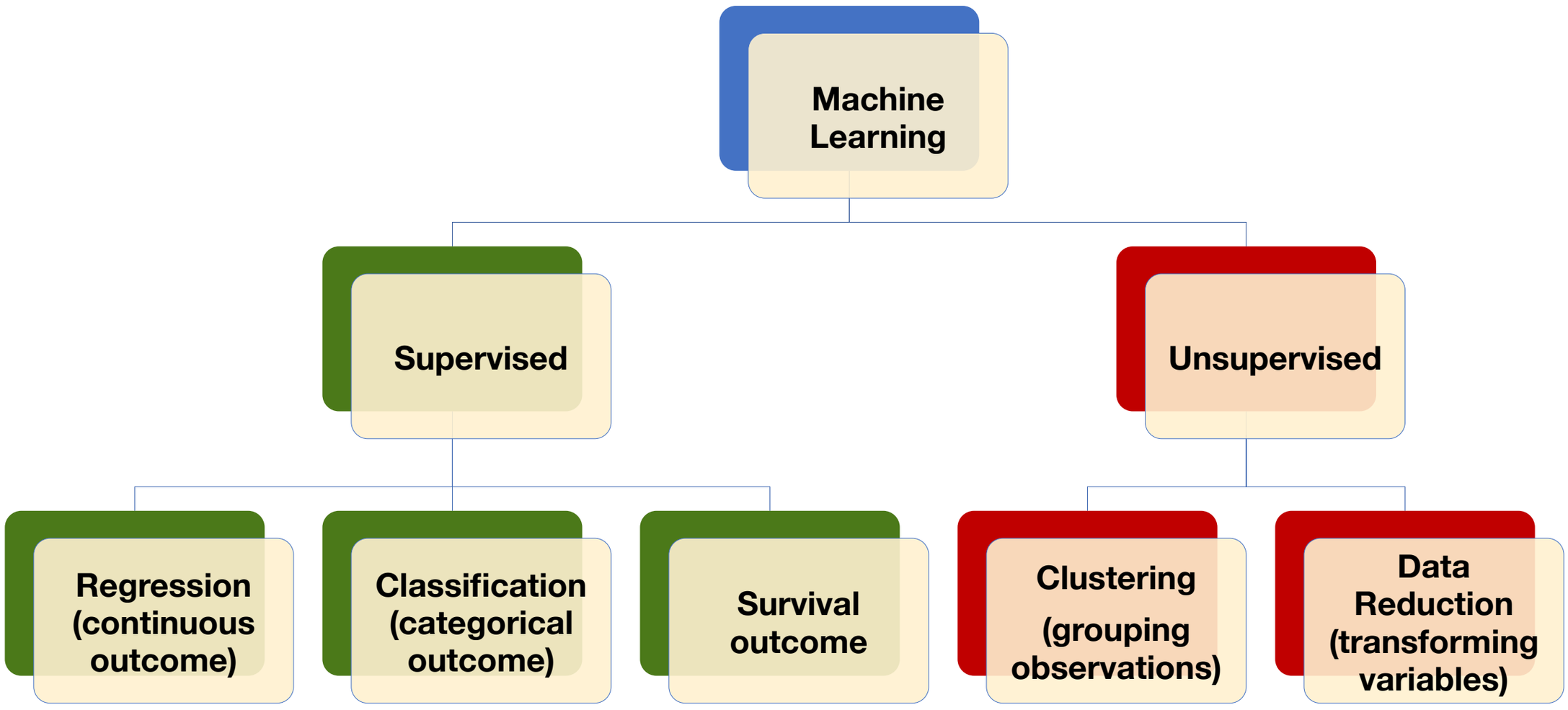
Machine Learning in R

Lecture 7 – Unsupervised Learning

Jean Feng – Epidemiology and Biostatistics

Supervised vs Unsupervised



Today's lecture

1. **Unsupervised learning**
2. Clustering
    A. K-means
    B. Hierarchical Clustering
3. Dimension Reduction
    A. PCA

Supervised vs Unsupervised Learning

Supervised learning

- Training data: $\{(X_i, Y_i) : i = 1,...,n\}$
- Goal: Predict an outcome Y given features X.

Unsupervised learning

- Training data: $\{X_i : i = 1,...,n\}$

No labels!

- Goals:
  - Discover some structure that characterizes the data, e.g. uncover subgroups and hierarchies
  - Generate hypotheses
  - Find relationships between features

Application: “Shared and distinct transcriptomic cell types across neocortical areas”

Tasic 2018

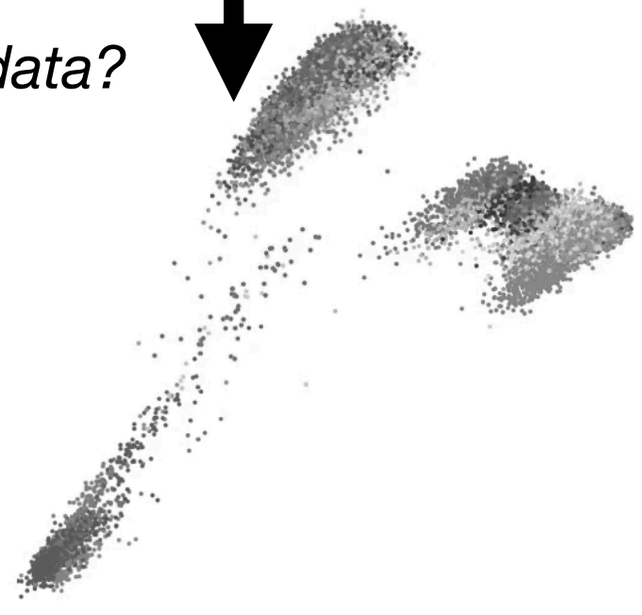
What cell types are in the brain?

Single-cell transcriptomics data:

n=23,000 cells, p=3,000 genes

$$X = \begin{array}{|c|c|c|c|c|c|} \hline 0.0 & 1.4 & -30 & 0 & -5.3 & \dots \\ \hline 0.2 & -1.3 & -4 & 1 & -4.2 & \dots \\ \hline 0.9 & 0.2 & -10 & 0 & -3.9 & \dots \\ \hline \dots & \dots & \dots & \dots & \dots & \dots \\ \hline \end{array}$$

Visualize the data?



Application: “Shared and distinct transcriptomic cell types across neocortical areas”

Tasic 2018

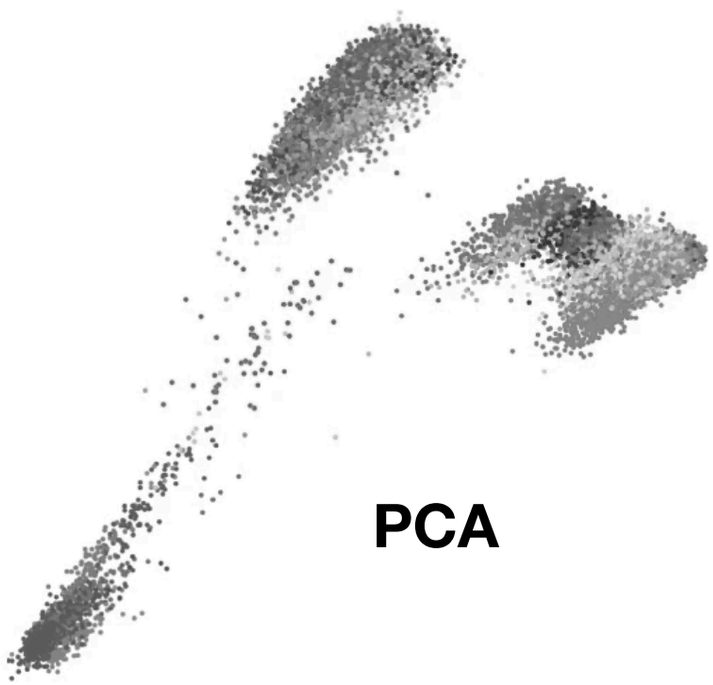
Single-cell transcriptomics data:
n=23,000 cells, p=3,000 genes

$$X = \begin{array}{|c|c|c|c|c|c|} \hline 0.0 & 1.4 & -30 & 0 & -5.3 & \dots \\ \hline 0.2 & -1.3 & -4 & 1 & -4.2 & \dots \\ \hline 0.9 & 0.2 & -10 & 0 & -3.9 & \dots \\ \hline \dots & \dots & \dots & \dots & \dots & \dots \\ \hline \end{array}$$

Dimension reduction along two axes

Latent factors

$$Z = \begin{array}{|c|c|} \hline 0.2 & 0.3 \\ \hline 0.8 & -0.6 \\ \hline -0.1 & 0.2 \\ \hline \dots & \dots \\ \hline \end{array}$$



Application: “Shared and distinct transcriptomic cell types across neocortical areas”

Tasic 2018

Single-cell transcriptomics data:
n=23,000 cells, p=3,000 genes

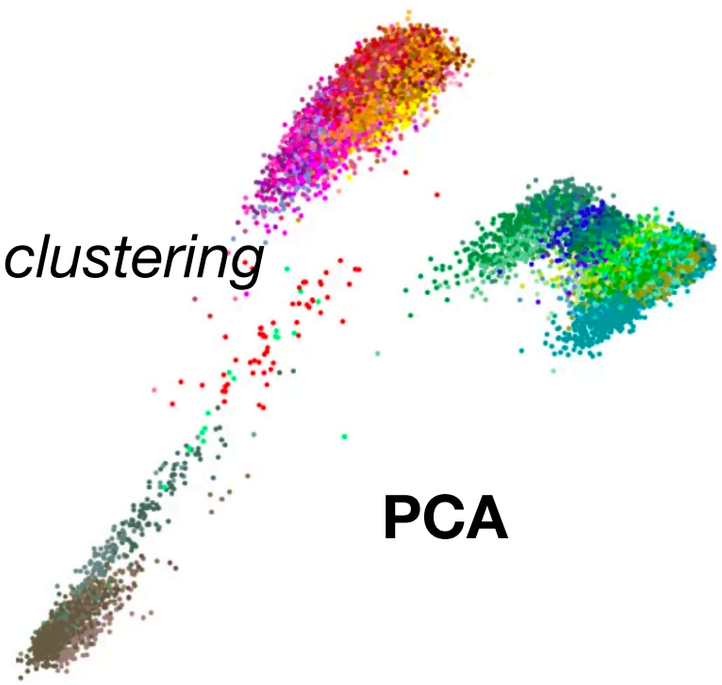
$$X = \begin{array}{|c|c|c|c|c|c|} \hline 0.0 & 1.4 & -30 & 0 & -5.3 & \dots \\ \hline 0.2 & -1.3 & -4 & 1 & -4.2 & \dots \\ \hline 0.9 & 0.2 & -10 & 0 & -3.9 & \dots \\ \hline \dots & \dots & \dots & \dots & \dots & \dots \\ \hline \end{array}$$

Dimension reduction along two axes

Latent factors

$$Z = \begin{array}{|c|c|} \hline 0.2 & 0.3 \\ \hline 0.8 & -0.6 \\ \hline -0.1 & 0.2 \\ \hline \dots & \dots \\ \hline \end{array}$$

Apply clustering



Application: “Shared and distinct transcriptomic cell types across neocortical areas”

Tasic 2018

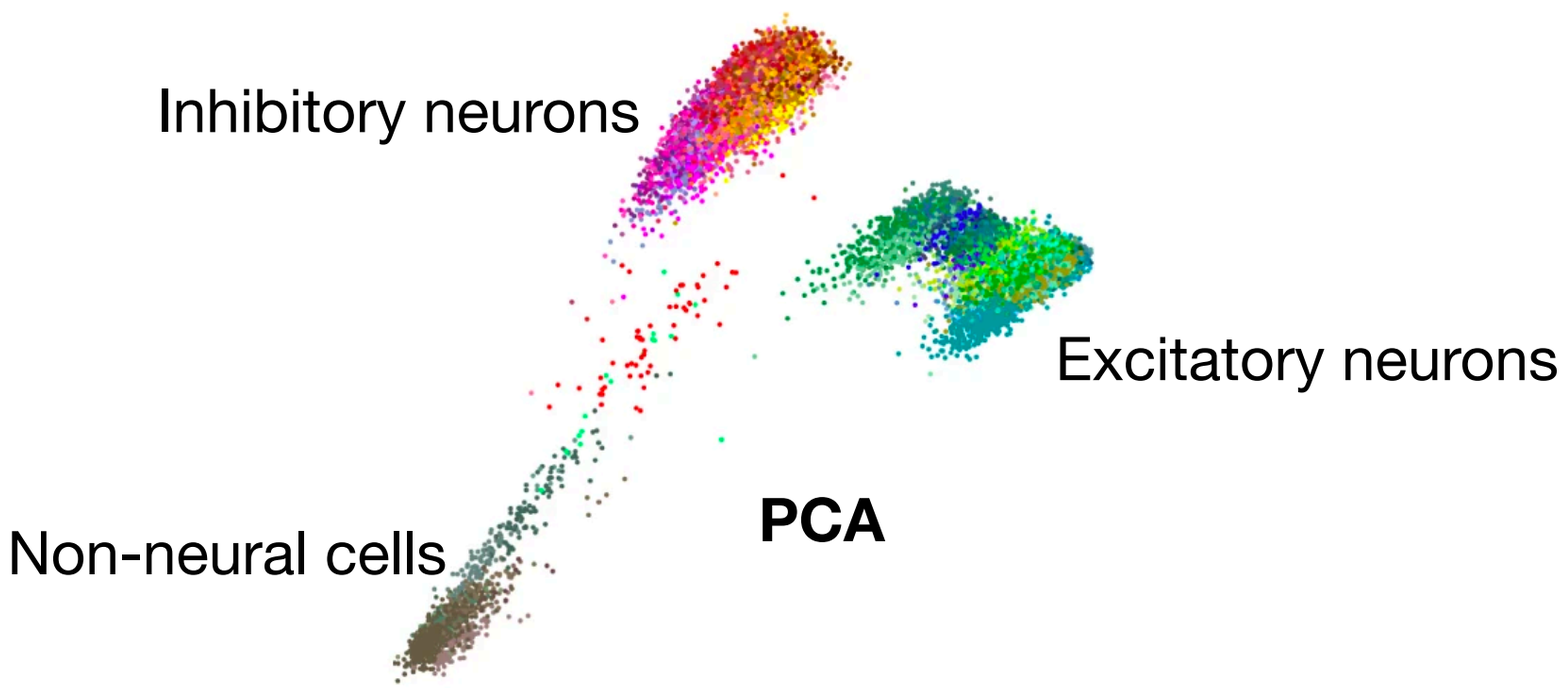
Single-cell transcriptomics data:
n=23,000 cells, p=3,000 genes

$$X = \begin{array}{|c|c|c|c|c|c|} \hline 0.0 & 1.4 & -30 & 0 & -5.3 & \dots \\ \hline 0.2 & -1.3 & -4 & 1 & -4.2 & \dots \\ \hline 0.9 & 0.2 & -10 & 0 & -3.9 & \dots \\ \hline \dots & \dots & \dots & \dots & \dots & \dots \\ \hline \end{array}$$

Dimension reduction along two axes

Latent factors

$$Z = \begin{array}{|c|c|} \hline 0.2 & 0.3 \\ \hline 0.8 & -0.6 \\ \hline -0.1 & 0.2 \\ \hline \dots & \dots \\ \hline \end{array}$$



Application: “Shared and distinct transcriptomic cell types across neocortical areas”

Tasic 2018

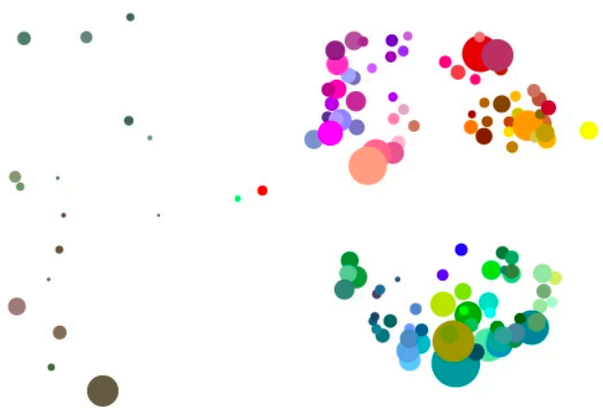
Single-cell transcriptomics data:
n=23,000 cells, p=3,000 genes

$$X = \begin{array}{|c|c|c|c|c|c|} \hline 0.0 & 1.4 & -30 & 0 & -5.3 & \dots \\ \hline 0.2 & -1.3 & -4 & 1 & -4.2 & \dots \\ \hline 0.9 & 0.2 & -10 & 0 & -3.9 & \dots \\ \hline \dots & \dots & \dots & \dots & \dots & \dots \\ \hline \end{array}$$

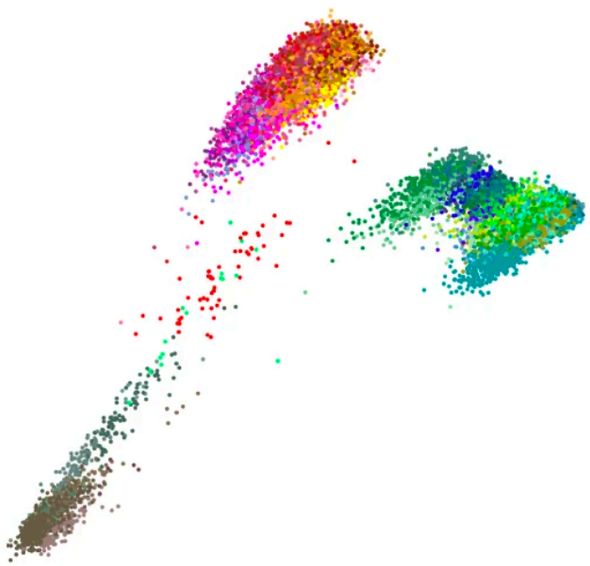
Dimension reduction along two axes

Latent factors

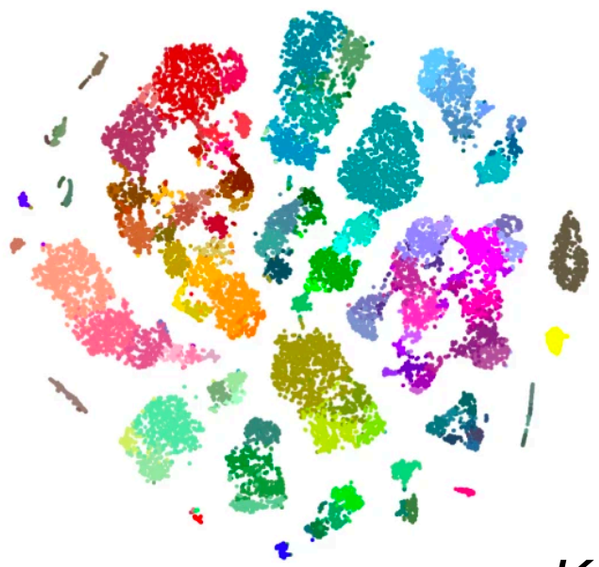
$$Z = \begin{array}{|c|c|} \hline 0.2 & 0.3 \\ \hline 0.8 & -0.6 \\ \hline -0.1 & 0.2 \\ \hline \dots & \dots \\ \hline \end{array}$$



MDS



PCA



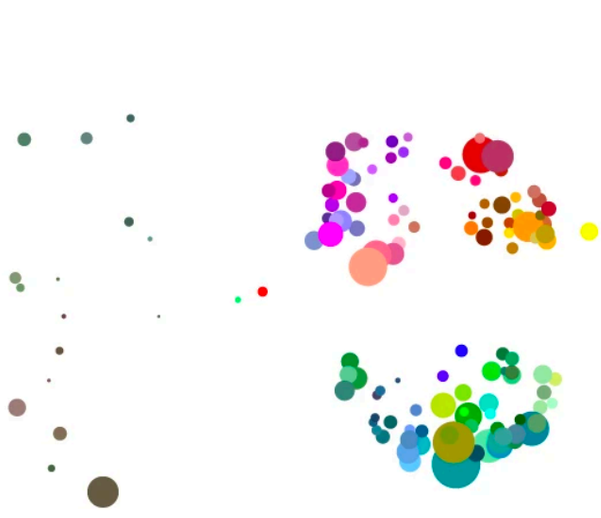
t-SNE

Kobak 2019

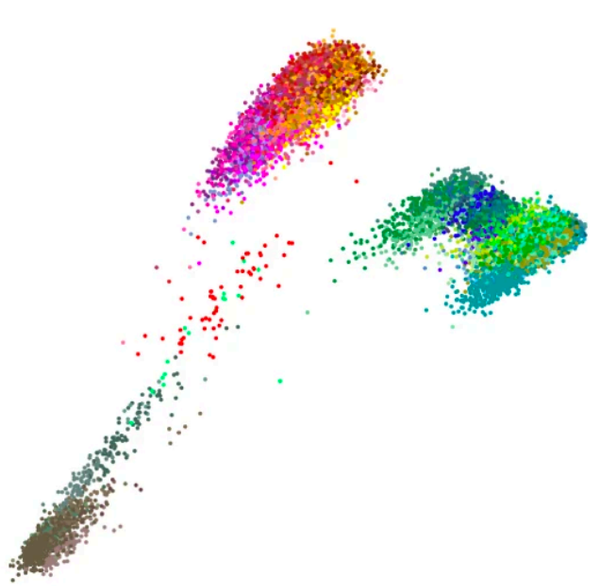
Challenges

- Difficult to validate and compare unsupervised learning results (we don't usually observe labels or ground truth)

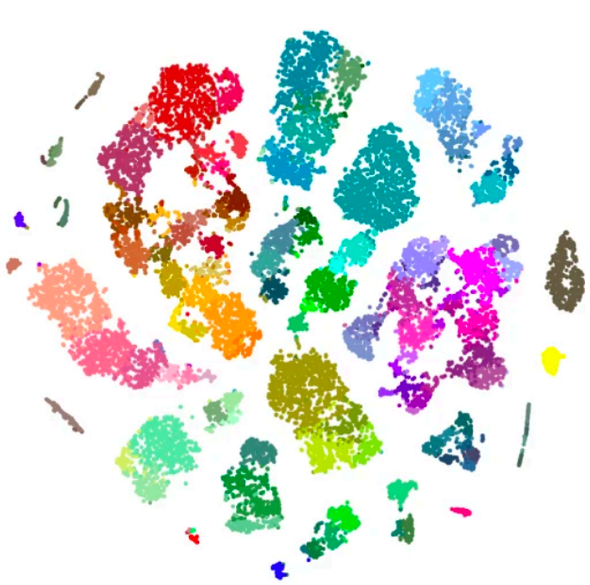
Which dimension reduction + clustering method is better?



MDS



PCA



t-SNE

Kobak 2019

Today's lecture

1. Unsupervised learning
2. **Clustering**
    A. K-means
    B. Hierarchical Clustering
3. Dimension Reduction
    A. PCA

Clustering

- The goal of clustering is to find subgroups or clusters in a dataset.

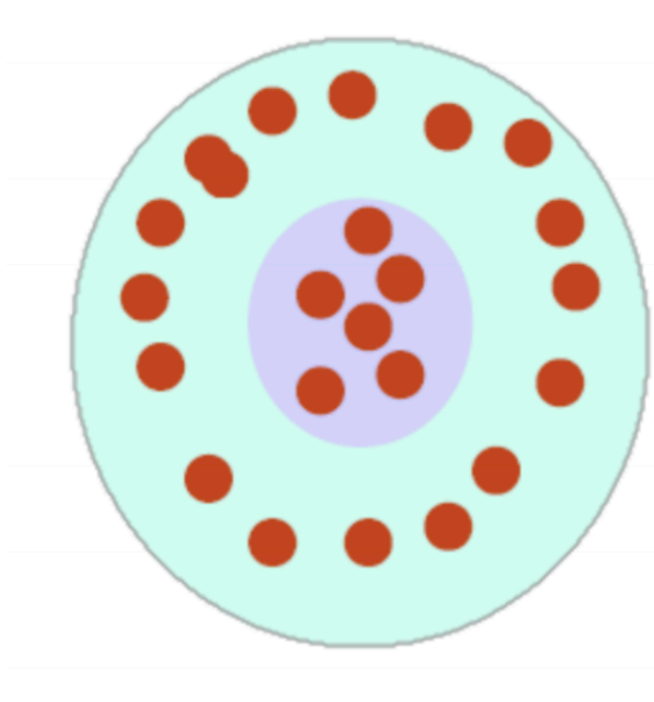
- Generally, we would like a partitioning where points within subgroups are **similar** to each other and points from different subgroups are **dissimilar** to each other.

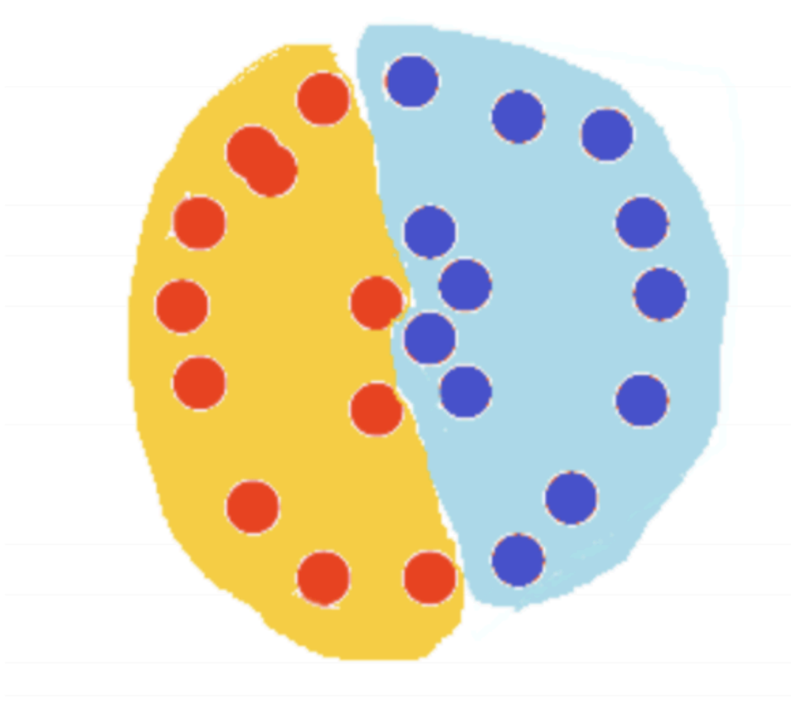
- Methods:
  - K-means
  - Hierarchical clustering
  - Spectral clustering
  - …

Which points are similar?

- To perform clustering, we need to define a dissimilarity measure between points.

- Our results are highly sensitive to this choice!





Dissimilarity measure

- Depending on your data, you'll need to consider distance between continuous, ordinal, and categorical variables.

- When X is p-dimensional, how do you aggregate the distances for the p variables into a single measure? Do you weight variables differently?

Euclidean distance: $d(x, x') = \sqrt{\sum_{i=1}^{p} (x_i - x_i')^2}$

Weighted Euclidean distance: $d(x, x') = \sqrt{\sum_{i=1}^{p} w_i (x_i - x_i')^2}$

Divisive vs agglomerative clustering

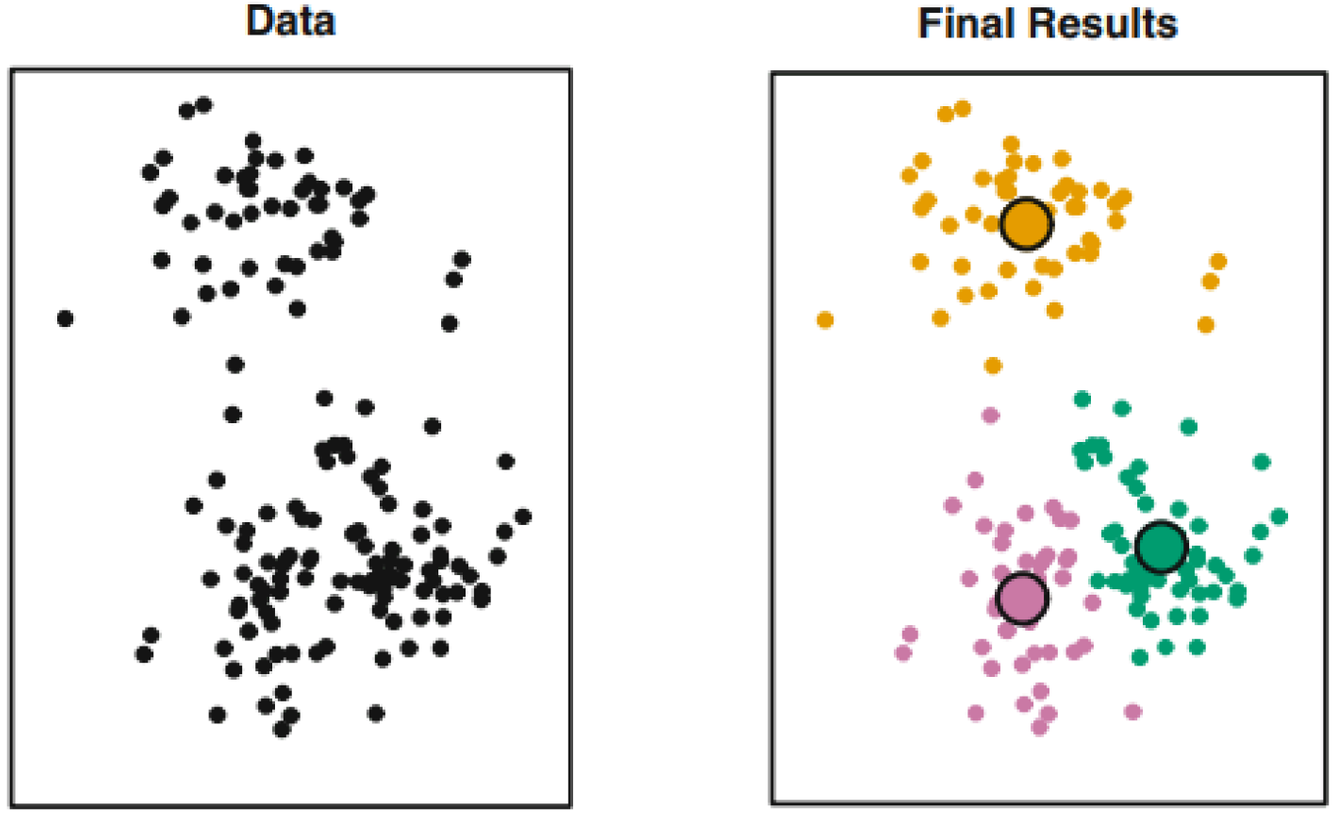
- **Divisive/top-down clustering**: Divide the data into clusters.
    - Uses global information to partition data.
    - Example: K-means
- **Agglomerative/bottom-up clustering**: Create clusters by merging similar observations.
    - Uses local information to partition data.
    - Example: Most hierarchical clustering methods

Today's lecture

1. Unsupervised learning
2. Clustering
    A. **K-means**
    B. Hierarchical Clustering
3. Dimension Reduction
    A. PCA

K-means

- Goal: Partition n observations into K clusters.
- Assumption: Dissimilarity measure is the Euclidean distance.
- Approach: Find a clustering such that the dissimilarity of points within each cluster is as low as possible.



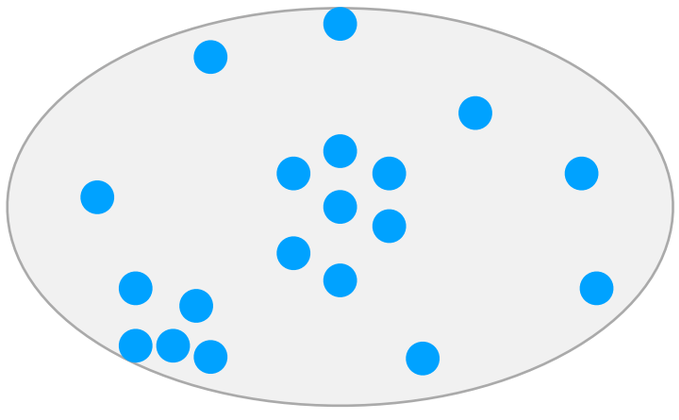
Notation

- Let $C_1, \ldots, C_K$ denote a partition of indices $\{1,...,n\}$.
- So the sets $C_1, \ldots, C_K$ satisfy the following properties:
  - $C_1 \cup C_2 \cup \ldots \cup C_K = \{1,...,n\}$ (all observations will belong to a cluster)
  - $C_i \cap C_j = \emptyset$ for all pairs $i \neq j$ (each observation belongs to only one cluster)
- Let $|C_k|$ be the number of indices in C_k.

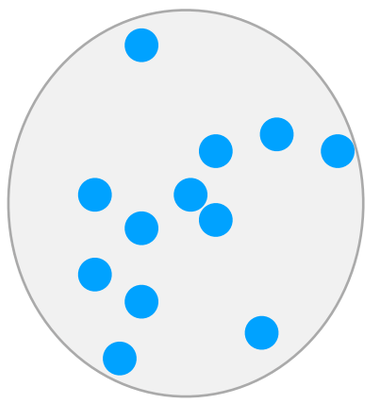
K-means: approach

- Approach: Find a clustering such that *the dissimilarity of points within each cluster* is as low as possible.

- Define the "**Within-cluster variation**" of a cluster C as the average squared distance between pairs of points within the cluster: $WCV(C) = \frac{1}{2|C|} \sum_{i \neq i' \in C} \|x_i - x_{i'}\|_2^2$



Big *WCV*

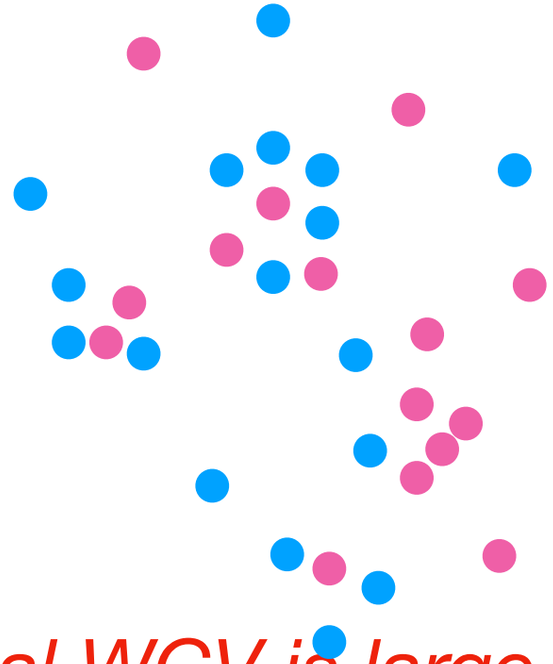


Small *WCV*

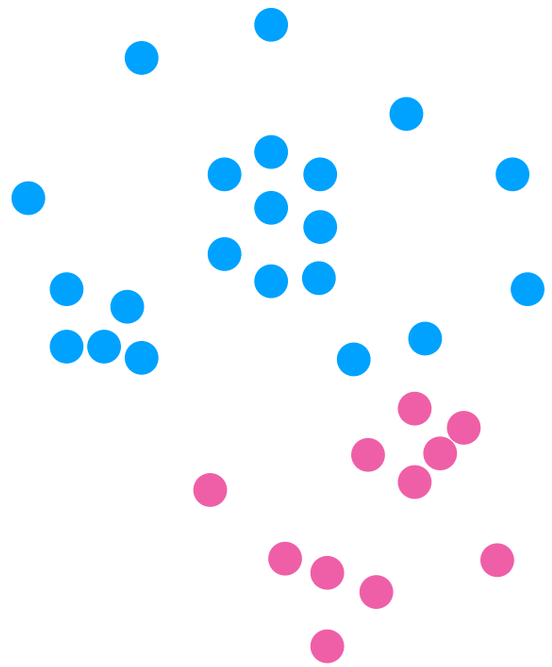
K-means: approach

- Approach: Find a clustering such that the dissimilarity of points within each cluster is *as low as possible.*

- Problem statement: Find a partition that minimizes the total within-cluster variation, i.e. $\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} WCV(C_k)$



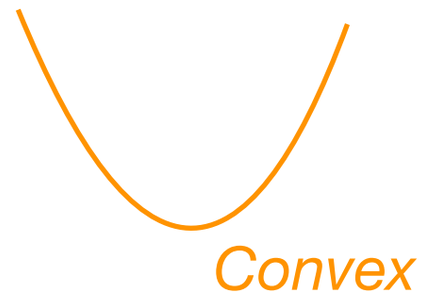
Total WCV is large



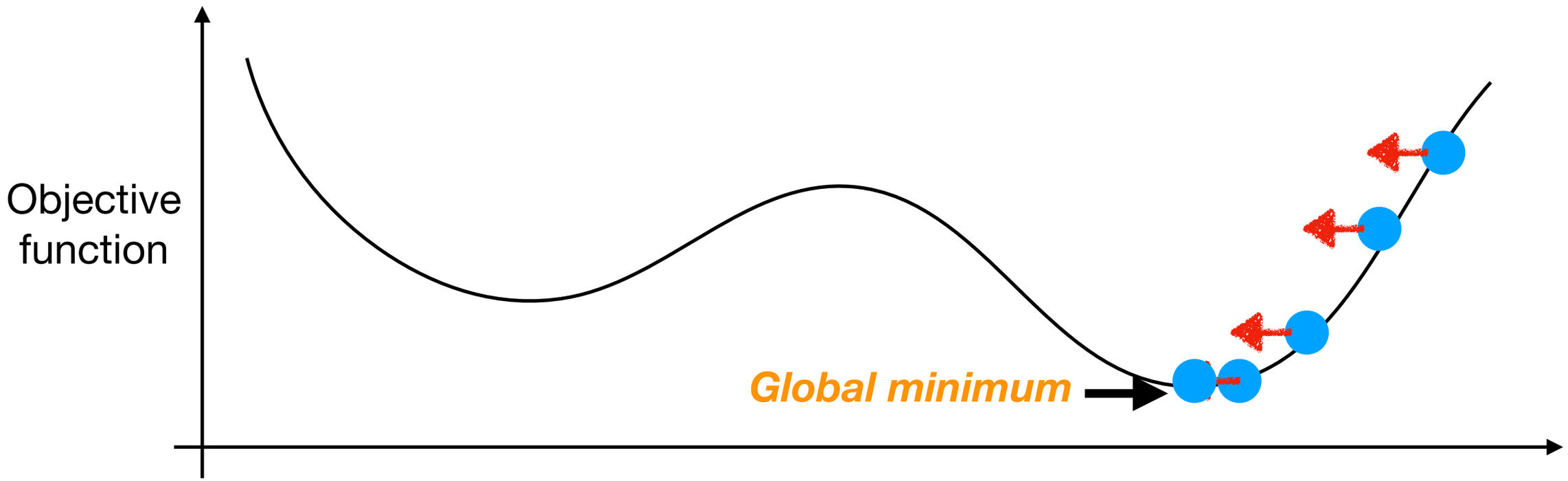
Total WCV is small

K-means: A difficult problem

Goal: $\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} WCV(C_k)$



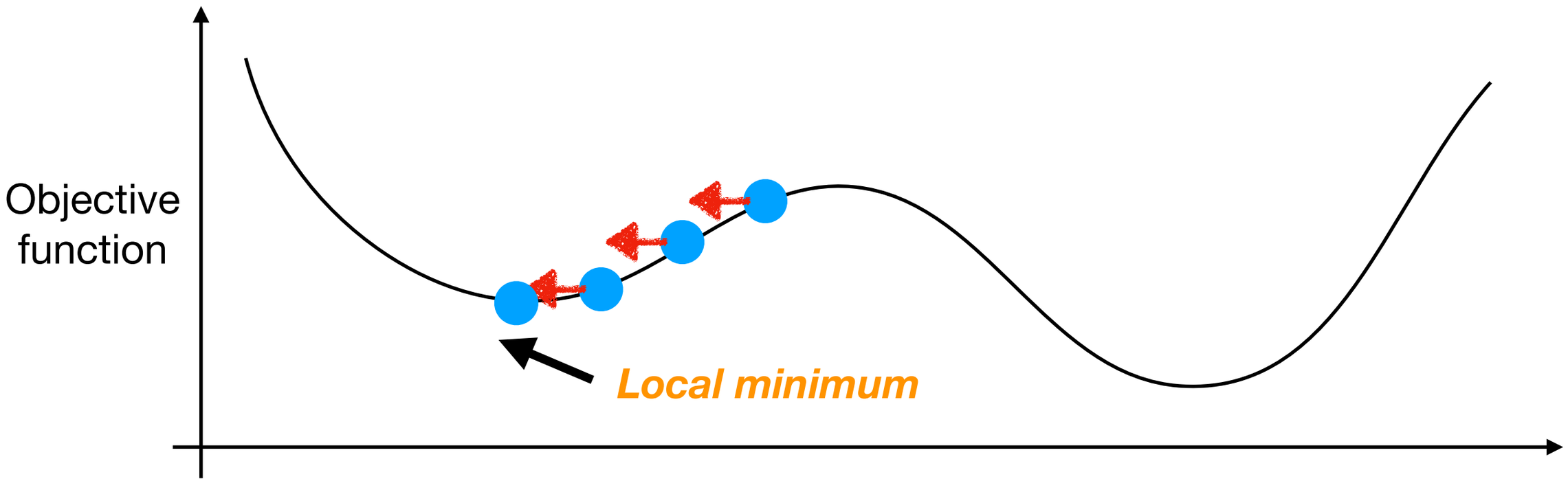
- Minimizing the total within-cluster variation is difficult because the objective function is not convex. Most algorithms are unable to guarantee that you'll find the best solution.



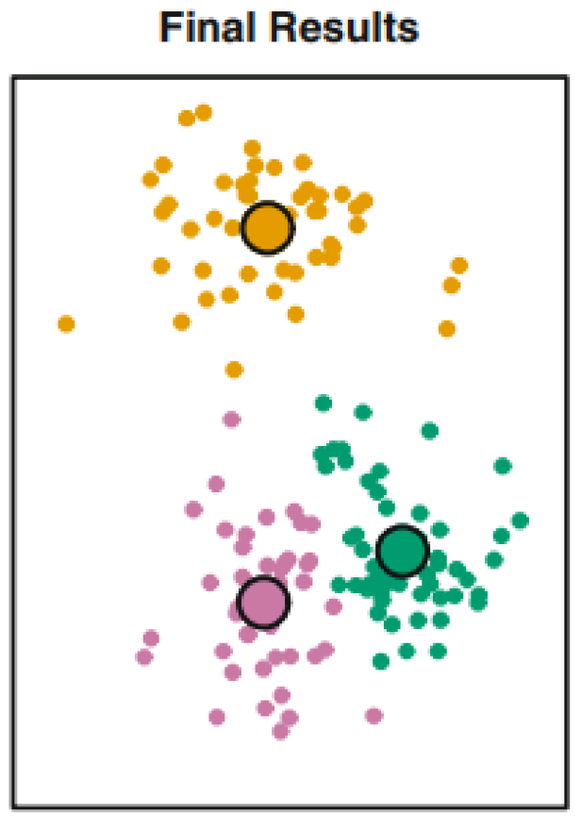
K-means: A difficult problem

Goal: $\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} WCV(C_k)$

- Minimizing the total within-cluster variation is difficult because the objective function is not convex. Most algorithms are unable to guarantee that you'll find the best solution.



K-means: A reformulation



The **centroid** of a cluster is the point where the *i*th element is the mean of each of the *i*th variable for all observations assigned to that cluster.

Let the centroid of cluster C_k be denoted μ_k. We can show that:

$$\underset{C_1,\ldots,C_K}{\arg\min} \sum_{k=1}^{K} WCV(C_k) \quad \Leftrightarrow \quad \underset{C_1,\ldots,C_K}{\arg\min} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2$$

Minimizing the total within-cluster variation is equivalent to minimizing the total average squared distance to the centroid.

Average squared Euclidean distance to the centroid

K-means: A greedy approach

- Using the idea of centroids, we can approximately solve the K-means problem using an iterative greedy algorithm.

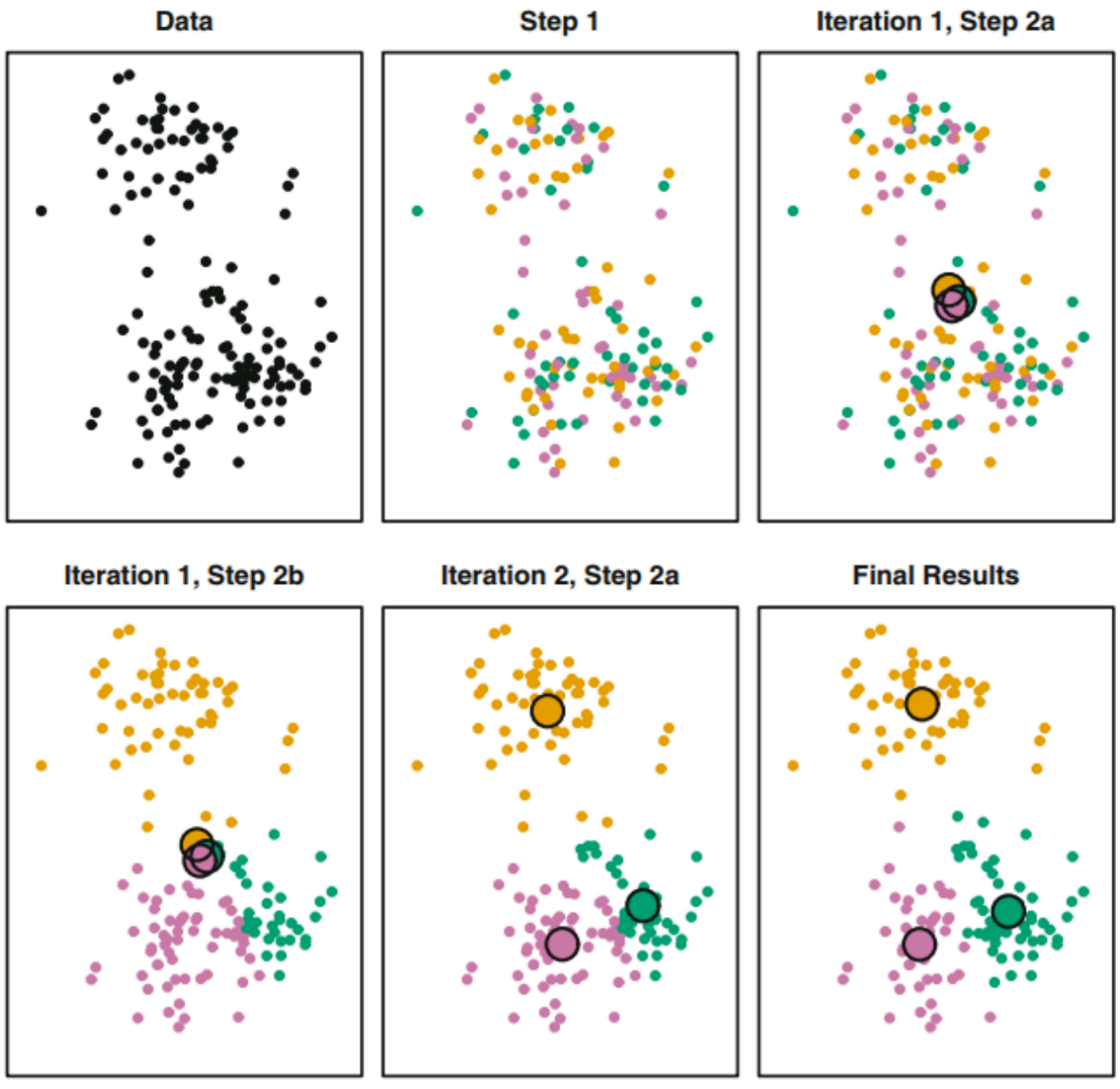
1. Randomly assign each observation to one of K clusters.

2. Iterate until convergence:

   A. Compute the centroid of each cluster given the current cluster assignments.

   B. Assign each observation to the cluster whose centroid is the closest.

Example

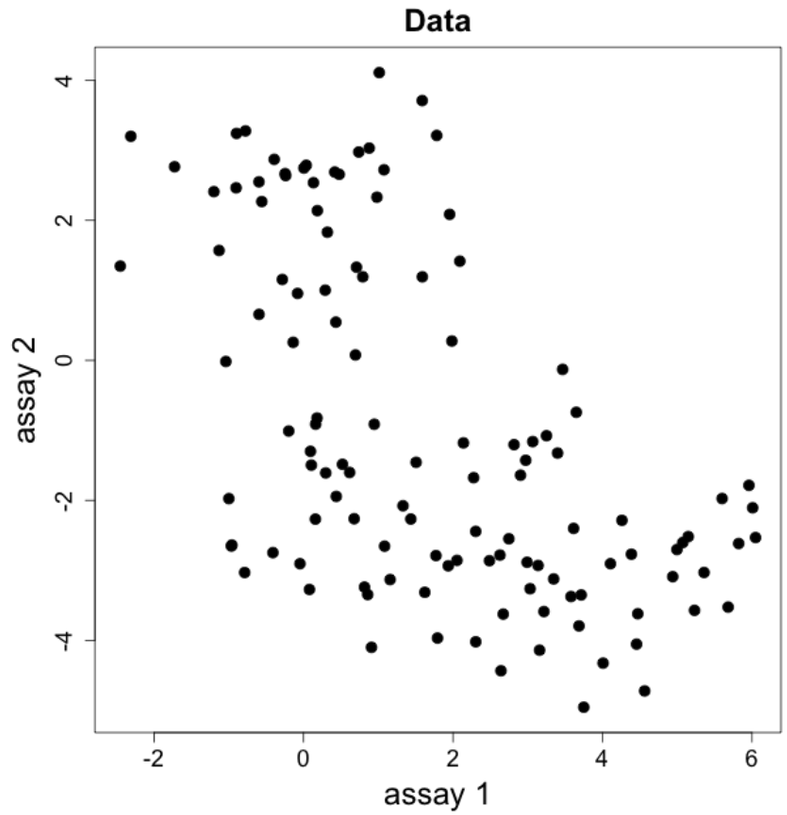

Data
Step 1
Iteration 1, Step 2a
Iteration 1, Step 2b
Iteration 2, Step 2a
Final Results

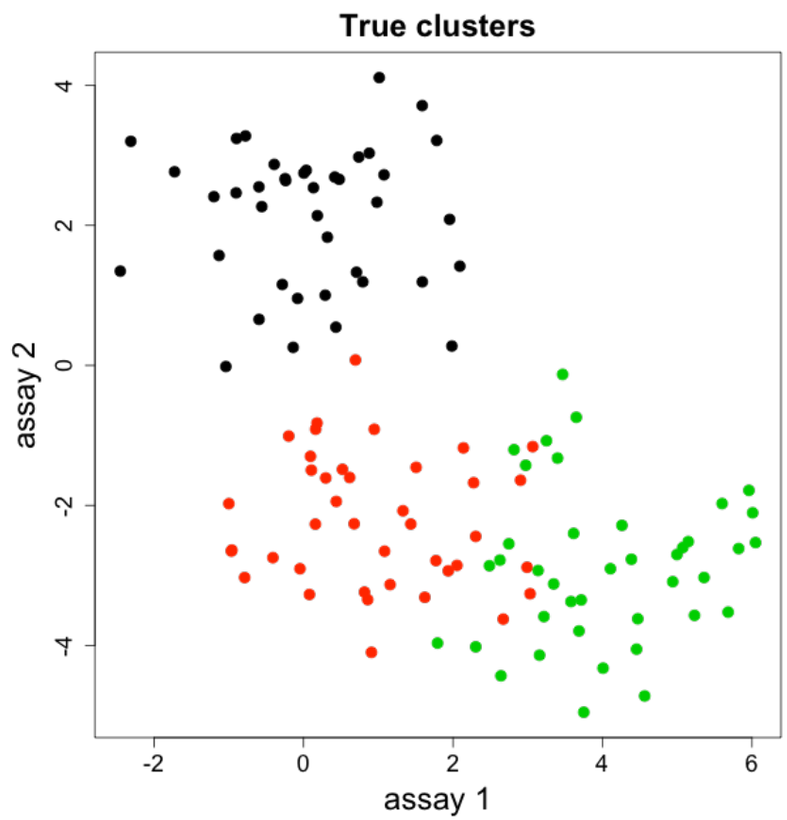
Example: Different initializations

Because the objective function is not convex, different initializations will give different solutions!

Try multiple initializations.

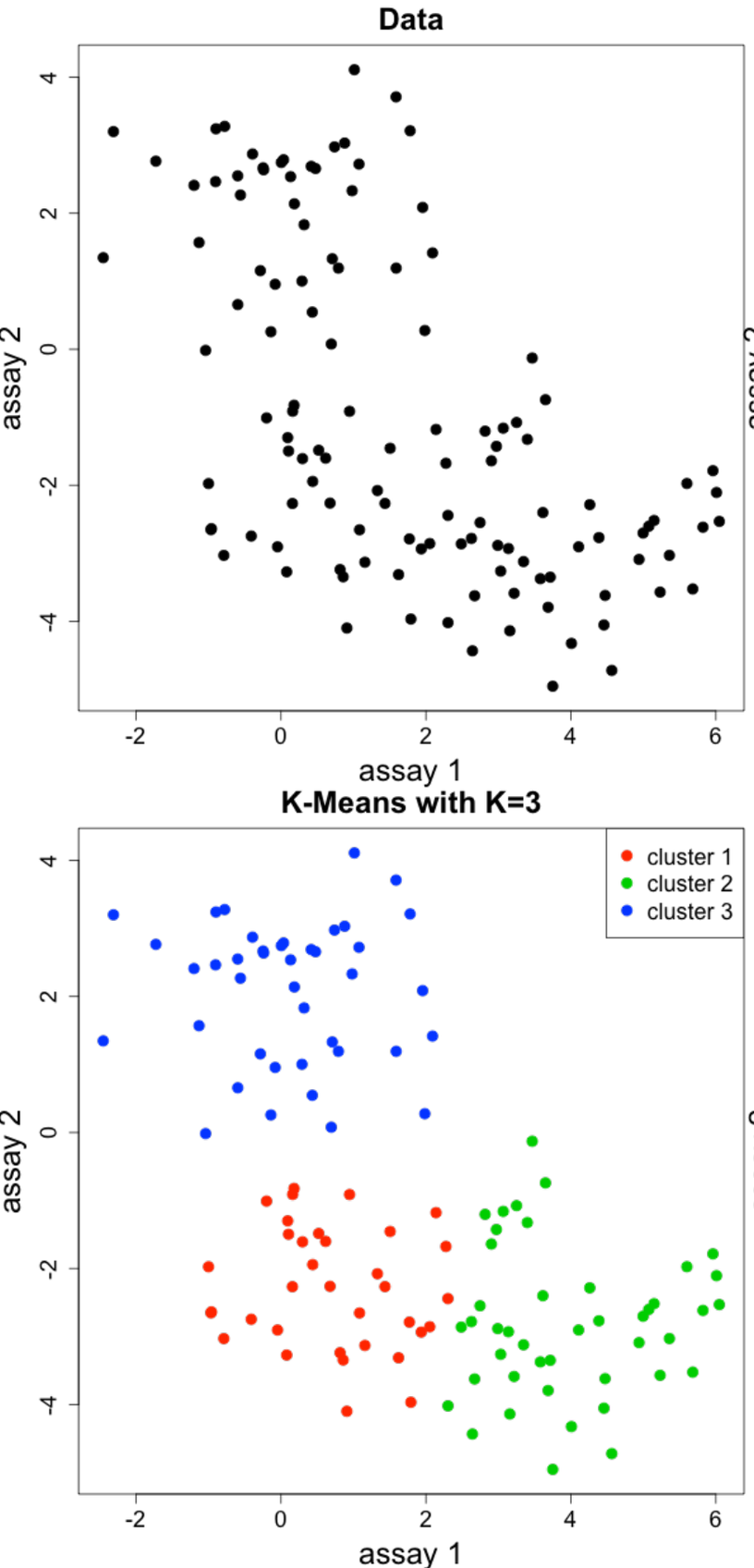
Example: K-means in 2 dimensions

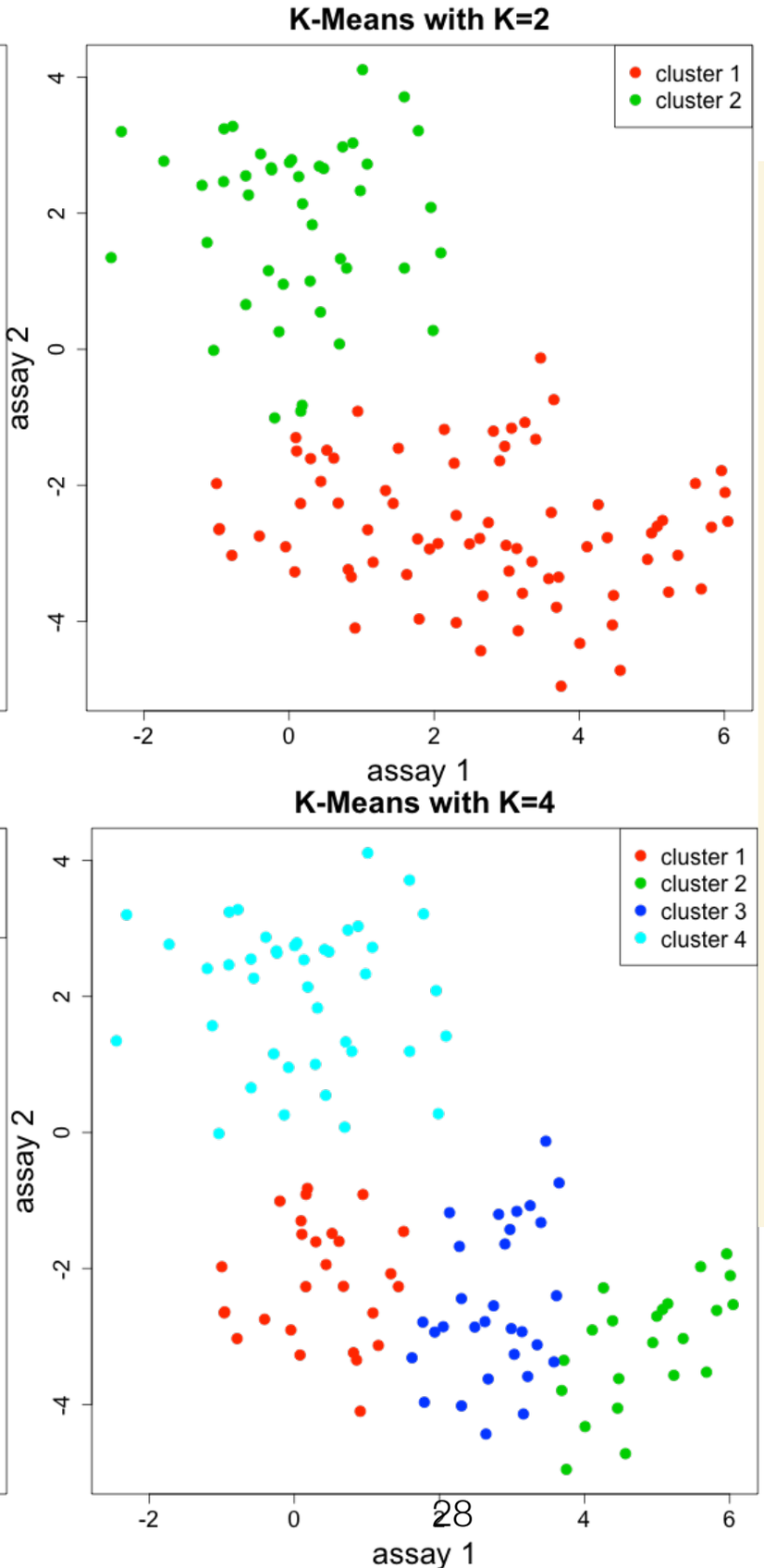




- Let's generate data from three classes.
- The points from each class are normally distributed around different means.

Example: K-means in 2 dimensions



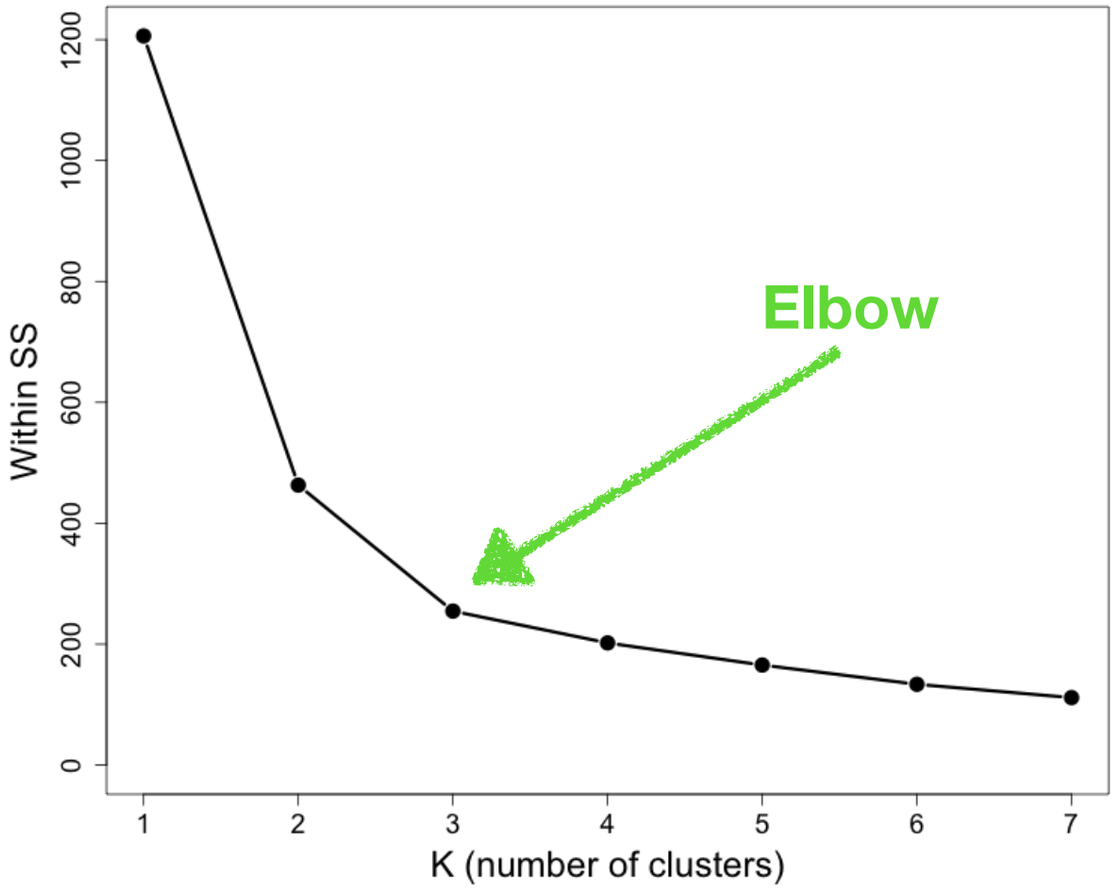


nstart = Number of initializations

```
> km.out2 <- kmeans(x, centers=2, nstart=20)
> 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
 [24] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1
 [47] 2 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
 [70] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [93] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[116] 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"
[4] "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
```

K-means: Picking the number of clusters

- **Q**: How can we pick the number of clusters *K*?



Example: How to get the total within cluster sum of squares for *K*=5

```
> km.out5 <- kmeans(x,5, nstart=20)
> km.out5$tot.withinss
[1] 165.1528
```

- Can we use a validation set?

K-means: Picking the number of clusters

- Elbow: Plot within-cluster variation versus K and select the "elbow"

- Gap Statistic: compares fitted cluster on observed data versus random uniform points

- Silhouette Statistic: compares within-cluster dissimilarity to between-cluster dissimilarity

- Cluster Stability: perturb data to evaluate stability of cluster assignments

- …

K-means: Summary

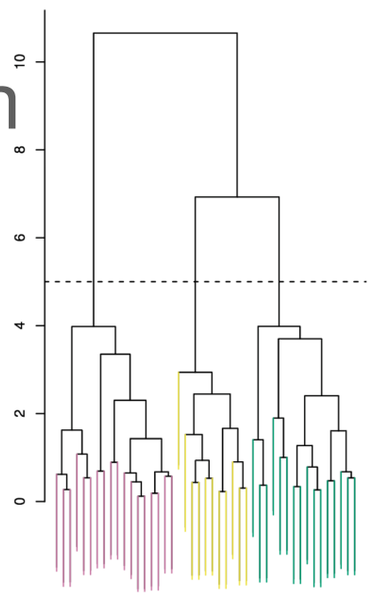
- Strengths:
  - Simple and fast algorithm
- Weaknesses:
  - Picking the number of clusters can be challenging
  - Only suitable when clusters are hyper-ellipsoids
  - Sensitive to the cluster initializations
  - Not suitable if number of features is much bigger than the number of observations

Today's lecture

1. Unsupervised learning
2. Clustering
    A. K-means
    B. Hierarchical Clustering
3. Dimension Reduction
    A. PCA

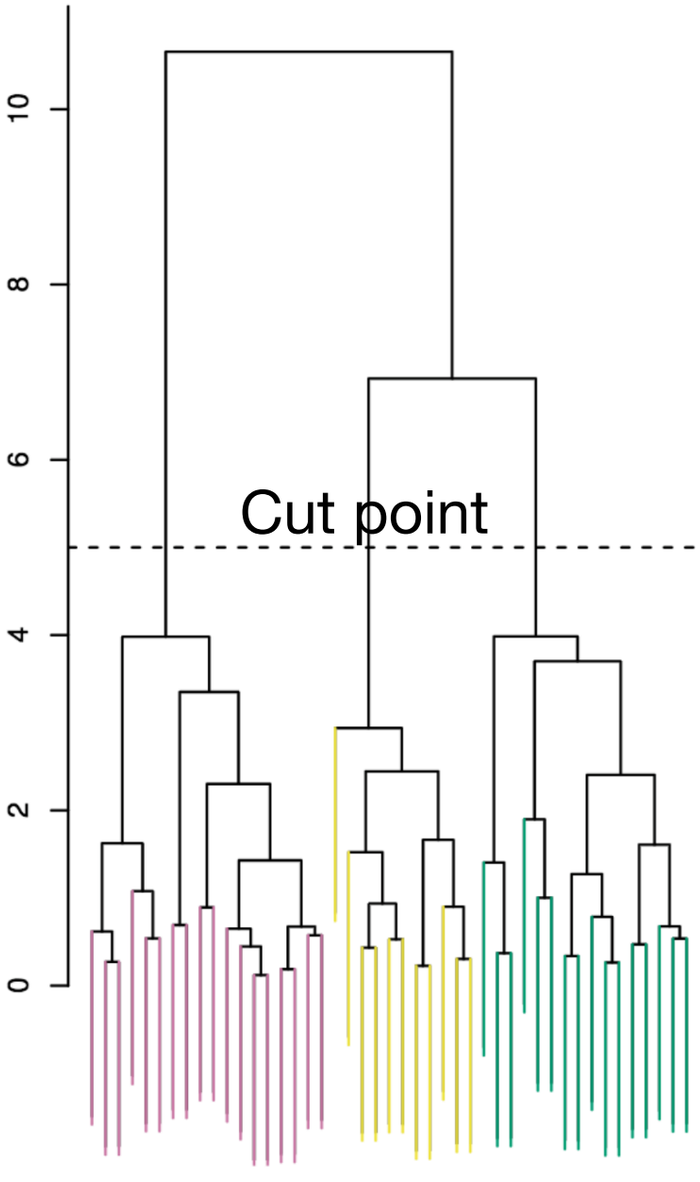
Hierarchical clustering

- Idea: Build a hierarchy of clusters from the observed data, where going up a level of the hierarchy merges clusters at the current level.
  - Bottom level: Each cluster corresponds to a single observation
  - Top level: The cluster contains all observations
- Assumption: No assumptions on the dissimilarity measure
- Output: Typically a dendrogram

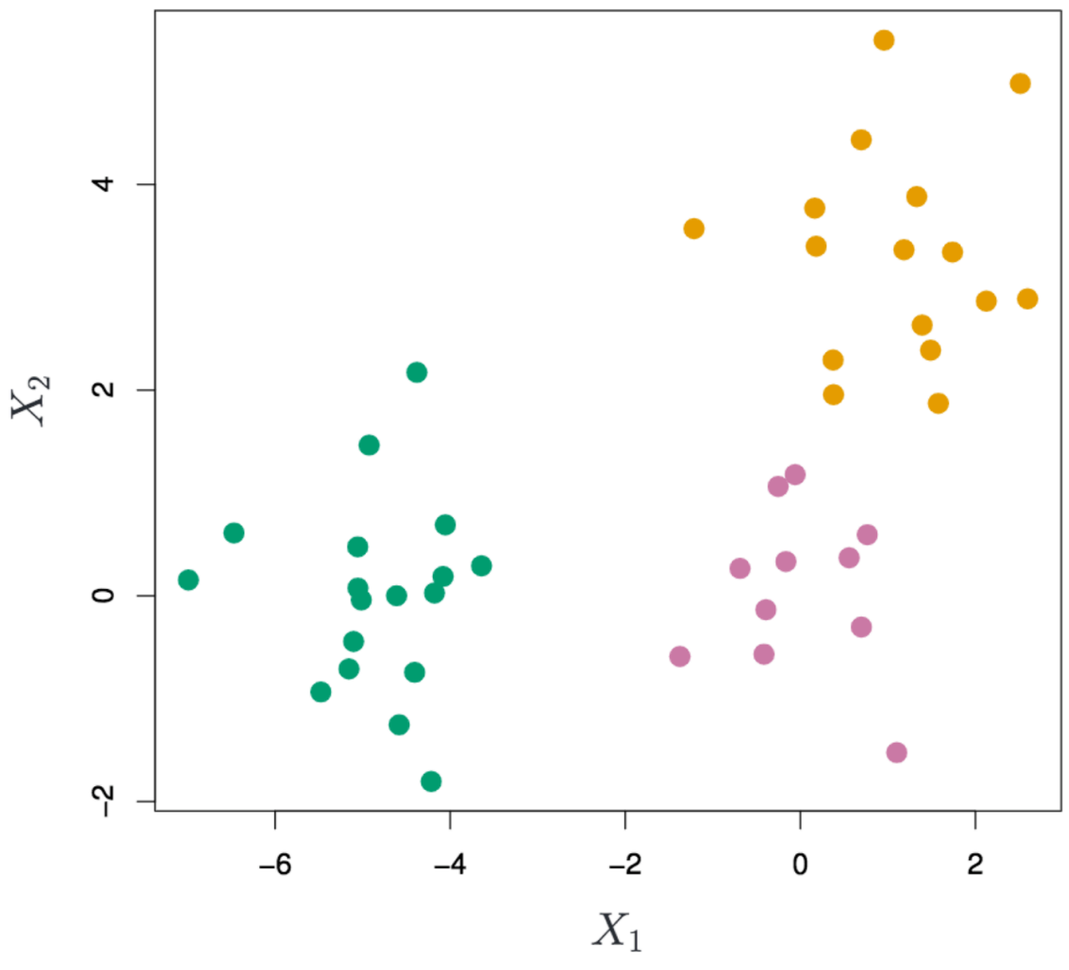


Interpreting a dendrogram

Node height indicates similarity of objects



Leaves are observations



Hierarchical clustering

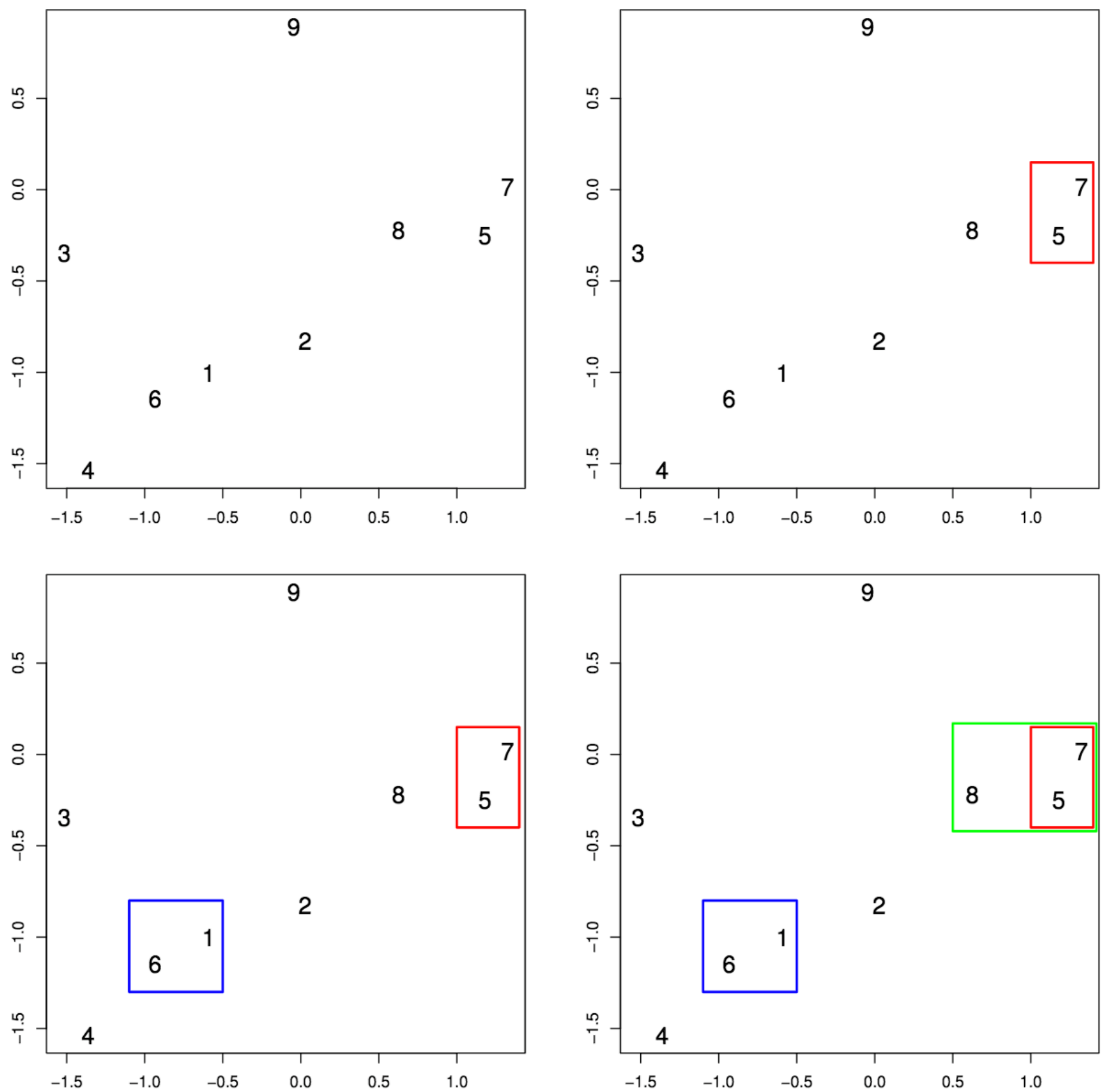
1. Begin with every observation representing a singleton cluster

2. Repeat until there is only a single cluster:

   A. Merge the two closest clusters into a single cluster.

Monotonicity property:
The dissimilarity between merged clusters is monotonically nondecreasing with the level of the merger.

Hierarchical clustering



Hierarchical clustering

1. Begin with every observation representing a singleton cluster

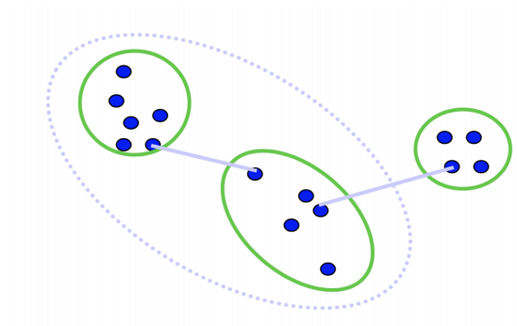
2. Repeat until there is only a single cluster:

   A. Merge the two **“*closest*”** clusters into a single cluster.

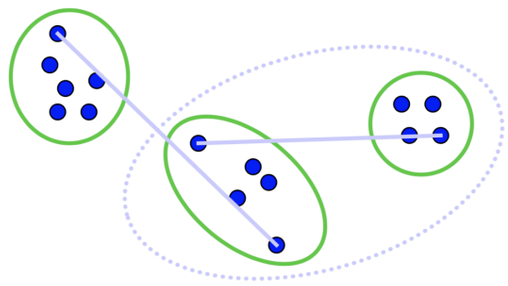
How do we measure dissimilarity between two clusters?

Ways to measure cluster distance

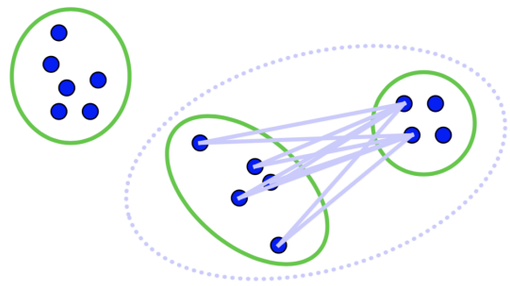
- Define cluster distance between C and C' using:
  - Single linkage: the minimum distance between points in C and C'



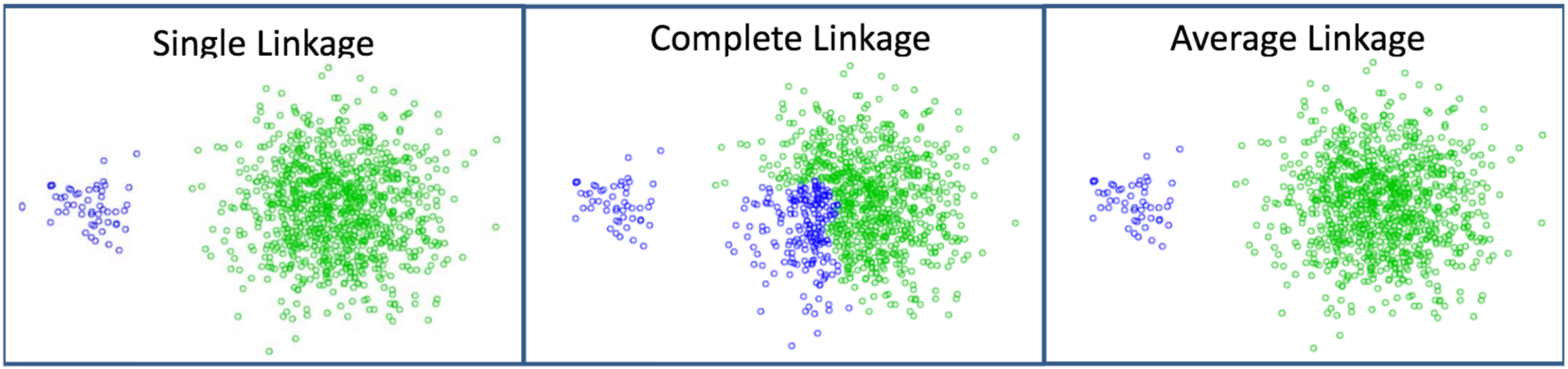
  - Complete linkage: the maximum distance between points in C and C'



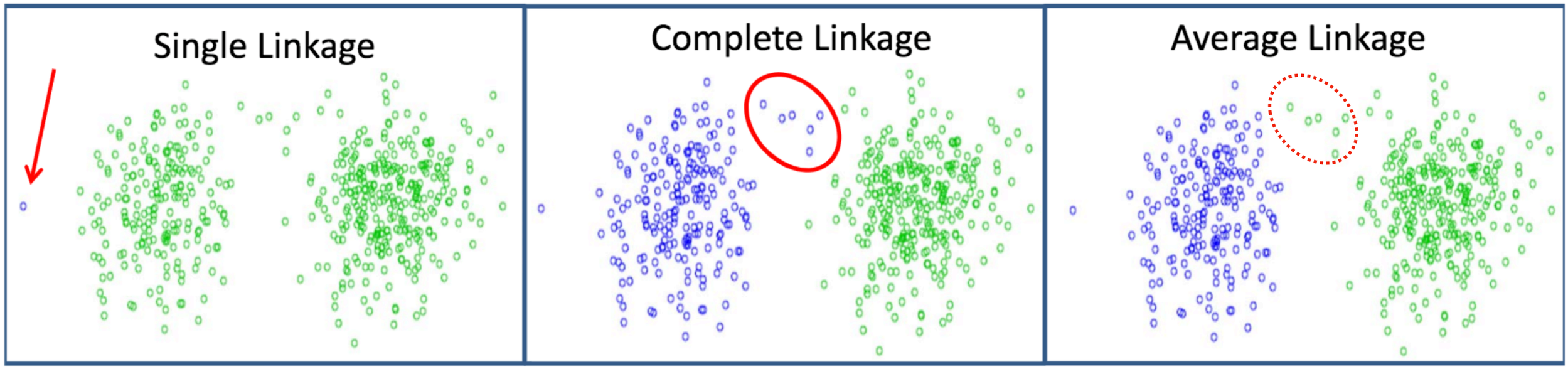
  - Average linkage: mean distance between points in C and C'



Linkage Examples

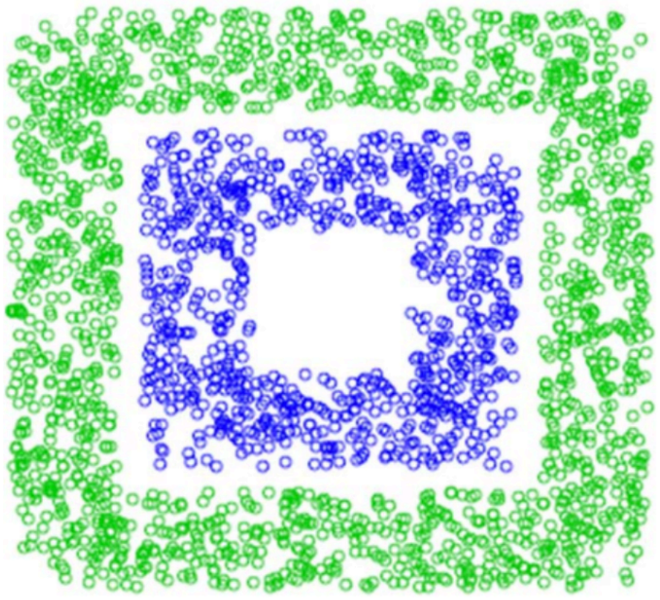


Linkage Examples

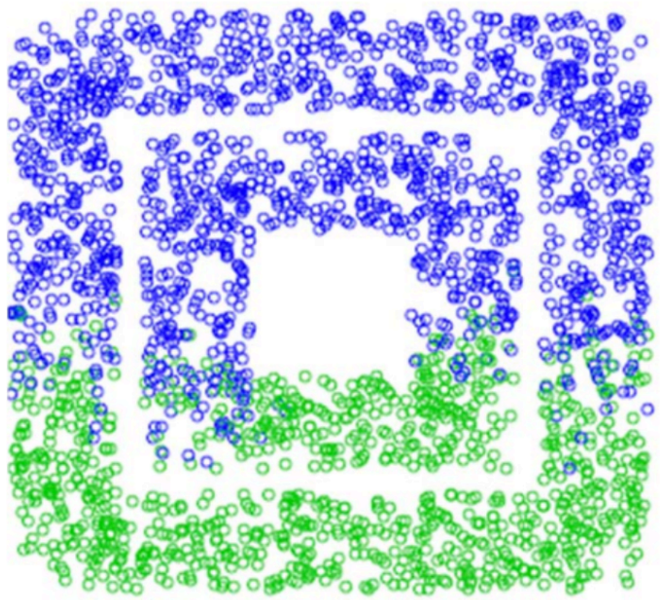


Linkage Examples

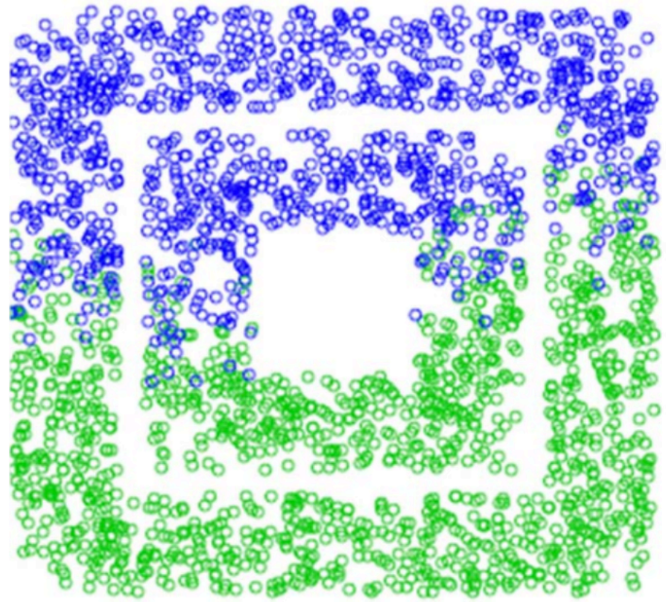
Single Linkage



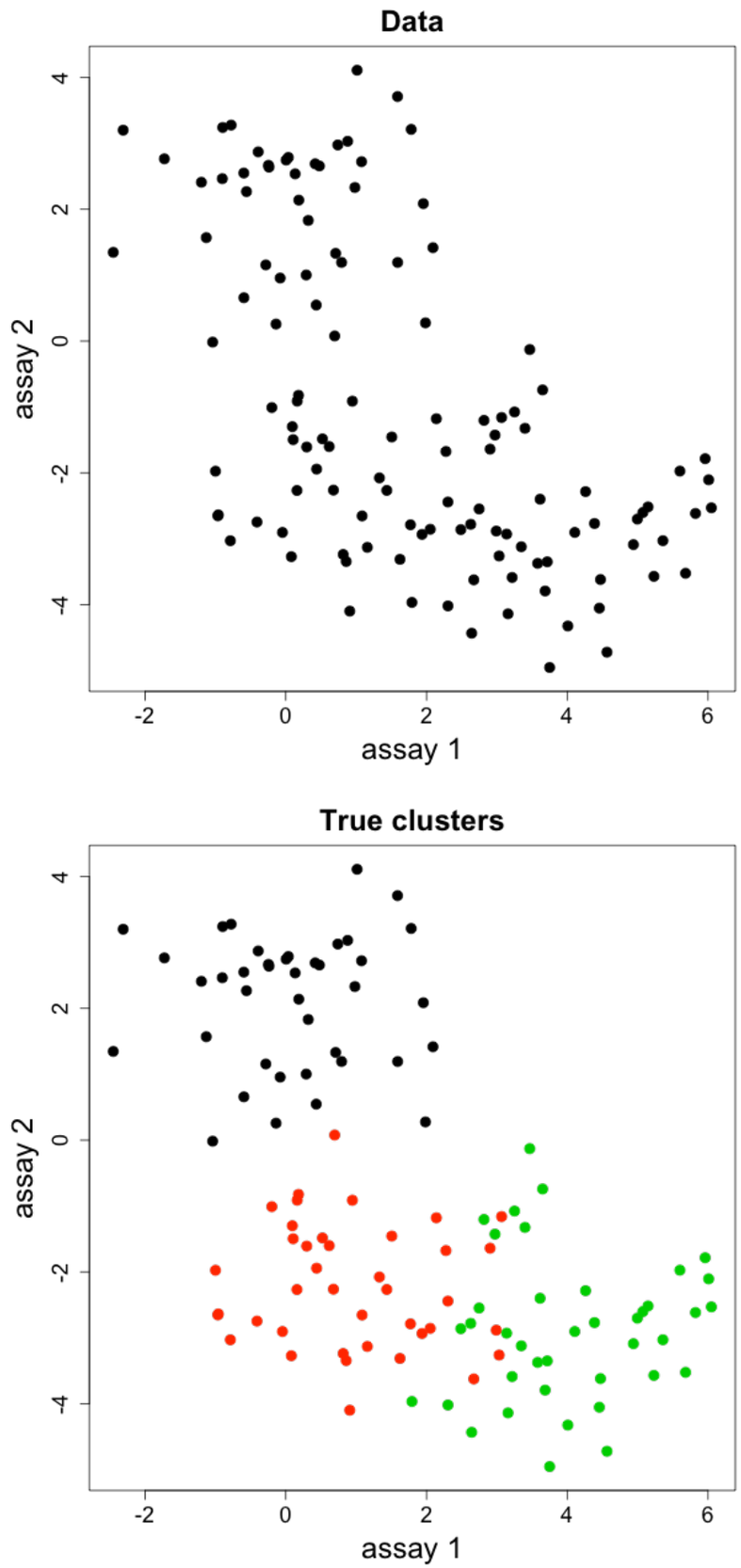
Complete Linkage



Average Linkage



Example: Hierarchical clustering



- Let's use the same data from the K-means section. Data is generated from three classes. The points from each class are normally distributed around different means.

- **Q**: What linkage do you think will be most effective?

- **A**: Complete

- **B**: Single

- **C**: Average

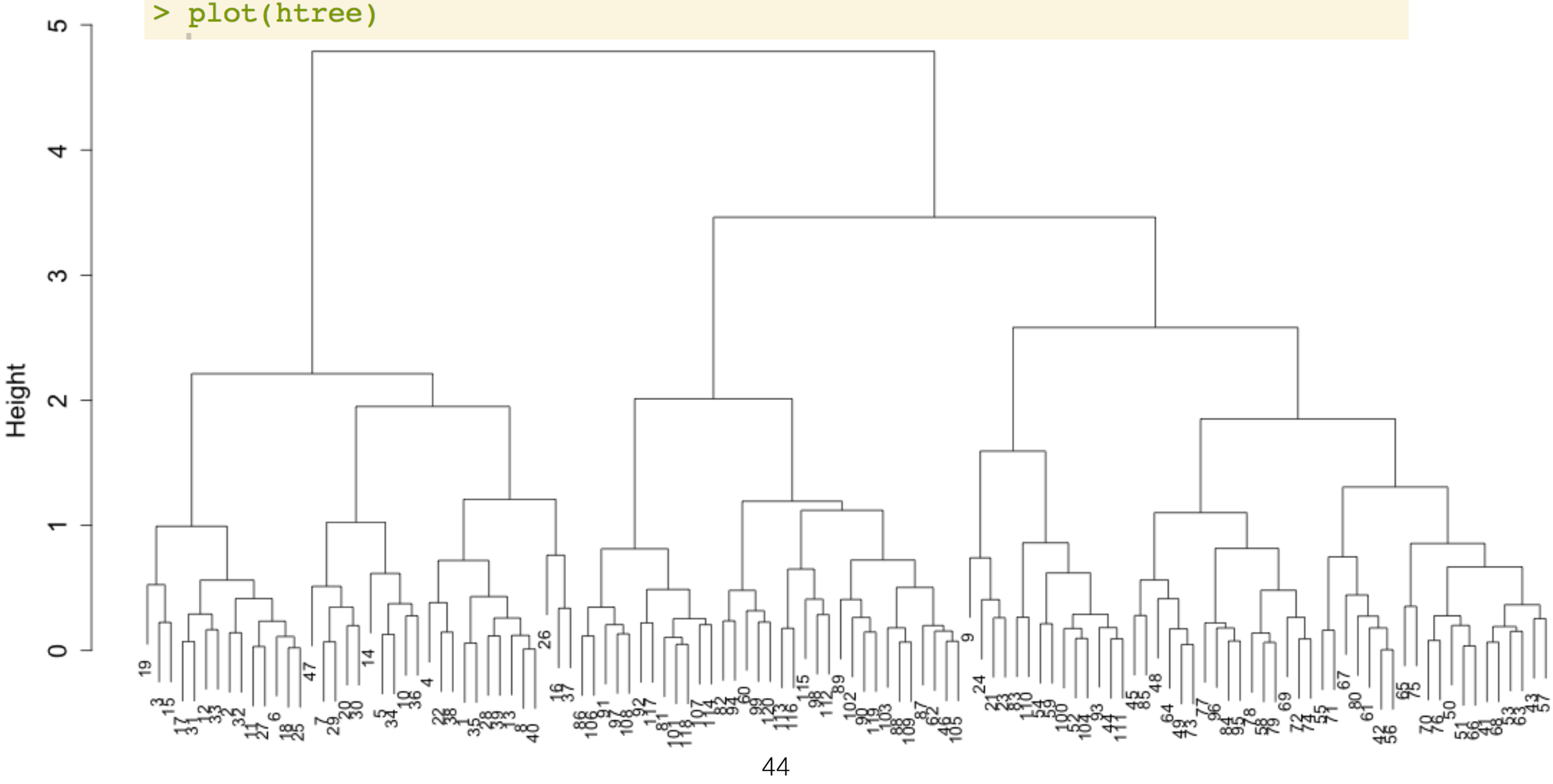
Example: Hierarchical clustering

```
> xsc <- scale(x)
> htree <- hclust(dist(xsc), method="complete")
> cutree(htree, 2)
  [1] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 2 2 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 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 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
> plot(htree)
```

1. Scale the points so that the standard deviation of each variable is 1.
2. Calculate the distance between points using `dist`. Perform hierarchical clustering with complete linkage.
3. Get the cluster assignments at depth 2.
4. Plot!

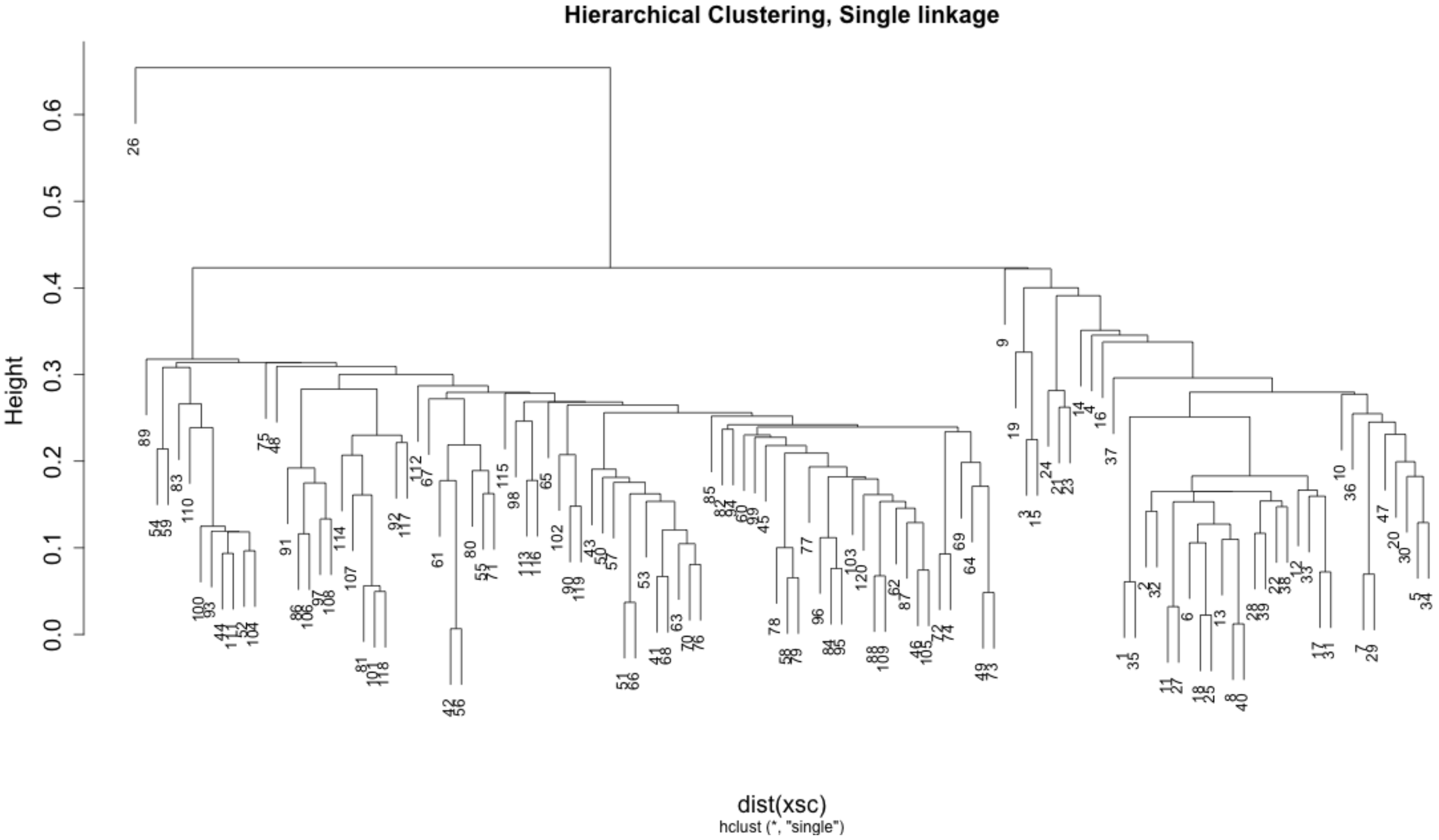
Example: Hierarchical clustering

```
> xsc <- scale(x)
> htree <- hclust(dist(xsc), method="complete")
> cutree(htree, 2)
  [1] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 2 2 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 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 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
> plot(htree)
```



Example: Hierarchical clustering

```
> htree <- hclust(dist(xsc), method="single")
```



Example: Comparing clusterings on this dataset

- Since we know the true labels on this dataset, let's compare how well the different linkage methods performed.

- The Rand Index computes the similarity between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings.

- A Rand Index of 1 means that the clusterings are in perfect agreement.

```
> s_assign <- cutree(hclust(dist(xsc),method="single"), 3)
> c_assign <- cutree(hclust(dist(xsc),method="complete"), 3)
> a_assign <- cutree(hclust(dist(xsc),method="average"), 3)
> adjustedRandIndex(s_assign, dat$group)
[1] 0.5430842
> adjustedRandIndex(c_assign, dat$group)
[1] 0.602639
> adjustedRandIndex(a_assign, dat$group)
[1] 0.6645349
```

Average linkage on this dataset performed the best

Hierarchical clustering: Summary

- Strengths:
  - Simple, intuitive
  - Clusters are nested
- Weaknesses:
  - Can be unstable
  - Depends heavily on type of linkage
  - Difficult to analyze theoretical properties

Today's lecture

1. Unsupervised learning
2. Clustering
    A. K-means
    B. Hierarchical Clustering
3. **Dimension Reduction**
    A. PCA

Dimension reduction

"Vector space embeddings"

"Manifold learning"

- In high-dimensional datasets, the features are typically related (e.g. correlated).

- Dimension reduction tries to retain the most important information in a small number of features.

Big data matrix

$$X = \begin{array}{|c|c|c|c|c|c|} \hline 0.0 & 1.4 & -30 & 0 & -5.3 & \dots \\ \hline 0.2 & -1.3 & -4 & 1 & -4.2 & \dots \\ \hline 0.9 & 0.2 & -10 & 0 & -3.9 & \dots \\ \hline \dots & \dots & \dots & \dots & \dots & \dots \\ \hline \end{array}$$

Dimension reduction →

Latent factors

$$Z = \begin{array}{|c|c|} \hline 0.2 & 0.3 \\ \hline 0.8 & -0.6 \\ \hline -0.1 & 0.2 \\ \hline \dots & \dots \\ \hline \end{array}$$

Dimension reduction

Big data matrix

$$X = \begin{array}{|c|c|c|c|c|c|} \hline 0.0 & 1.4 & -30 & 0 & -5.3 & \dots \\ \hline 0.2 & -1.3 & -4 & 1 & -4.2 & \dots \\ \hline 0.9 & 0.2 & -10 & 0 & -3.9 & \dots \\ \hline \dots & \dots & \dots & \dots & \dots & \dots \\ \hline \end{array}$$

Dimension reduction →

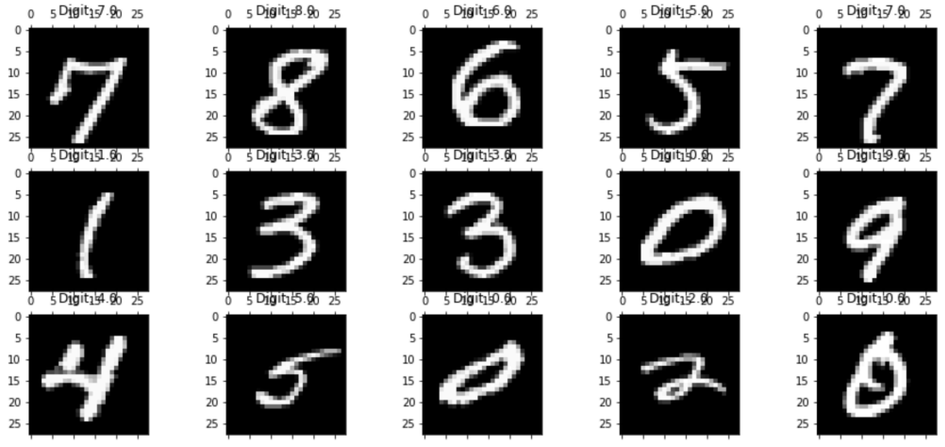
Latent factors

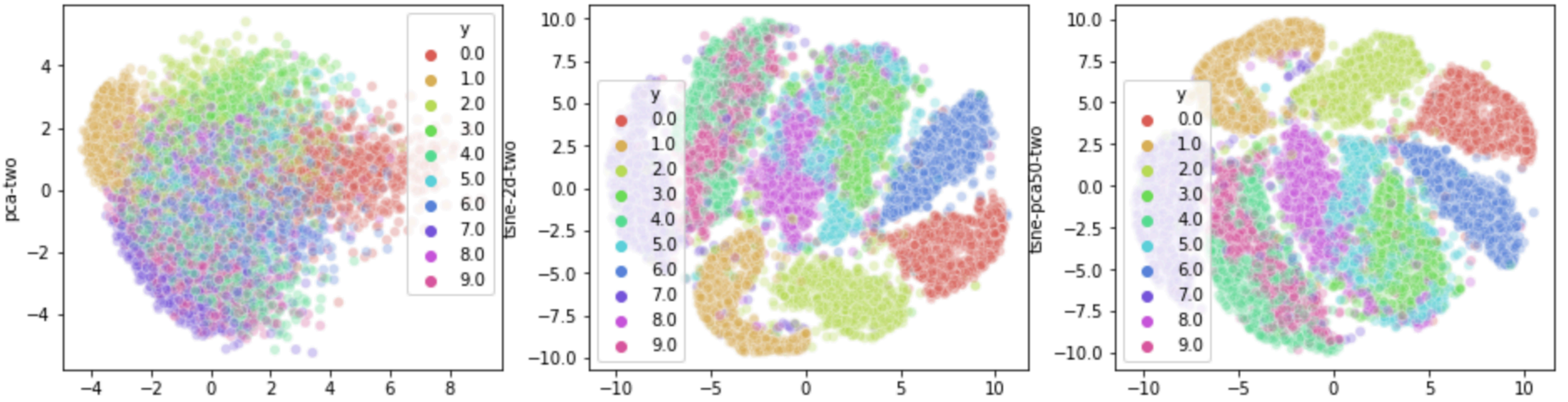
$$Z = \begin{array}{|c|c|} \hline 0.2 & 0.3 \\ \hline 0.8 & -0.6 \\ \hline -0.1 & 0.2 \\ \hline \dots & \dots \\ \hline \end{array}$$

- Linear methods: PCA, NMF, MDS, …
- Non-linear methods: Kernel PCA, Autoencoder, t-SNE, …

Dimension reduction: Application

- Apply dimension reduction prior to clustering





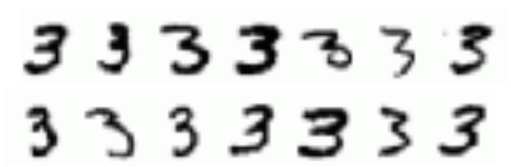
PCA **t-SNE** **PCA and then t-SNE**

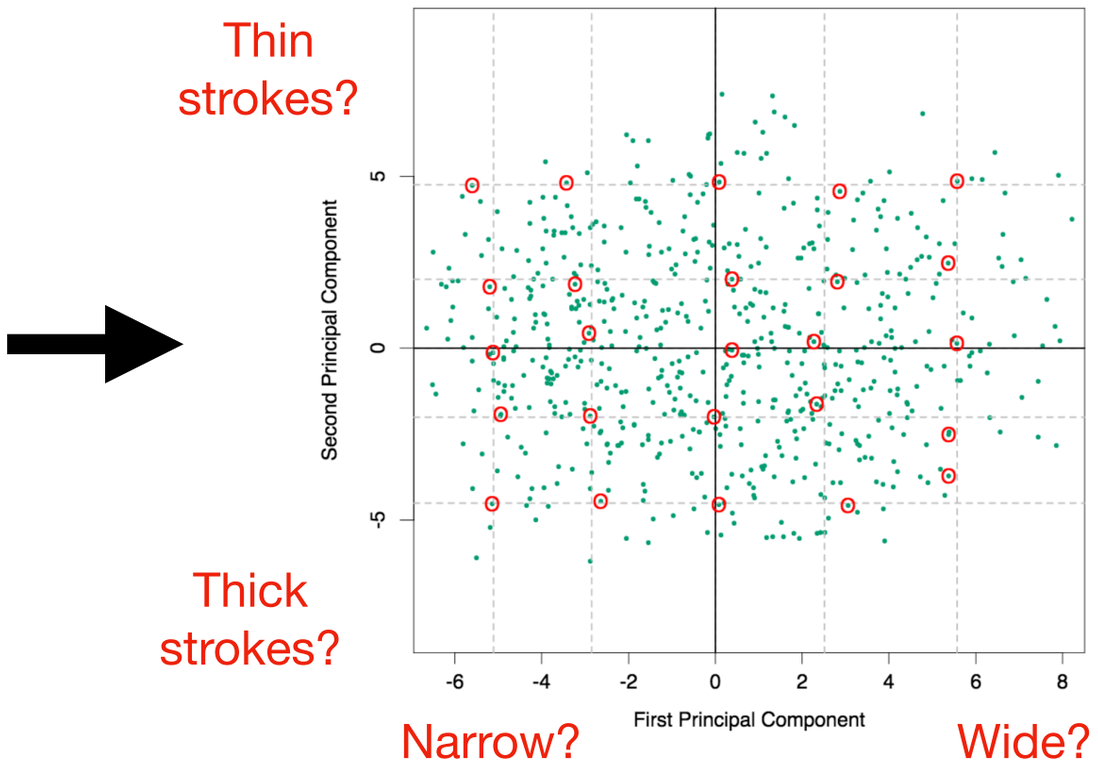
https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

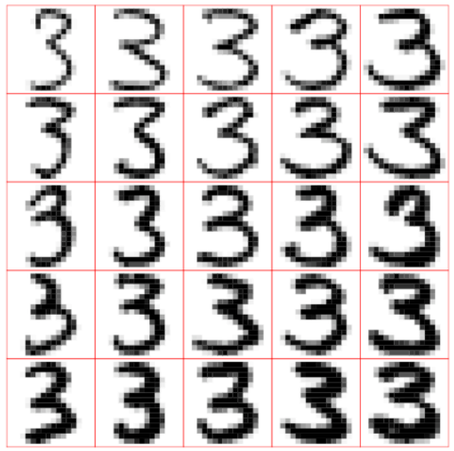
Dimension reduction: Applications

- Understand main sources of variation in the data

High-dimensional image data



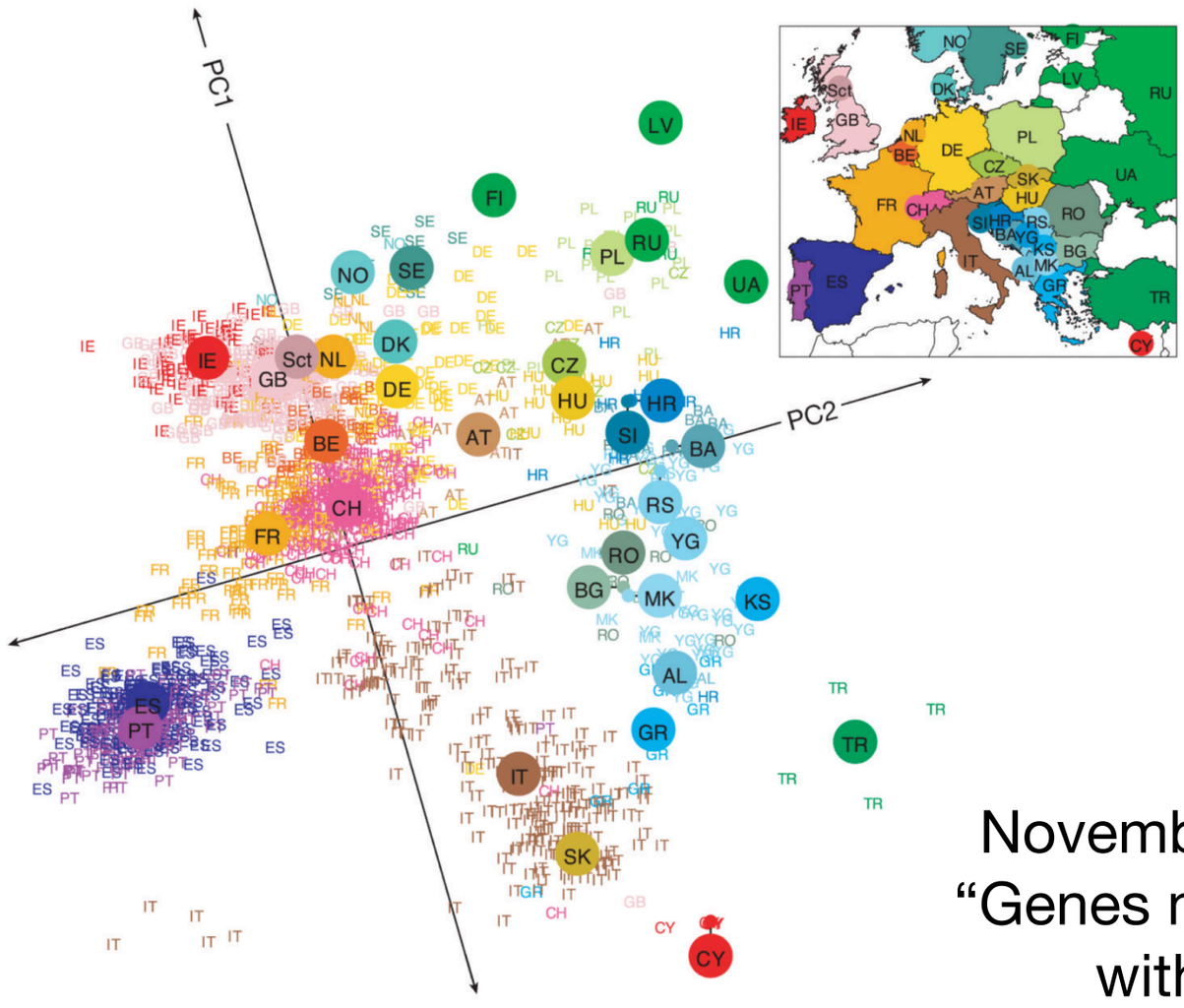




Dimension reduction: Applications

- Understand main sources of variation in the data

High-dimensional genetic data



Novembre, et. al. 2008. “Genes mirror geography within Europe”

Dimension reduction: Applications

- Pre-process data to use as input into a supervised learning algorithm

High-dimensional image data

$X \in \mathbb{R}^{784}$

$Y \in \{0,...,9\}$

Latent factors

0.2	0.3
0.8	-0.6
-0.1	0.2
…	…

Apply linear regression to this instead

$X \in \mathbb{R}^{2}$

$Y \in \{0,...,9\}$

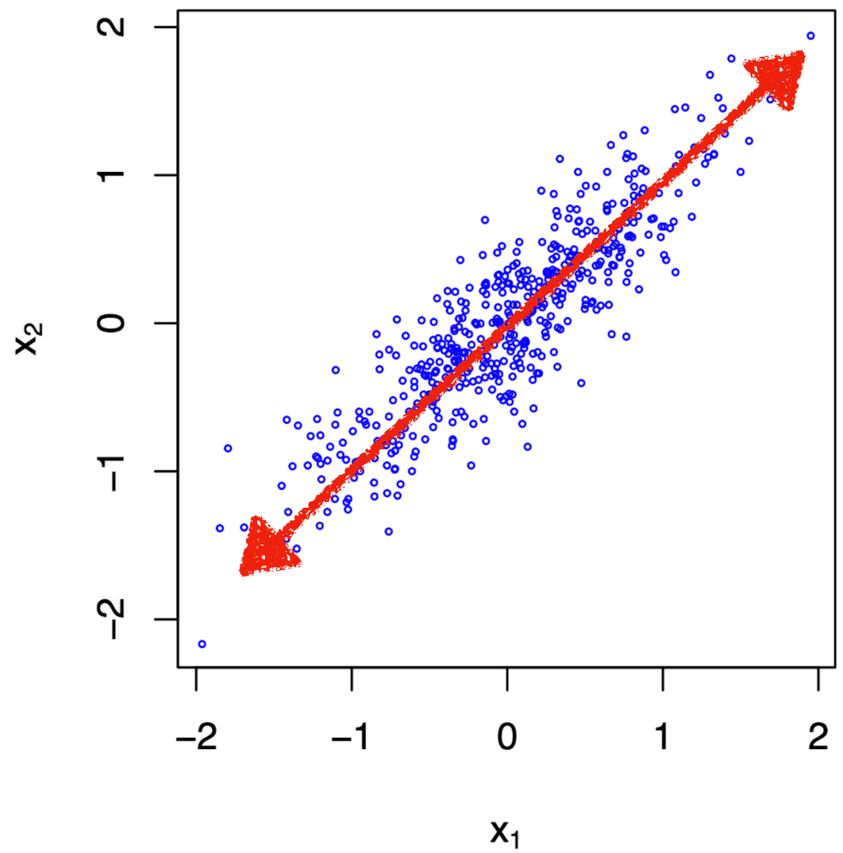
Today's lecture

1. Unsupervised learning
2. Clustering
    A. K-means
    B. Hierarchical Clustering
3. Dimension Reduction
    A. PCA

Principal Components Analysis (PCA)

- Goal: Find K orthogonal axes (principal components) that capture the maximum variance of the data and are mutually uncorrelated.

Example from 2D to 1D



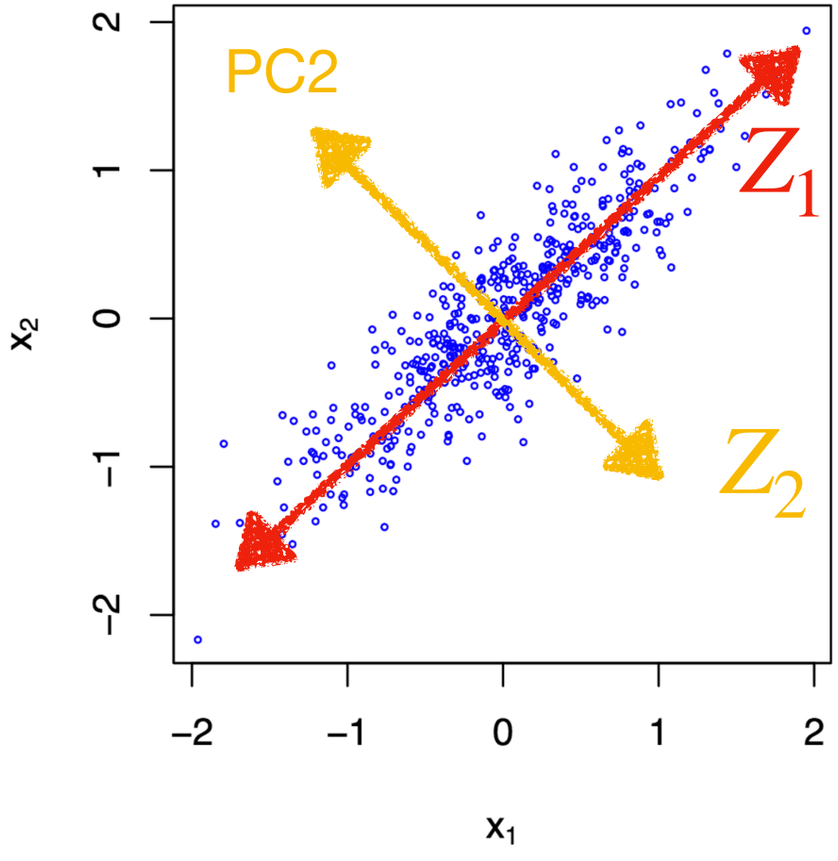
$$Z_1 = v_1 X_1 + v_2 X_2$$

Principal component 1 (PC1)

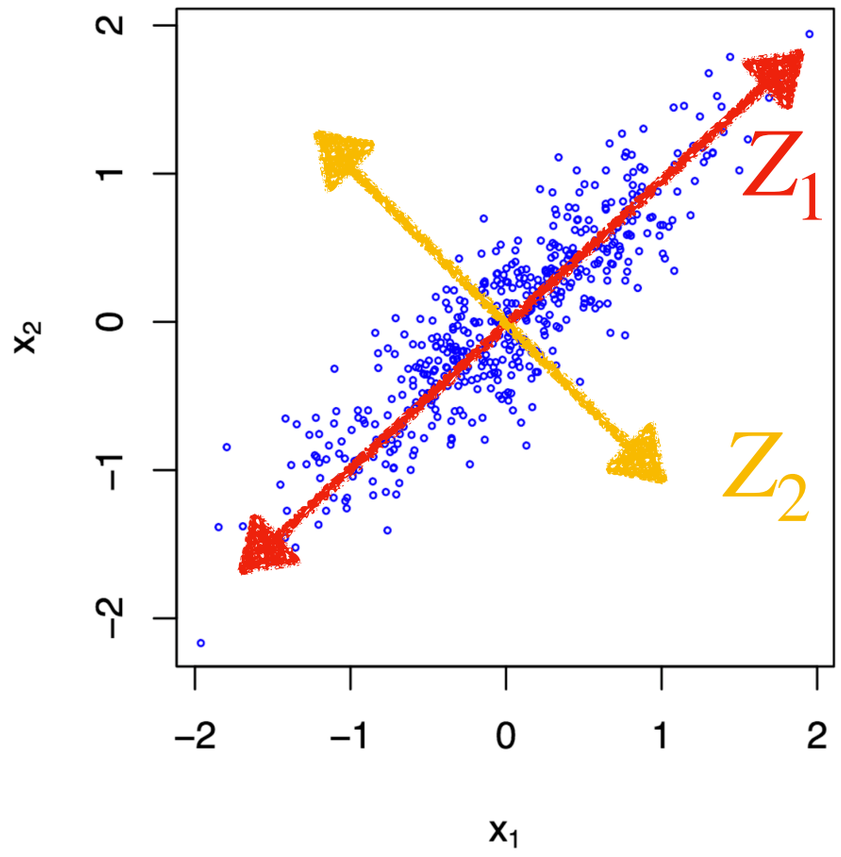
Principal Components Analysis (PCA)

- To find an additional principal component, choose a linear combination that:
  - Is orthogonal/perpendicular to previous ones
  - Explains the maximum variation in the data

Example:
PCA from 2D
to 2D



PCA concepts



$$Z_1 = v_{1,1}X_1 + v_{1,2}X_2$$

$$Z_2 = v_{2,1}X_1 + v_{2,2}X_2$$

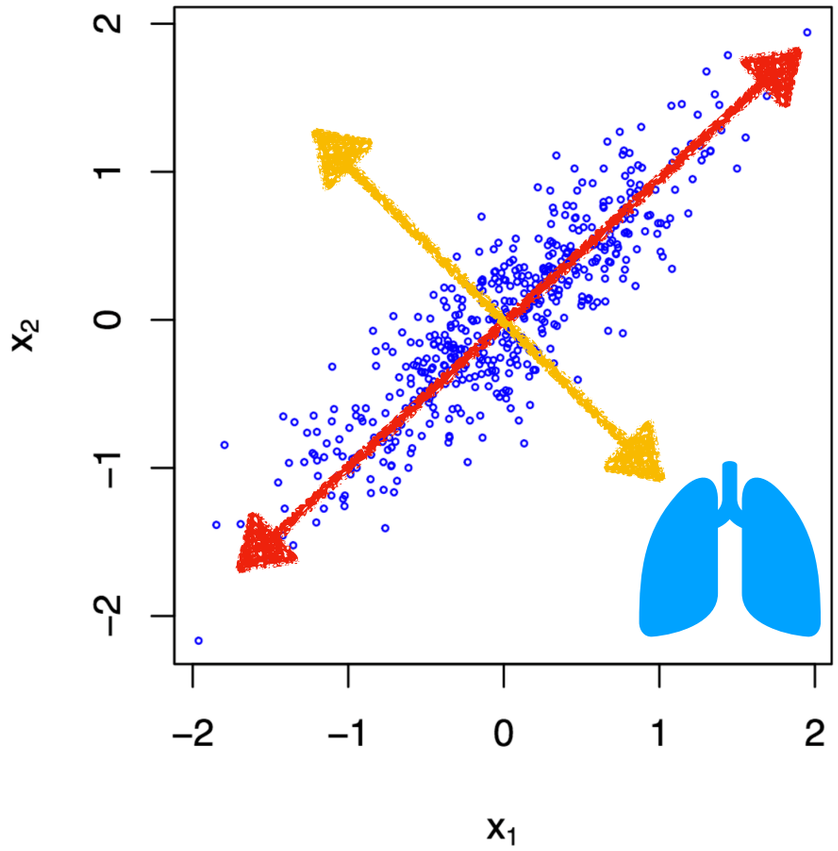
The PC loadings are $v_1 = (v_{1,1}, v_{1,2})$ and $v_2 = (v_{2,1}, v_{2,2})$. They are the directions in which the data vary the most.

The PC scores are given by Z_1 and Z_2. They are the new coordinates if we project the data onto the loading vectors.

Example: PC loadings and scores

Health of heart

Suppose we find that the first PC in our dataset correspond to the health of a patient's heart and the second PC corresponds to the health of the patient's lungs.



Health of lungs

Interpretation:

The <u>loadings</u> for the first PC contains the association between each variable and the health of the heart.

The <u>scores</u> for the first PC tell us how healthy the patient's heart is.

Principal Components Analysis (PCA)

For $k = 1, \ldots, K$

Solve $\max_{v_k : \|v_k\|_2 = 1} \underbrace{v_k^\top X^\top X v_k}$ subject to $v_k^\top v_j = 0 \quad \forall j < k$

Variance of X along axis defined by v_k

Define the kth PC as $Z_k = X v_k$.

Turns out this is simply doing a Singular Value Decomposition!

(Descending order along diagonal)

Perform Singular Value Decomposition $X = UDV^\top$

The PC scores are given by $Z_1, \ldots, Z_K$ where $Z = XV$.

The PC loadings are $v_1, \ldots, v_k$.

Principal Components Analysis (PCA)

- Mathematically, PCA is accomplished using singular value decomposition.

- There are many packages you can use to perform PCA:
  - prcomp() (stats)
  - princomp() (stats)
  - PCA() (FactoMineR)
  - dudi.pca() (ade4)
  - acp() (amap)

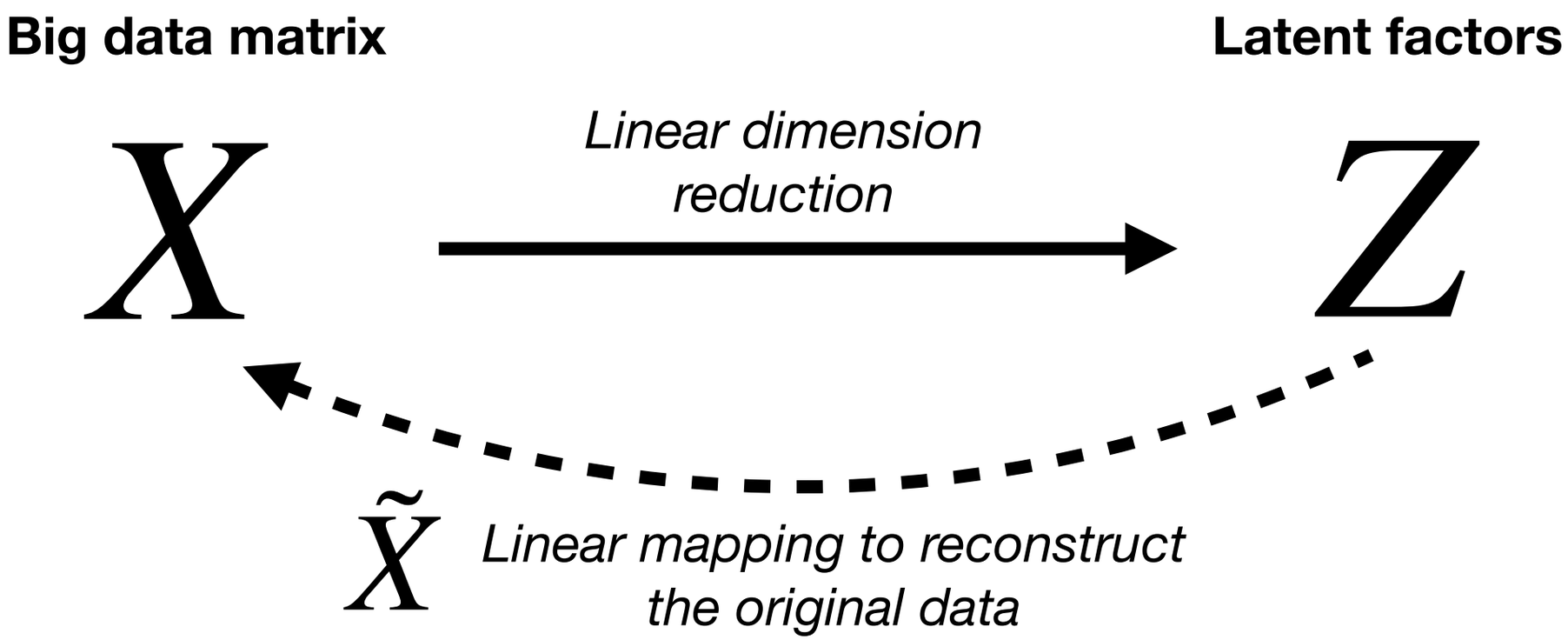
All of these should give similar results. They are just slightly different implementations of PCA.

We'll use `prcomp` in the lab.

PCA: Reconstruction error

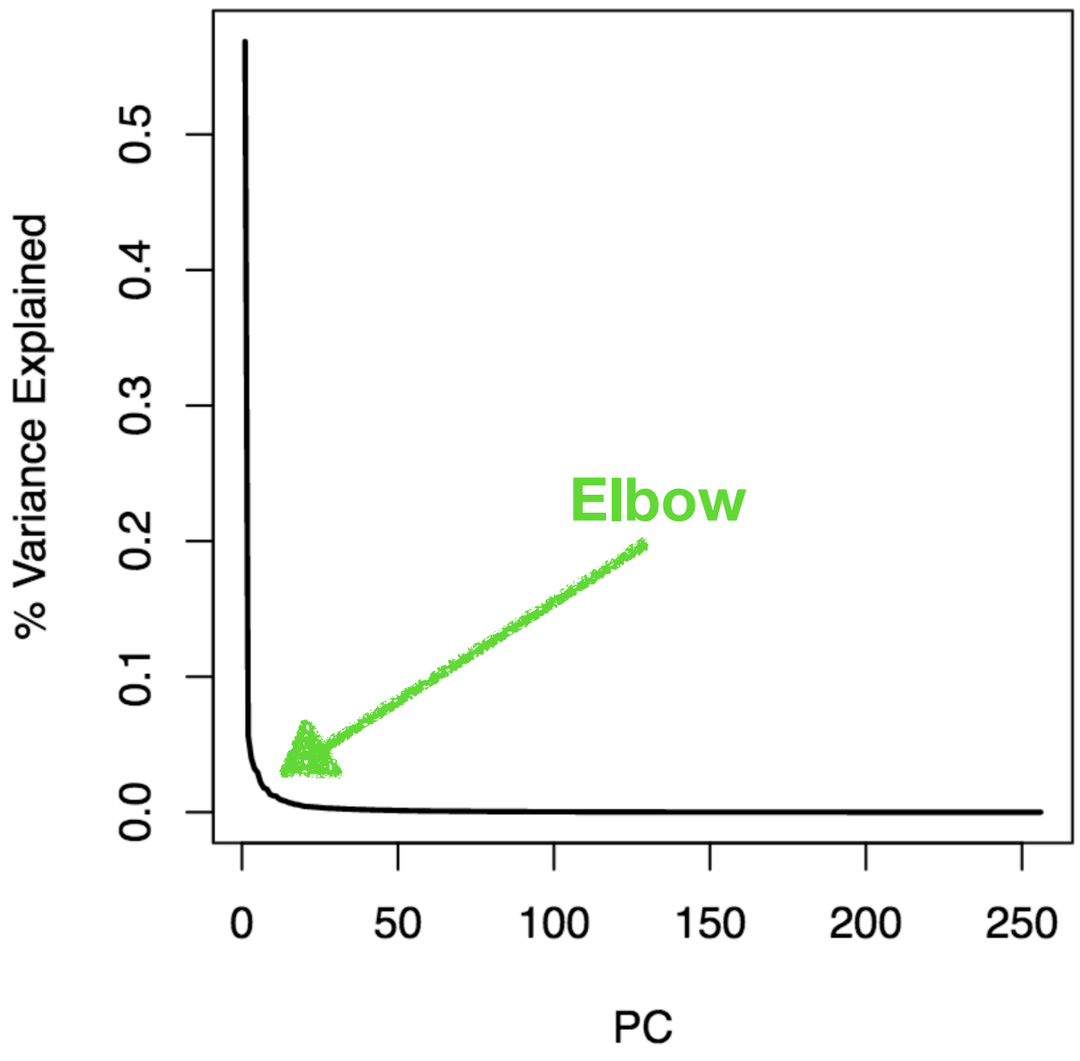
- It turns out that PCA can also be thought of as the best linear data compression onto K dimensions that minimizes the reconstruction error:

$$\min_{\tilde{X}} \|X - \tilde{X}\|_F^2 \text{ subject to } \text{rank}(\tilde{X}) = k$$



Picking the number of PCs

Scree plot:



- The total variance for a dataset is

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

- The variance explained by the *m*th PC is

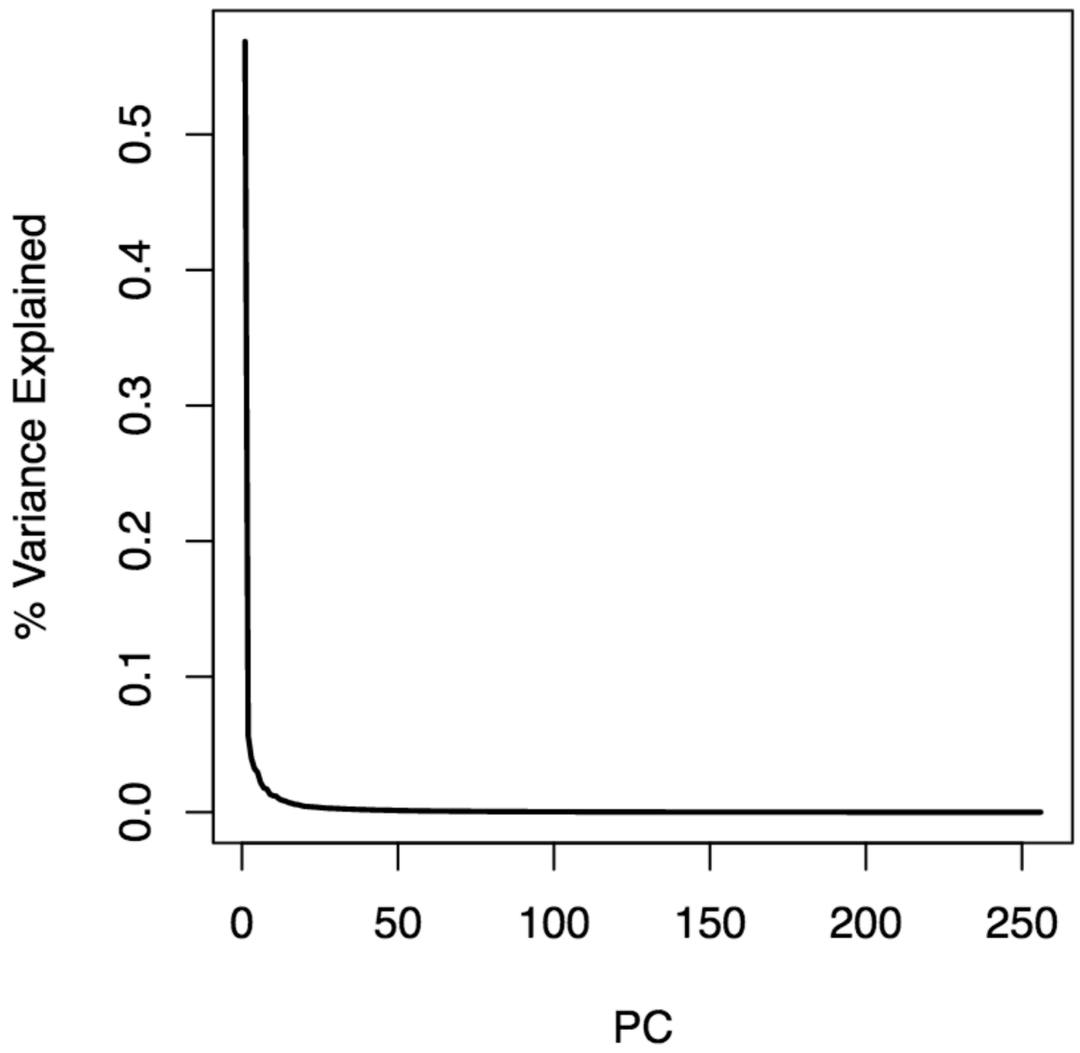
$$Var(Z_m) = \frac{1}{n} \sum_{i=1}^{n} z_{i,j}^2.$$

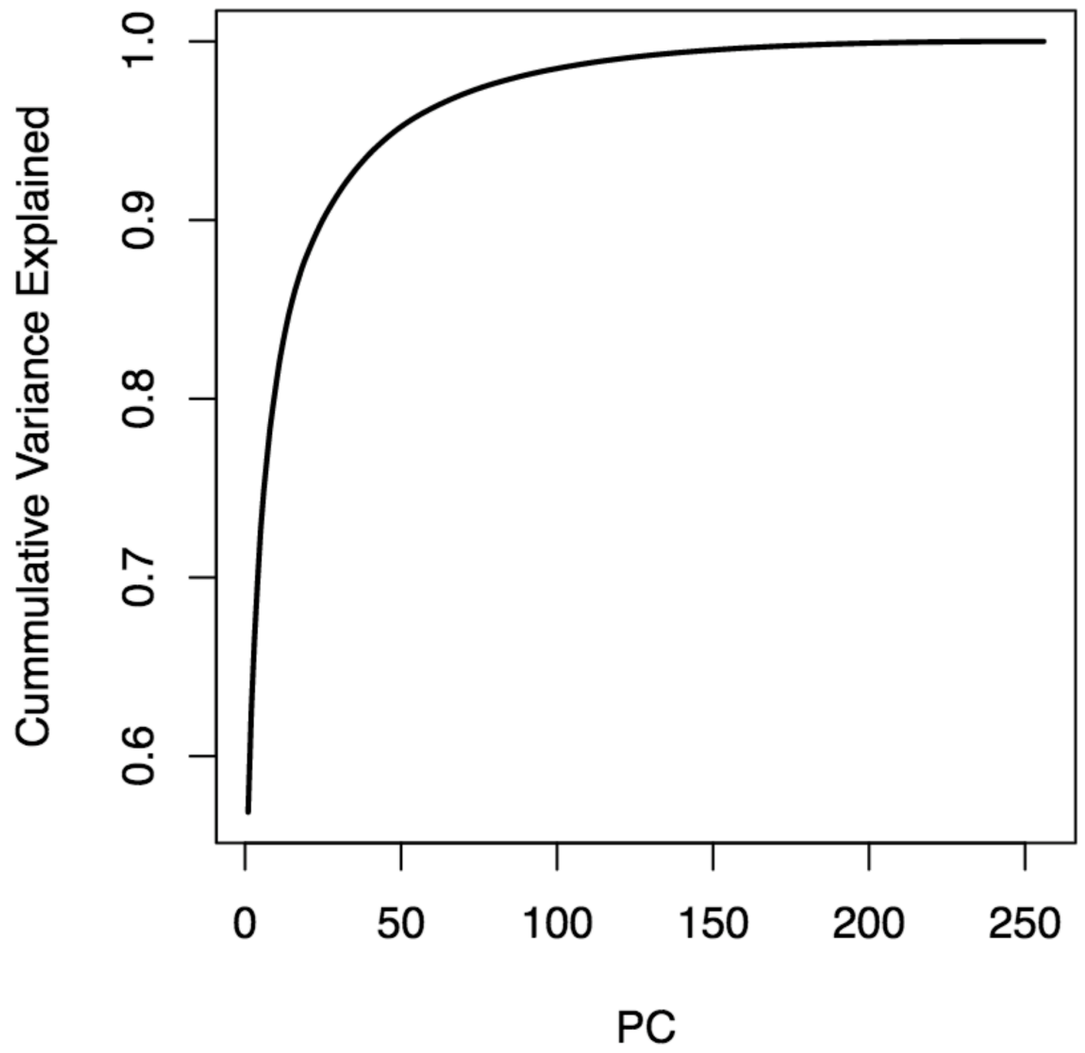
- The proportion of variance explained is

$$\frac{\sum_{i=1}^{m} \text{Variance explained by } m\text{th PC}}{\text{Total variance of the dataset}}$$

Picking the number of PCs

Scree plot:





Picking the number of PCs

- Options:
  - Elbow in scree plot
  - Take K that explains at least, say, 90% of the variance.
  - Cross-validation based on reconstruction error

PCA: Summary

- Strengths:
  - Best linear dimension reduction
  - Simple and easy
  - Unique global solution
- Weaknesses:
  - Cannot handle non-linearities
  - Ultra-high-dimensional settings

Today's lecture

1. Unsupervised learning
2. Clustering
    A. K-means
    B. Hierarchical Clustering
3. Dimension Reduction
    A. PCA