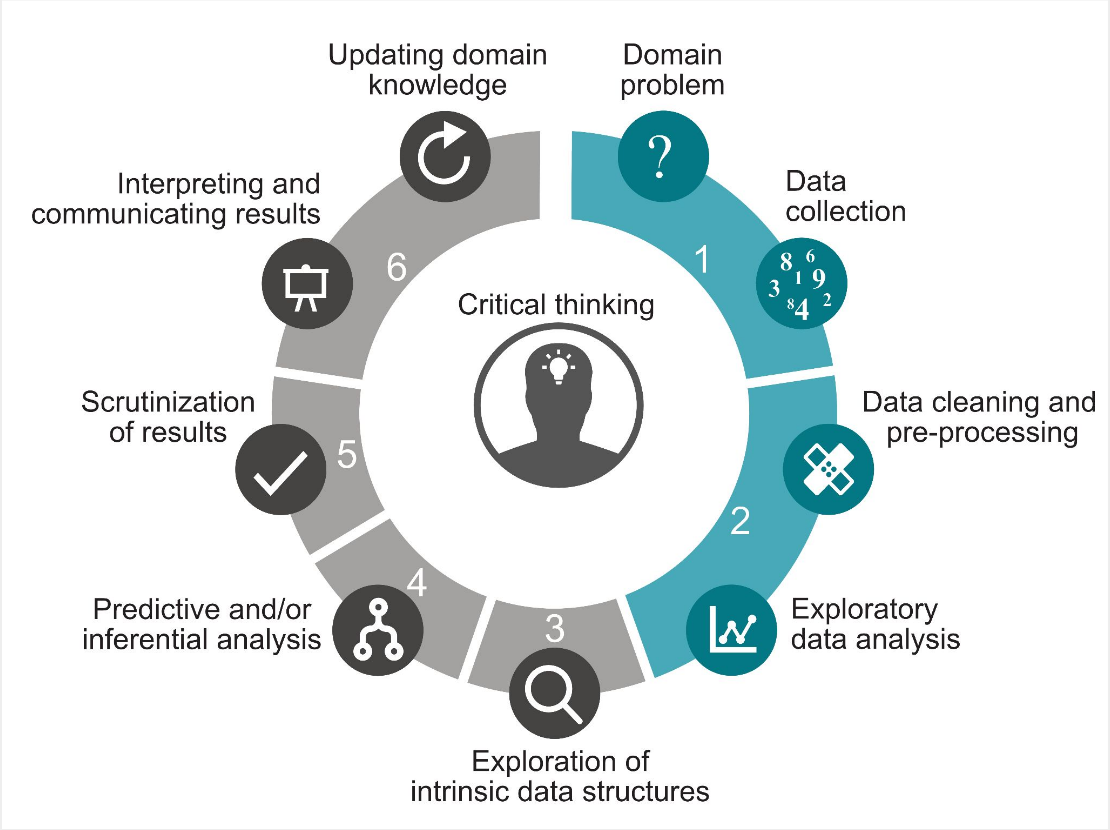
Introduction to Unsupervised Learning

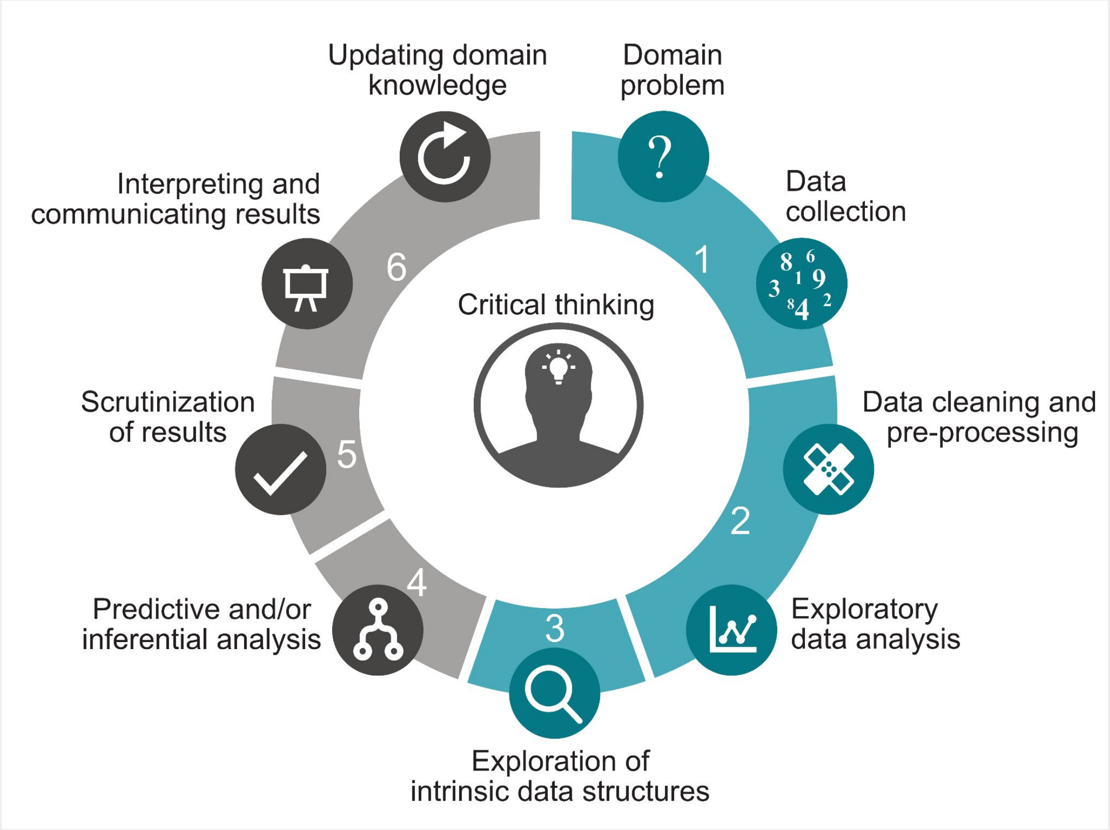
February 10, 2025

The Big Picture: Data Science Life Cycle



Credits: [Veridical Data Science (VDS) Textbook]

The Big Picture: Data Science Life Cycle



Credits: [Veridical Data Science (VDS) Textbook]()

Today’s plan: Introduction to Unsupervised Learning

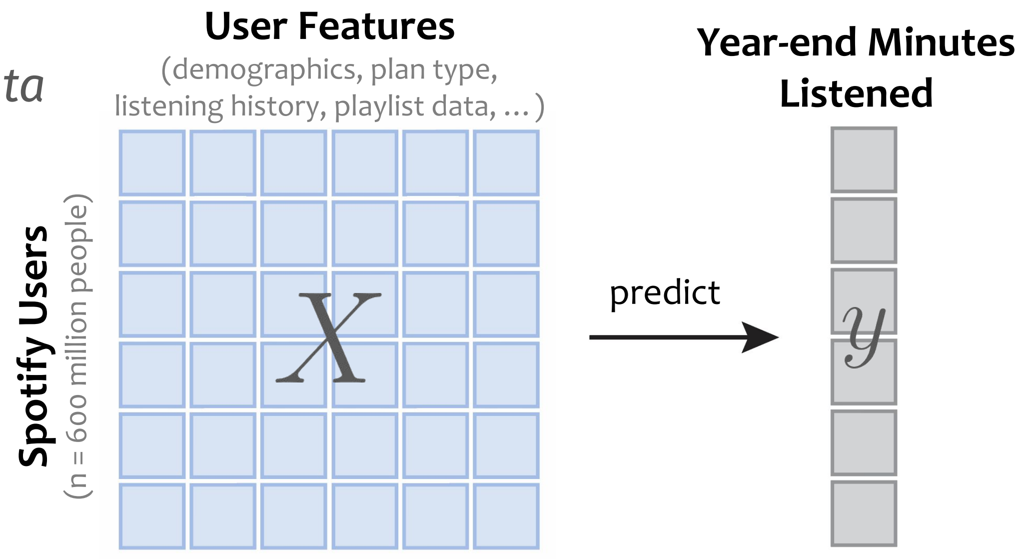
1. What is **unsupervised learning**?
2. **Applications** of unsupervised learning?
3. Overview of popular **dimension reduction** methods
4. Overview of popular **clustering** methods

Supervised vs Unsupervised Learning

In the **supervised learning** setting, we typically have some *covariate/feature* data matrix $X \in \mathbb{R}^{n \times p}$ and want to predict a *label/response* $y \in \mathbb{R}^n$

Example:

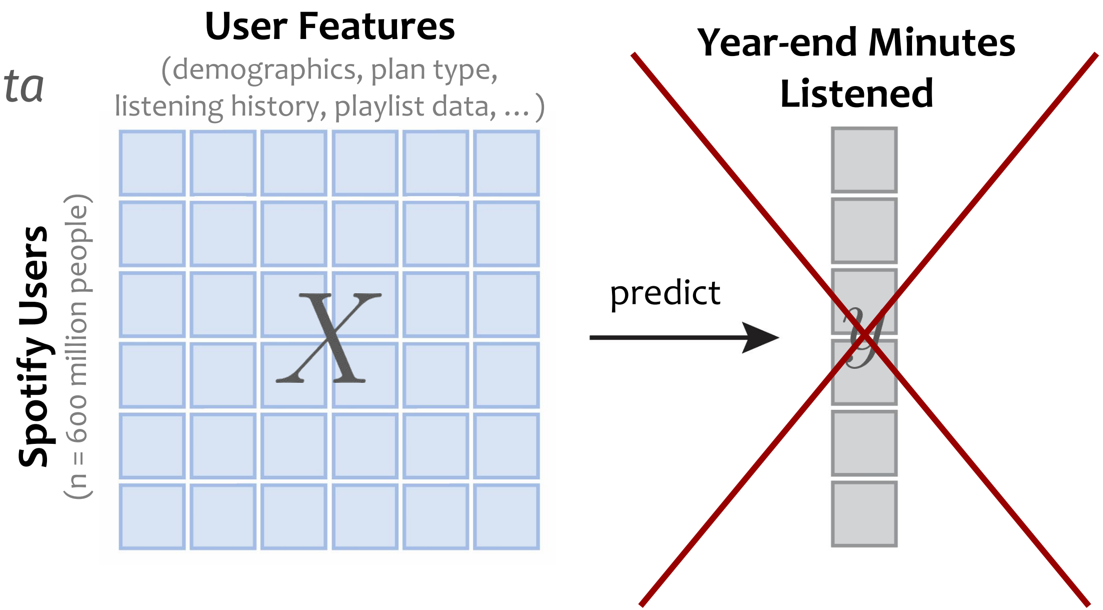
Spotify data



Supervised vs Unsupervised Learning

In the **unsupervised learning** setting, we have some *covariate/feature* data matrix $X \in \mathbb{R}^{n \times p}$ and but **no label/response** $y \in \mathbb{R}^n$

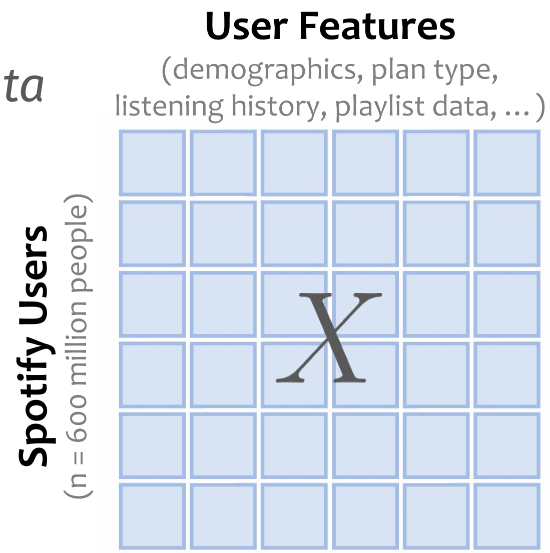
Example:
Spotify data



Applications of Unsupervised Learning

In the **unsupervised learning** setting, we have some *covariate/feature* data matrix $X \in \mathbb{R}^{n \times p}$ and but **no label/response** $y \in \mathbb{R}^n$

Example:
Spotify data



* similar applications in Netflix, Amazon, Youtube, Tiktok, and ad recommendation systems

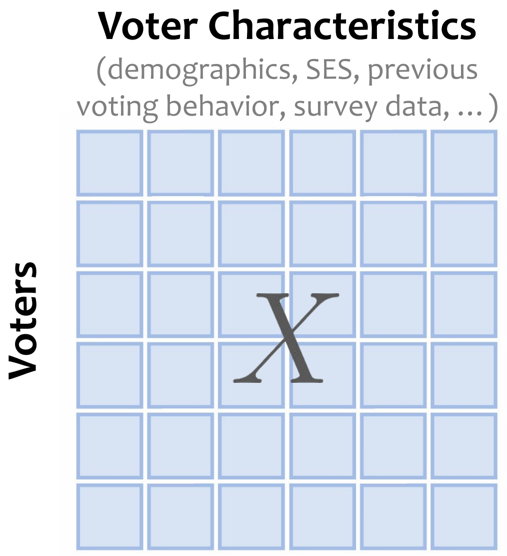
What can we do without labels/responses?

- **Descriptive statistics**
  - Ex. mean age of spotify users
- **Pattern recognition:** discover patterns among observations and/or features
  - Ex. popular song/genre mashups
- **Clustering:** identify groups of similar observations and/or features
  - Ex. groups of people with similar listening histories

Applications of Unsupervised Learning

In the **unsupervised learning** setting, we have some *covariate/feature* data matrix $X \in \mathbb{R}^{n \times p}$ and but **no label/response** $y \in \mathbb{R}^n$

Example:
Political Behavior



Clustering/pattern recognition applications:

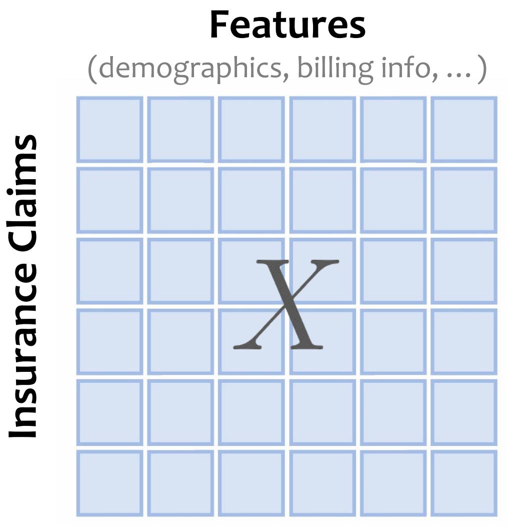
- Can we identify groups of similar voters so that we can create targeted messages?
- Who are the swing voters? And what issues are most likely to sway them?

Applications of Unsupervised Learning

In the **unsupervised learning** setting, we have some *covariate/feature* data matrix $X \in \mathbb{R}^{n \times p}$ and but **no label/response** $y \in \mathbb{R}^n$

Example:

Anomaly/ Fraud Detection



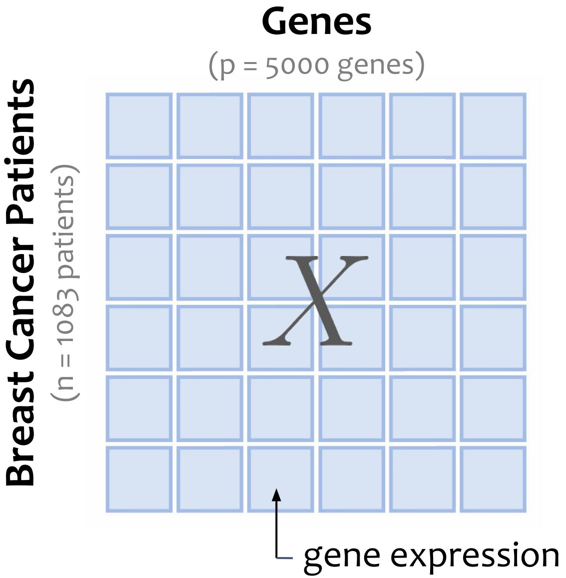
Clustering/pattern recognition applications:

+ Are there any anomalies in the claims/billing data? Maybe these are fraudulent.
+ Can perform clustering to identify similar claims that may be billed incorrectly

Applications of Unsupervised Learning

In the **unsupervised learning** setting, we have some *covariate/feature* data matrix $X \in \mathbb{R}^{n \times p}$ and but **no label/response** $y \in \mathbb{R}^n$

Example:
Cancer genomics



Clustering/pattern recognition applications:

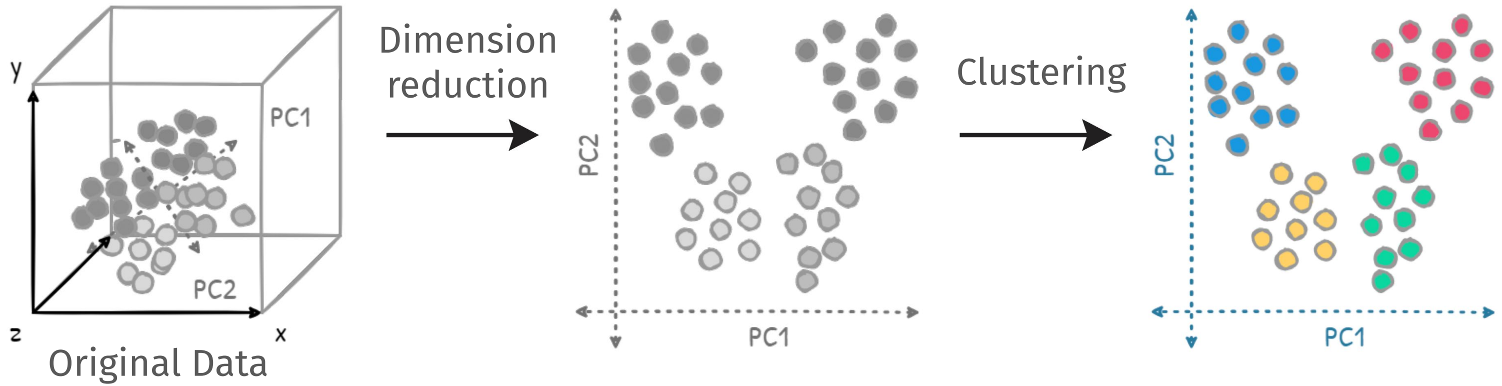
- Are there **subtypes** of patients who have similar tumors? Moreover, are there particular genes that drive these subtypes?
  - Hope is that these groups can be treated similarly and in a more personalized way than what's done for the whole group → *"personalized medicine"*

How do we "learn" from unsupervised data?

Two common unsupervised learning tools

1. **Dimension Reduction**
   - For pattern recognition, visualizing your data, data compression
2. **Clustering**
   - For identifying groups or clusters in your data

A common dimension reduction + clustering pipeline:



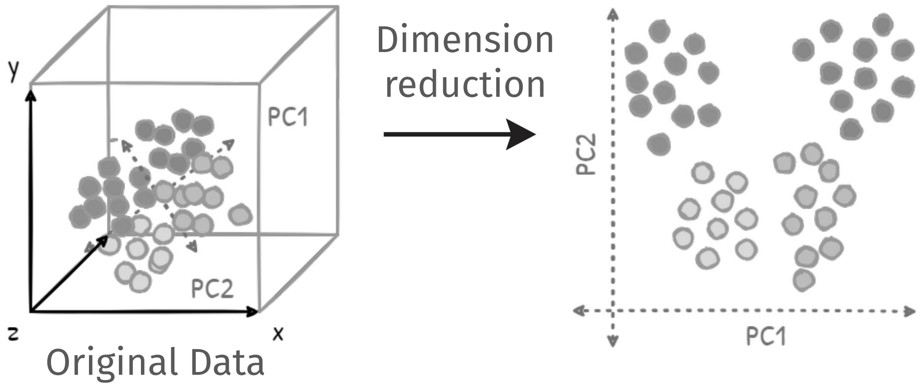
Dimension Reduction

Dimension Reduction

In reality, data is often **"high-dimensional"** (i.e., has many covariates/features)

How do we visualize data with > 3 features?

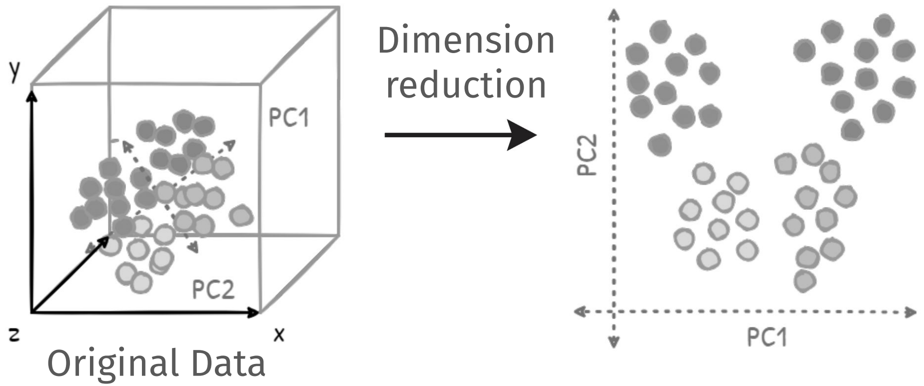
Dimension Reduction: aims to find a lower-dimensional representation of the data which preserves as much of the original information as possible



Principal Components Analysis (PCA)

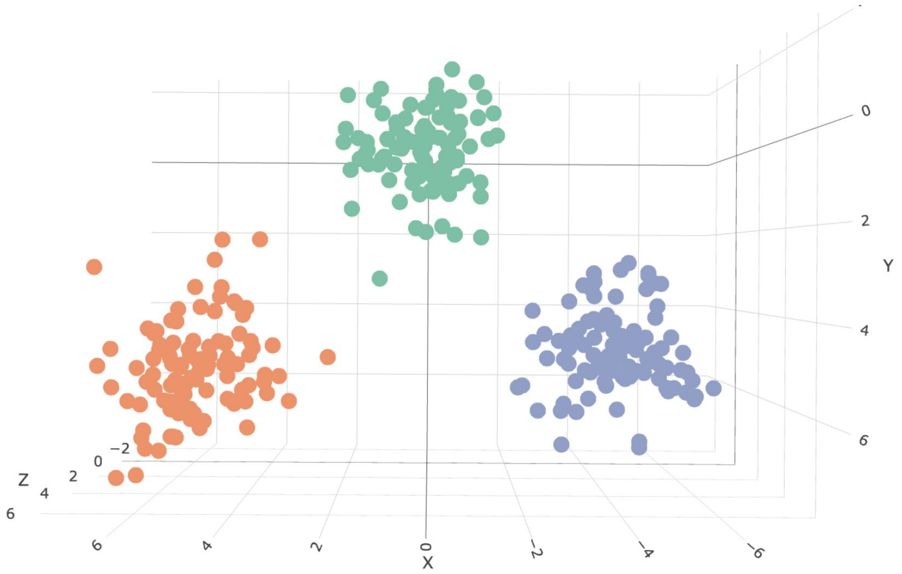
Principal Components Analysis (PCA): finds a lower-dimensional representation of the data which preserves as much of the **variance** in the data as possible

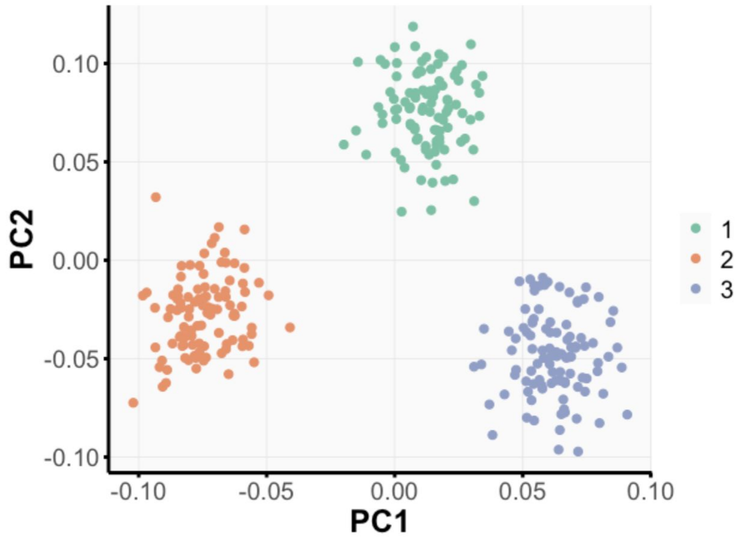
- More specifically, PCA finds a lower-dimensional hyperplane (or orthogonal directions) such that when the data is projected onto the hyperplane, the variance of the data is maximized
- PCA is a **linear projection**



When does PCA "work" and when does PCA "not work"?

Scenario: (High-dimensional) Gaussian data



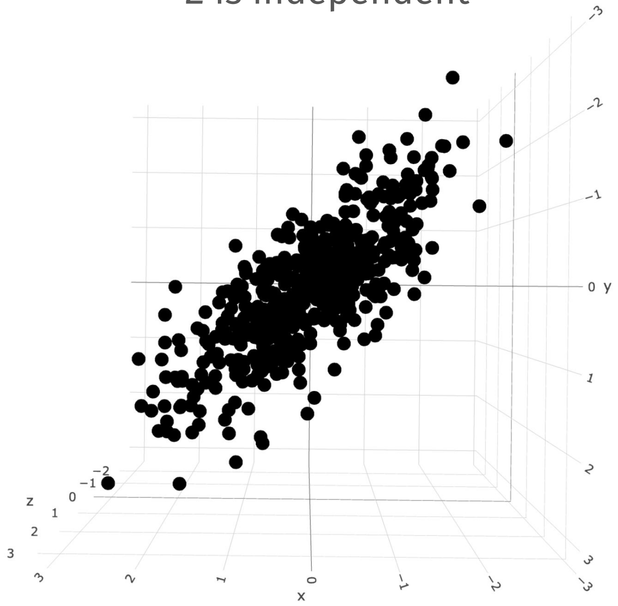


✔ This is the ideal scenario for PCA

When does PCA "work" and when does PCA "not work"?

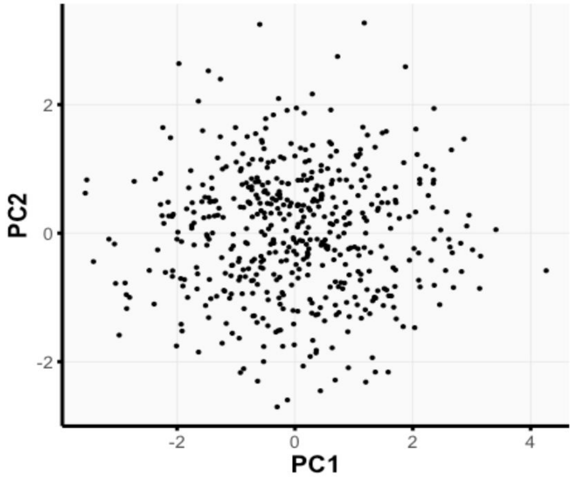
Scenario: Correlated Variables

X and Y are highly correlated;
Z is independent



PC Loadings:

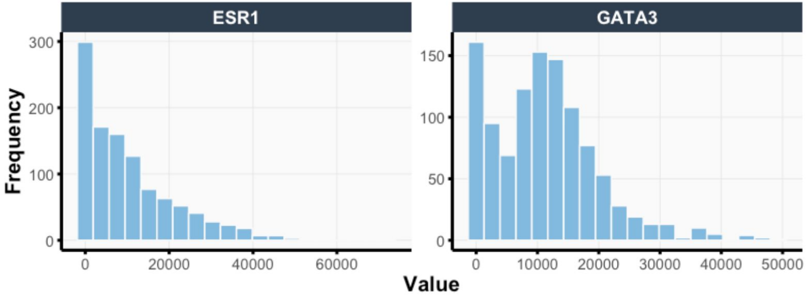
```
PC1 =  0.7X + 0.7Y + 0.1Z
PC2 = -0.1X - 0.1Y + 1.0Z
```

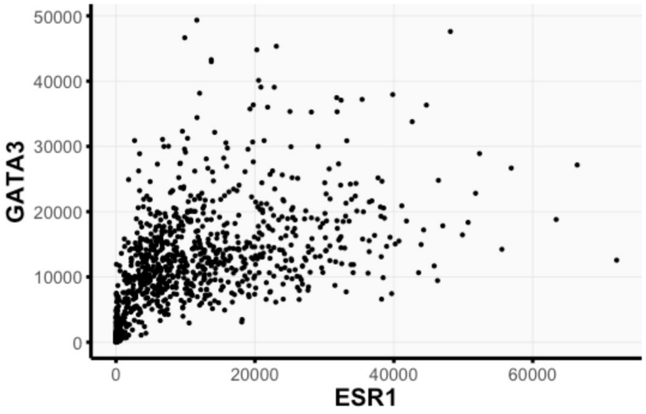


PCs typically group correlated variables together

When does PCA "work" and when does PCA "not work"?

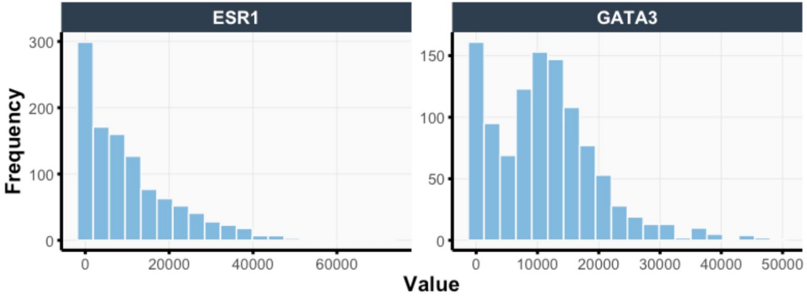
Scenario: Highly skewed data or data with outliers

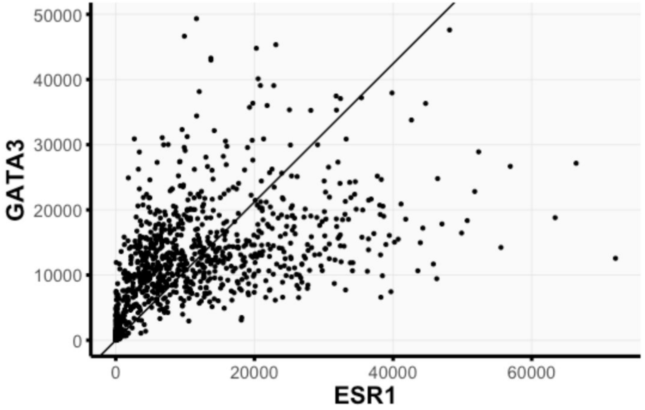


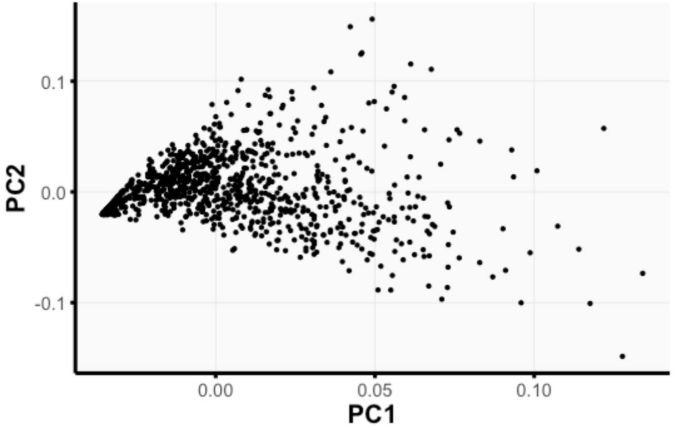


When does PCA "work" and when does PCA "not work"?

Scenario: Highly skewed data or data with outliers



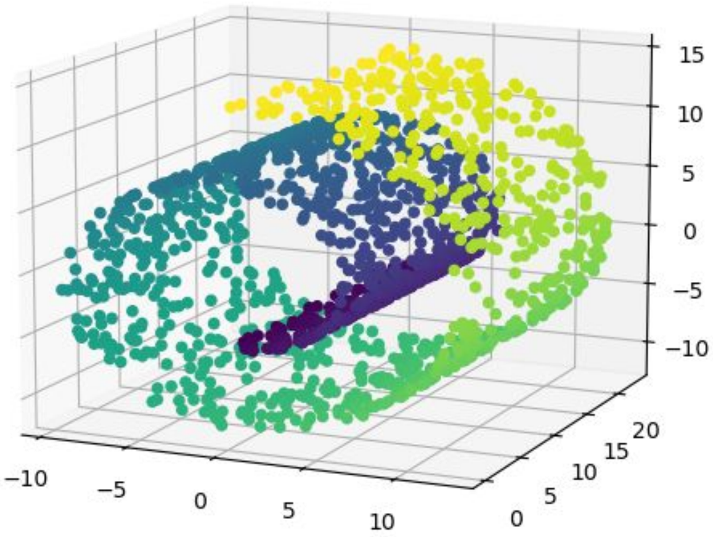


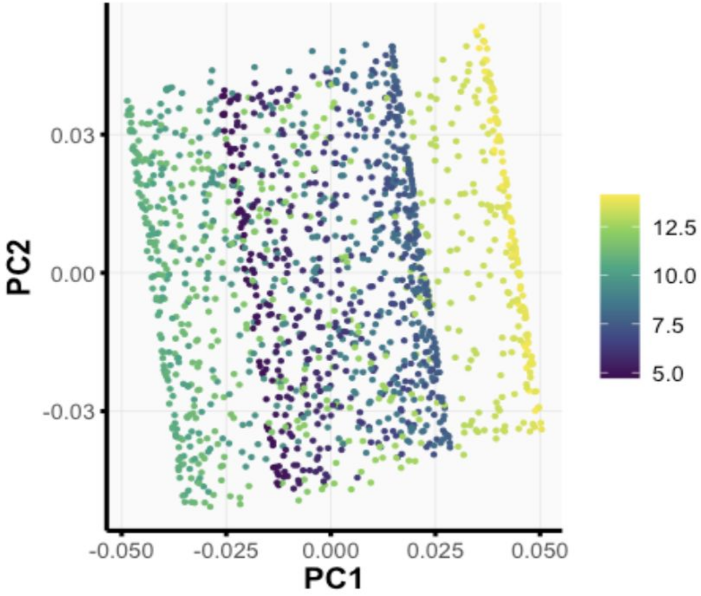


✔ **if** variance is still a meaningful measure of information

When does PCA "work" and when does PCA "not work"?

Scenario: Swiss roll





✘ not great for nonlinear manifolds

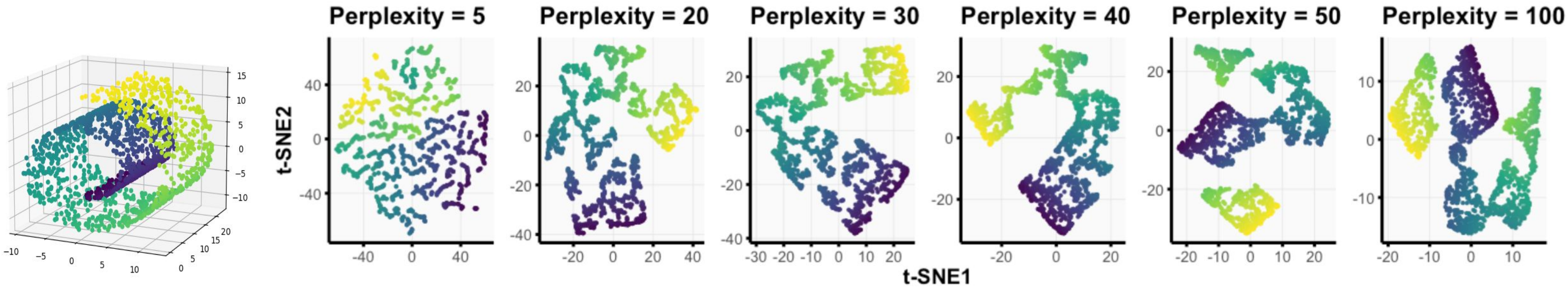
Nonlinear Dimension Reduction Methods

tSNE

1. Compute Euclidean distance between every pair of points in X
2. Translate these pairwise distances into probability of being neighbors
   + Large pairwise distance → low probability of being neighbors
3. Find lower-dimensional representation such that

Prob(i and j are neighbors) in **original high-dimensional** space	≈	Prob(i and j are neighbors) in **new low-dimensional** space

Hyperparameter: perplexity

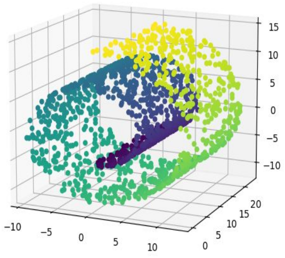


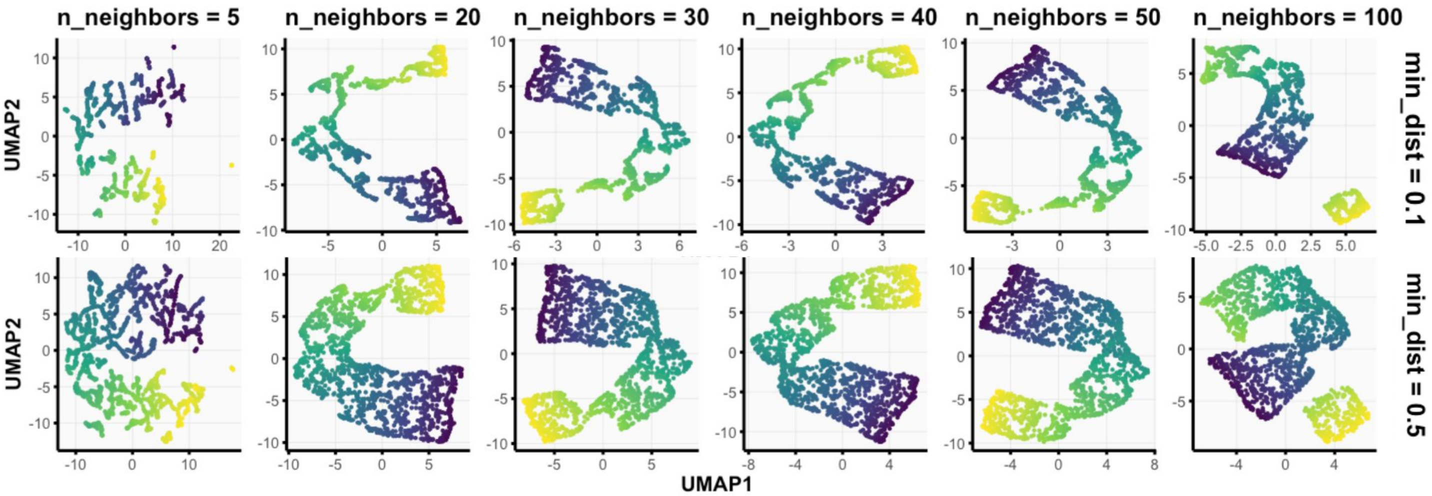
Nonlinear Dimension Reduction Methods

UMAP

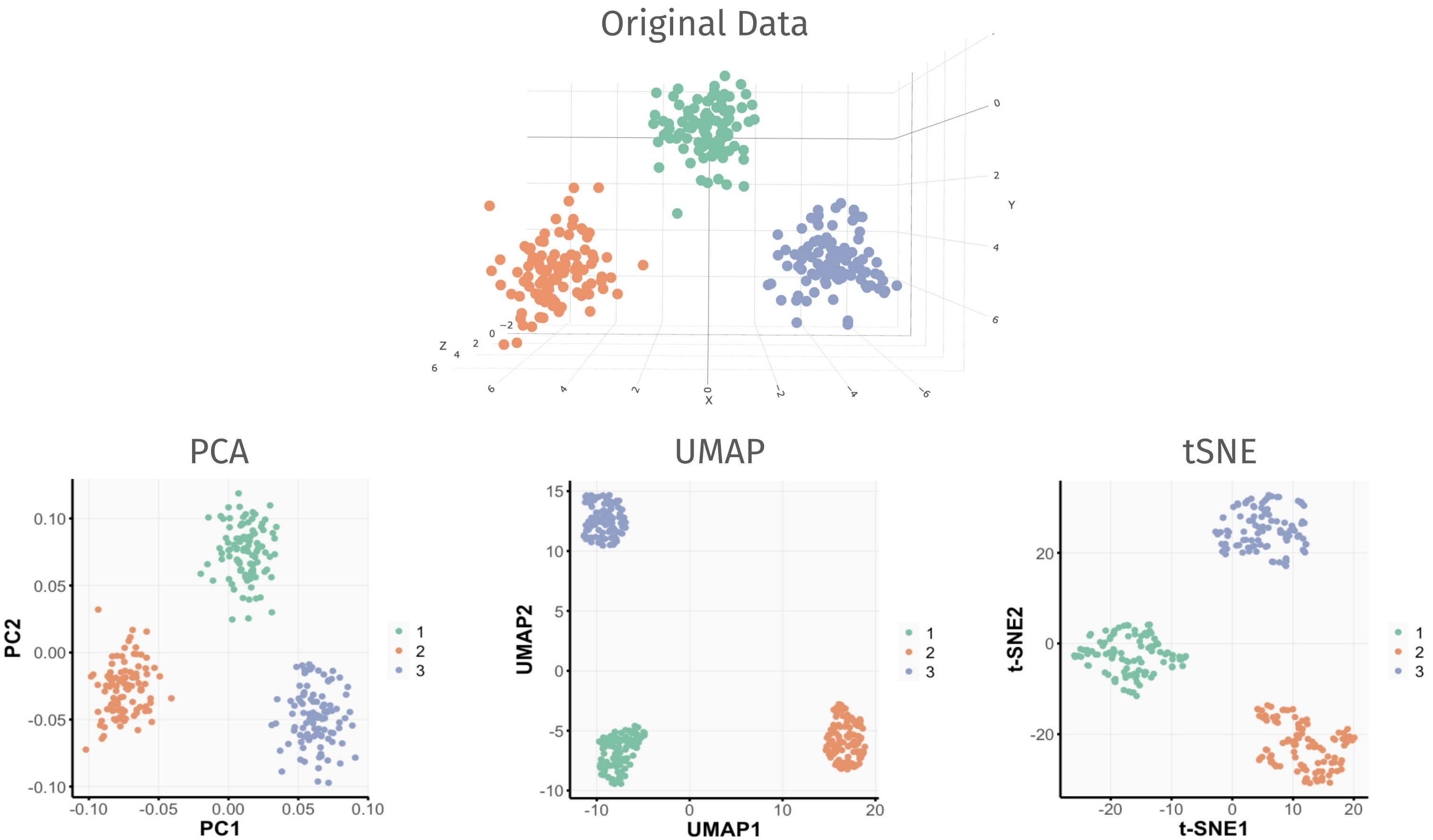
- UMAP often does better than tSNE at preserving the global structure
- Like tSNE, the idea is that pairs of points that are close in the original high-dimensional space should also be close in the new low-dimensional space
- How does UMAP differ from tSNE? Different similarity metrics, loss function, optimization algorithm

Hyperparameter: n_neighbors and min_distance





Word of caution: tSNE and UMAP can exaggerate clusters



Recap: Dimension Reduction Methods

	PCA	**tSNE**	**UMAP**
Feature Interpretability	Yes	No	No
Linear/nonlinear	Linear	Nonlinear	Nonlinear
Number of components	Orthogonal, nested; Can compute all p components at once	Non-nested and need to re-run for each chosen rank; Typically only 2-3 components	Non-nested and need to re-run for each chosen rank; Typically only 2-3 components
Computation	Fast	Slower	Slower but faster than tSNE
Unique, global solution	Yes	Converges to local solution	Converges to local solution
Other considerations?	No hyperparameters	Results can change drastically depending on hyperparameters; Not good at preserving global structure; "Curse of dimensionality"	Results can change drastically depending on hyperparameters; Better at preserving global structure than tSNE; "Curse of dimensionality"

Other dimension reduction methods: MDS, NMF, ICA, Isomap, LLE, Autoencoders, …

Additional Resources

A more detailed review of these unsupervised learning methods + more can be found at https://tiffanymtang.github.io/dsip-s25/#unsupervised-learning

- Also includes R and Python code for implementing these methods

The quarto notebook that generated this walkthrough can be found here:

https://github.com/tiffanymtang/dsip-s25/blob/main/unsupervised_learning/notebooks/unsupervised_learning.qmd

Additional Resources for Unsupervised Learning Methods

- Elements of Statistical Learning Textbook Chapter 14

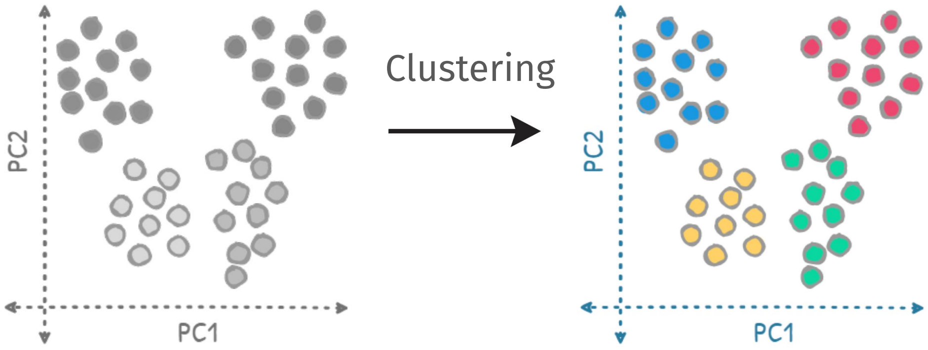
Clustering

Clustering

Clustering: aims to identify groups/clusters of samples (and/or features) that are "similar"

Two main approaches:

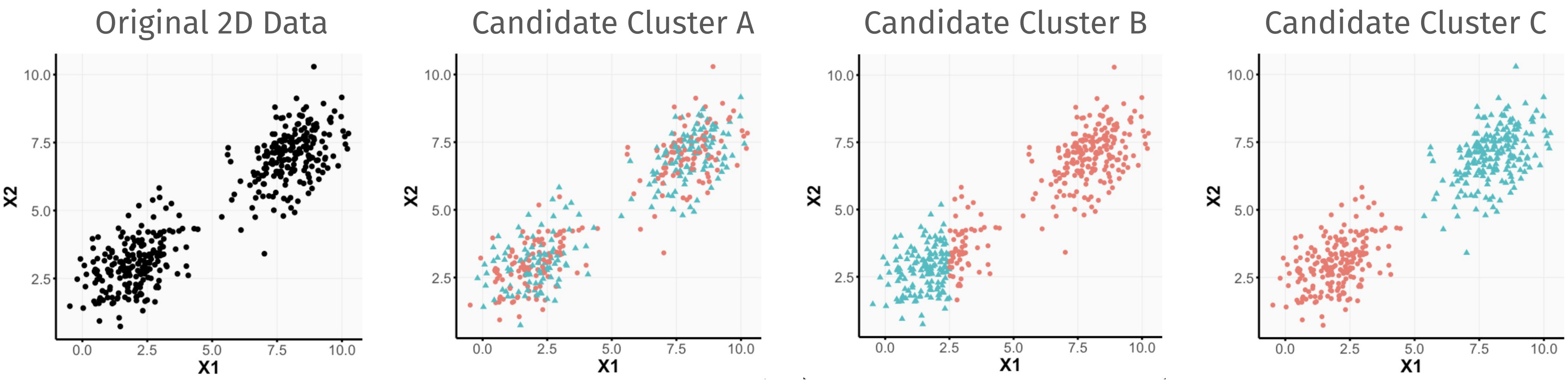
+ K-means clustering
+ Hierarchical clustering



K-means Clustering

For a pre-specified *K*:

- **Idea:** find *K* clusters which result in the "tightest" groups (i.e., has the smallest within-cluster variance)

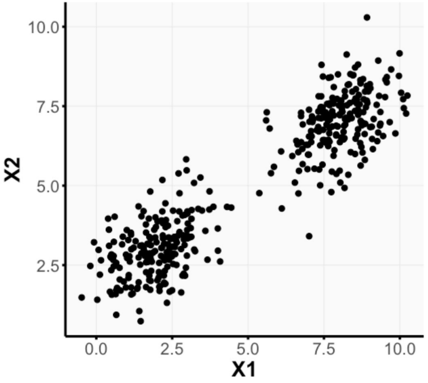


K-means Clustering

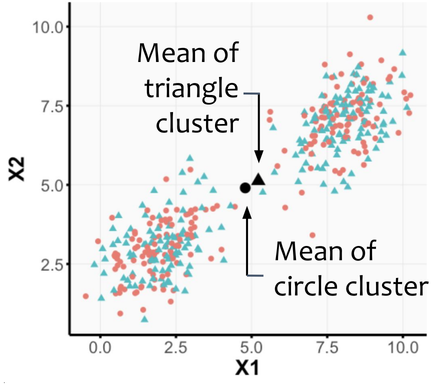
For a pre-specified *K:*

- **Idea:** find *K* clusters which result in the "tightest" groups (i.e., has the smallest within-cluster variance)

Original 2D Data

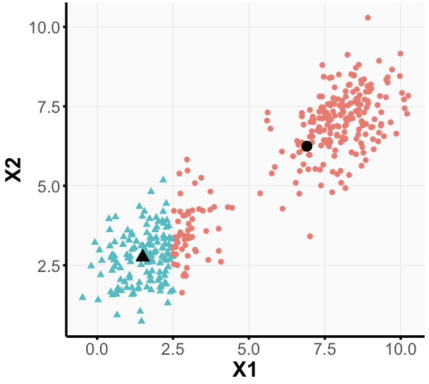


Candidate Cluster A



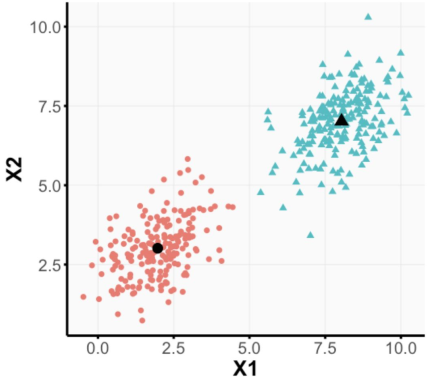
Blue: high variance
Red: high variance

Candidate Cluster B



Blue: low variance
Red: high variance

Candidate Cluster C

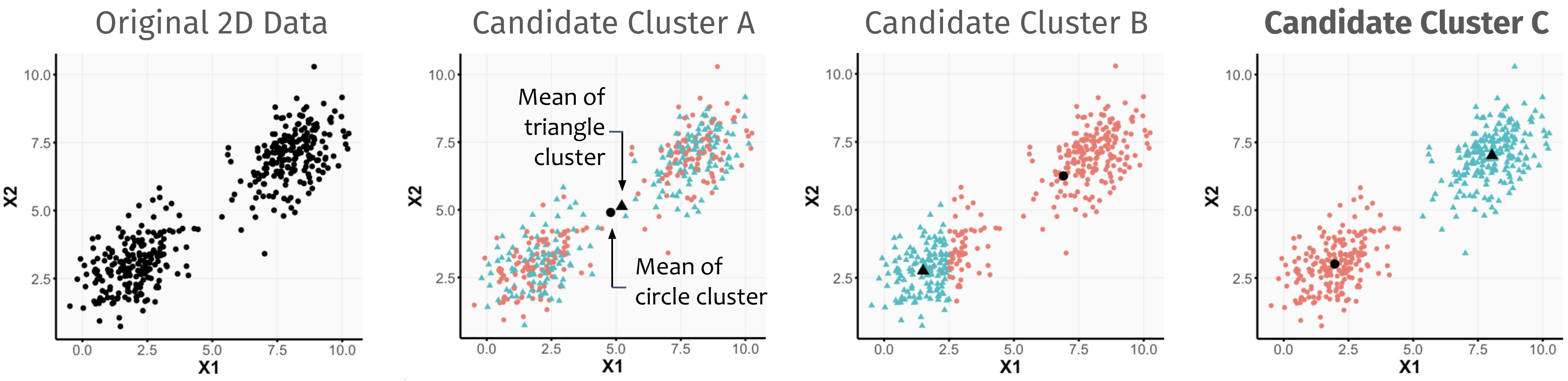


Blue: low variance
Red: low variance

K-means Clustering

For a pre-specified *K:*

+ **Idea:** find *K* clusters which result in the "tightest" groups (i.e., has the smallest within-cluster variance)



Blue: high variance
Red: high variance

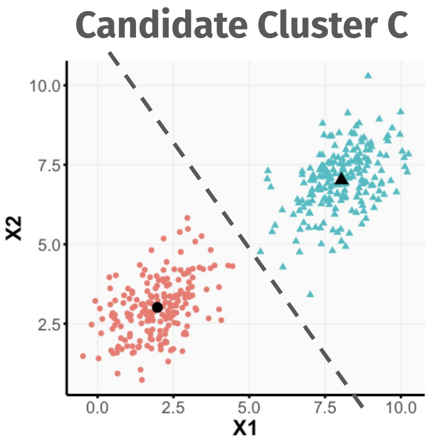
Blue: low variance
Red: high variance

Blue: low variance
Red: low variance

K-means Clustering

For a pre-specified *K*:

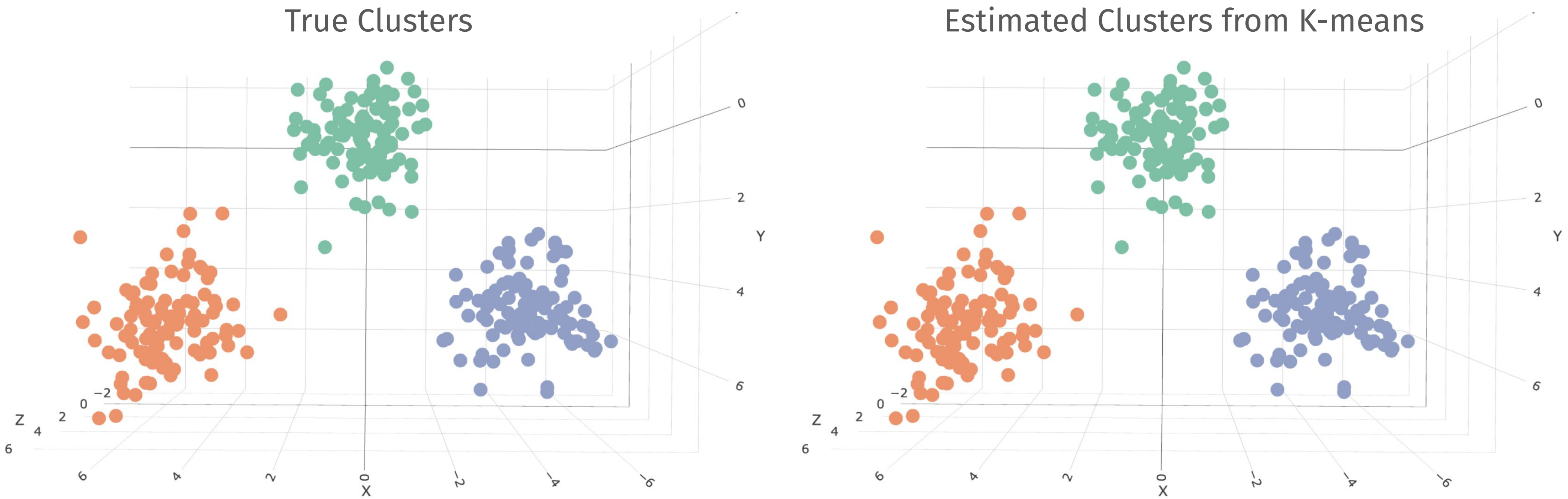
- **Idea:** find *K* clusters which result in the "tightest" groups (i.e., has the smallest within-cluster variance)



Points get clustered to the closest centroid

When does K-means "work" and when does K-means "not work"?

Scenario: Spherical, linearly-separable clusters

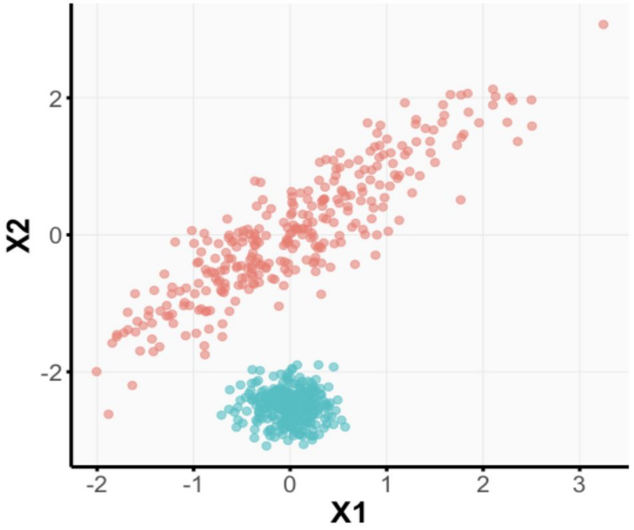


✔ This is the ideal scenario for K-means

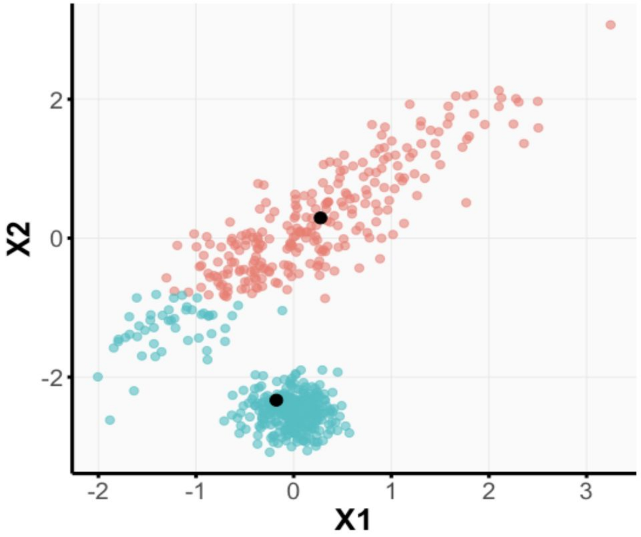
When does K-means "work" and when does K-means "not work"?

Scenario: Non-spherical clusters

True Clusters



Estimated Clusters from K-means

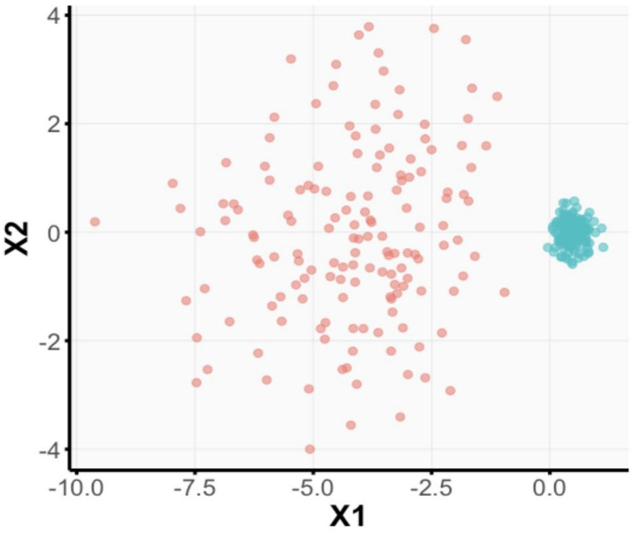


✘ Not great for non-spherical clusters

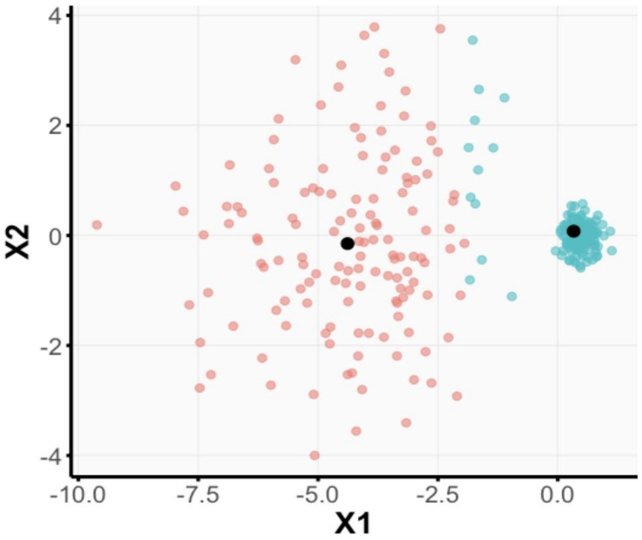
When does K-means "work" and when does K-means "not work"?

Scenario: Clusters with different variances

True Clusters



Estimated Clusters from K-means

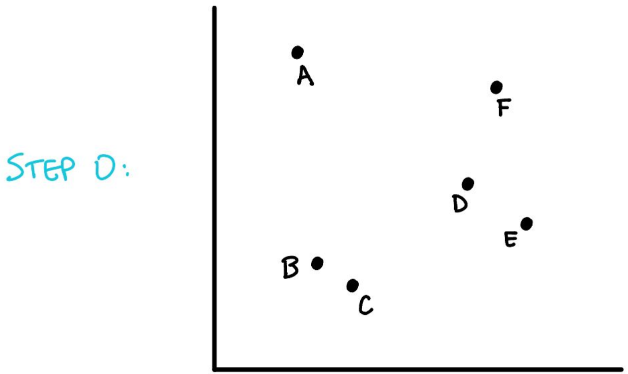


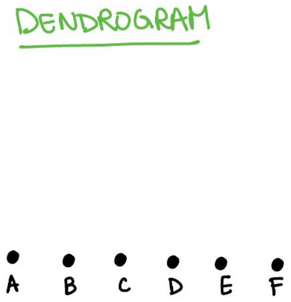
✘ Not great for clusters with different variances

Hierarchical Clustering

- A **greedy, agglomerative** algorithm
- Gives family of nested clusterings, presented as a tree
- At the lowest level, each cluster contains a single observation
- As we move up the tree, some leaves begin to fuse into branches – these are observations that are most **similar** to each other

Initialization (Step 0): Each point starts as its own singleton cluster

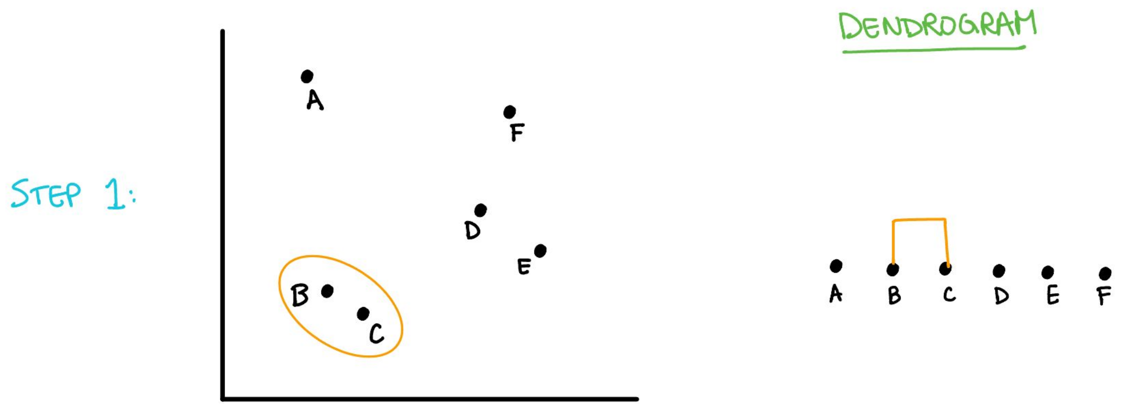




Hierarchical Clustering

- A **greedy, agglomerative** algorithm
- Gives family of nested clusterings, presented as a tree
- At the lowest level, each cluster contains a single observation
- As we move up the tree, some leaves begin to fuse into branches – these are observations that are most **similar** to each other

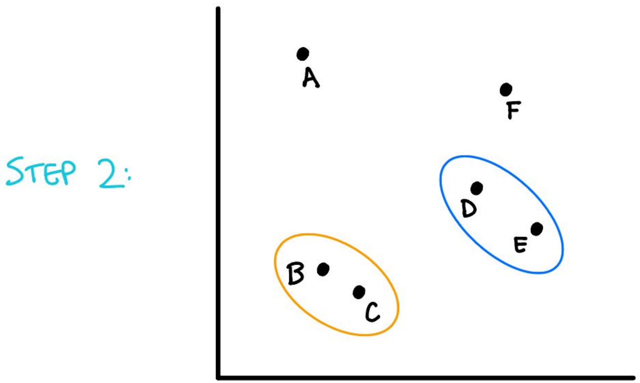
Next Step: Join the two points/clusters that are "closest" together

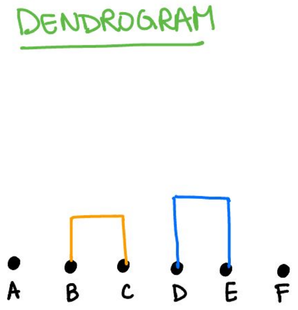


Hierarchical Clustering

- A **greedy, agglomerative** algorithm
- Gives family of nested clusterings, presented as a tree
- At the lowest level, each cluster contains a single observation
- As we move up the tree, some leaves begin to fuse into branches – these are observations that are most **similar** to each other

Next Step: Join the two points/clusters that are "closest" together



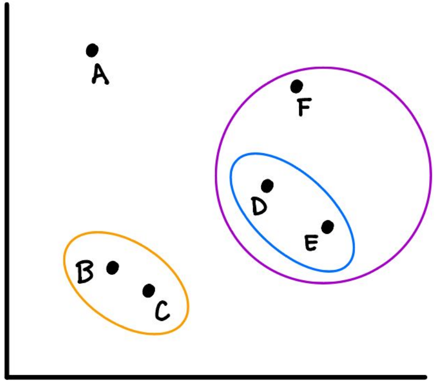


Hierarchical Clustering

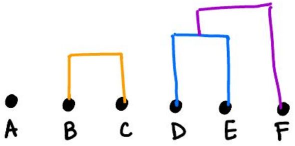
- A **greedy, agglomerative** algorithm
- Gives family of nested clusterings, presented as a tree
- At the lowest level, each cluster contains a single observation
- As we move up the tree, some leaves begin to fuse into branches – these are observations that are most **similar** to each other

Next Step: Join the two points/clusters that are "closest" together

STEP 3:



DENDROGRAM

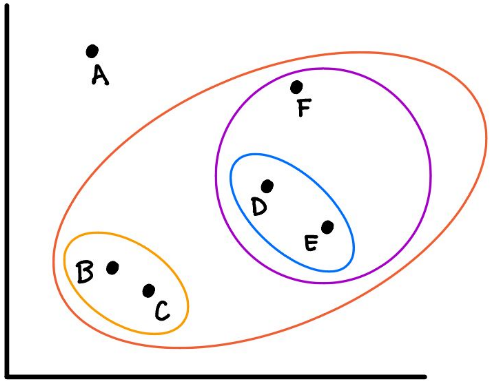


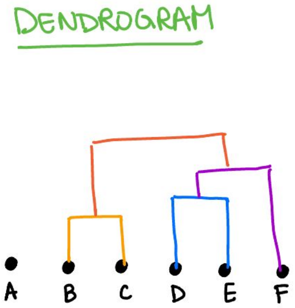
Hierarchical Clustering

- A **greedy, agglomerative** algorithm
- Gives family of nested clusterings, presented as a tree
- At the lowest level, each cluster contains a single observation
- As we move up the tree, some leaves begin to fuse into branches – these are observations that are most **similar** to each other

Next Step: Join the two points/clusters that are "closest" together

STEP 4:



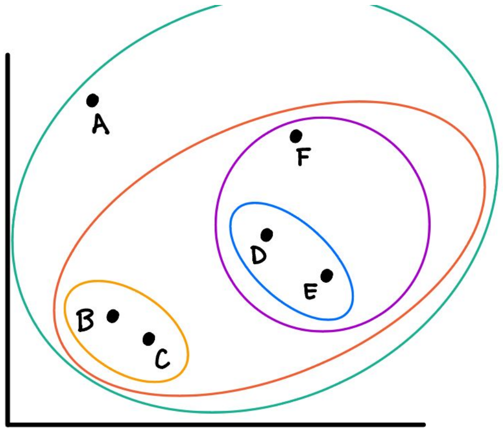


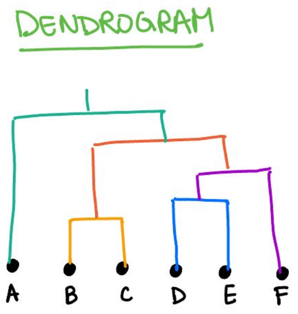
Hierarchical Clustering

- A **greedy, agglomerative** algorithm
- Gives family of nested clusterings, presented as a tree
- At the lowest level, each cluster contains a single observation
- As we move up the tree, some leaves begin to fuse into branches – these are observations that are most **similar** to each other

Next Step: Join the two points/clusters that are "closest" together

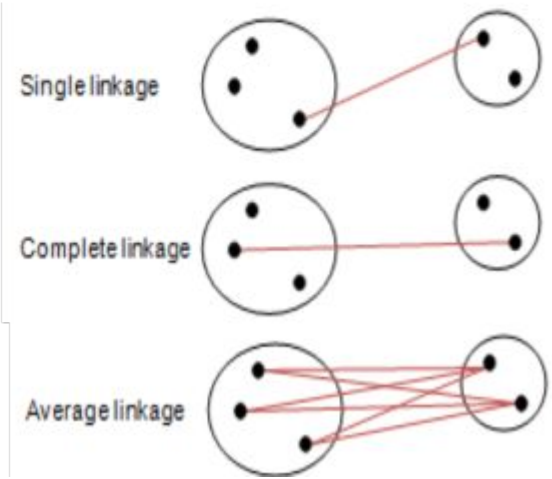
STEP 5 :



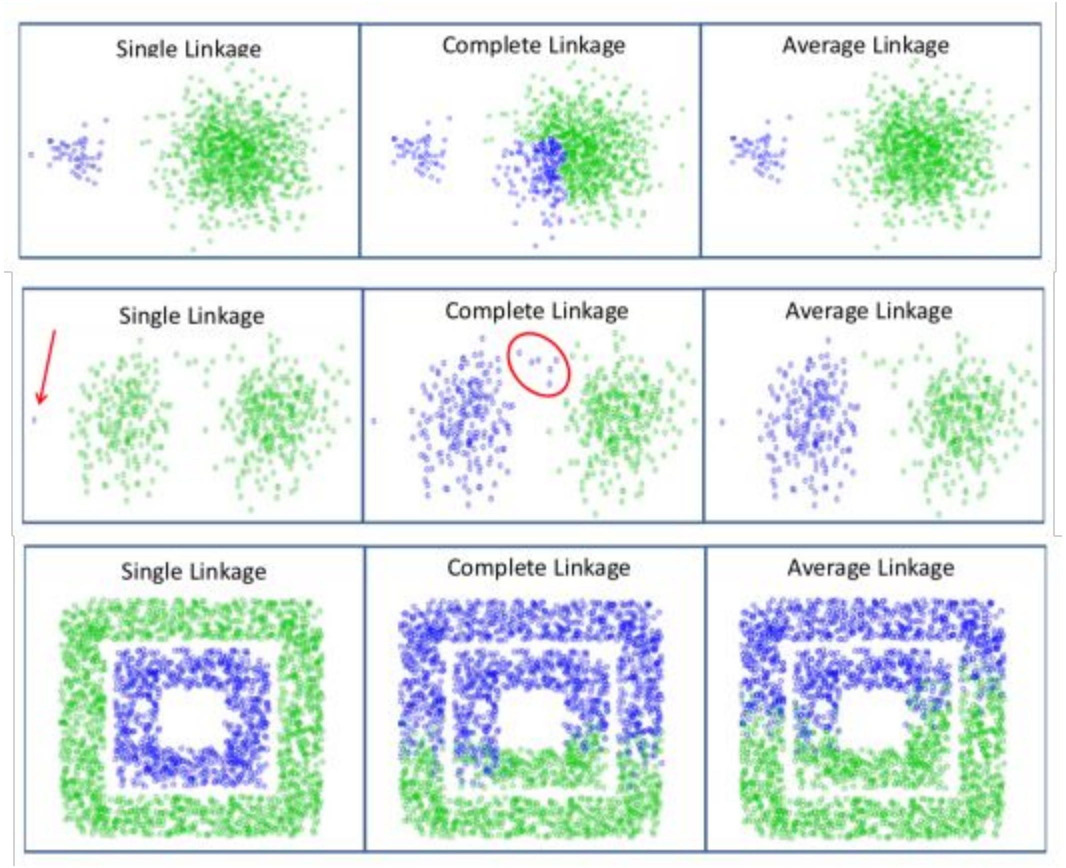


How to join clusters/observations

1. **Distance metric**: a measure of dissimilarity between two observations
    a. Examples: l_2, l_1, any of your favorite norms, 1 – cor(x, y)
2. **Linkage metric**: rule for joining two clusters
    a. Single Linkage (min)
    b. Complete Linkage (max)
    c. Average Linkage (average)
    d. Ward's Linkage (min variance)



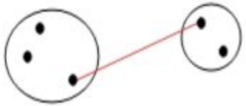
Linkage Examples



Linkages

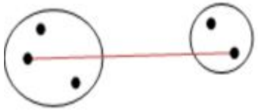
Single Linkage (min)

- Can handle diverse shapes
- Very sensitive to outliers or noise
- Often results in unbalanced clusters
- Extended, trailing clusters in which observations are fused one at a time – chaining



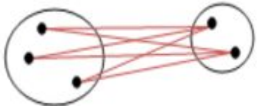
Complete Linkage (max)

- Often gives cluster with similar sizes
- Less sensitive to outliers
- Works better with spherical distributions



Average Linkage

- Compromise between single & complete linkage
- Less sensitive to outliers than single linkage, but not as robust as single complete linkage
- Works better with spherical distributions

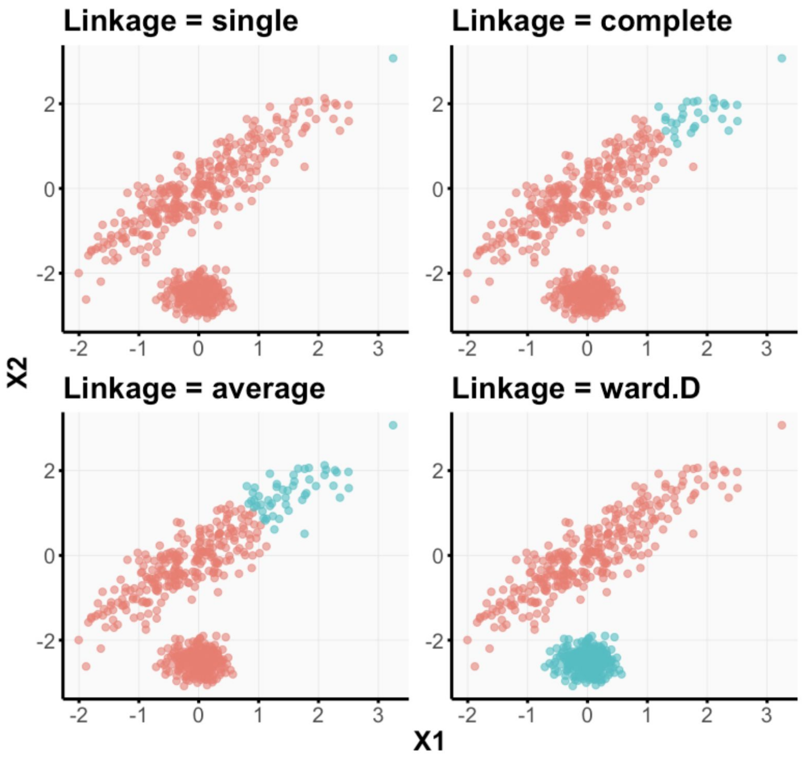


Ward's linkage: join sets that minimize the Euclidean distance between all pairs of points

Average and Ward's linkage are the most widely used in practice

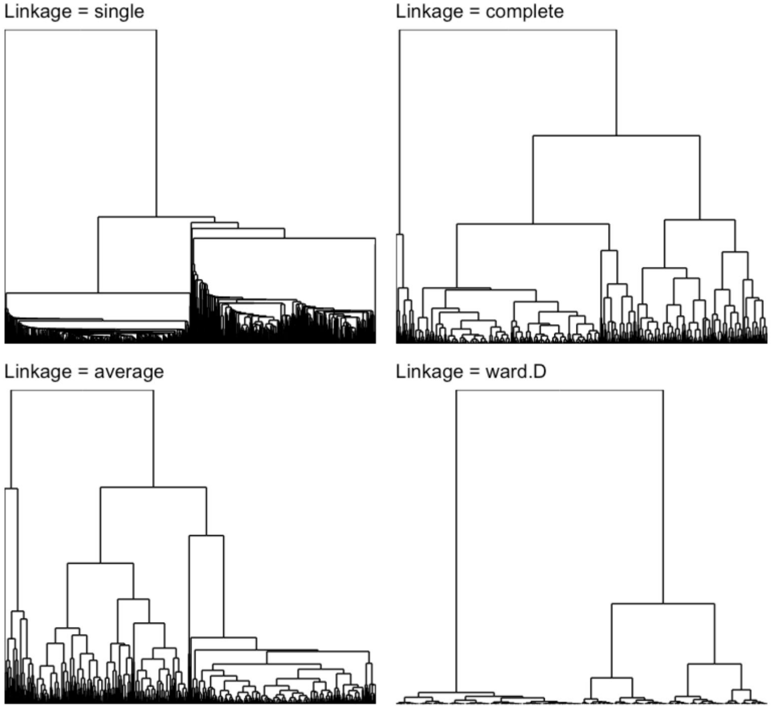
When does hierarchical clustering "work" or "not work"?

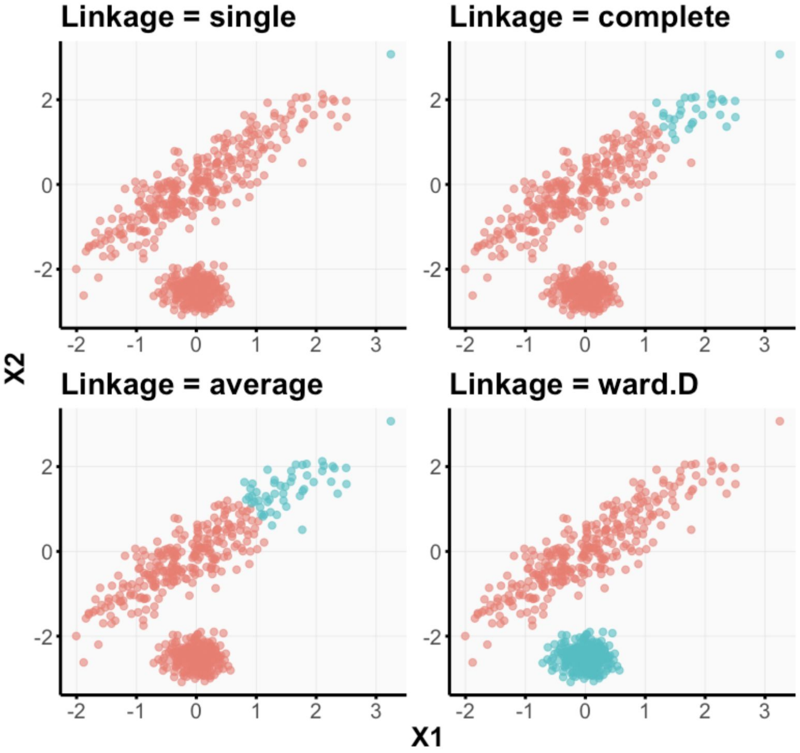
Scenario: Non-spherical clusters



When does hierarchical clustering "work" or "not work"?

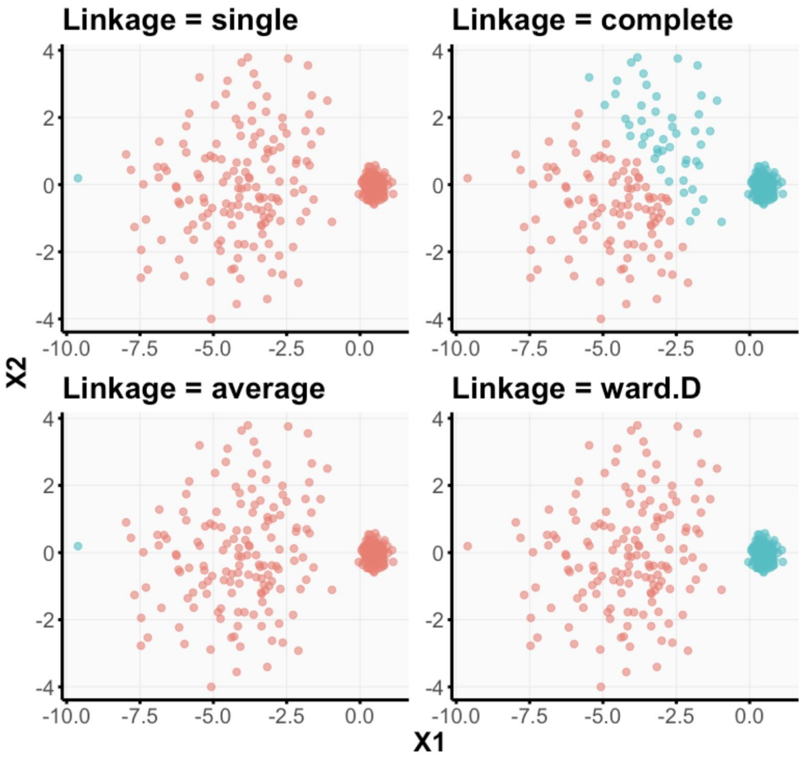
Scenario: Non-spherical clusters





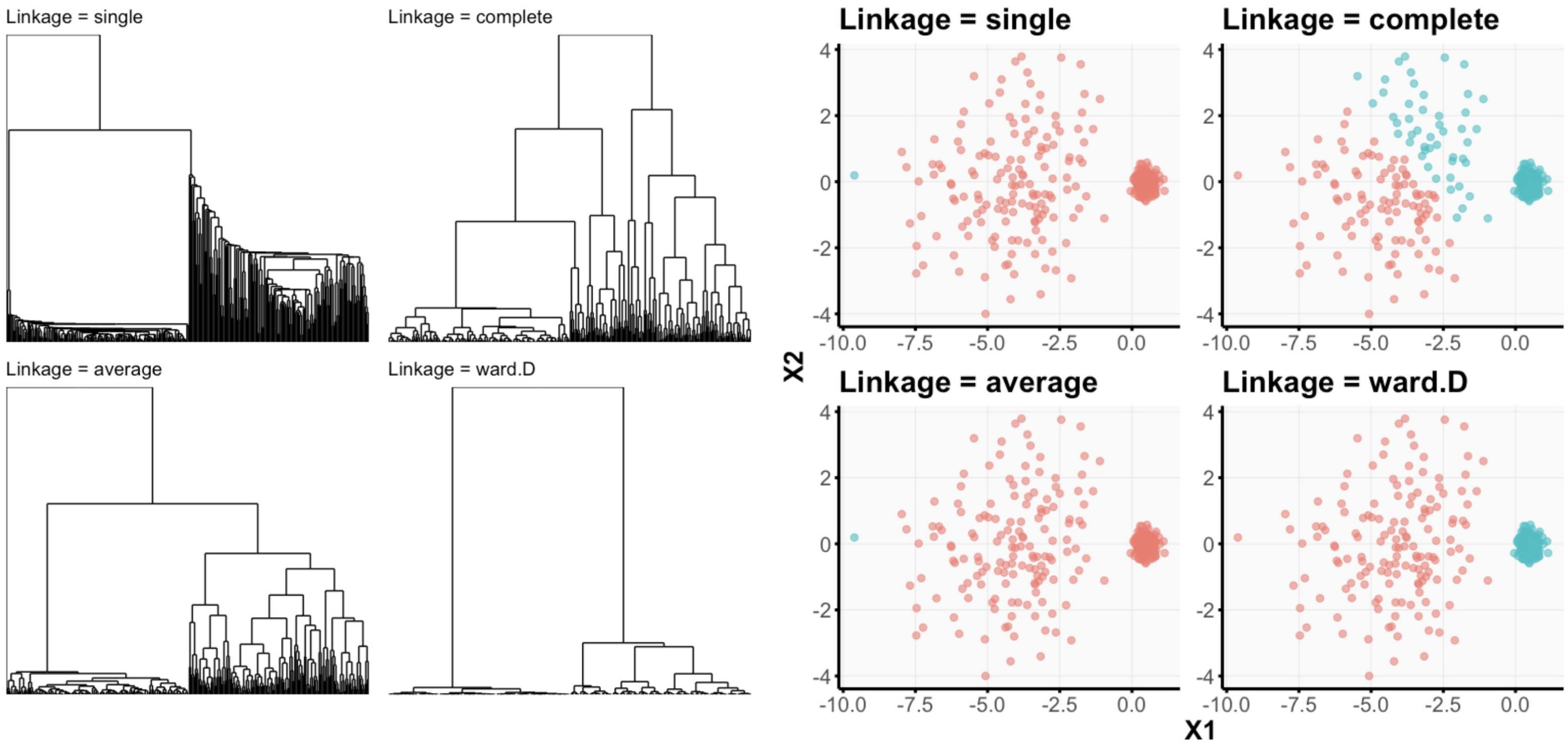
When does K-means "work" and when does K-means "not work"?

Scenario: Clusters with different variances



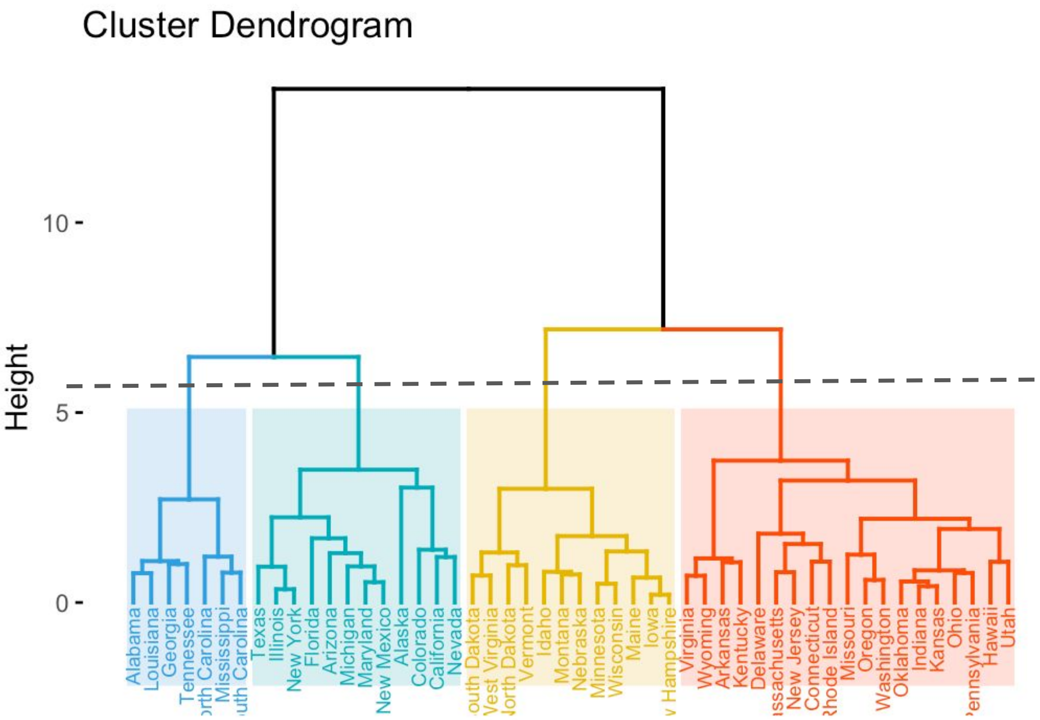
When does K-means "work" and when does K-means "not work"?

Scenario: Clusters with different variances



Hierarchical Clustering

- Gives family of nested clusterings, presented as a tree



Recap: Clustering Methods

	K-means Clustering	**Hierarchical Clustering**
Advantages	• Super fast and intuitive • Good when clusters are spherical and linearly separable	• Gives nested family of clusterings • Convenient visualizations with dendrograms
Disadvantages	• Bad when clusters are not spherical or have different variances • Must choose K a priori • Local solution; depends on initialization	• Depends *heavily* on linkage (single, complete, average, Ward's linkage) • Greedy search
Shared Disadvantages	• Irrelevant variables are treated as equals with relevant ones • Suffers from "Curse of Dimensionality": computing distances between two points in high dimensions is hard and inaccurate	

Do dimension reduction first and then clustering using the dimension-reduced data

Other clustering methods: mixture models, DBSCAN, spectral clustering, K-mediods

Additional Resources

A more detailed review of these unsupervised learning methods + more can be found at https://tiffanymtang.github.io/dsip-s25/#unsupervised-learning

- Also includes R and Python code for implementing these methods

The quarto notebook that generated this walkthrough can be found here:

https://github.com/tiffanymtang/dsip-s25/blob/main/unsupervised_learning/notebooks/unsupervised_learning.qmd

Additional Resources for Unsupervised Learning Methods

- Elements of Statistical Learning Textbook Chapter 14