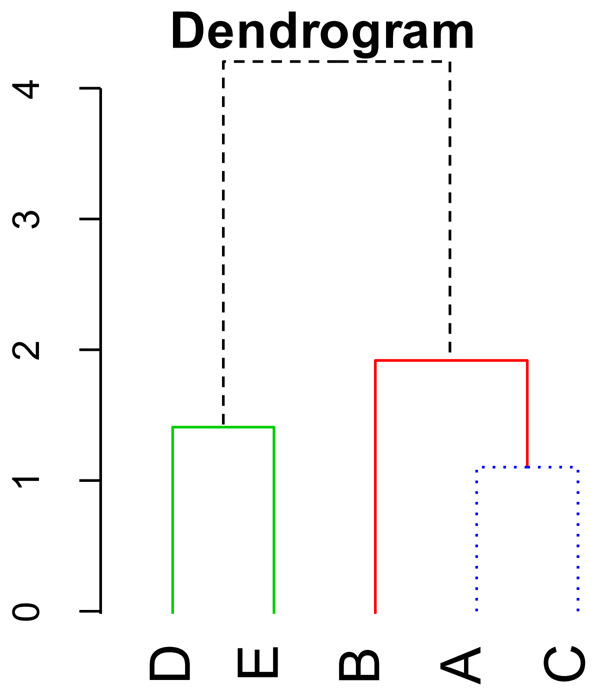




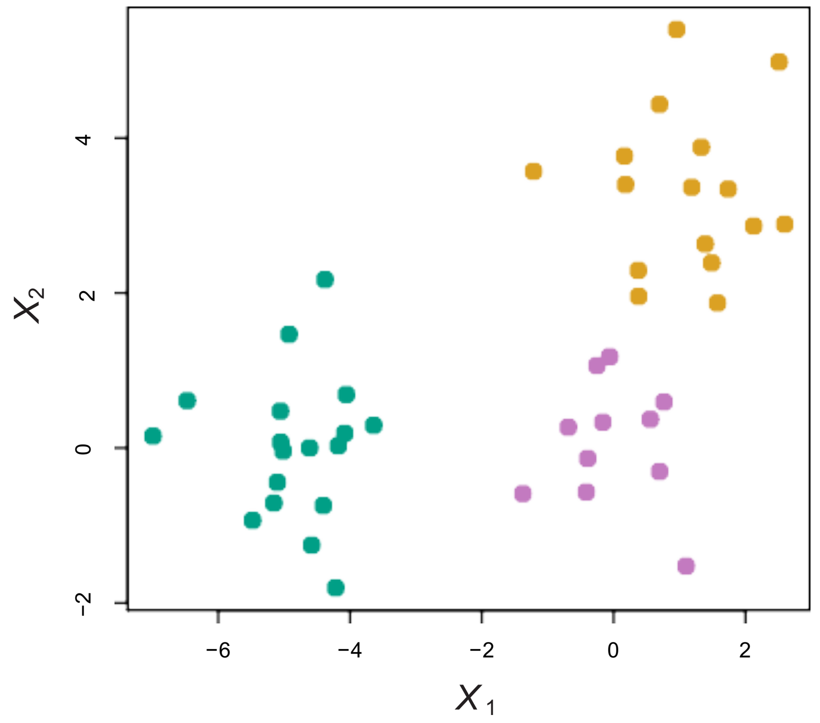
Hierarchical Clustering Algorithm

The height of each node in the plot is proportional to the value of the intergroup dissimilarity between its two daughters



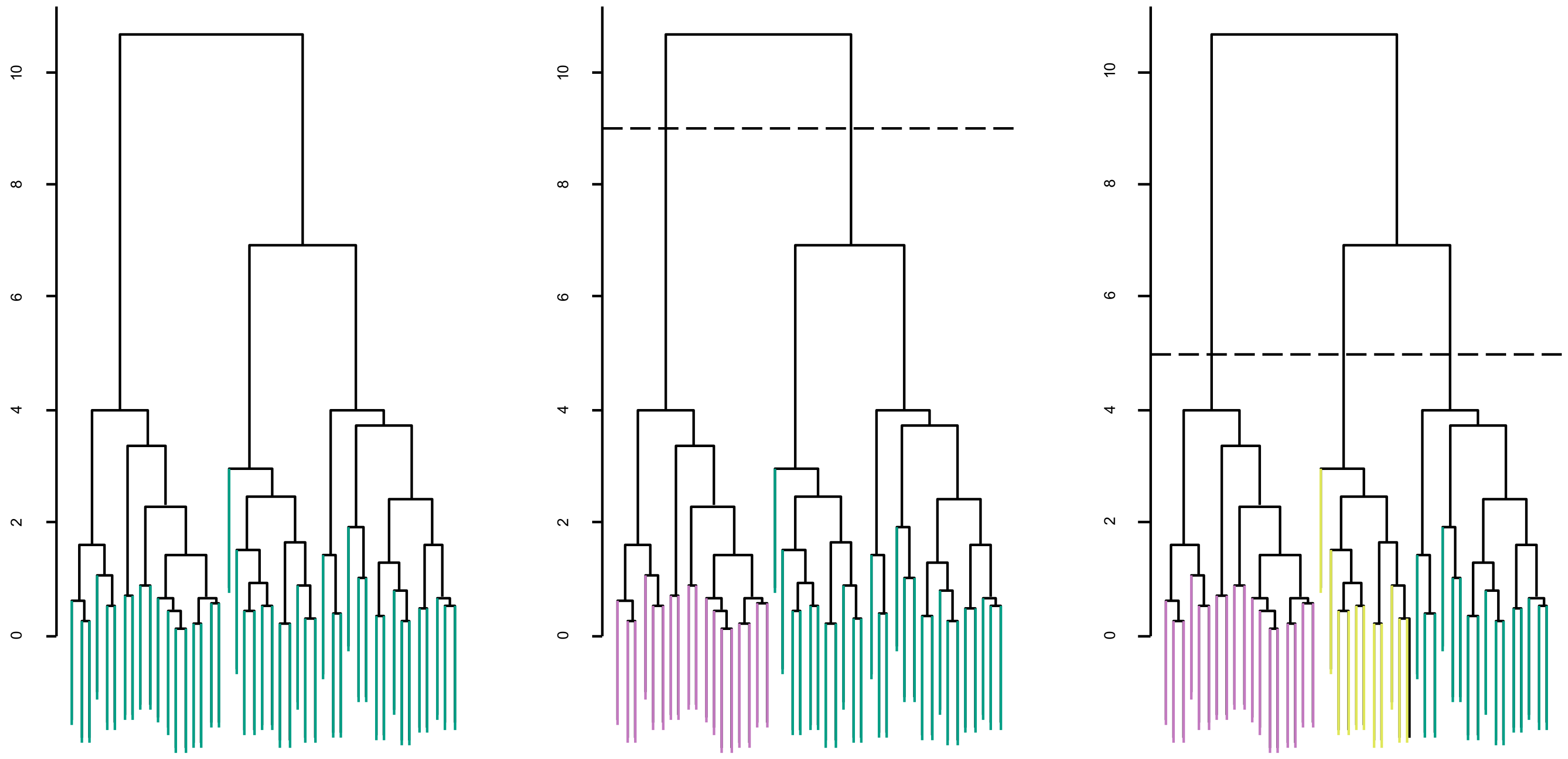


An Example



45 observations generated in 2-dimensional space. In reality there are three distinct classes, shown in separate colors.
However, we will treat these class labels as unknown and will seek to cluster the observations in order to discover the classes from the data.

Application of hierarchical clustering



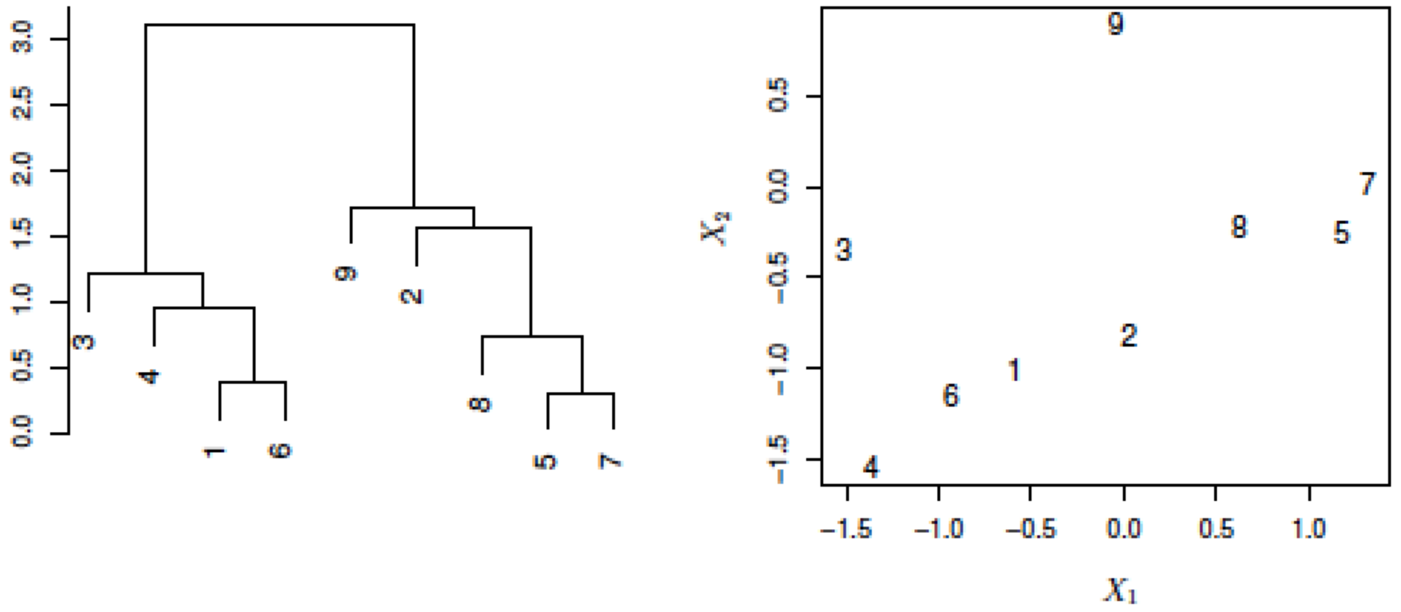
Details of previous figure

- *Left:* Dendrogram obtained from hierarchically clustering the data from previous slide, with complete linkage and Euclidean distance.
- *Center:* The dendrogram from the left-hand panel, cut at a height of 9 (indicated by the dashed line). This cut results in two distinct clusters, shown in different colors.

Details of previous figure

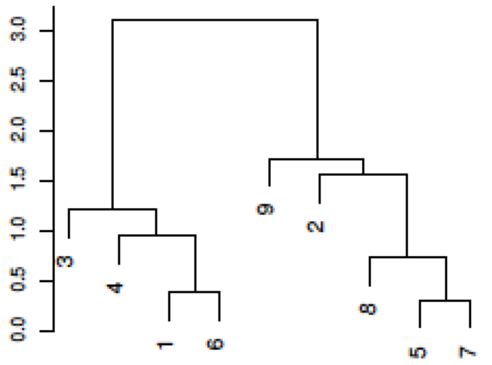
- *Right:* The dendrogram from the left-hand panel, now cut at a height of 5. This cut results in three distinct clusters, shown in different colors.
- Note that the colors were not used in clustering, but are simply used for display purposes in this figure

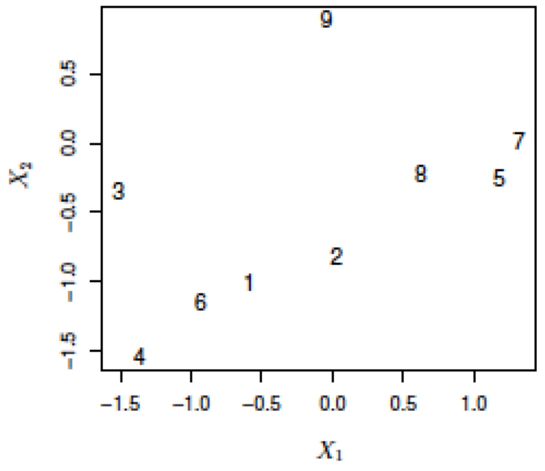
Another Example



- An illustration of how to properly interpret a dendrogram with nine observations in two-dimensional space. The raw data on the right was used to generate the dendrogram on the left.
- Observations 5 and 7 are quite similar to each other, as are observations 1 and 6.

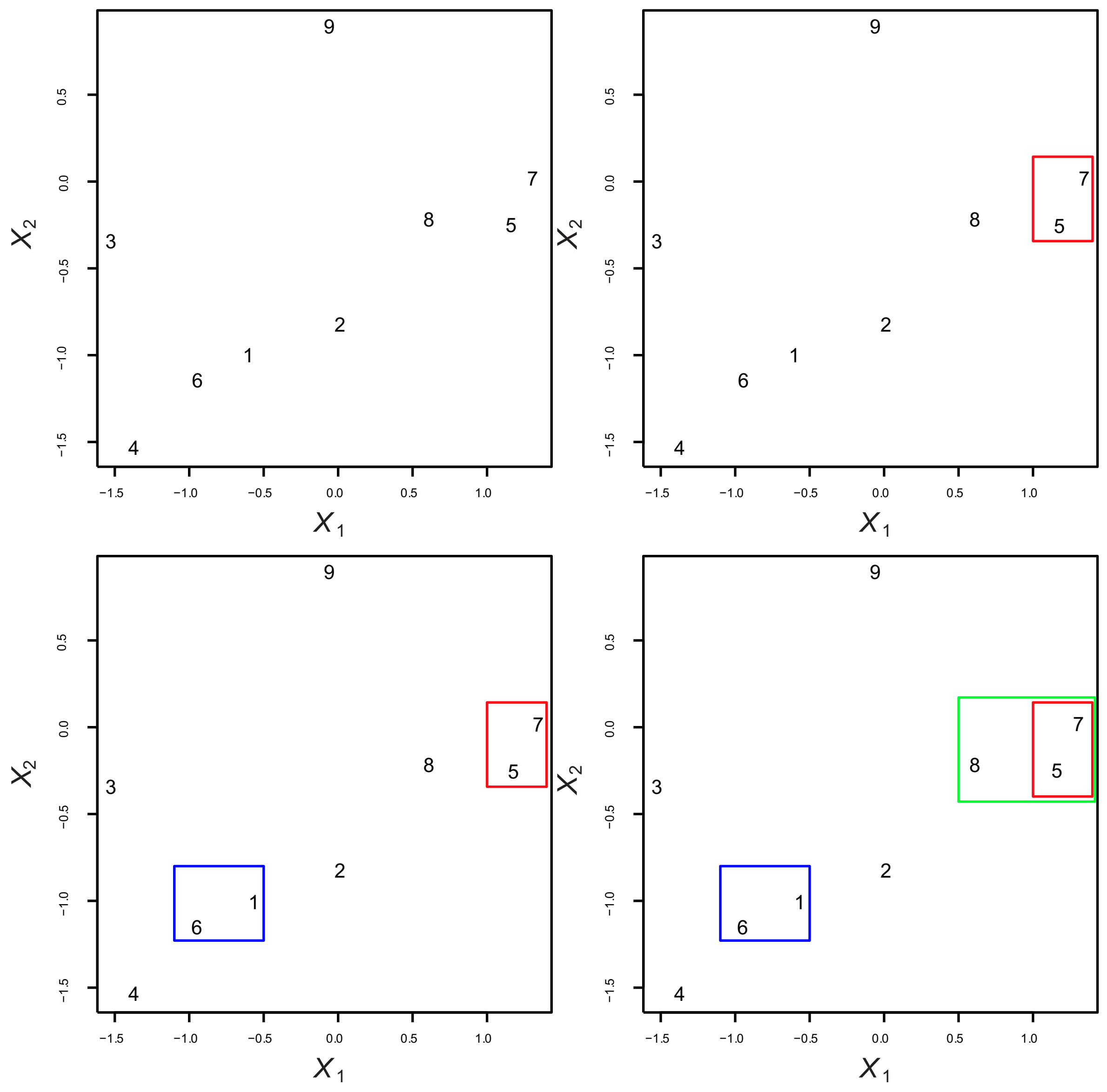
Another Example



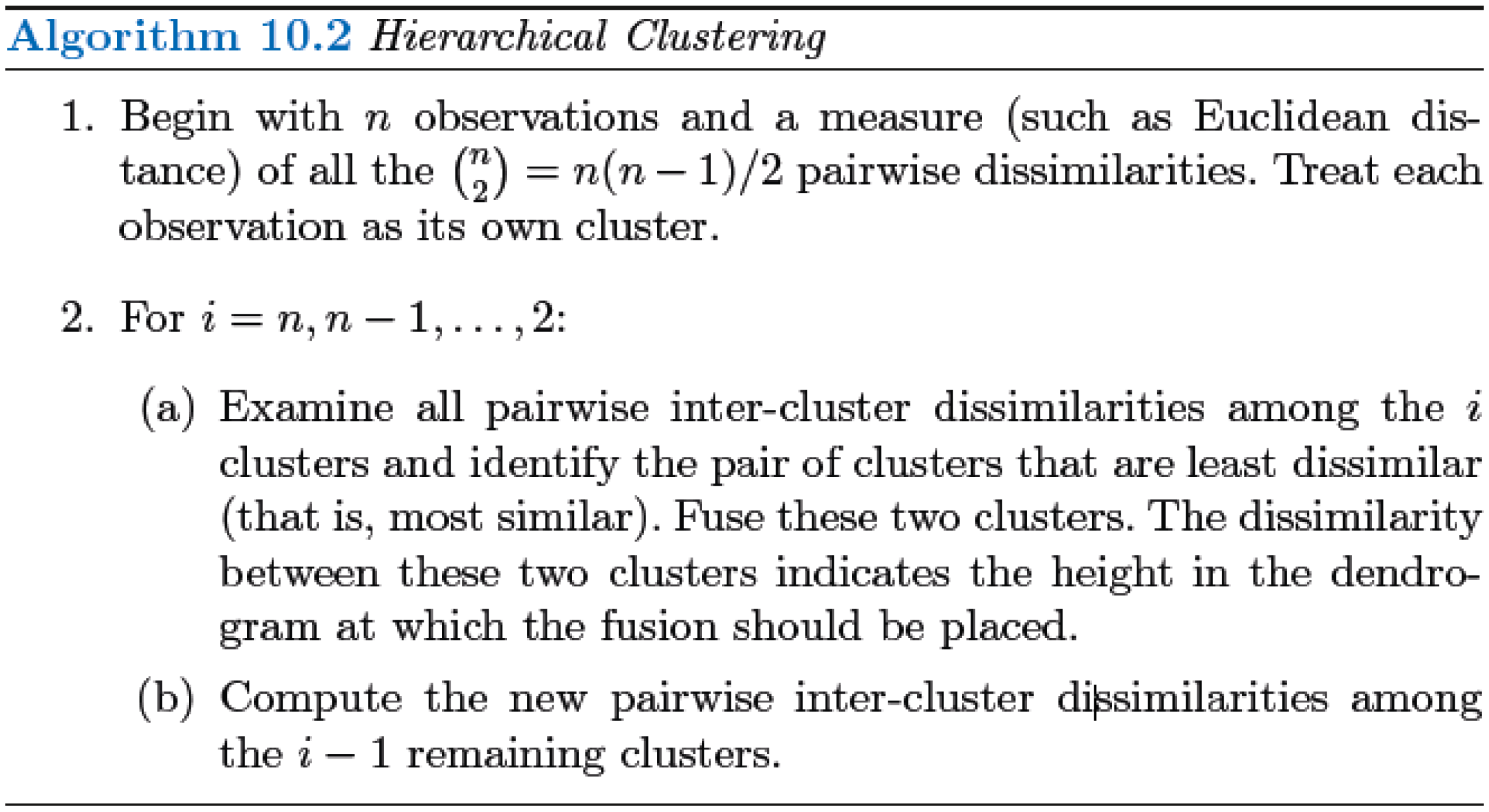


- However, observation 9 is *no more similar to* observation 2 than it is to observations 8, 5, and 7, even though observations 9 and 2 are close together in terms of horizontal distance.
- This is because observations 2, 8, 5, and 7 all fuse with observation 9 at the same height, approximately 1.8.

Merges in previous example



Algorithm



Algorithm 10.2 *Hierarchical Clustering*

1. Begin with n observations and a measure (such as Euclidean distance) of all the $\binom{n}{2} = n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.

2. For $i = n, n-1, \ldots, 2$:

   (a) Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.

   (b) Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

Nested Clusters

- The term hierarchical refers to the fact that clusters obtained by cutting the dendrogram at a given height are necessarily nested within the clusters obtained by cutting the dendrogram at any greater height.
- However, on an arbitrary data set, this assumption of hierarchical structure might be unrealistic
- Hierarchical clustering can sometimes yield worse (i.e. less accurate) results than K - means clustering for a given number of clusters

Dissimilarity between Groups

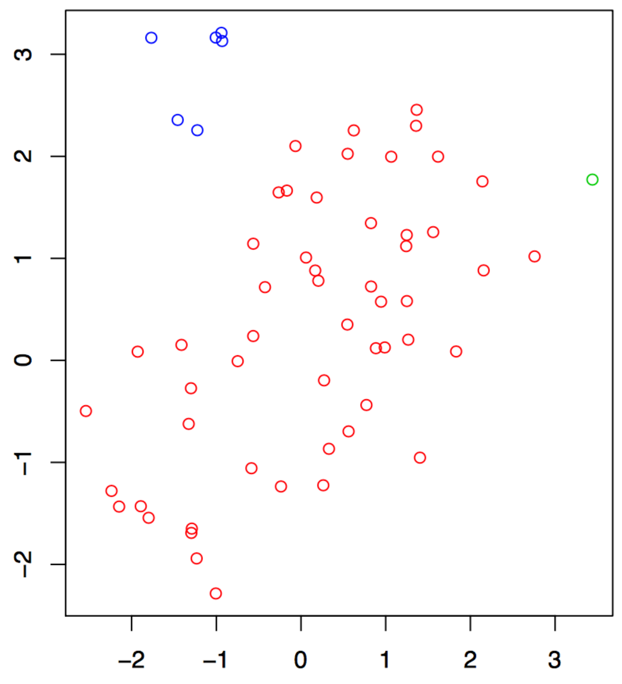
- The concept of dissimilarity between a pair of observations needs to be extended to a pair of groups of observations .
- This extension is achieved by developing the notion of linkage, which defines the dissimilarity between two groups of observations. The four most common types of linkage—complete, average, single, and centroid

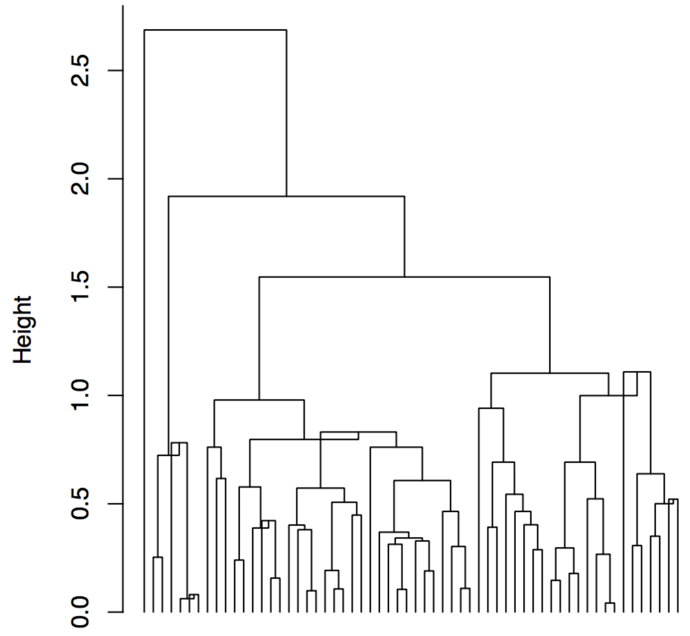
Types of Linkage

Linkage	*Description*
Complete	Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities.
Average	Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

Types of Linkage

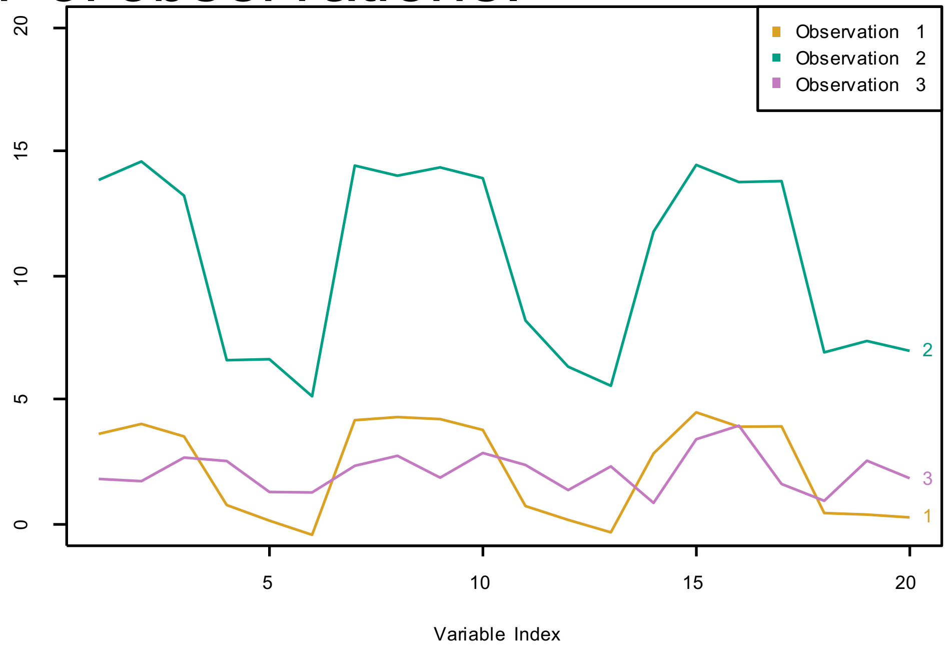
- For centroid dissimilarity, inversion can occur, whereby two clusters are fused at a height below either of the individual clusters in the dendrogram. This can lead to difficulties in visualization as well as in interpretation of the dendrogram.





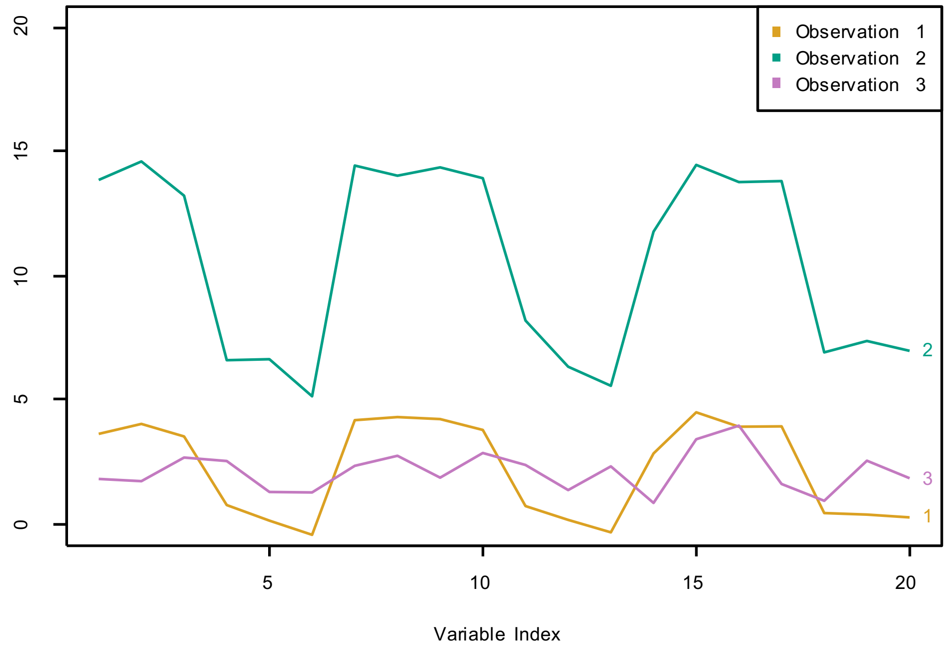
Choice of Dissimilarity Measure

- 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.



Choice of Dissimilarity Measure

- Correlation-based distance focuses on the shapes of observation profiles rather than their magnitudes.
- Observations 1, 3 have small Euclidean distance but weak correlation (low correlation similarity).
- Observations 1, 2 have large Euclidean distance but strong correlation (high correlation similarity).



Practical issues

- Should the observations or features first be standardized in some way? For instance, maybe the variables should be centered to have mean zero and scaled to have standard deviation one.
- In the case of hierarchical clustering,
  - What dissimilarity measure should be used?
  - What type of linkage should be used?
- How many clusters to choose? (in both *K*-means or hierarchical clustering). Difficult problem. No agreed-upon method. See Elements of Statistical Learning, chapter 13 for more details.

How many clusters?

- Sometimes, we might have no problem specifying the number of clusters K ahead of time, e.g.,
  - Segmenting a client database into K clusters for *K* salesmen
- Other times, K is implicitly defined by cutting a hierarchical clustering tree at a given height
- In most exploratory applications, *K* is unknown.
- What is the “right” value of K?

This is a hard problem

Determining the number of clusters is a hard problem!

Why is it hard?

- Determining the number of clusters is a hard task for humans to perform (unless the data are low-dimensional). Not only that, it's just as hard to explain what it is we're looking for.

  Usually, statistical learning is successful when at least one of these is possible

This is a hard problem

- Why is it important?
  - E.g., it might mean a big difference scientifically if we were convinced that there were K = 2 subtypes of breast cancer vs. K = 3 subtypes
  - One of the (larger) goals of data mining/statistical learning is automatic inference; choosing K is certainly part of this.

Reminder: within-cluster variation

We focus on K-means, but most ideas apply to other settings

Recall: given the number of clusters *K*, the *K*-means algorithm approximately minimizes the within-cluster variation:

$$W = \sum_{k=1}^{K} \sum_{C(i)=k} \|X_i - \bar{X}_k\|_2^2$$

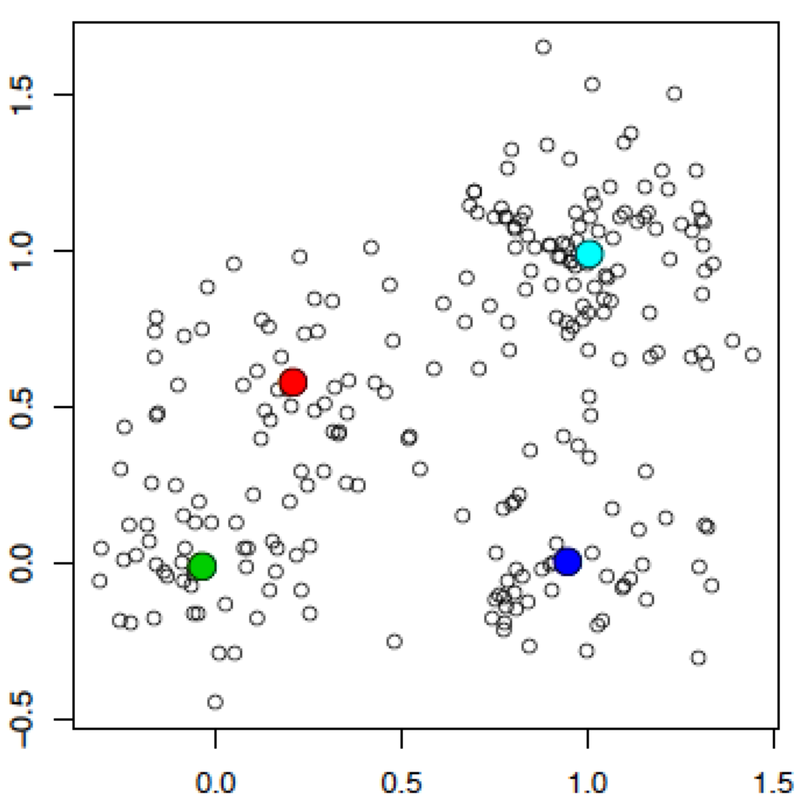
over clustering assignments *C*, where $\overline{X}_k$ is the average of points in group *k*

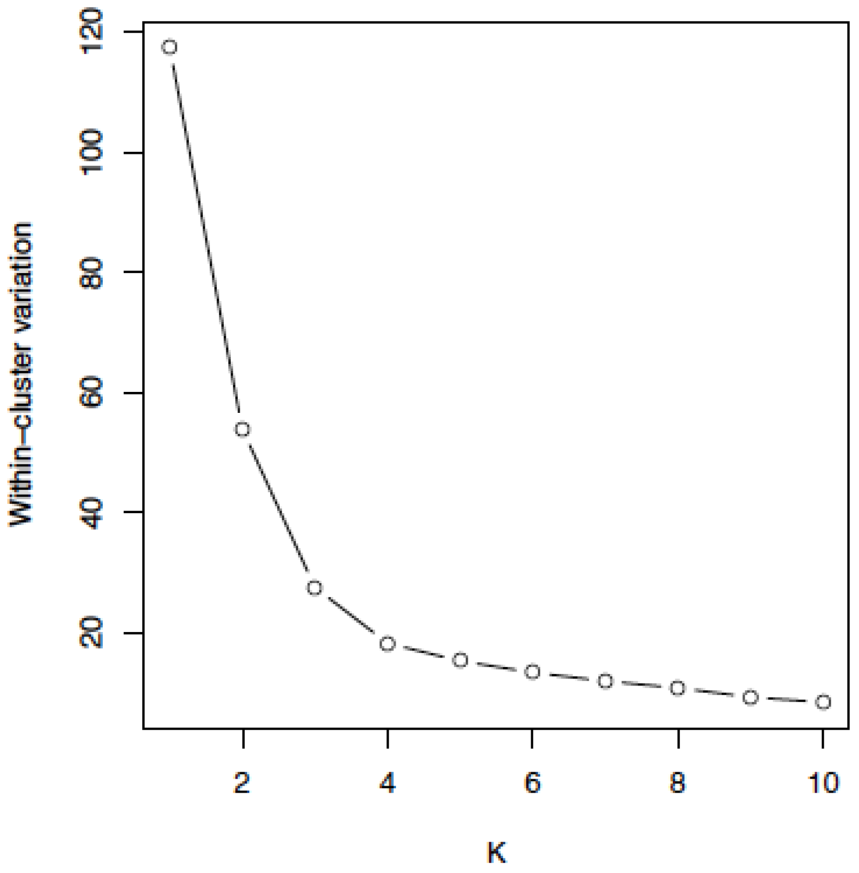
Clearly a lower value of W is better. So why not just run *K*-means for a bunch of different values of K, and choose the value of K that gives the smallest *W* (*K*)?

That’s not going to work

Problem: within-cluster variation just keeps decreasing: scree plot

Example: $n = 250$, $p = 2$, $K = 1, \ldots 10$





Between-cluster variation

Within-cluster variation measures how tightly grouped the clusters are. As we increase the number of clusters *K*, this just keeps going down. What are we missing?

Between-cluster variation measures how spread apart the groups are from each other:

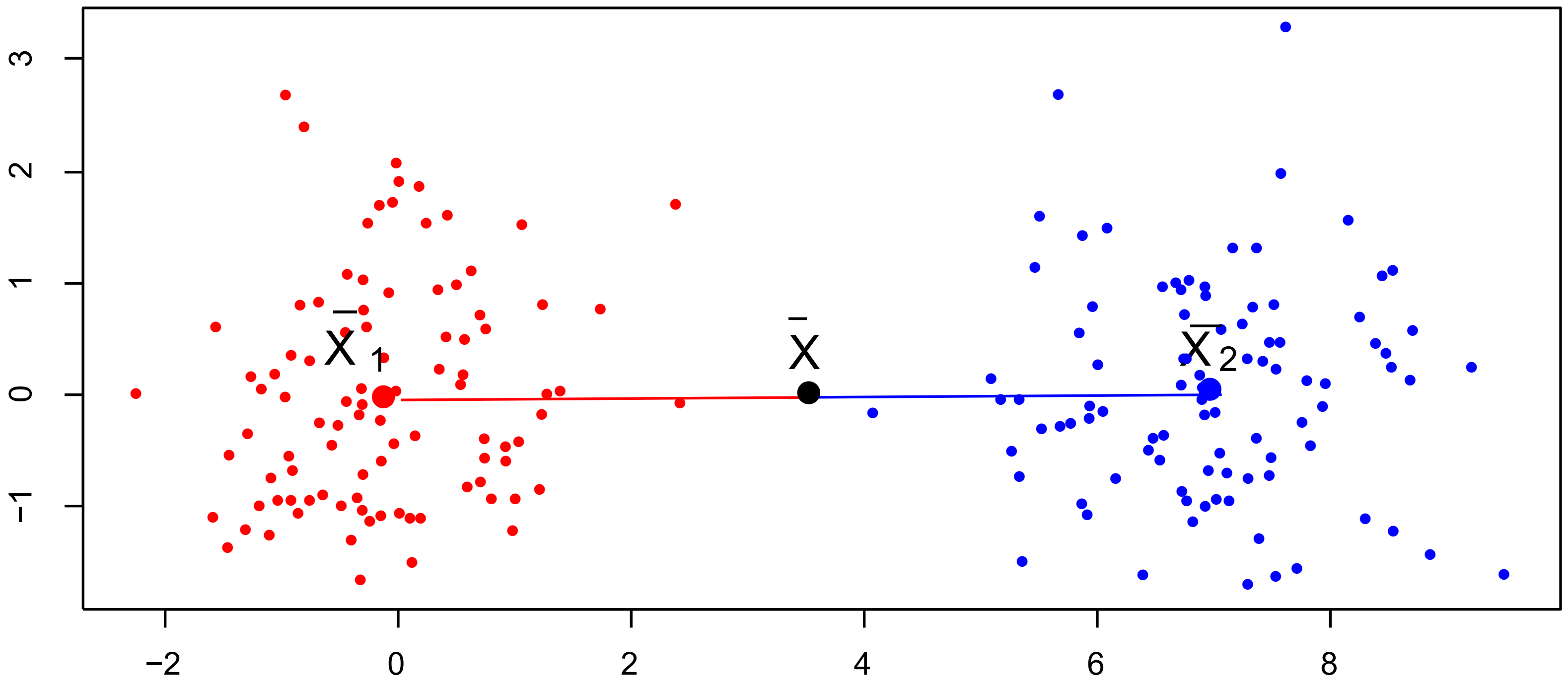
$$B = \sum_{k=1}^{K} n_K \|\bar{X}_k - \bar{X}\|_2^2$$

where as before X^{-}_{k} is the average of points in group k, and X^{-} is the overall average, i.e.

$$\bar{X}_k = \frac{1}{n_k} \sum_{C(i)=k} X_i \quad \text{and} \quad \bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

Example: between-cluster variation

Example: $n = 100, p = 2, K = 2$



$$B = n_1 \|\bar{X}_1 - \bar{X}\|_2^2 + n_2 \|\bar{X}_2 - \bar{X}\|_2^2$$

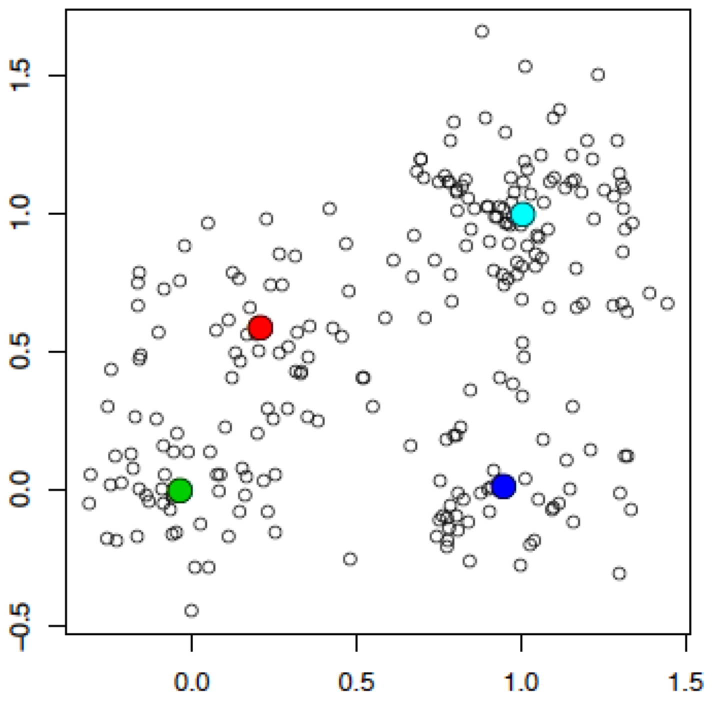
$$W = \sum_{C(i)=1} \|X_i - \bar{X}_1\|_2^2 + \sum_{C(i)=2} \|X_i - \bar{X}_2\|_2^2$$

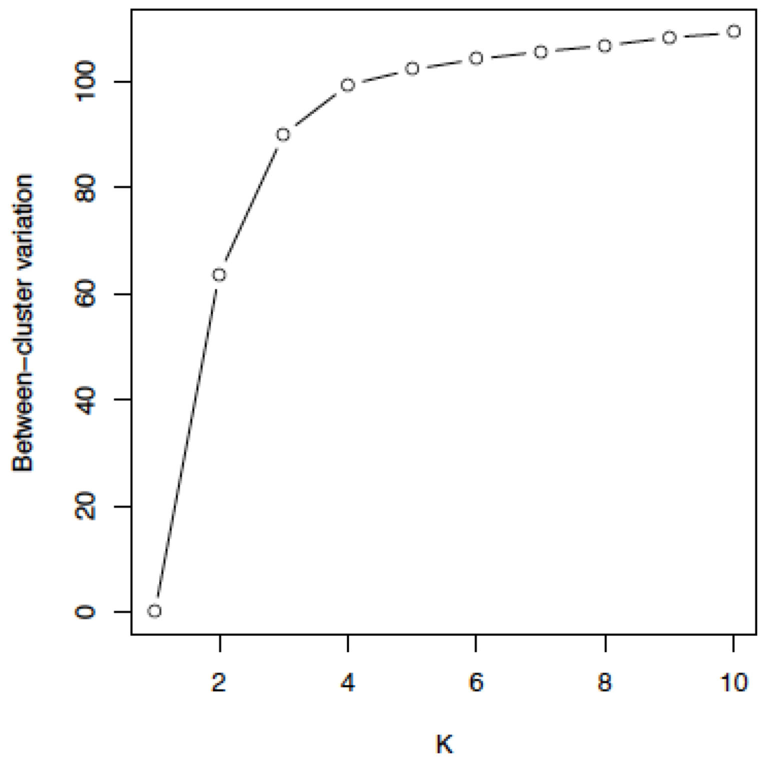
Still not going to work

Bigger B is better, can we use it to choose K?

Problem: between- cluster variation just keeps increasing

Running example: n = 250, p = 2, K = 1, . . . 10





CH index

- Ideally we'd like our clustering assignments *C* to simultaneously have a small W and a large *B*

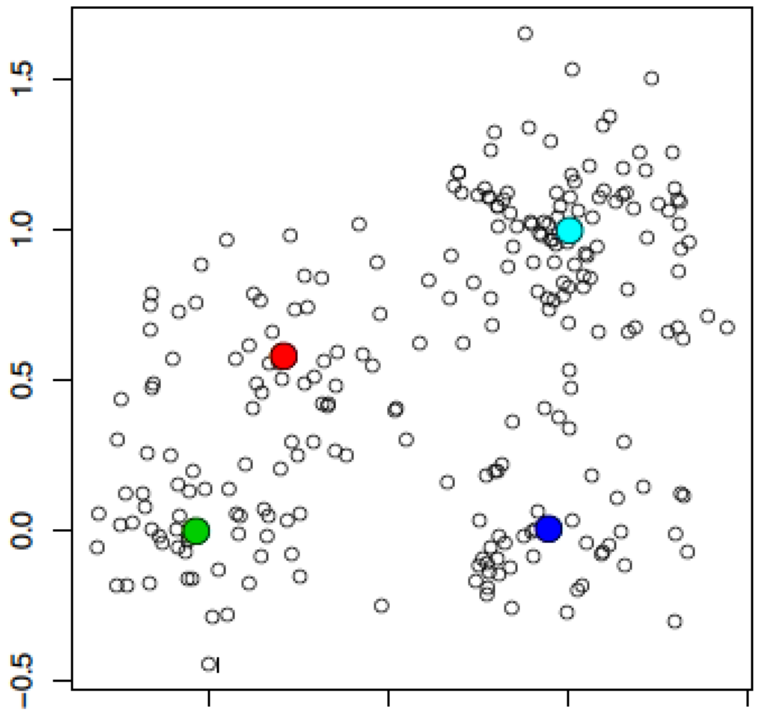
- This is the idea behind the CH index proposed by Calinski and Harabasz (1974), in "A dendrite method for cluster analysis"
- For clustering assignments coming from K clusters, we record CH score:

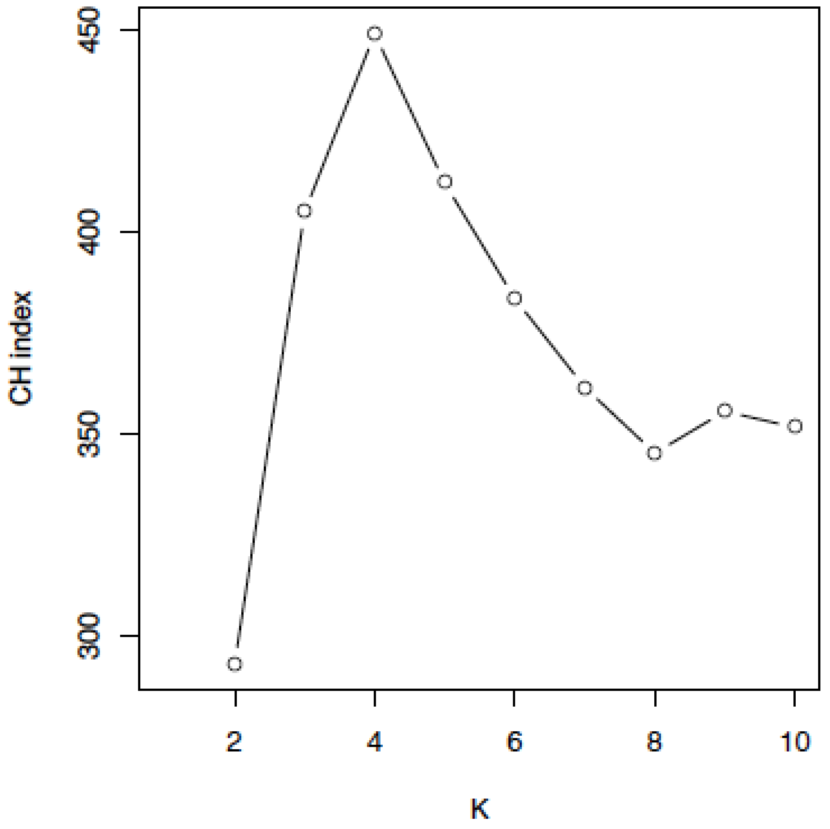
$$\mathrm{CH}(K) = \frac{B(K)/(K-1)}{W(K)/(n-K)}$$

- To choose *K*, just pick some maximum number of clusters to be considered $K_{\max}$ (e.g., $K = 20$), and choose the value of *K* with the largest score CH(*K*).

Example: CH index

Running example: $n = 250$, $p = 2$, $K = 2, \ldots 10$.





We would choose K = 4 clusters, which seems reasonable

General problem: the CH index is not defined for K = 1. We could never choose just one cluster (the null model)!

Gap statistic

It’s true that W(K) keeps dropping, but how much it drops at any one *K* should be informative

The gap statistic was introduced by Tibshirani et al. (2001), “Estimating the number of clusters in a data set via the gap statistic”

Gap statistic

It is based on this idea. We compare the observed within-cluster variation $W(K)$ to $W_{\text{unif}}(K)$, the within-cluster variation we'd see if we instead had points distributed uniformly (over an encapsulating box). The gap for K clusters is defined as

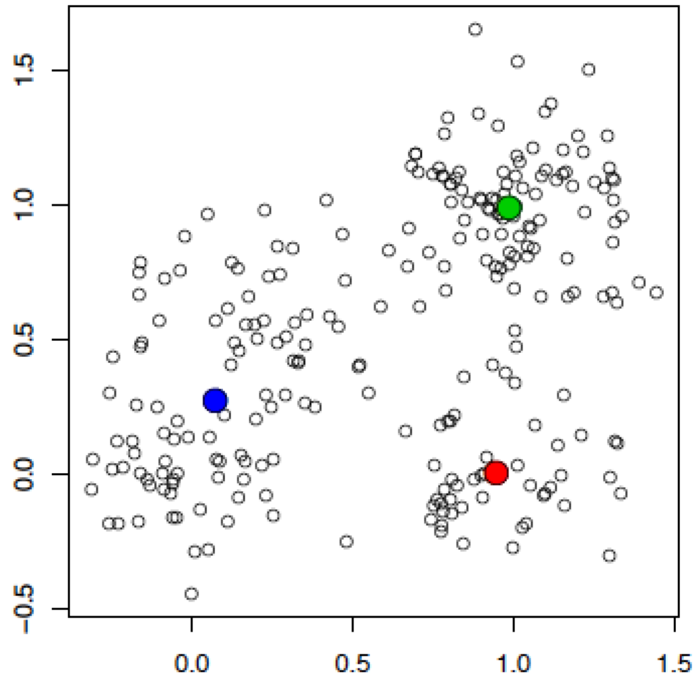
$$\text{Gap}(K) = \log W_{\text{unif}}(K) - \log W(K)$$

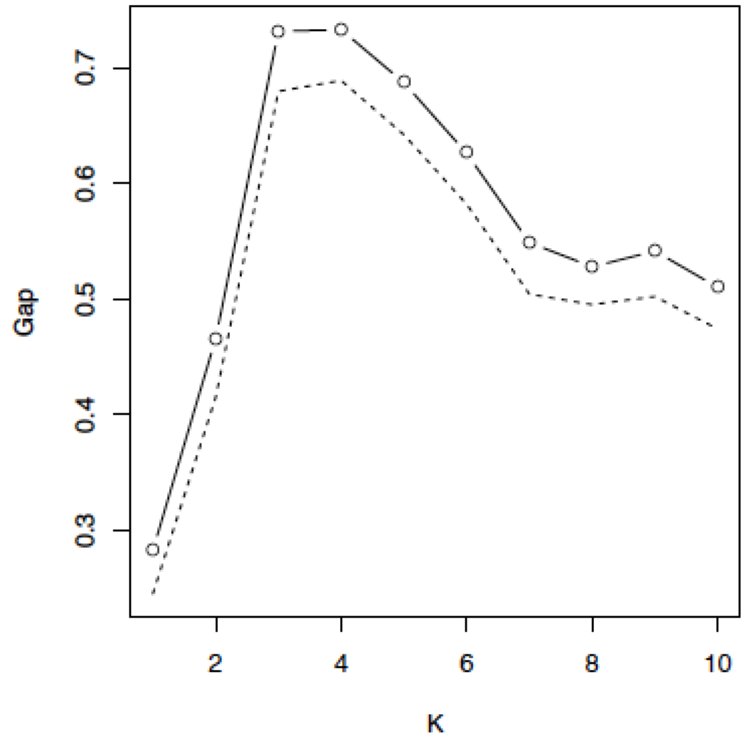
Gap statistic

The quantity log $W_{\mathrm{unif}}(K)$ is computed by simulation: we average the log within-cluster variation over, say, 20 simulated uniform data sets. We also compute the standard error of $s(K)$ of log Wunif(K) over the simulations. Then we choose K by K ˆ = min ,K $\in$ {1, . . . Kmax} : Gap(K) ≥ Gap(K +1) − s(K +1),

Example: gap statistic

Running example: $n = 250, p = 2, K = 1, \ldots 10$





We would choose K = 3 clusters, which is also reasonable
The gap statistic does especially well when the data fall into one cluster. (Why? Hint: think about the null distribution that it uses)

CH index and gap statistic in R

The CH index can be computed using the kmeans function in the base distribution, which returns both the within-cluster variation and the between-cluster varation

E.g.,

```
k = 5
km = kmeans(x, k, alg="Lloyd")
names(km)
# Now use some of these return items to compute ch
```

The gap statistic is implemented by the function gap in the package lga, and by the function gap in the package SAGx. (Beware: these functions are poorly documented ... it’s unclear what clustering method they’re using)

Silhouette Analysis

- A method of interpretation and validation of consistency within clusters of data
- Provides a succinct graphical representation of how well each object lies within its cluster

Silhouette Analysis

- Assume the data have been clustered via any technique, such as k-means, into *k* clusters.
- For each sample $\mathbf{x}_i$, let a_i be the average distance between $\mathbf{x}_\mathrm{i}$ and all other data within the same cluster.
- Can interpret a_i as a measure of how well $\mathbf{x}_i$ is assigned to its cluster (the smaller the value, the better the assignment).

Silhouette Analysis

- We then define the average dissimilarity of point $\mathbf{x}_i$ to a cluster *c* as the average of the distance from $\mathbf{x}_i$ i to all points in *c*.

- Let b_i be the lowest average distance of $\mathbf{x}_i$ to all points in any other cluster, of which $\mathbf{x}_i$ is not a member.

Silhouette Analysis

- The cluster with this lowest average dissimilarity is said to be the "neighboring cluster" of $\mathbf{x}_i$, because it is the next best fit cluster for point $\mathbf{x}_i$. We now define a silhouette:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

- or

$$s_i = \begin{cases} 1 - a_i / b_i & a_i < b_i \\ 0 & a_i = b_i \\ b_i / a_i - 1 & a_i > b_i \end{cases}$$

Silhouette Analysis

- From

$$s_i = \begin{cases} 1 - a_i / b_i & a_i < b_i \\ 0 & a_i = b_i \\ b_i / a_i - 1 & a_i > b_i \end{cases}$$

it is obvious that $-1 \leq s_i \leq 1$.

Silhouette Analysis

- s_i close to 1 requires $a_i \ll b_i$.
- a_i is a measure of dissimilarity of $\mathbf{x}_i$ to its own cluster
- Thus a small a_i means $\mathbf{x}_i$ is well matched.
- Large b_i implies that $\mathbf{x}_i$ is badly matched to its neighboring cluster.
- Thus an s_i close to one means that $\mathbf{x}_i$ is appropriately clustered.

Silhouette Analysis

- s_i close to negative one, by the same logic means that $\mathbf{x}_i$ would be more appropriate if it was clustered in its neighboring cluster.
- An s_i near zero means that $\mathbf{x}_i$ is on the border of two natural clusters.

Silhouette Analysis

- The average s_i over a cluster measures how tightly grouped all the data in the cluster are.
- Thus the average s_i over the entire dataset is a measure of how appropriately the data have been clustered.

Silhouette Analysis

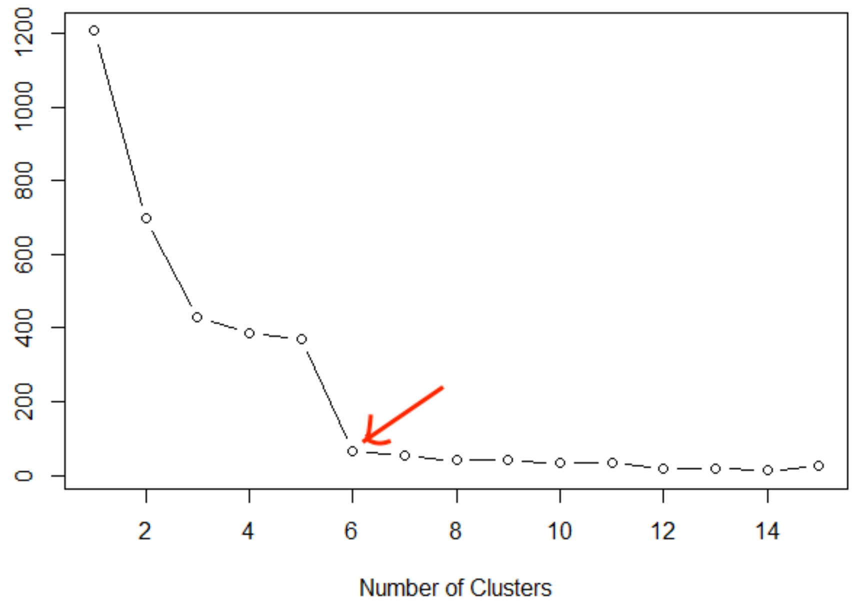
- If there are too many or too few clusters, some of the clusters will display narrower silhouettes than the rest.
- So silhouette plots and averages may be used to determine the natural number of clusters within a dataset.
- One can also increase the likelihood of the silhouette being maximized at the correct number of clusters by re-scaling the data using feature weights that are cluster specific.

Cross Validation

- The data is partitioned into *v* folds.
- Each of the folds is then held out at turn as a test set, a clustering model computed on the other *v* − 1 training sets
- The value of an objective function is calculated for the test set.
- Example of objective function: the sum of the squared distances to the centroids for *k*-means

Cross Validation

- These *v* values are calculated and averaged for each alternative number of clusters *c*, and the cluster number selected such that further increase in number of clusters leads to only a small reduction in the objective function (scree plots)



More

- https://stackoverflow.com/questions/15376075/cluster-analysis-in-r-determine-the-optimal-number-of-clusters/15376462#15376462

Once again, it really is a hard problem

Background

Just How Many Clusters are there in the Galaxy Data?

- Galaxy Data from Postman *et al.* (1986): measurements of velocities in 10^3 km/sec of 82 galaxies from a survey of the Corona Borealis region.
- Roeder (1990): at least 3, no more than 7 modes (Confidence set)
- Others are in consensus

9172	9350	9483	9558	9775	10227
10406	16084	16170	18419	18552	18600
18927	19052	19070	19330	19343	19349
19440	19473	19529	19541	19547	19663
19846	19856	19863	19914	19918	19973
19989	20166	20175	20179	20196	20215
20221	20415	20629	20795	20821	20846
20875	20986	21137	21492	21701	21814
21921	21960	22185	22209	22242	22249
22314	22374	22495	22746	22747	22888
22914	23206	23241	23263	23484	23538
23542	23666	23706	23711	24129	24285
24289	24366	24717	24990	25633	26960
26995	32065	32789	34279		

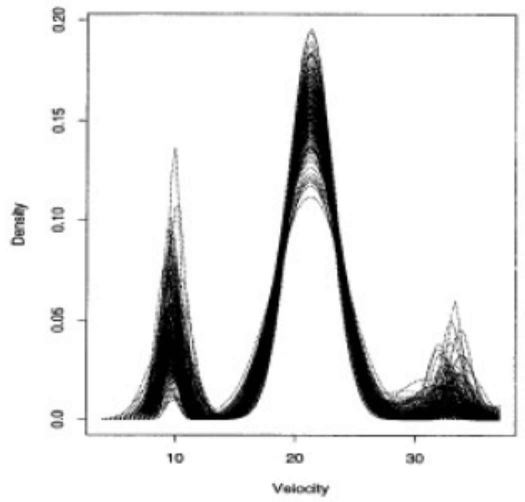


Figure 1. Densities Obtained From the Markov Chain Monte Carlo Sampler Using the Astronomy Data From Roeder (1992).

- Histogram from Roeder and Wasserman (1997)

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.

Principal Components Analysis: details

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

$$Z_1 = \varphi_{11}X_1 + \varphi_{21}X_2 + \ldots + \varphi_{p1}X_p$$

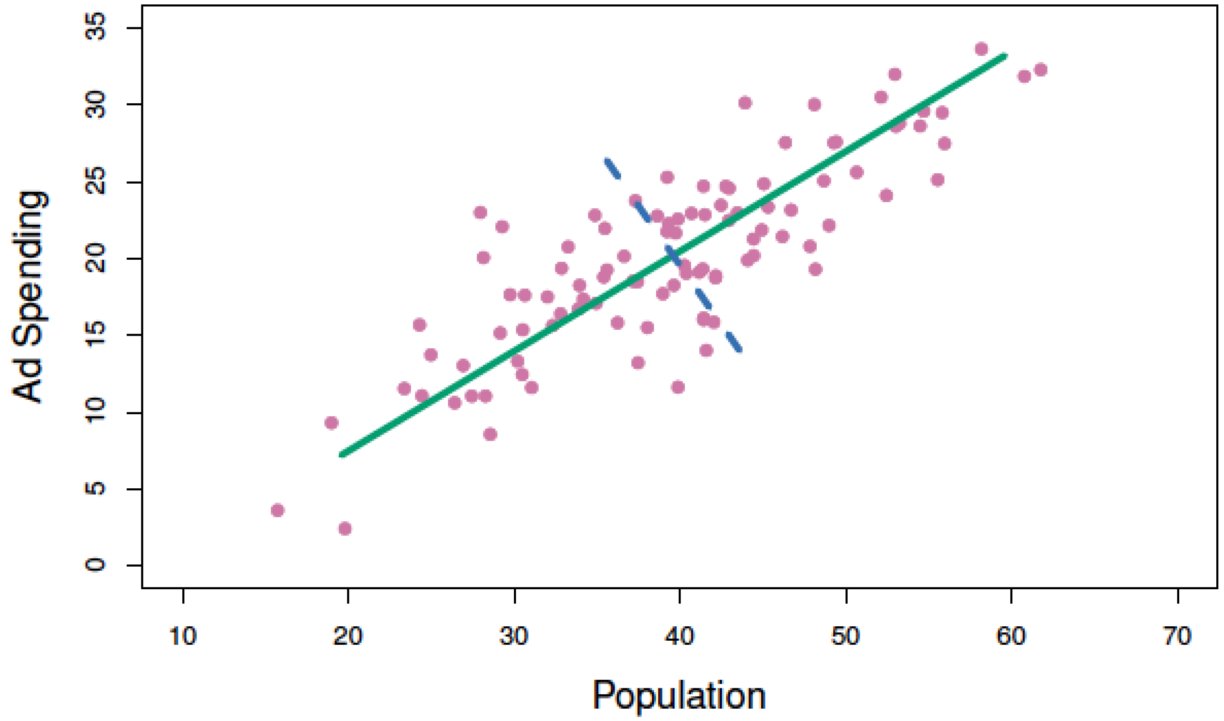
that has the largest variance. By normalized, we mean that

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1$$

Principal Components Analysis: details

- We refer to the elements $\varphi_{11}, \ldots, \varphi_{p1}$ as the loadings of the first principal component; together, the loadings make up the principal component loading vector, $\varphi_1 = (\varphi_{11}\, \varphi_{21} \ldots \varphi_{p1})^T$.
- 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: example



The population size (pop) and ad spending (ad) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component direction, and the blue dashed line indicates the second principal component direction.

Computation of Principal Components

- Suppose we have a $n \times p$ data set. Since we are only interested in variance, we assume **X** that each of the variables in **X** has been centered to have mean zero (that is, the column means of **X** are zero).
- We then look for the linear combination of the sample feature values of the form $z_{i1} = \varphi_{11}x_{i1} + \varphi_{21}x_{i2} + \ldots + \varphi_{p1}x_{ip}$

  for i = 1, . . . , n that has largest sample variance, subject to the constraint that

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1$$

Computation of Principal Components

- 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: continued

- Therefore, the first principal component loading vector solves the optimization problem

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

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

Geometry of PCA

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

Further principal components

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

$$z_{i2} = \varphi_{12}x_{i1} + \varphi_{22}x_{i2} + \ldots + \varphi_{p2}x_{ip},$$

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

Further principal components: 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 φ_1, φ_2, $\varphi_3, \ldots$ are the ordered sequence of right singular vectors of the matrix **X**, and the variances of the components are 1/n times the squares of the singular values. There are at most $\min(n - 1, p)$ principal components.

Illustration

- USAarrests data: For each of the fifty states in the United States, the data set contains the number of arrests per 100, 000 residents for each of three crimes: Assault, Murder, and Rape. We also record UrbanPop (the percent of the population in each state living in urban areas).

Illustration

- The principal component score vectors have length $n = 50$, and the principal component loading vectors have length $p = 4$.
- PCA was performed after standardizing each variable to have mean zero and standard deviation one.

USAarrests data: PCA plot

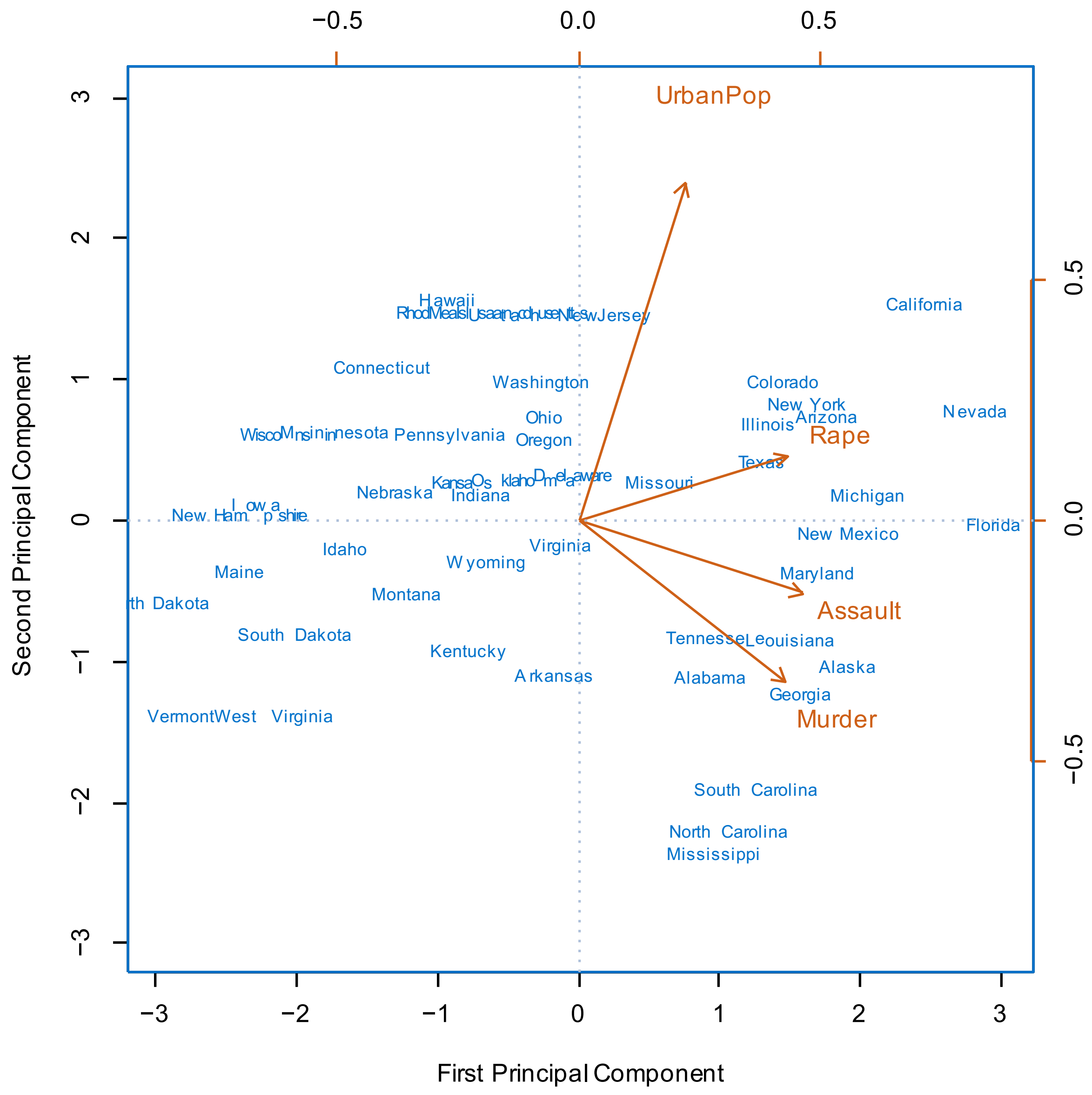
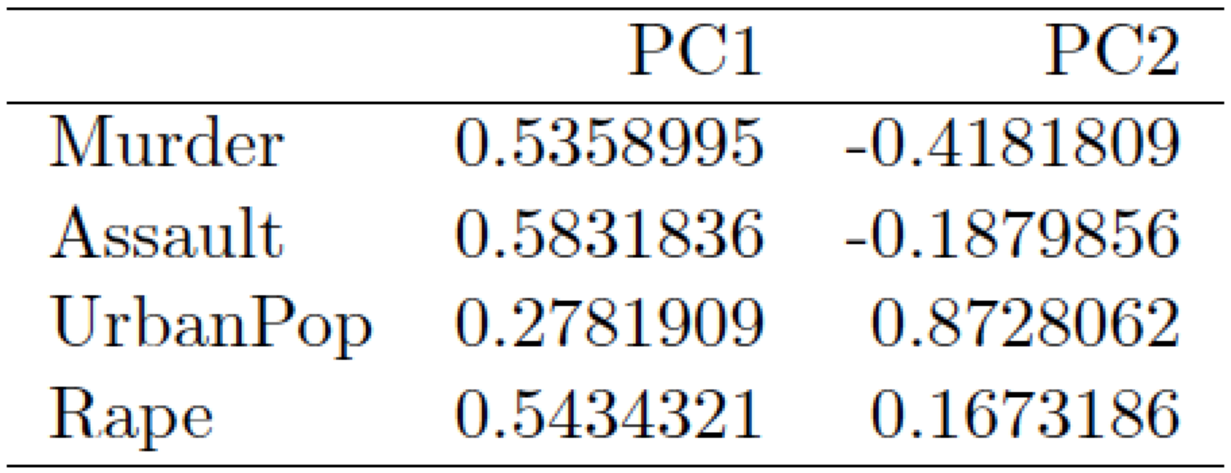


Figure details

The first two principal components for the USArrests data.

- The blue state names represent the scores for the first two principal components.

PCA loadings



	PC1	PC2
Murder	0.5358995	-0.4181809
Assault	0.5831836	-0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

The contribution of each variable in each of the principal components PC1 and PC2 is shown as a vector. For example, Assault is shown as $[0.58,-0.19]^T$

Figure details

- The orange arrows indicate the first two principal component loading vectors (with axes on the top and right). For example, the loading for Rape on the first component is 0.54, and its loading on the second principal component 0.17 [the word Rape is centered at the point (0.54, 0.17)].
- This figure is known as a *biplot*, because it displays both the principal component scores and the principal component loadings.

Figure details

- The first loading vector places approximately equal weight on Assault, Murder, and Rape, with much less weight on UrbanPop.
- Hence this component roughly corresponds to a measure of overall rates of serious crimes.
- The second loading vector places most of its weight on UrbanPop and much less weight on the other three features.
- Hence, this component roughly corresponds to the level of urbanization of the state.

Figure details

- The crime-related variables (Murder, Assault , and Rape) are located close to each other
- The UrbanPop variable is far from the other three.
- The crime-related variables are correlated with each other: states with high murder rates tend to have high assault and rape rates
- The UrbanPop variable is less correlated with the other three.

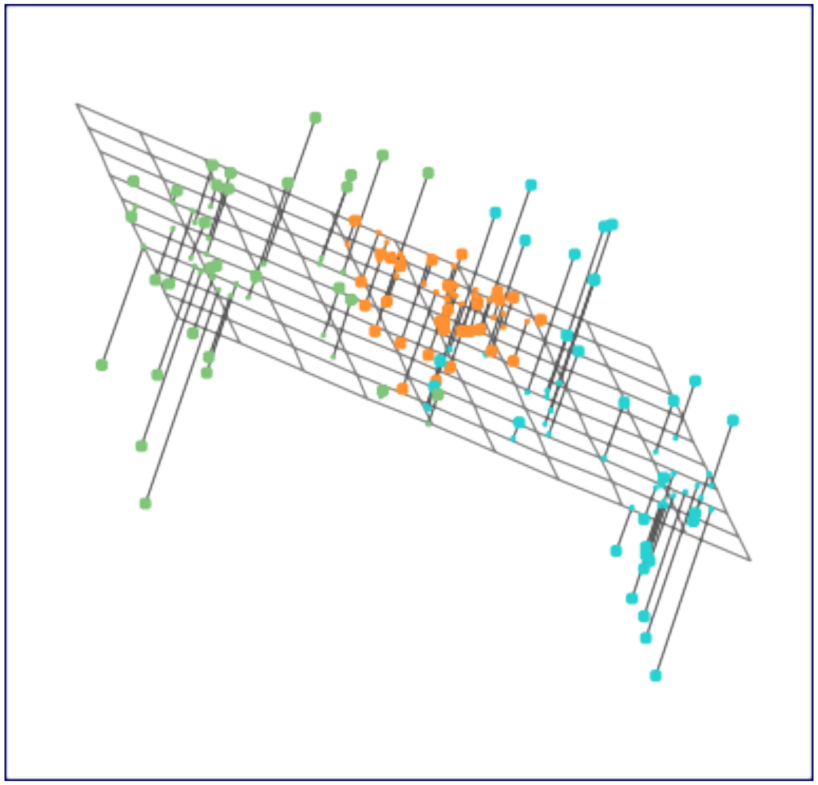
Figure details

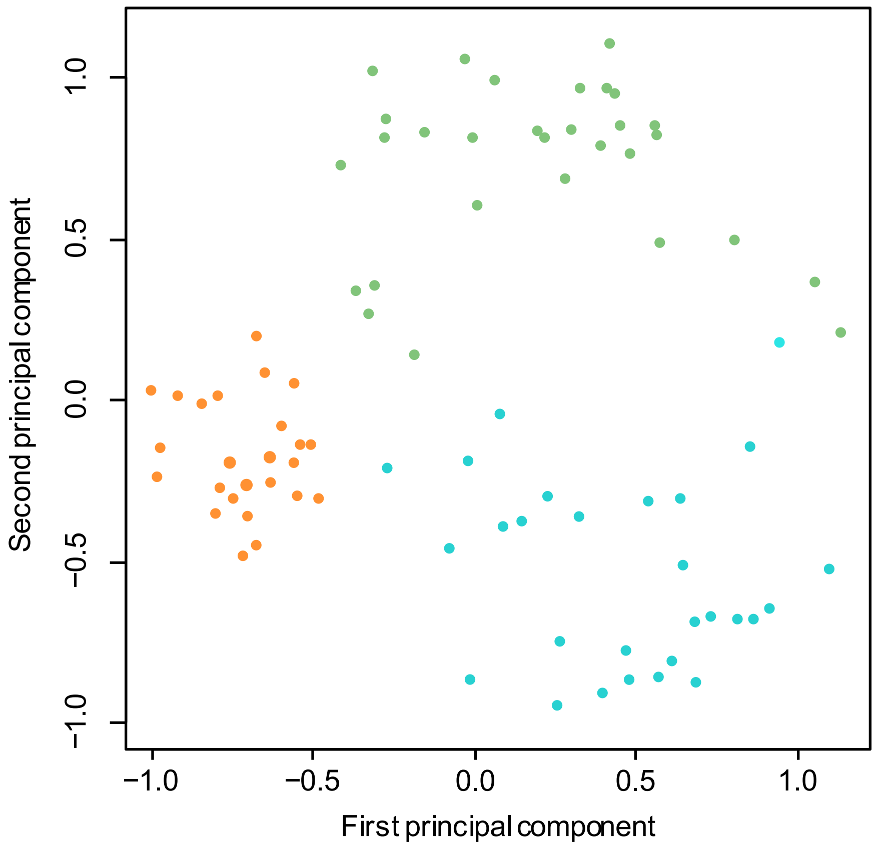
- States with large positive scores on the first component, such as California, Nevada and Florida, have high crime rates
- States like North Dakota, with negative scores on the first component, have low crime rates.

Figure details

- California has a high score on the second component, hence a high level of urbanization
- The opposite is true for states like Mississippi.
- States close to zero on both components, such as Virginia, have approximately average levels of both crime and urbanization.

Another Interpretation of Principal Components





The first three principal components of a data set span the three-dimensional hyperplane that is closest to the n observations,

PCA find the hyperplane closest to the observations

- The first principal component loading vector has a very special property: it defines the line in *p*-dimensional space that is *closest* to the *n* observations (using average squared Euclidean distance as a measure of closeness)

PCA find the hyperplane closest to the observations

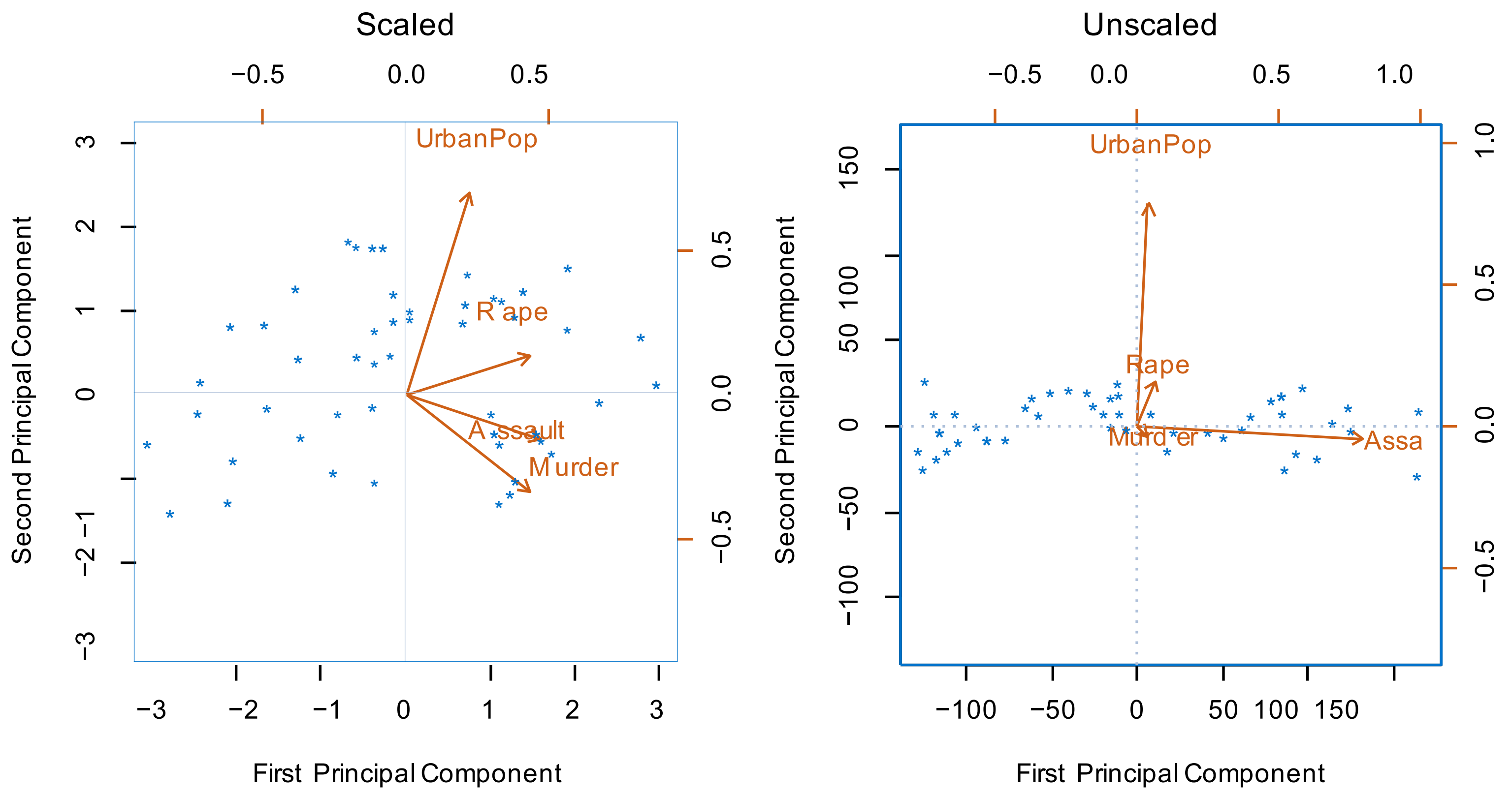
- The notion of principal components as the dimensions that are closest to the *n* observations extends beyond just the first principal component.
- For instance, the first two principal components of a data set span the plane that is closest to the *n* observations, in terms of average squared Euclidean distance.

Scaling of the variables matters

- Murder, Rape, Assault, and UrbanPop have variance 18. 97, 87. 73, 6945. 16, and 209. 5, respectively.
- In PCA on the unscaled variables, the first principal component loading vector will have a very large loading for Assault
- This is simply a consequence of the scales of the variables.
  - For instance, if Assault were measured in units of the number of occurrences per 100 people (rather than number of occurrences per 100,000 people), then this would amount to dividing all of the elements of that variable by 1,000. Then the variance of the variable would be tiny

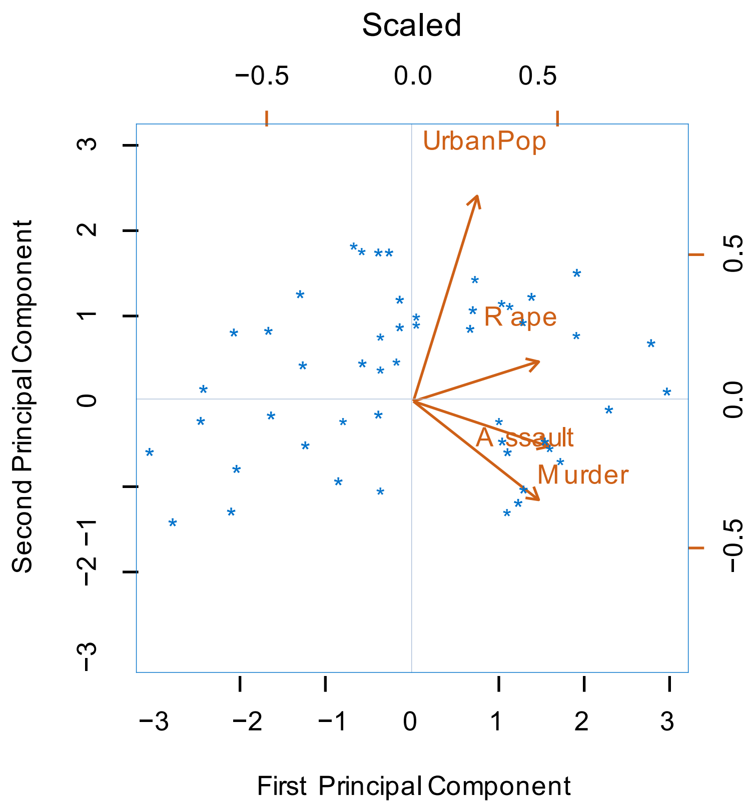
Scaling of the variables matters

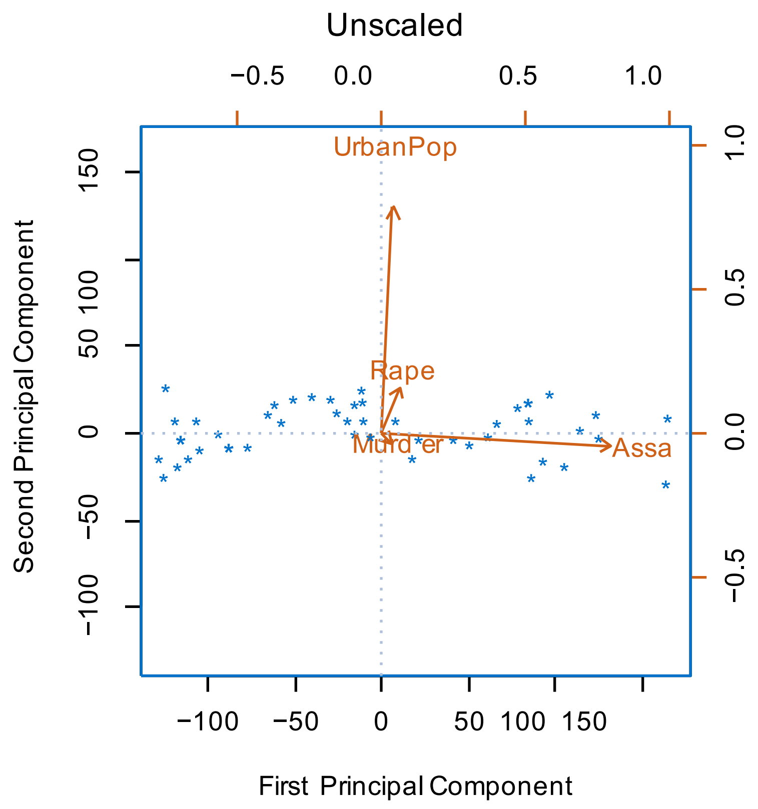
- It is undesirable for the principal components obtained to depend on an arbitrary choice of scaling
- We typically scale each variable to have standard deviation one before we perform PCA.



Scaling of the variables matters

- If the variables are in different units, scaling each to have standard deviation equal to one is recommended.
- If they are in the same units, you might or might not scale the variables.





Proportion Variance Explained

- We can now ask a natural question: how much of the information in a given data set is lost by projecting the observations onto the first few principal components?
- That is, how much of the variance in the data is not contained in the first few 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

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

Proportion Variance Explained

The variance explained by the *m*th principal component is

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

- It can be shown that

$$\textstyle\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{m=1}^{M} \mathrm{Var}(Z_m)$$

with $M = \min(n-1, p)$.

Proportion Variance Explained: continued

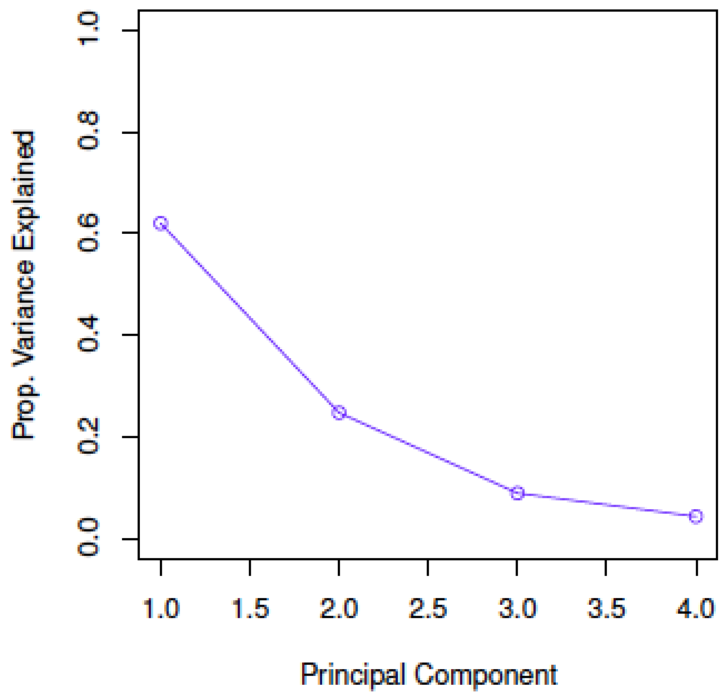
- Therefore, the PVE of the *m*th principal component is given by the positive quantity between 0 and 1

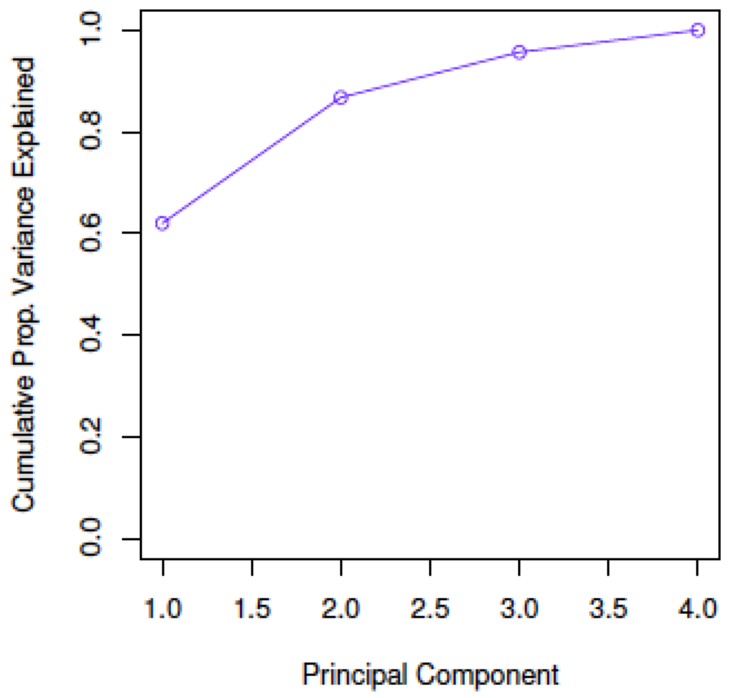
$$\frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}$$

Principal Component Principal Component

Proportion Variance Explained: continued

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





How many principal components should we use?

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?*

How many principal components should we use?

- 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".

How many principal components should we use?

- If we compute principal components in a supervised analysis, then we can treat the number of principal component score vectors to be used as a tuning parameter to be selected via cross-validation

PCA vs Clustering

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

Conclusions

- *Unsupervised learning* is important for understanding the variation and grouping structure of a set of unlabeled data, and can be a useful pre-processor for supervised learning
- It is intrinsically more difficult than *supervised learning* because there is no gold standard (like an outcome variable) and no single objective (like test set accuracy)

Conclusions

- It is an active field of research, with many recently developed tools such as *self-organizing maps*, *independent components analysis* and *spectral clustering*.

  See *The Elements of Statistical Learning*, chapter 14.

Appendix 1

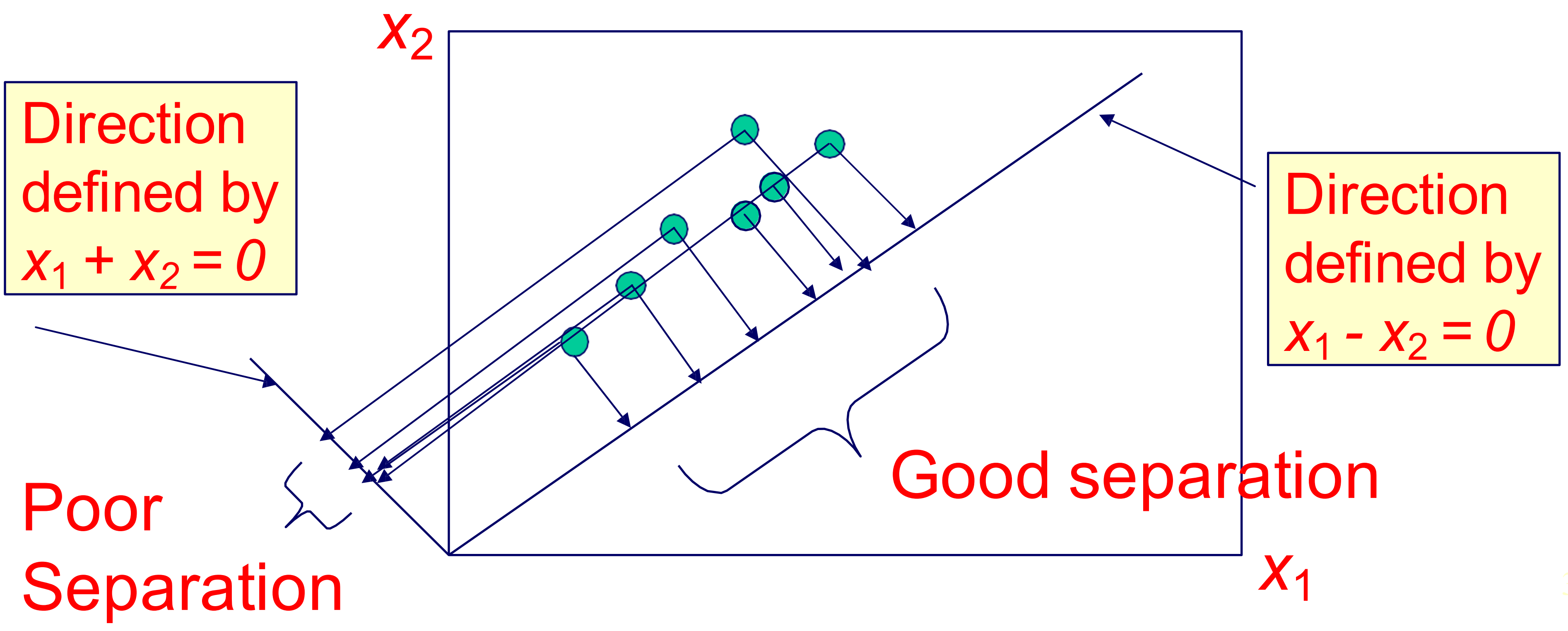
- In this appendix we try to perform PCA more analytically

PCA: Linear Projections of Data

- Excessive dimensionality **x**= [x_1, x_2, … x_d] causes
  - Computational difficulty
  - Visualization issues
- Solution:
  - Combine features to reduce dimensionality
  - Linear combinations, *e.g.,* $2x_1+3x_2+x_3$
    - are simple to compute and tractable
    - Project high dimensional data onto a lower dimensional space

Projection to a lower dimensional space

- Allow computer to search for interesting directions


x_2
Direction defined by $x_1 + x_2 = 0$
Direction defined by $x_1 - x_2 = 0$
Good separation
Poor Separation
x_1

Linear Projection

- Equation of line through origin: $x_1 + 2x_2 + 4x_3 = 0$

$$[1\ 2\ 4] \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_1 + 2x_2 + 4x_3 = 0$$

Can be written as $\mathbf{w}^T\mathbf{x} = 0$ where $\mathbf{w} = [1\ 2\ 4]$

- Projection of a point x along line with projection weights **a** is given by:

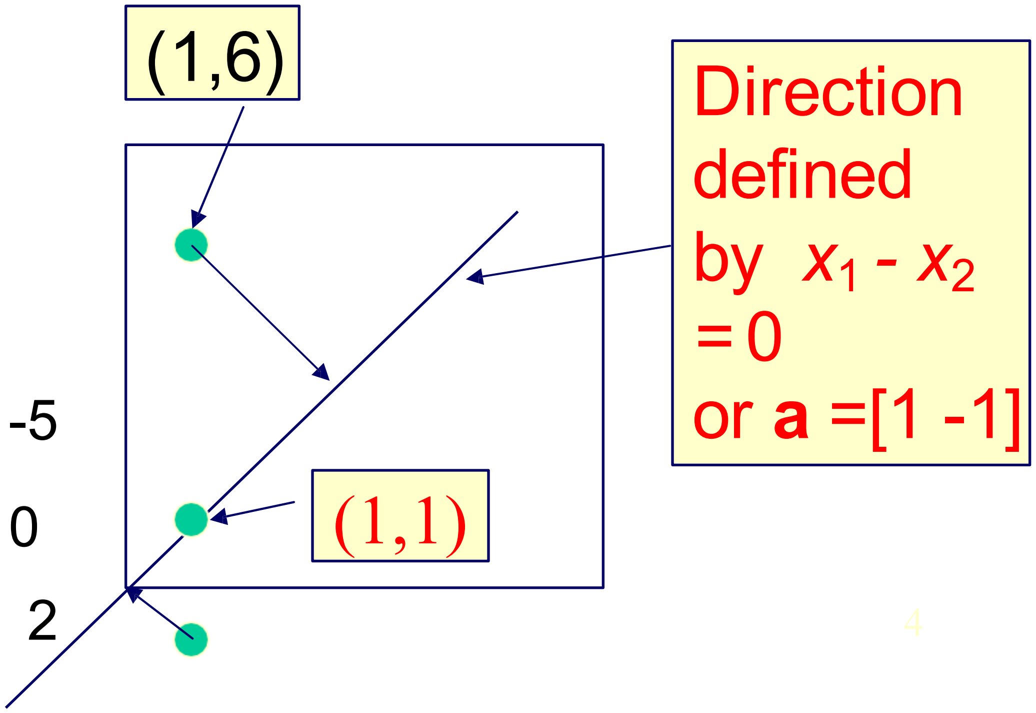
$$\mathrm{a}^T\mathrm{x} = \sum_{j=1}^{d} \mathrm{a}_j x_j$$

- Example:

Projection of (1,6) along [1 -1]: 1 - 6 = -5

Projection of (1,1) along [1 -1]: 1 - 1 = 0

Projection of (1,-1) along [1 -1]: 1 + 1 = 2



Principal Components Analysis

- We will look at Data Matrix, Scatter Matrix and Covariance Matrix to arrive at best projection of data

Data Matrix

Let **X** be a *n x p* *data matrix* of *n* samples

$$\begin{array}{cc} & p \text{ variables} \\ \begin{matrix}\mathbf{x}^{T}(1) \\ \\ \mathbf{x}^{T}(i) \\ \\ \mathbf{x}^{T}(n)\end{matrix} & \left(\begin{matrix} \qquad\qquad\qquad \\ \\ \\ \\ \qquad\qquad\qquad \end{matrix}\right) \end{array}$$

$\mathbf{x}(i)$ is a p x 1 column vector

Each row of matrix is of the form $\mathbf{x}^{T}(i)$

Assume **X** is centralized, so that the value of each variable is subtracted for that variable

Scatter Matrix

$$\mathbf{S}_{p\times p} = \sum_{i=1}^{n}\left(\mathbf{x}(i)-\mathbf{m}\right)\left(\mathbf{x}(i)-\mathbf{m}\right)^{T}$$

- The Scatter Matrix **S** is (*n*-1) times the Covariance Matrix
- Relationship Between Data Matrix and Scatter Matrix:
    - **S** = **X** ^{T} **X** is the *p* x *p* scatter matrix of the data since **X** has zero mean

Projection

Let **a** be a *p* x *1* column vector of projection weights that result in the largest variance when the data matrix X are projected along **a.**
The projection of any data vector **x** is the linear combination.

$$\mathbf{a}^T\mathbf{x} = \sum_{j=1}^{p} a_j x_j$$

Projected values of all data vectors in data matrix **X** onto **a** can be expressed as **Xa** which is an *n* x 1 column vector.

Variance along Projection

Variance along **a** is

$$\mathrm{Var}_{\mathbf{a}}\left(\mathbf{X}\right) \propto \left(\mathbf{Xa}\right)^T \left(\mathbf{Xa}\right)$$
$$= \mathbf{a}^T \mathbf{S} \mathbf{a}$$

where **S** is is the scatter matrix of **S**.

Therefore, the variance is a function of both **X** and **a**.
Maximizing variance along **a** is not well-defined since we can increase it without limit by increasing the size of the components of **a**.

Optimization Problem

Impose a normalization constraint on the **a** vectors such that $\mathbf{a}^T\mathbf{a} = 1$

Optimization problem is to maximize

$$u(\mathbf{a}) = \mathbf{a}^T\mathbf{S}\mathbf{a} - \lambda\left(\mathbf{a}^T\mathbf{a} - 1\right)$$

Variance Criterion

Normalization Criterion

Where λ is a Lagrange multiplier.

Solution: Differentiating wrt **a** yields

which reduces to

Characteristic Equation of S!

Characteristic Equation

Given a p x p matrix $\mathbf{M}$, a very important class of linear equations is of the form $\mathbf{M}_{pxp}\,\mathbf{x}_{px1} = \lambda\mathbf{x}_{px1}$

which can be rewritten as $(\mathbf{Mx}-\lambda\mathbf{I})\ \mathbf{x} = \mathbf{0}$.

If $\mathbf{M}$ is real and symmetric there are p possible solution vectors (vectors $\mathbf{x}$ that satisfy the characteristic equation) called eigenvectors,$\mathbf{e}_1, .., \mathbf{e}_p$ and associated eigenvalues λ_1 ,...., λ_p. The eigenvectors are linearly independent and can be chosen so that they are orthogonal/ orthonormal.

First Principal Component

If the matrix **M** is the Scatter matrix **S**, the Characteristic Equation is:

$$(\mathbf{S} - \lambda\mathbf{I})\mathbf{a} = 0$$

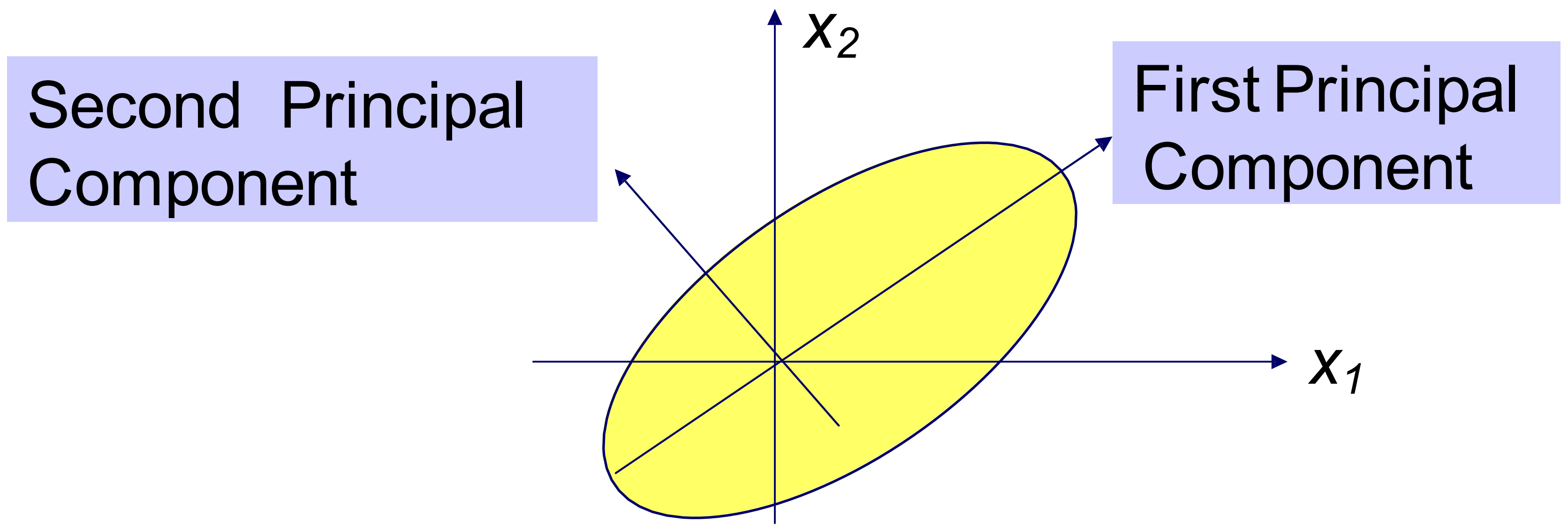
The roots are eigenvalues of **S.**
The corresponding eigenvectors are principal components.
First principal component is the eigenvector associated with the largest eigenvalue of **S**.

First Principal Component

- First principal component is the eigenvector associated with the largest eigenvalue of **S**.
- It is the direction along which **X** has the largest variation.
- Second Principal component is in direction orthogonal to first. Therefore, it is uncorrelated with the first principal component.
- Has second largest eigenvalue.

Other Principal Components

- All principal components are in direction orthogonal to one another, so they are uncorrelated.
- The variation along each principal component depends on how large its eigenvalue is.



Alternative Derivation of PCA

- DHS Text gives another way to derive the fact that eigen vectors of scatter matrix are principal components
- By setting up the squared error criterion function of the data and the vector **e** that minimizes it satisfies the characteristic equation
- Can be generalized to from one-dimensional projection to a dimension *p'* which are the eigenvectors of **S** forming the principal components

Projection into *k* Eigenvectors

Variance of data projected into first *k* eigenvectors is proportional to

$$\sum_{j=1}^{k} \lambda_j$$

That is because:

$$\mathrm{Var}_{\mathbf{e}_j}\left(\mathbf{X}\right) \propto \left(\mathbf{X}\mathbf{e}_j\right)^T \left(\mathbf{X}\mathbf{e}_j\right)$$

$$=$$

Projection into *k* Eigenvectors

and:

$$\mathrm{Var}_{\mathbf{e}_1,\mathbf{e}_2,\ldots,\mathbf{e}_k}\left(\mathbf{X}\right)\propto$$

Projection into *k* Eigenvectors

Variance of data projected into first *k* eigenvectors is

$$\sum_{j=1}^{k} \lambda_j$$

The proportion of variance lost in approximating true data matrix **X** using only first *k* eigenvectors is

$$\frac{\sum_{j=k+1}^{p} \lambda_j}{\sum_{l=1}^{p} \lambda_l}$$

Usually 5 to 10 principal components capture 90% of variance in the data

How Many Principal Components Should One Use?

When Using PCA for classification, one can consider the number of components used as a hyper parameter and use cross-validation to select the number of components.

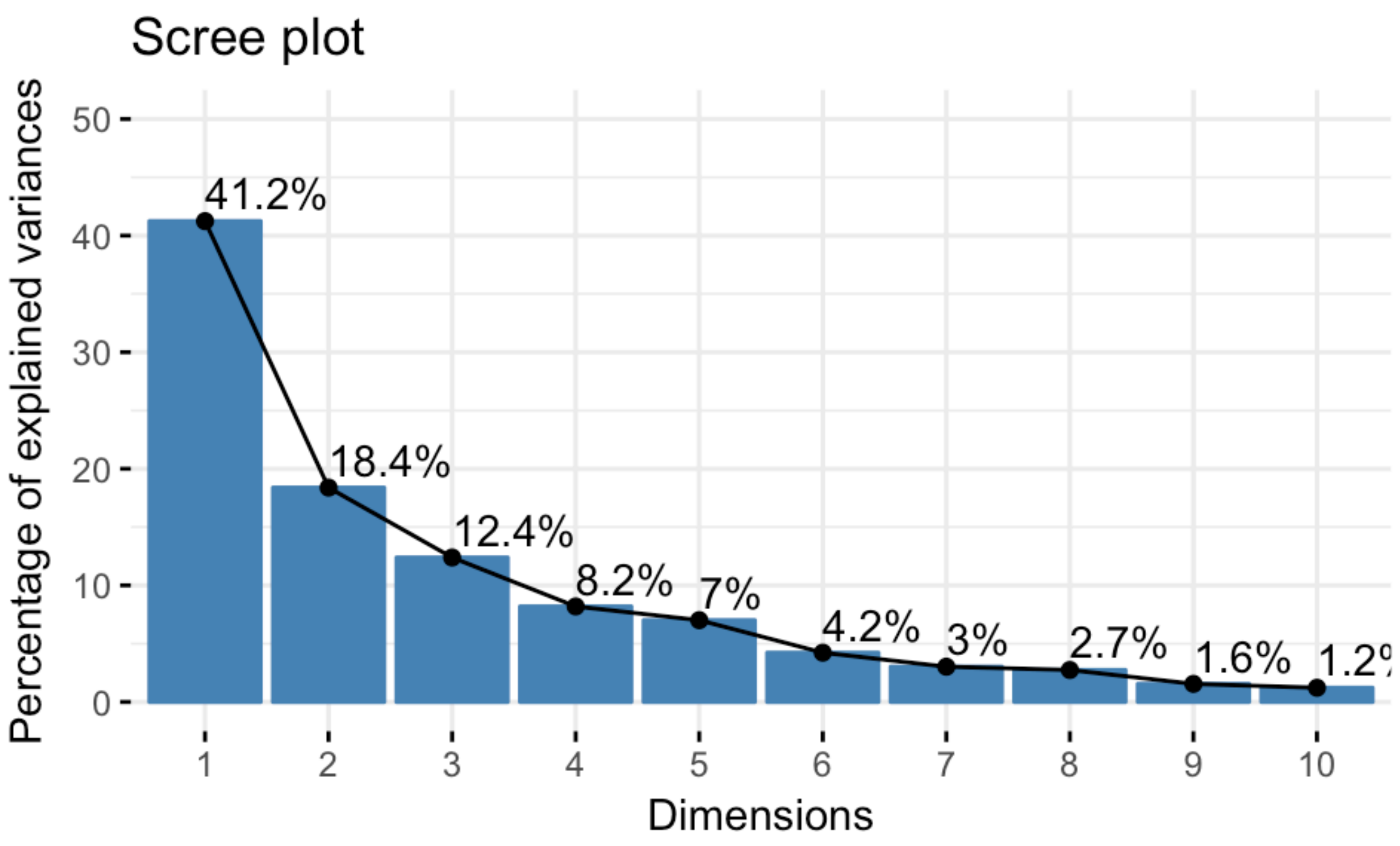
How Many Principal Components Should One Use?

An alternative method to determine the number of principal components merely from unlabeled **X** is to look at a Scree Plot, which is the plot of eigenvalues ordered from largest to the smallest.

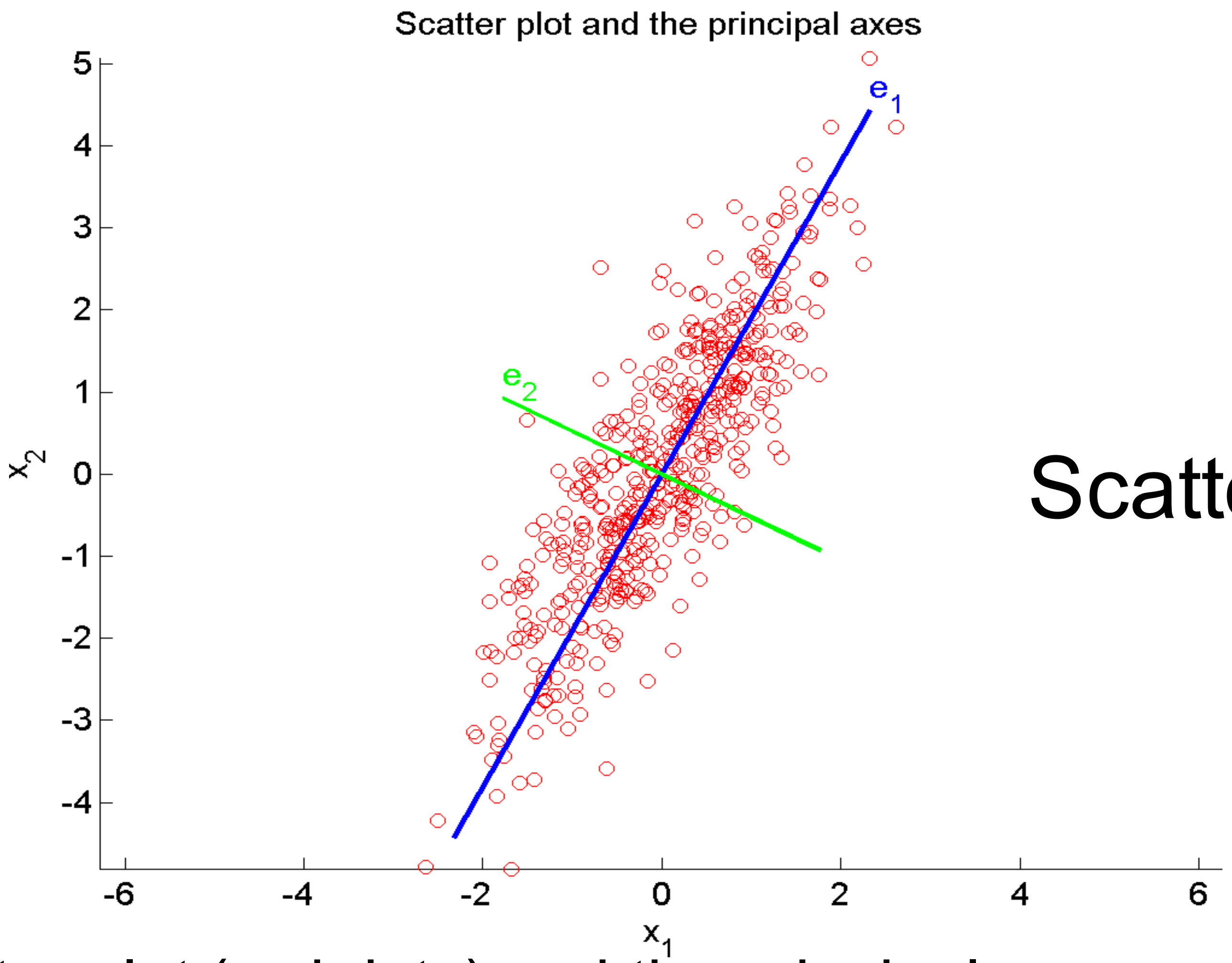
The number of component is determined at the point, beyond which the remaining eigenvalues are all relatively small and of comparable size (Jollife 2002, Peres-Neto, Jackson, and Somers (2005)).

How Many Principal Components Should One Use?

Scree plot: alternative form



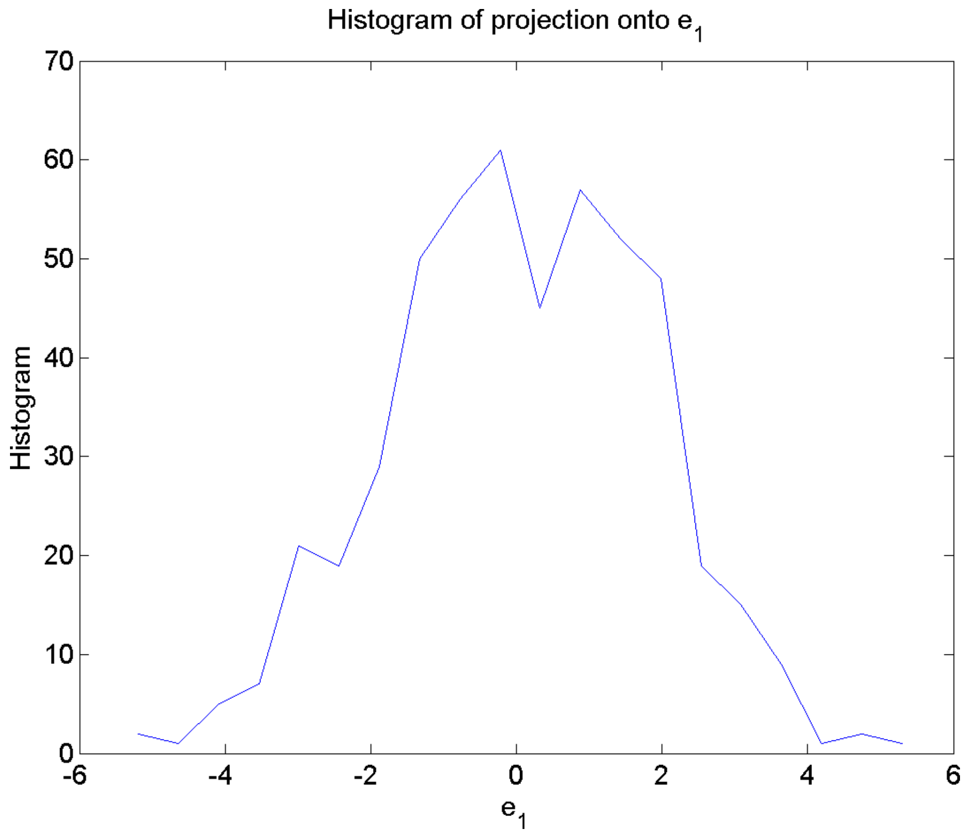
Examples

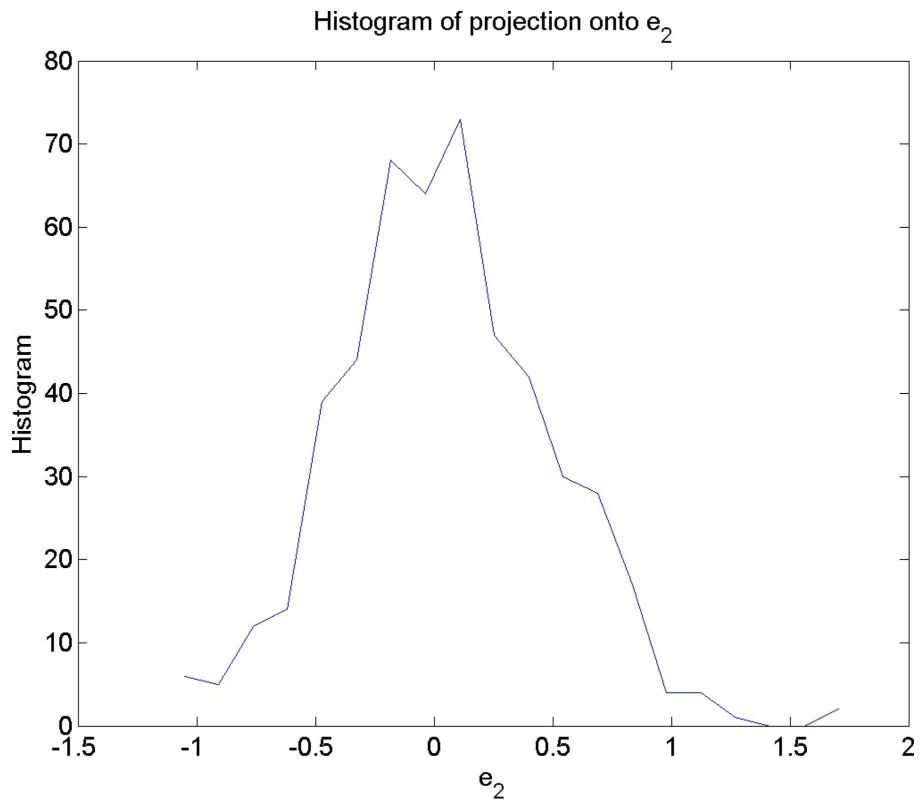


Scatter plot.

Scatter plot (red dots) and the principal axes for a bivariate sample. The blue line shows the axis e_1 with the greatest variance and the green line shows the axis e_2 with the smallest variance. Features are now uncorrelated.

Examples





Projection onto e_1.

(c) Projection onto e_2.

Appendix 2

In this Appendix, we introduce SOMs

Self-Organizing Maps

Key idea (Kohonen, 1982)

- Idea: to learn a *map* of the data, in a low-dimensional embedding.
- Self-Organizing Maps can be thought of as a constrained version of *k*-means clustering, in which the prototypes are encouraged to lie in a one- or two-dimensional manifold in the feature space.

Self-Organizing Maps

Key idea (Kohonen, 1982)

- This manifold is also referred to as *constrained topological map*, since the original high-dimensional observations can be mapped down onto the two-dimensional coordinate system.
- The original SOM algorithm was online, but batch versions have also been proposed.

Self-Organizing Maps

- We consider a SOM with a two-dimensional rectangular grid of *k* prototypes *with p components*. The choice of a grid is arbitrary, other choices like hexagonal grids are also possible.
- Each of the *k* prototypes are parameterized with respect to an integer coordinate pair $l_j \in Q_1 \times Q_2$.

Self-Organizing Maps

- Here $Q_1 = \{1, 2, . . . , q_1\}$, $Q_2 = \{1, 2, . . . , q_2\}$, and $k = q_1q_2$.
- One can think of the protoypes as "buttons" sewn on the principal compnent plane in a regular pattern.
- Intuitively, the SOM tries to bend the plane so that the buttons approximate the data points as well as possible.
- Once the model is fit, the observations can be mapped onto the two-dimensional grid.

Self-Organizing Maps

- The ***mj*** (prototypes) are initialized, for example, to lie in the two-dimensional principal component plane of the data
- The observations $\mathbf{x}_i$ are processed one at a time.
- We find the closest prototype $\mathbf{m}_j$ to $\mathbf{x}_i$ in Euclidean distance in R^p, and then for all neighbors $\mathbf{m}_k$ of $\mathbf{m}_j$, we move $\mathbf{m}_k$ toward $\mathbf{x}_i$ via the update $\mathbf{m}_k \leftarrow \mathbf{m}_k + \alpha(\mathbf{x}_i - \mathbf{m}_k)$

Self-Organizing Maps

- The neighbors of $\mathbf{m}_j$ are defined to be all $\mathbf{m}_k$ such that the distance between l_j and l_k is small.
- $\alpha \in R$ is the learning rate and determines the scale of the step. The simplest approach uses Euclidean distance, and "small" is determined by a threshold s. The neighborhood always includes the closest prototype itself.

Self-Organizing Maps

- Notice that the distance is defined in the space $Q_1 \times Q_2$ of integer topological coordinates of the prototypes, rather than in the feature space R^d.
- The effect of the update is to move the prototypes closer to the data, but also to maintain a smooth two-dimensional spatial relationship between the prototypes.

SOMs: Practical Considerations

- The performance is influenced by the learning rate *α* and the distance threshold *s*. Typically *α* is decreased from 1 to 0 in a few thousand iterations.
- Similarly *s* is decreased linearly from starting value *R* to 1 over a few thousand iterations. The description above refers to the simplest version of SOM.

SOMs: Practical Considerations

- In more advanced approaches, the update step is changed with distance:
- $m_k \leftarrow m_k + \alpha h(\|l_j - l_k\|)(x_i - m_k)$, where the neighborhood function h gives more weight to prototypes m_k with indices l_k closer to l_j than those further away.

SOMs: Link to k-means

- If *s* is chosen so small that each neighborhood contains only one point, then the spatial connection between prototypes is lost. In that case, SOM is an online version of k-means, and converges to a local minimum of the k-means objective.

Self-Organizing Maps: Summary

- Self-Organizing Maps are a variant of k-means for nonlinear dimensionality reduction and visualisation.
- As in k-means, points are iteratively assigned to cluster means.
- Unlike k-means, updates of cluster means are constrained and synchronized based on a user-defined lattice between cluster means.

Self-Organizing Maps: Video

- https://www.youtube.com/watch?v=H9H6s-x-0YE