Cluster Analysis: Methods and Applications

A 4-Hour Introduction to Unsupervised Learning

MSc Statistical Data Analysis

January 2, 2026

Course Overview

Learning Objectives:

- Understand basic ideas of cluster analysis
- Know when to apply clustering methods
- Explain and apply k-Means algorithm
- Explain and apply k-Medoids (PAM)
- Perform hierarchical cluster analysis
- Select optimal number of clusters
- Compare clustering methods

Prerequisites:

- Basic statistics knowledge
- Familiarity with R programming
- Understanding of distance metrics

Course Structure (4 Hours):

1. Hour 1: Introduction and k-Means
2. Hour 2: k-Medoids and Distance Metrics
3. Hour 3: Hierarchical Clustering
4. Hour 4: Optimal k Selection

Materials Included:

- Slide deck with visualizations
- R code examples
- Practice exercises
- Reference materials

Roadmap: Hour 1

Topics Covered:

- What is cluster analysis?
- Types of clustering methods
- When to use clustering
- k-Means: Intuition
- k-Means: Algorithm
- k-Means: Mathematics
- k-Means: Implementation
- Initialization and convergence

Learning Outcomes:

- Define clustering and unsupervised learning
- Identify appropriate use cases
- Explain k-Means algorithm steps
- Implement k-Means in R
- Recognize limitations

What is Clustering?

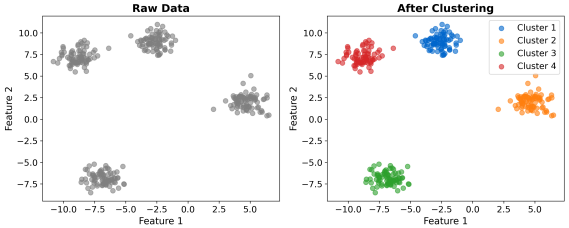
Intuition ("Birds of a feather flock together")
Clustering finds natural groups in data without being told what to look for.

Definition:

- Group similar objects together
- Unsupervised learning method
- No predefined labels
- Discover hidden patterns

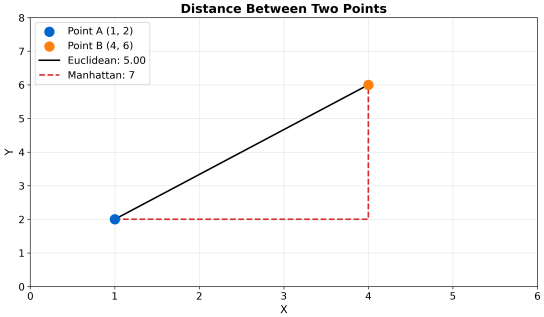
Key Concepts:

- **Within-cluster similarity**: Objects in same cluster are similar
- **Between-cluster dissimilarity**: Objects in different clusters are dissimilar
- **Distance measure**: Quantifies similarity



Natural groupings in data: three distinct clusters visible in this 2D scatter plot

Distance Between Points

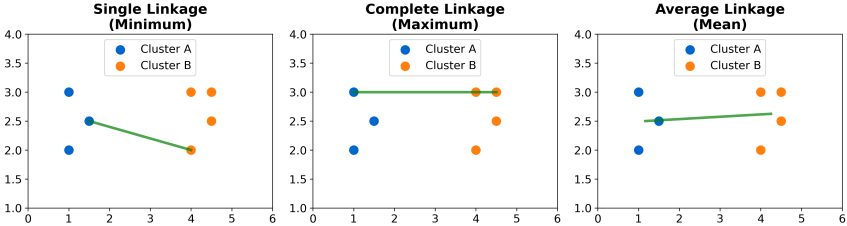


Euclidean (L2): $d = \sqrt{\sum (x_i - y_i)^2}$ – straight line

Manhattan (L1): $d = \sum |x_i - y_i|$ – grid distance

Minkowski (Lp): $d = (\sum |x_i - y_i|^p)^{1/p}$ – general family

Distance: Point-to-Cluster and Cluster-to-Cluster



Point-to-Cluster Distance:

- **Single linkage**: $d(P, C) = \min_{q \in C} d(P, q)$
- **Complete linkage**: $d(P, C) = \max_{q \in C} d(P, q)$
- **Average linkage**: $d(P, C) = \frac{1}{|C|} \sum_{q \in C} d(P, q)$

Cluster-to-Cluster Distance:

- **Single**: $d(A, B) = \min_{a \in A, b \in B} d(a, b)$
- **Complete**: $d(A, B) = \max_{a \in A, b \in B} d(a, b)$
- **Average**: $d(A, B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$
- Used in hierarchical clustering

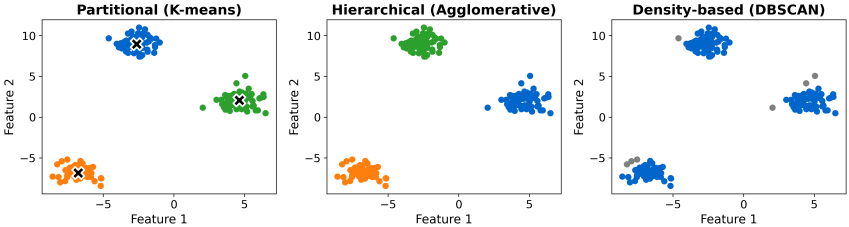
Types of Clustering Methods

1. **Partitioning Methods:**
   - Divide data into k clusters
   - Each point belongs to one cluster
   - Examples: k-Means, k-Medoids

2. **Hierarchical Methods:**
   - Build hierarchy of clusters
   - Tree-like structure (dendrogram)
   - Examples: Agglomerative, Divisive

3. **Density-Based Methods:**
   - Clusters as high-density regions
   - Can find arbitrary shapes
   - Examples: DBSCAN, OPTICS

4. **Model-Based Methods:**
   - Assume statistical model
   - Estimate parameters
   - Examples: Gaussian Mixture Models



k-Means Clustering

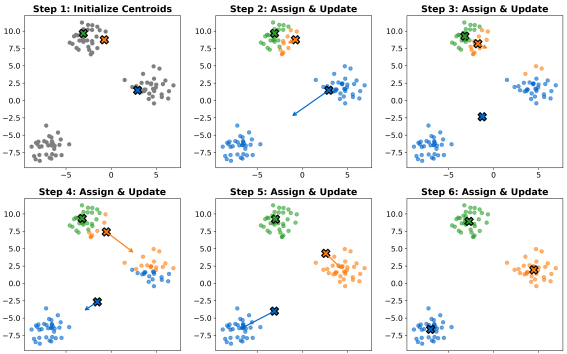
k-Means: Intuition

Main Idea:

- Partition data into k clusters
- Each cluster has a center (centroid)
- Assign points to nearest centroid
- Update centroids iteratively

Key Properties:

- Simple and fast
- Works well for spherical clusters
- Requires specifying k in advance
- Sensitive to initialization



Algorithm iterations: random initialization, assignment, update, convergence

k-Means is the most popular clustering algorithm due to its simplicity

k-Means Algorithm: Step 4 - Iterate

Iteration Process:

1. Repeat assignment and update steps
2. Continue until convergence

Convergence Criteria:

- Centroids stop moving
- Assignments stop changing
- Maximum iterations reached
- Objective function change below threshold

Pseudo-Code:

1. Initialize: Select k random centroids
2. **Repeat:**
   - Assignment: Assign each point to nearest centroid
   - Update: Recalculate centroids as cluster means
3. **Until:** Convergence

Typically converges in 10-20 iterations

k-Means: Objective Function

What k-Means Minimizes:
Plain English: "How far are points from their cluster centers?"
Within-Cluster Sum of Squares (WCSS):

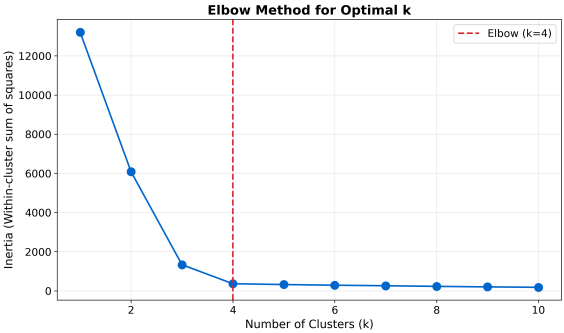
$$J = \sum_{j=1}^{k} \sum_{x_i \in C_j} ||x_i - \mu_j||^2$$

Interpretation:

- Sum of squared distances
- From points to their centroids
- Measures cluster compactness
- Lower is better

Properties:

- J decreases with each iteration
- Algorithm always converges
- May converge to local optimum
- Not guaranteed to find global optimum



k-Means is a coordinate descent algorithm minimizing WCSS

Objective function decreases until convergence

k-Means: Initialization Problem

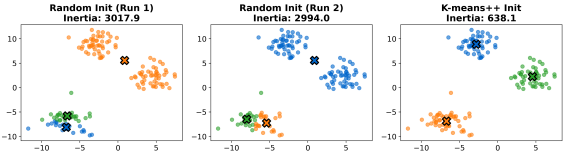
The Problem:

- Random initialization can lead to poor results
- Different runs give different clusterings
- May converge to local optima

Solutions:

1. Run multiple times (nstart parameter)
2. Choose best result (lowest WCSS)
3. Use smart initialization (k-means++)

Always use nstart ¿ 1 to avoid poor local optima



Three different random initializations leading to different final clusterings

k-Means++: Smart Initialization

Algorithm:

1. Choose first centroid uniformly at random from data points
2. For each remaining centroid:
   - Calculate $D(x)$ = distance from each point to nearest existing centroid
   - Choose next centroid with probability $\propto D(x)^2$
3. Repeat until k centroids selected

Advantages:

- Spreads initial centroids out
- Reduces chance of poor initialization
- Provably better than random
- Often converges faster
- Still allows multiple runs

In R:

- Some implementations use it automatically
- Or specify algorithm="Lloyd" vs algorithm="Hartigan-Wong"

k-Means: Advantages and Disadvantages

Advantages:

- Simple to understand
- Fast: $O(nkd \cdot i)$ complexity
- Works well for spherical clusters
- Scales to large datasets
- Easy to implement
- Widely used and tested

where:

- n = data points
- k = clusters
- d = dimensions
- i = iterations

Disadvantages:

- Must specify k in advance
- Sensitive to initialization
- Assumes spherical clusters
- Sensitive to outliers
- Not suitable for non-convex shapes
- Assumes equal cluster sizes
- Mean may not be representative

When to be Cautious:

- Clusters have different sizes
- Clusters have non-spherical shapes
- Data contains outliers

Section 1 Summary

Key Concepts Covered:

- Clustering as unsupervised learning
- Types of clustering methods
- Applications of clustering
- k-Means algorithm steps
- Objective function (WCSS)
- Implementation in R

k-Means Algorithm:

1. Initialize k centroids
2. Assign points to nearest centroid
3. Update centroids as cluster means
4. Repeat until convergence

Remember:

- Use nstart ¿ 1
- k-Means minimizes WCSS
- Works best for spherical clusters

k-Medoids in R: Implementation

```r
# Load required package
library(cluster)

# Generate example data with outlier
set.seed(123)
x <- rbind(
  matrix(rnorm(100, mean=0, sd=0.3), ncol=2),
  matrix(rnorm(100, mean=1, sd=0.3), ncol=2),
  c(5, 5) # outlier
)

# Apply PAM
pam_result <- pam(x, k=2)

# Extract results
medoids <- pam_result$medoids
clusters <- pam_result$clustering
```

```r
# Visualize
plot(x, col=clusters, pch=19,
     main="PAM Clustering",
     xlab="Feature 1", ylab="Feature 2")
points(medoids, col=1:2, pch=8, cex=3, lwd=2)

# Compare with k-Means
km_result <- kmeans(x, centers=2, nstart=25)
plot(x, col=km_result$cluster, pch=19,
     main="k-Means Clustering")
points(km_result$centers, col=1:2, pch=3,
       cex=3, lwd=3)
```

PAM is available in the cluster package in R

Distance Metrics

Why Distance Metrics Matter

Central to Clustering:

- Clustering depends on similarity
- Similarity measured by distance
- Different distances $\rightarrow$ different clusters
- No "correct" distance in general

Choice Depends On:

- Data type (continuous, categorical, mixed)
- Feature scales
- Domain knowledge
- Computational constraints

Distance Function Properties:

A function $d(x, y)$ is a distance metric if:

1. Non-negativity: $d(x, y) \geq 0$
2. Identity: $d(x, x) = 0$
3. Symmetry: $d(x, y) = d(y, x)$
4. Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$

Always consider which distance metric is appropriate for your data

Euclidean Distance

Definition:
Straight-line distance between two points:

$$d_E(x, y) = \sqrt{\sum_{j=1}^{d} (x_j - y_j)^2}$$

Example (2D):

$$d_E((1,2),(4,6)) = \sqrt{(4-1)^2 + (6-2)^2}$$

$$= \sqrt{9+16} = \sqrt{25} = 5$$

Properties:

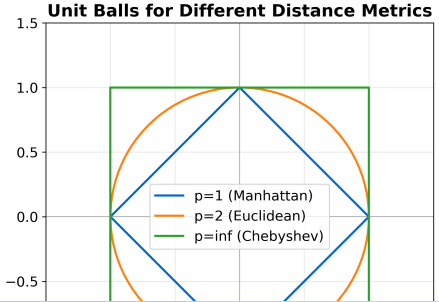
- Most common distance
- Intuitive geometric interpretation
- Corresponds to L_2 norm

When to Use:

- Continuous features
- Features on similar scales
- Geometric interpretation important
- Most clustering algorithms assume it

When to be Careful:

- Features on different scales
- Presence of outliers
- High-dimensional data (curse of dimensionality)



Minkowski Distance Family

Definition:
Generalized L_p norm:

$$d_p(x, y) = \left(\sum_{j=1}^{d} |x_j - y_j|^p \right)^{1/p}$$

Special Cases:

- $p = 1$: Manhattan distance
- $p = 2$: Euclidean distance
- $p \to \infty$: Chebyshev distance

$$d_\infty(x, y) = \max_j |x_j - y_j|$$

Choice of p is a hyperparameter that can be tuned

Effect of p:

- Small p: more robust, sparse
- Large p: dominated by largest difference
- $p = 2$ is compromise

Example:
For points $(0, 0)$ and $(3, 4)$:

$$p = 1: \quad d_1 = 7$$
$$p = 2: \quad d_2 = 5$$
$$p = \infty: \quad d_\infty = 4$$

Other Distance Metrics

Cosine Distance:

$$d_{cos}(x, y) = 1 - \frac{x \cdot y}{||x|| \cdot ||y||}$$

Measures angle between vectors, not magnitude
Use for: Text data, high dimensions

Correlation Distance:

$$d_{cor}(x, y) = 1 - \text{cor}(x, y)$$

Measures linear relationship
Use for: Time series, gene expression

Hamming Distance:

$$d_H(x, y) = \sum_{j=1}^{d} \mathbb{1}(x_j \neq y_j)$$

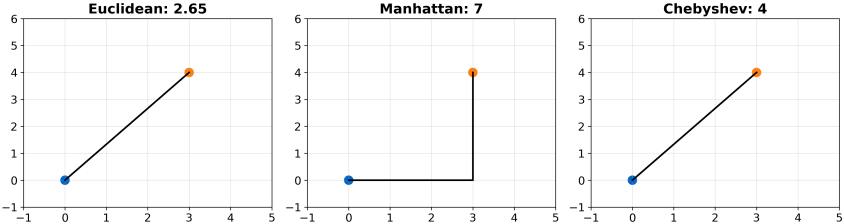
Counts number of differences
Use for: Categorical data, binary data

Mahalanobis Distance:

$$d_M(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}$$

Accounts for correlations via covariance S
Use for: Correlated features, different scales

Distance Metrics: Visual Comparison



Euclidean: Circular contours **Manhattan:** Diamond-shaped contours **Chebyshev:** Square contours

Same data, different distance metrics lead to different clusterings

Data Standardization: Why It Matters

The Problem:

- Features on different scales
- Distance dominated by large-scale features
- Small-scale features ignored

Example:
Customer data:

- Age: 20-80 (range 60)
- Income: 20,000-200,000 (range 180,000)

Distance heavily influenced by income

Solution: Standardization
Z-score normalization:

$$z_j = \frac{x_j - \mu_j}{\sigma_j}$$

After standardization:

- Mean $= 0$
- Standard deviation $= 1$
- All features on same scale

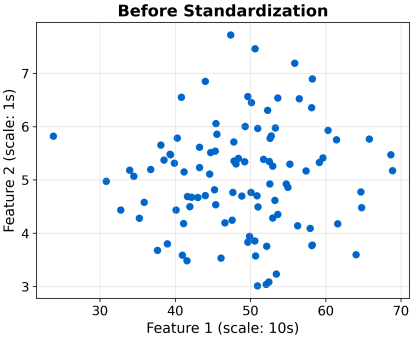
Alternative: Min-Max Scaling

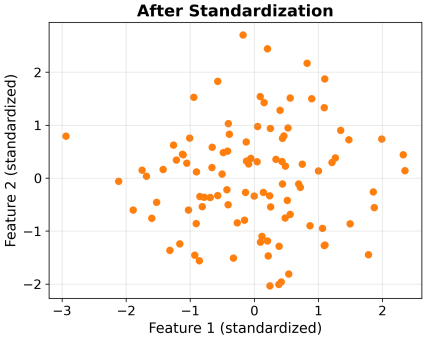
$$x'_j = \frac{x_j - \min(x_j)}{\max(x_j) - \min(x_j)}$$

Scales to [0, 1] range

Always standardize features before clustering unless scales are comparable

Standardization: Visual Impact





Left (Unstandardized):

- Clustering dominated by X axis
- Y axis variation ignored

Right (Standardized):

- Both dimensions contribute equally
- More meaningful clusters

Distance Metrics in R

```r
# Example data
x <- matrix(c(1,2,4,5,2,3,6,7), ncol=2, byrow=TRUE)

# Euclidean (default)
d_euclidean <- dist(x, method="euclidean")

# Manhattan
d_manhattan <- dist(x, method="manhattan")

# Minkowski with p=3
d_minkowski <- dist(x, method="minkowski", p=3)

# Maximum (Chebyshev)
d_max <- dist(x, method="maximum")
```

```r
# Standardization
x_scaled <- scale(x)

# Clustering with different distances
km_euclidean <- kmeans(x, centers=2)
km_after_scaling <- kmeans(x_scaled, centers=2)

# PAM with Manhattan distance
library(cluster)
pam_manhattan <- pam(x, k=2, metric="manhattan")

# Custom distance matrix
custom_dist <- as.dist(my_distance_function(x))
hclust_result <- hclust(custom_dist)
```

dist() function supports multiple distance metrics

Roadmap: Hour 3

Topics Covered:

- Hierarchical clustering overview
- Agglomerative vs Divisive
- Dendrograms
- Linkage methods: Single
- Linkage methods: Complete
- Linkage methods: Average
- Linkage methods: Ward's
- Comparing linkage methods
- Implementation in R

Learning Outcomes:

- Explain hierarchical clustering
- Interpret dendrograms
- Understand linkage methods
- Choose appropriate linkage
- Implement in R with hclust
- Cut dendrograms to extract clusters

Hierarchical Clustering

Agglomerative Hierarchical Clustering Algorithm

Algorithm Steps:

1. Start: Each point is its own cluster
2. Compute distance matrix between all clusters
3. Merge two closest clusters
4. Update distance matrix
5. Repeat steps 2-4 until one cluster remains

Complexity:

- Time: $O(n^3)$ naive, $O(n^2 \log n)$ optimized
- Space: $O(n^2)$ for distance matrix
- More expensive than k-Means

Example with 4 Points:

1. Initial: $\{A\}, \{B\}, \{C\}, \{D\}$
2. Merge closest: $\{A, B\}, \{C\}, \{D\}$
3. Merge next closest: $\{A, B\}, \{C, D\}$
4. Final merge: $\{A, B, C, D\}$

Key Question:

How do we measure distance between clusters (not just points)?
$\rightarrow$ Linkage methods!

Dendrograms: Visual Representation

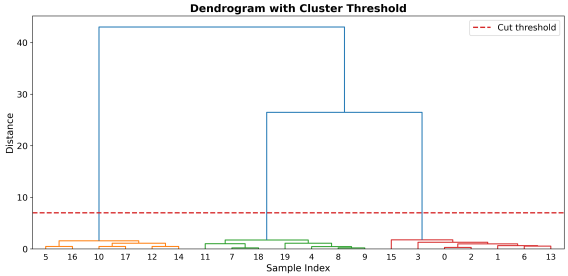
What is a Dendrogram?

- Tree diagram showing cluster hierarchy
- Vertical axis = distance/dissimilarity
- Horizontal axis = observations
- Height of merge = distance between clusters

How to Read:

- Bottom: individual observations
- Moving up: clusters merge
- Higher merge = more dissimilar
- Top: all in one cluster

Dendrograms provide a complete picture of the clustering structure



Dendrogram showing cluster hierarchy with merge heights

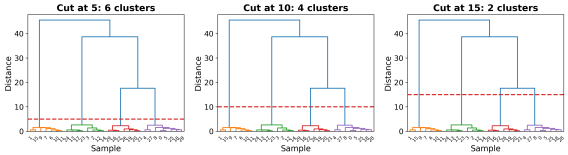
Cutting Dendrograms

Extracting Clusters:

- Draw horizontal line through dendrogram
- Number of branches crossed = number of clusters
- Can cut at any height
- Higher cut = fewer clusters
- Lower cut = more clusters

How to Choose Cut Height:

- Look for long vertical lines
- Gap in merge distances
- Domain knowledge
- Use validation metrics



Same dendrogram cut at different heights: $k = 2$ (top), $k = 3$ (middle), $k = 4$ (bottom)

Linkage Methods

Linkage Methods: The Core Question

The Problem:
Given:

- Cluster $C_i = \{x_1, x_2, \ldots\}$
- Cluster $C_j = \{y_1, y_2, \ldots\}$

Question: What is distance $d(C_i, C_j)$?

We Know:

- Distance between points: $d(x_a, y_b)$

We Need:

- Distance between clusters

Linkage = Aggregation Rule
Different rules give different results:

- Single linkage
- Complete linkage
- Average linkage
- Ward's method

Each has different properties and use cases

Choice of linkage significantly impacts results

Linkage method is the key parameter in hierarchical clustering

Complete Linkage (Furthest Neighbor)

Definition:
Distance = maximum distance between any two points:

$$d_{CL}(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y)$$

Properties:

- Most restrictive
- Creates compact, spherical clusters
- Resistant to chaining
- Tends to produce balanced clusters

Advantages:

- Robust to outliers (to some extent)
- Produces tight clusters
- Good for well-separated groups

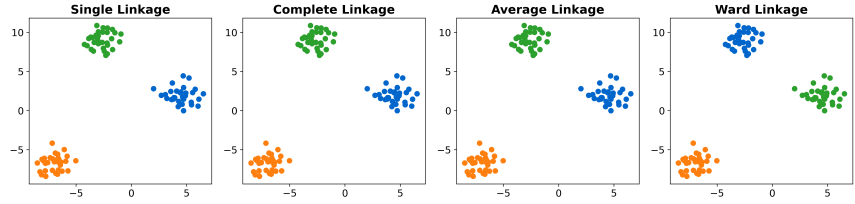
Disadvantages:

- May break natural elongated clusters
- Can be too restrictive
- Sensitive to outliers in other way

When to Use:

- Want compact clusters
- Clusters well-separated
- Default choice often

Linkage Methods: Visual Comparison



Single: Chain-like, elongated **Complete:** Compact, spherical **Average:** Balanced compromise **Ward's:** Very compact, equal size

Same data, four different linkage methods produce different clusterings

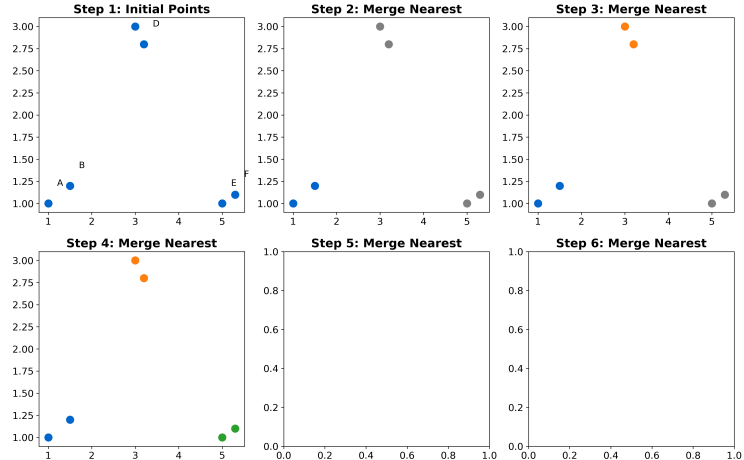
Hierarchical Clustering in R

```r
# Generate example data
set.seed(123)
x <- rbind(
  matrix(rnorm(50, mean=0), ncol=2),
  matrix(rnorm(50, mean=3), ncol=2)
)

# Compute distance matrix
d <- dist(x, method="euclidean")

# Hierarchical clustering with different linkages
hc_single <- hclust(d, method="single")
hc_complete <- hclust(d, method="complete")
hc_average <- hclust(d, method="average")
hc_ward <- hclust(d, method="ward.D2")
```

```r
# Plot dendrogram
plot(hc_ward, main="Dendrogram (Ward's)",
     xlab="", sub="", cex=0.7)

# Cut tree to get k clusters
clusters <- cutree(hc_ward, k=2)

# Draw rectangles around clusters
rect.hclust(hc_ward, k=2, border="red")

# Visualize clusters in original space
plot(x, col=clusters, pch=19,
     main="Hierarchical Clustering Results")
```

hclust() is the main function for hierarchical clustering in R

Step-by-Step Dendrogram Example



Agglomerative clustering process shown step-by-step: (1) Initial individual points, (2) First merges, (3) Continued merging, (4) Final dendrogram

Hierarchical Clustering: Advantages and Disadvantages

Advantages:

- No need to specify k in advance
- Dendrogram provides complete picture
- Deterministic (no random initialization)
- Works with any distance metric
- Can cut at any level
- Captures hierarchical structure
- Easy to interpret visually

Disadvantages:

- Computationally expensive: $O(n^2 \log n)$
- Memory intensive: $O(n^2)$
- Not suitable for very large datasets
- Cannot undo previous merges
- Sensitive to noise and outliers (single linkage)
- May produce imbalanced clusters

Typical Size Limits:

- Practical: $n < 10{,}000$
- Large: $n < 50{,}000$ (with optimization)

Roadmap: Hour 4

Topics Covered:

- The k-selection problem
- Elbow method
- Silhouette analysis
- Gap statistic
- Visual validation
- Cluster quality metrics
- Method comparison
- Best practices
- Common pitfalls

Learning Outcomes:

- Understand k-selection challenge
- Apply multiple methods
- Interpret elbow plots
- Calculate silhouette coefficients
- Compare different k values
- Validate clustering results
- Avoid common mistakes

The k-Selection Problem

The Fundamental Question: "How many groups are really there?"

The Challenge:

- k-Means and PAM require specifying k
- True k is usually unknown
- Different k gives different clusterings
- No universally correct answer

Why It's Hard:

- Objective function always improves with larger k
- $k = n$ gives perfect clustering (but useless)
- Need balance: fit vs complexity
- Clusters may not exist at all

Approaches to Finding k:

1. Statistical methods
   - Elbow method
   - Silhouette analysis
   - Gap statistic
2. Visual inspection
3. Domain knowledge
4. Multiple methods combined

Important:

- Use multiple methods
- Consider interpretability
- Domain knowledge is key

k-selection is more art than science; combine multiple approaches

Silhouette Analysis

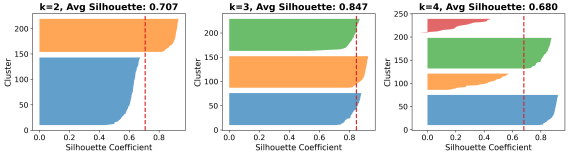
Silhouette Plots: Visual Interpretation

Silhouette Plot:

- One bar per data point
- Grouped by cluster
- Sorted within cluster
- Longer bars = better fit
- Negative values = poor fit

Good Clustering:

- Most bars positive and long
- Similar widths across clusters
- High average silhouette



Silhouette plots for $k = 2, 3, 4$. Higher average and fewer negative values indicate better clustering.

Silhouette analysis provides both global and point-wise clustering quality

Other Cluster Quality Metrics

Dunn Index:

$$D = \frac{\min_{i \neq j} d(C_i, C_j)}{\max_k d'(C_k)}$$

- Ratio of minimum inter-cluster distance to maximum intra-cluster distance
- Higher is better
- Favors well-separated, compact clusters

Davies-Bouldin Index:

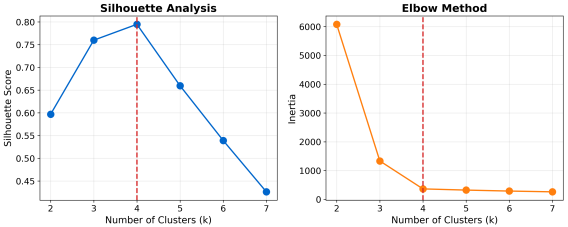
$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$$

- Average similarity between each cluster and its most similar one
- Lower is better

Calinski-Harabasz Index:

$$CH = \frac{BSS/(k-1)}{WSS/(n-k)}$$

- Ratio of between-cluster to within-cluster variance
- Higher is better
- Similar to F-statistic in ANOVA



Multiple validation metrics across different k values

Common Pitfalls in Clustering

1. Not Standardizing Features

- Different scales dominate
- Solution: Always standardize

2. Ignoring Outliers

- Can distort clusters
- Solution: Remove or use robust methods

3. Using Single Method

- May miss true structure
- Solution: Try multiple algorithms

4. Forgetting Initialization

- Random starts matter
- Solution: Use nstart ¿ 1

5. Overinterpreting Results

- Algorithms always find clusters
- Solution: Validate, validate, validate

6. Ignoring Domain Knowledge

- Statistical optimum may not be practical
- Solution: Consult experts

7. Not Checking Assumptions

- k-Means assumes spherical clusters
- Solution: Match method to data

8. Skipping Visualization

- Numbers don't show everything
- Solution: Always plot your clusters

Key Takeaways

1. **Clustering is exploratory** - discover patterns, don't confirm hypotheses
2. **No "correct" clustering** - multiple valid ways to group data
3. **Always standardize** - feature scales matter greatly
4. **Try multiple methods** - different algorithms, different insights
5. **Validate extensively** - use multiple metrics and visual inspection
6. **Domain knowledge is key** - statistics guide, expertise decides
7. **Interpretation matters** - clusters must be understandable and actionable
8. **Check assumptions** - match method to data characteristics
9. **Document everything** - reproducibility and transparency essential

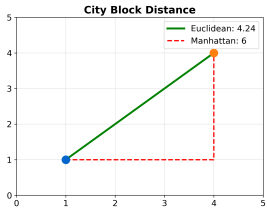
Thank You!

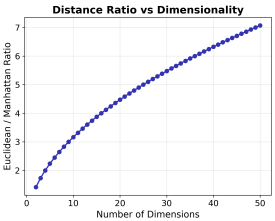
Questions?

Good luck with your clustering projects!

Distance Metrics: Manhattan vs Euclidean

Visual Comparison:





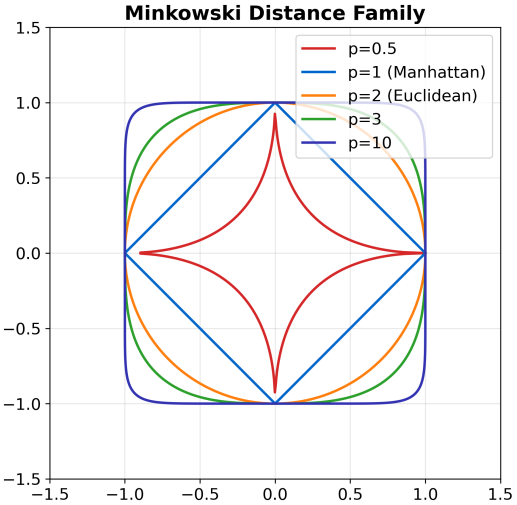
Key Differences:

- **Euclidean**: Straight line (circular contours)
- **Manhattan**: Grid path (diamond contours)
- Same center, different distance calculations
- Choice affects clustering results

When to use:

- *Euclidean*: General-purpose, geometric data
- *Manhattan*: Grid-like data, city blocks

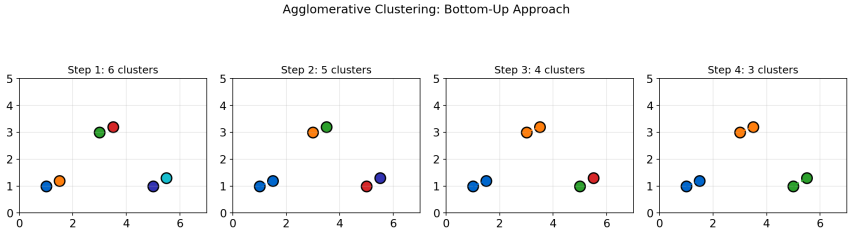
Minkowski Distance Family



Formula: $d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$

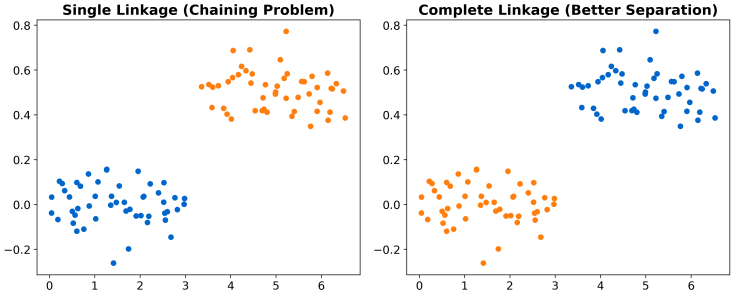
- $p = 1$: Manhattan distance (L1 norm)
- $p = 2$: Euclidean distance (L2 norm)
- $p = \infty$: Chebyshev distance (L∞ norm, max difference)

Agglomerative Clustering: Step-by-Step



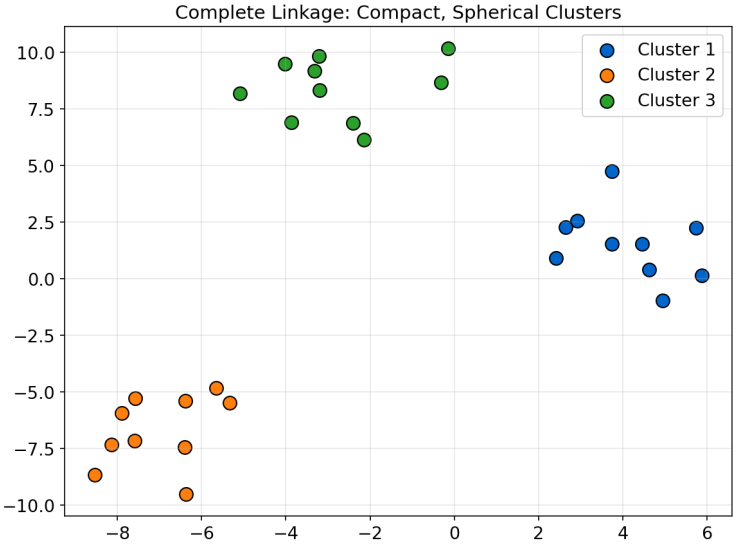
Process: Start with individual points, merge closest pairs until single cluster

Single Linkage: Chaining Effect



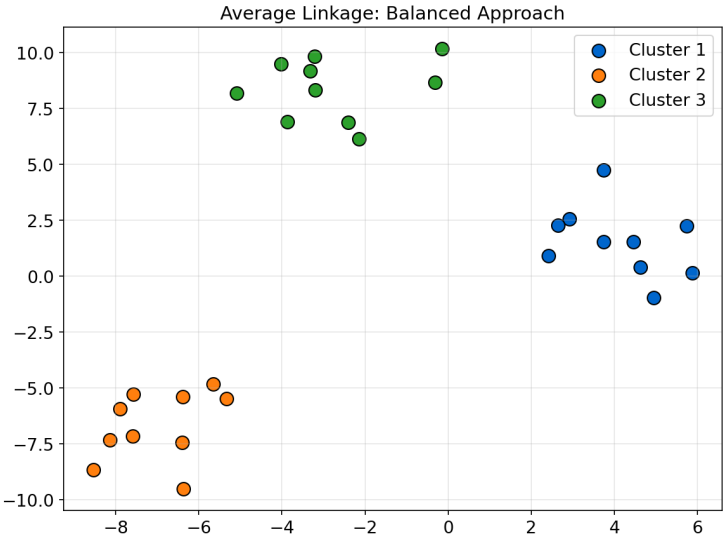
Problem: Chains can connect distinct clusters inappropriately

Complete Linkage: Compact Clusters



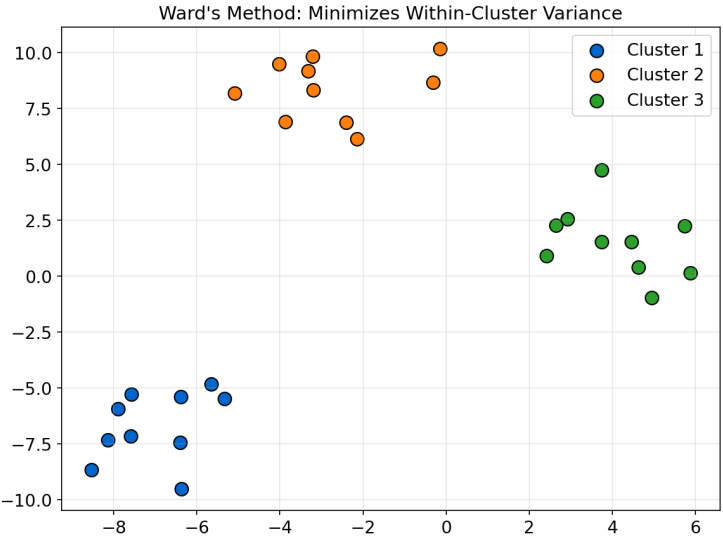
Advantage: Produces tight, spherical clusters; minimizes maximum distance

Average Linkage: Balanced Approach



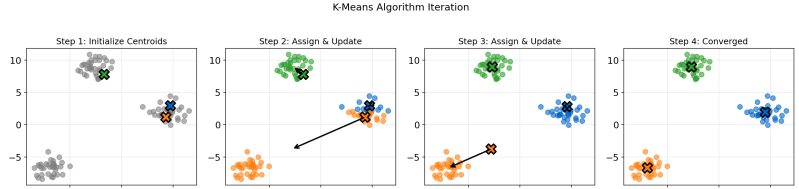
Balance: Compromise between single (chaining) and complete (compactness)

Ward's Method: Equal-Size Bias



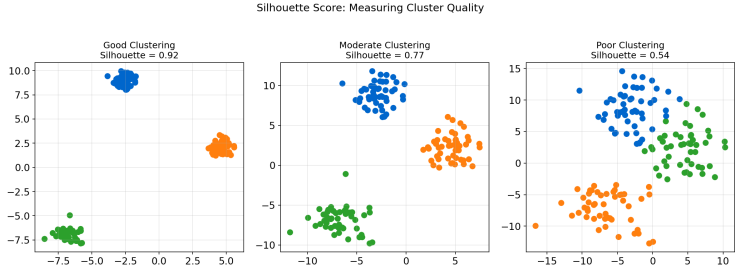
Tendency: Both k-Means and Ward's prefer equal-sized clusters

k-Means Algorithm: Visual Process



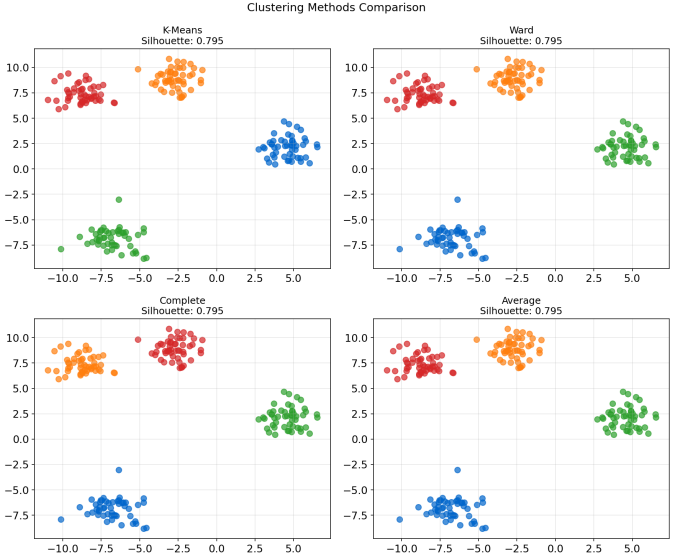
Four steps: Initialize → Assign → Update → Converge

Silhouette Coefficient: Intuition



Formula: $s(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))}$ where $a(i)$ = avg distance to same cluster, $b(i)$ = avg distance to nearest other cluster

All Methods Comparison



No single correct answer: Choice depends on data structure and domain knowledge

Appendix

Detailed Content and Advanced Topics

This appendix contains:

- k-Medoids (PAM) algorithm details
- Step-by-step algorithm walkthroughs
- Detailed R code examples
- Advanced clustering topics
- Additional comparison matrices
- Common pitfalls and solutions

Table of Contents

Supervised vs Unsupervised Learning

Supervised Learning:

- Training data with labels
- Learn mapping: $X \rightarrow Y$
- Examples: Classification, Regression
- Goal: Predict labels for new data

Example: Email spam detection

- X = email features
- Y = spam/not spam (known)

Unsupervised Learning:

- Data without labels
- Discover structure: $X \rightarrow$ patterns
- Examples: Clustering, Dimensionality reduction
- Goal: Find hidden patterns

Example: Customer segmentation

- X = customer features
- Y = ? (unknown groups)

Clustering is exploratory: we discover groups rather than predict them

When to Use Clustering

Applications:

- Customer segmentation
- Image segmentation
- Document organization
- Gene expression analysis
- Anomaly detection
- Data compression
- Recommendation systems

When Clustering is Appropriate:

- No labels available
- Exploratory analysis needed
- Want to understand data structure
- Reduce data complexity
- Find natural groupings

When to be Careful:

- Data has no structure
- Need interpretable results
- Groups are not well-separated

Clustering is a tool for discovery, not confirmation

k-Means Algorithm: Step 1 - Initialize

Initialization Step:

1. Choose number of clusters k
2. Randomly select k data points as initial centroids
3. OR use k-means++ initialization

Why it Matters:

- Different initializations can lead to different results
- Bad initialization can trap algorithm in local optimum
- k-means++ provides better starting points

Notation:

- n = number of data points
- k = number of clusters
- $X = \{x_1, x_2, \ldots, x_n\}$ = data points
- $\mu_1, \mu_2, \ldots, \mu_k$ = centroids

Initial Centroids:

$$\mu_j^{(0)} \text{ for } j = 1, 2, \ldots, k$$

Chosen randomly from data points or using k-means++

k-Means Algorithm: Step 2 - Assign

Assignment Step:
For each data point x_i:

1. Calculate distance to all centroids
2. Assign to nearest centroid

Mathematical Formulation:

$$c_i = \arg\min_j ||x_i - \mu_j||^2$$

where c_i is the cluster assignment for point x_i

Distance is typically Euclidean:

$$||x_i - \mu_j||^2 = \sum_{d=1}^{D} (x_{id} - \mu_{jd})^2$$

Example:
Point $x = (2, 3)$
Centroids:

- $\mu_1 = (1, 2)$
- $\mu_2 = (5, 6)$

Distances:

$$||x - \mu_1||^2 = (2-1)^2 + (3-2)^2 = 2$$
$$||x - \mu_2||^2 = (2-5)^2 + (3-6)^2 = 18$$

$\Rightarrow$ Assign x to cluster 1

Each point is assigned to exactly one cluster

k-Means Algorithm: Step 3 - Update

Update Step:
For each cluster $j = 1, 2, \ldots, k$:

1. Find all points assigned to cluster j
2. Calculate mean of these points
3. This mean becomes new centroid

Mathematical Formulation:

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

where $C_j = \{x_i : c_i = j\}$ is the set of points in cluster j

Centroids move to the center of mass of their assigned points

Example:
Cluster 1 contains points:

- $(1, 2)$
- $(2, 3)$
- $(1, 3)$

New centroid:

$$\begin{aligned} \mu_1 &= \frac{1}{3}[(1, 2) + (2, 3) + (1, 3)] \\ &= \frac{1}{3}(4, 8) \\ &= (1.33, 2.67) \end{aligned}$$

k-Means: Worked Example

Given Data:

- Points: $(1,1), (2,1), (4,3), (5,4)$
- $k = 2$
- Initial centroids: $\mu_1 = (1,1), \mu_2 = (2,1)$

Iteration 1 - Assignment:

- $(1,1)$: nearest to $\mu_1 \rightarrow$ C1
- $(2,1)$: nearest to $\mu_2 \rightarrow$ C2
- $(4,3)$: nearest to $\mu_2 \rightarrow$ C2
- $(5,4)$: nearest to $\mu_2 \rightarrow$ C2

Iteration 1 - Update:

C1: $\{(1,1)\}$

$$\mu_1 = (1,1)$$

C2: $\{(2,1), (4,3), (5,4)\}$

$$\mu_2 = \frac{1}{3}[(2,1) + (4,3) + (5,4)] = (3.67, 2.67)$$

Iteration 2 - Assignment:

Reassign points to updated centroids, continue until no changes occur

k-Means in R: Basic Implementation

```r
# Generate example data
set.seed(123)
x <- rbind(
  matrix(rnorm(100, mean=0, sd=0.3), ncol=2),
  matrix(rnorm(100, mean=1, sd=0.3), ncol=2),
  matrix(rnorm(100, mean=c(1,0), sd=0.3), ncol=2)
)

# Apply k-Means
km_result <- kmeans(x, centers=3, nstart=25)

# Extract results
centers <- km_result$centers
clusters <- km_result$cluster
wcss <- km_result$tot.withinss
```

```r
# Visualize results
plot(x, col=clusters, pch=19,
     main="k-Means Clustering",
     xlab="Feature 1", ylab="Feature 2")
points(centers, col=1:3, pch=3, cex=3, lwd=3)
legend("topright", legend=c("Cluster 1","Cluster 2","Cluster 3"),
       col=1:3, pch=19)

# Print summary
cat("Within-cluster SS:", wcss, "\n")
cat("Between-cluster SS:", km_result$betweenss, "\n")
cat("Cluster sizes:", km_result$size, "\n")
```

nstart parameter runs k-Means multiple times with different initializations

Roadmap: Hour 2

Topics Covered:

- Limitations of k-Means
- k-Medoids (PAM) algorithm
- Comparison: k-Means vs k-Medoids
- Distance metrics
- Euclidean distance
- Manhattan distance
- Minkowski distance
- Choosing distance metrics
- Data standardization

Learning Outcomes:

- Recognize when k-Means fails
- Explain PAM algorithm
- Implement k-Medoids in R
- Understand distance metrics
- Choose appropriate metrics
- Apply standardization

k-Medoids (PAM)

Limitations of k-Means

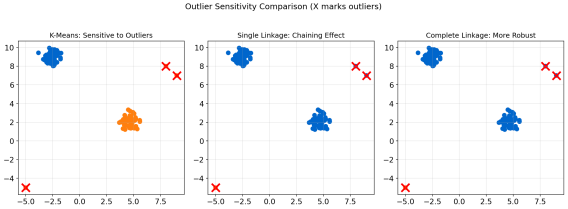
Problem 1: Outlier Sensitivity

- Mean is sensitive to outliers
- Single outlier can pull centroid
- Entire cluster affected

Problem 2: Non-Representative Centers

- Centroid may not be an actual data point
- May fall in empty space
- Hard to interpret

k-Medoids addresses these limitations by using medoids instead of means



k-Means (left) pulls centroid toward outlier; k-Medoids (right) is robust

k-Medoids: Main Idea

Key Concept:

- Use medoid instead of mean
- Medoid = most centrally located point
- Must be an actual data point
- More robust to outliers

Definition:
Medoid of cluster C_j is the point m_j that minimizes:

$$\sum_{x_i \in C_j} d(x_i, m_j)$$

where d is a distance function

Mean vs Median:

- Mean = average (can be non-robust)
- Median = middle value (robust)
- Medoid = multidimensional median

Example (1D):
Data: 1, 2, 3, 4, 100

- Mean = 22 (pulled by outlier)
- Median = 3 (robust)

Medoid extends this to multiple dimensions

PAM Algorithm: Partitioning Around Medoids

Step 1 - Build:

1. Choose k initial medoids
2. For each pair (medoid, non-medoid):
   - Swap them
   - Calculate total cost
   - Keep swap if cost decreases
3. Repeat until no improvement

Step 2 - Assign:

- Assign each point to nearest medoid
- Same as k-Means assignment

Objective Function:
Minimize total dissimilarity:

$$J = \sum_{j=1}^{k} \sum_{x_i \in C_j} d(x_i, m_j)$$

where m_j is the medoid of cluster j

Key Difference from k-Means:

- k-Means: μ_j can be any point
- PAM: m_j must be a data point

PAM is more robust but computationally more expensive than k-Means

k-Means vs k-Medoids: Comparison

k-Means:

- Centers are means (centroids)
- Not necessarily data points
- Fast: $O(nkd \cdot i)$
- Minimizes sum of squared distances
- Assumes Euclidean distance
- Sensitive to outliers
- Best for spherical clusters

When to Use k-Means:

- Large datasets
- No outliers
- Spherical clusters
- Speed is important

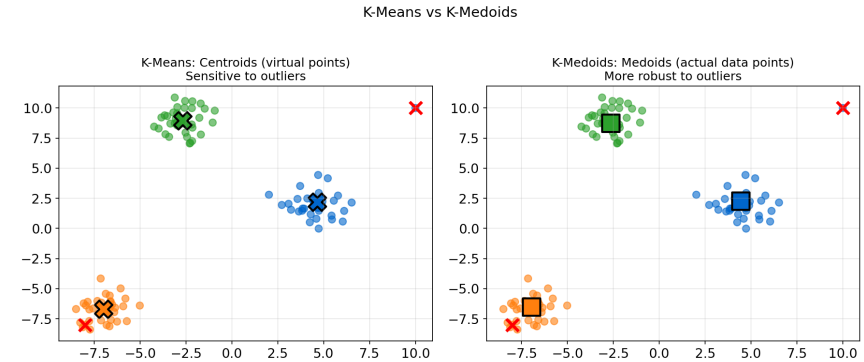
k-Medoids (PAM):

- Centers are medoids
- Must be data points
- Slower: $O(k(n-k)^2 \cdot i)$
- Minimizes sum of distances
- Works with any distance metric
- Robust to outliers
- More flexible

When to Use PAM:

- Moderate-sized datasets
- Outliers present
- Non-Euclidean distances needed
- Interpretable centers wanted

Visual Comparison: k-Means vs k-Medoids



Left (k-Means):

- Centroids (crosses) not data points
- Affected by outliers

Right (k-Medoids):

- Medoids are actual data points
- Robust to outliers

Manhattan Distance

Definition:
Sum of absolute differences (taxicab distance):

$$d_M(x, y) = \sum_{j=1}^{d} |x_j - y_j|$$

Example (2D):

$$d_M((1,2),(4,6)) = |4-1| + |6-2|$$
$$= 3 + 4 = 7$$

Properties:

- Also called city block distance
- Corresponds to L_1 norm
- More robust to outliers than Euclidean

Geometric Interpretation:

- Distance along grid lines
- Like Manhattan street grid
- Multiple paths with same distance

When to Use:

- High-dimensional data
- Outliers present
- Features represent counts
- Grid-like structure

Comparison:

- Euclidean: $d_E((1,2),(4,6)) = 5$
- Manhattan: $d_M((1,2),(4,6)) = 7$

Section 2 Summary

Key Concepts Covered:

- k-Medoids (PAM) algorithm
- Robustness to outliers
- Distance metrics
- Euclidean vs Manhattan
- Minkowski family
- Data standardization

Important Takeaways:

- PAM more robust than k-Means
- Distance choice affects results
- Always standardize features
- No universally best distance
- Consider data characteristics

Hierarchical Clustering: Overview

Main Idea:

- Build hierarchy of clusters
- No need to specify k in advance
- Result is a tree (dendrogram)
- Can cut tree at any level to get clusters

Advantages over k-Means:

- Don't need to choose k beforehand
- Provides complete clustering hierarchy
- Deterministic (no random initialization)
- Works with any distance metric

We focus on agglomerative hierarchical clustering

Two Approaches:

1. **Agglomerative (Bottom-Up):**
   - Start with n clusters (one per point)
   - Merge closest clusters iteratively
   - Continue until one cluster remains
   - More common

2. **Divisive (Top-Down):**
   - Start with one cluster (all points)
   - Split clusters recursively
   - Continue until n clusters remain
   - Less common, more complex

Single Linkage (Nearest Neighbor)

Definition:
Distance = minimum distance between any two points:

$$d_{SL}(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$$

Properties:

- Most permissive
- Creates long, chain-like clusters
- Sensitive to noise and outliers
- Can produce unbalanced clusters

Advantages:

- Can handle non-spherical clusters
- Good for elongated clusters
- Simple to compute

Disadvantages:

- Chaining effect
- Outliers create bridges
- Often produces poor clusters

When to Use:

- Clusters are elongated
- No noise/outliers
- Exploratory analysis

Average Linkage

Definition:
Distance = average of all pairwise distances:

$$d_{AL}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y)$$

Properties:

- Compromise between single and complete
- Less susceptible to outliers than single
- Less restrictive than complete
- Often gives good results

Advantages:

- Balanced approach
- Reasonably robust
- Works in many situations
- Intuitive interpretation

Disadvantages:

- More computation than single/complete
- May not be best for any specific case

When to Use:

- General-purpose method
- Unsure about cluster shape
- Want reasonable default

Ward's Method (Minimum Variance)

Definition:
Merge clusters that minimize increase in total within-cluster variance:

$$\Delta(C_i, C_j) = \frac{|C_i||C_j|}{|C_i| + |C_j|}||\mu_i - \mu_j||^2$$

where μ_i is the centroid of cluster C_i

Properties:

- Minimizes within-cluster variance
- Similar philosophy to k-Means
- Creates compact, spherical clusters
- Tends toward equal-sized clusters

Advantages:

- Often produces very good clusters
- Statistical justification
- Popular in practice
- Connects to k-Means

Disadvantages:

- Biased toward spherical clusters
- Biased toward equal-sized clusters
- More computationally intensive

When to Use:

- Expecting spherical clusters
- Want compact groups
- Default choice in many fields

Ward's method is most similar to k-Means among linkage methods

Linkage Methods: Summary Table

Method	Definition	Cluster Shape	Best For
Single	Min distance	Elongated, chains	Non-spherical shapes (if no noise)
Complete	Max distance	Compact, spherical	Well-separated groups balanced sizes
Average	Mean distance	Intermediate	General purpose compromise
Ward's	Min variance	Very compact spherical	Spherical clusters similar to k-Means

Recommendation: Try multiple linkage methods and compare results

Section 3 Summary

Key Concepts Covered:

- Hierarchical clustering concept
- Agglomerative algorithm
- Dendrograms and interpretation
- Four main linkage methods
- Linkage method comparison
- Cutting dendrograms
- Implementation in R

Important Takeaways:

- Linkage choice significantly affects results
- Ward's method often works well
- Dendrograms provide full hierarchy
- More expensive than k-Means
- Try multiple linkage methods
- Visual interpretation powerful

Choosing Optimal k

Elbow Method: Within-Cluster Sum of Squares

Method:

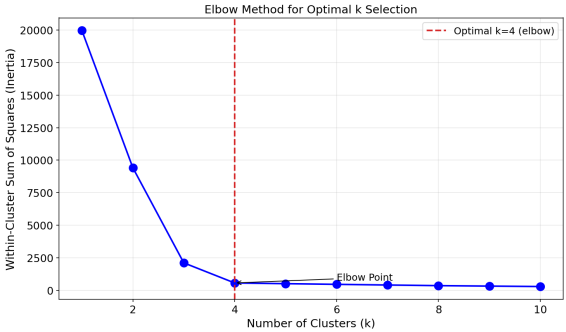
1. Run clustering for $k = 1, 2, \ldots, K$
2. Calculate WCSS for each k:

$$WCSS(k) = \sum_{j=1}^{k} \sum_{x_i \in C_j} ||x_i - \mu_j||^2$$

3. Plot WCSS vs k
4. Look for "elbow" - point where decrease slows

Intuition:

- WCSS always decreases with larger k
- Initially steep decline
- Then marginal gains diminish
- Elbow = best trade-off



Elbow at $k = 3$ suggests optimal number of clusters

Limitations:

- Elbow not always clear
- Subjective interpretation
- May not exist
- Need other methods too

Elbow Method in R

```r
# Function to compute WCSS
compute_wcss <- function(data, k) {
  km <- kmeans(data, centers=k, nstart=25)
  return(km$tot.withinss)
}

# Try different values of k
k_values <- 1:10
wcss_values <- sapply(k_values, function(k) {
  compute_wcss(x, k)
})

# Create data frame
elbow_data <- data.frame(
  k = k_values,
  WCSS = wcss_values
)
```

```r
# Plot elbow curve
plot(elbow_data$k, elbow_data$WCSS,
     type="b", pch=19, col="blue",
     xlab="Number of Clusters (k)",
     ylab="Within-Cluster Sum of Squares",
     main="Elbow Method")
grid()

# Or using factoextra package
library(factoextra)
fviz_nbclust(x, kmeans, method="wss",
             nstart=25)
```

Run k-Means for each k and plot WCSS to find elbow

Silhouette Coefficient: Concept

Idea:
Measure how well each point fits in its cluster

For Point i:
$a(i)$ = average distance to points in same cluster (cohesion)
$b(i)$ = average distance to points in nearest other cluster (separation)

Silhouette Coefficient:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Interpretation:

- $s(i) \approx 1$: Well clustered
- $s(i) \approx 0$: On border
- $s(i) \approx -1$: Misclassified

Average Silhouette:
For clustering with k clusters:

$$\bar{s}(k) = \frac{1}{n} \sum_{i=1}^{n} s(i)$$

Higher $\bar{s}(k)$ indicates better clustering

Choosing k:
Select k that maximizes $\bar{s}(k)$

Silhouette Analysis in R

```r
library(cluster)
library(factoextra)

# Compute silhouette for k=3
km <- kmeans(x, centers=3, nstart=25)
sil <- silhouette(km$cluster, dist(x))

# Average silhouette width
mean(sil[,3])

# Plot silhouette
plot(sil, col=1:3, border=NA,
     main="Silhouette Plot (k=3)")

# Or with factoextra
fviz_silhouette(sil)
```

```r
# Compare different k values
k_values <- 2:8
avg_sil <- sapply(k_values, function(k) {
  km <- kmeans(x, centers=k, nstart=25)
  sil <- silhouette(km$cluster, dist(x))
  mean(sil[,3])
})

# Plot average silhouette vs k
plot(k_values, avg_sil, type="b",
     xlab="Number of Clusters",
     ylab="Average Silhouette Width",
     main="Silhouette Method")
abline(v=k_values[which.max(avg_sil)],
       lty=2, col="red")
```

silhouette() function in cluster package computes silhouette coefficients

Gap Statistic Method

Idea:
Compare clustering structure to random uniform data

Method:

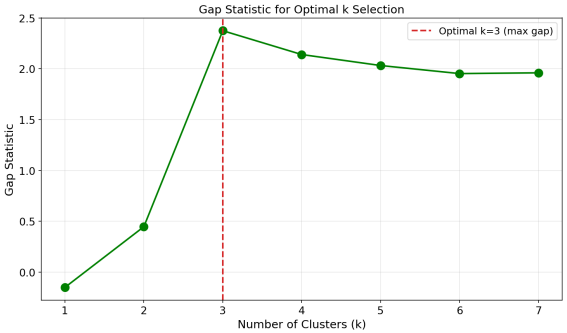
1. Cluster observed data: compute $\log(WCSS_k)$
2. Generate B random uniform datasets
3. Cluster each: compute $\log(WCSS_k^*)$
4. Calculate gap:

$$Gap(k) = \frac{1}{B}\sum_{b=1}^{B} \log(WCSS_k^{*b}) - \log(WCSS_k)$$

5. Choose k that maximizes $Gap(k)$

Intuition:

- Random data has no structure
- If data has structure, gap should be large
- Larger gap = better clustering



Gap statistic with standard error bands. Peak indicates optimal k.

Gap statistic provides statistical justification for chosen k

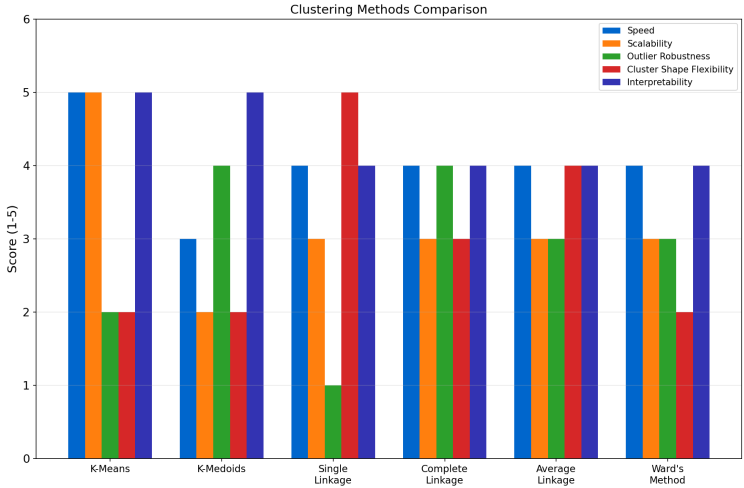
Gap Statistic in R

```r
library(cluster)

# Compute gap statistic
set.seed(123)
gap_stat <- clusGap(x,
                    FUN = kmeans,
                    nstart = 25,
                    K.max = 10,
                    B = 50)

# View results
print(gap_stat, method="firstmax")

# Optimal k
k_optimal <- maxSE(gap_stat$Tab[,"gap"],
                   gap_stat$Tab[,"SE.sim"],
                   method="firstmax")
```

```r
# Plot gap statistic
plot(gap_stat, main="Gap Statistic")

# Or with factoextra
library(factoextra)
fviz_gap_stat(gap_stat)

# Access gap values
gap_values <- gap_stat$Tab[,"gap"]
se_values <- gap_stat$Tab[,"SE.sim"]

cat("Optimal k:", k_optimal, "\n")
```

Gap statistic is computationally intensive due to bootstrap resampling

Comparing k-Selection Methods



Elbow Method: Suggests $k = 3$

Silhouette: Maximum at $k = 3$

Gap Statistic: Peak at $k = 3$

Consensus: $k = 3$ **is optimal for this dataset**

Visual Validation Techniques

1. Cluster Scatter Plots:

- Plot data with cluster colors
- Check for separation
- Look for overlap

2. PCA Projection:

- Project high-D data to 2D
- Visualize clusters
- Check principal components

3. Pair Plots:

- Matrix of scatter plots
- All feature pairs
- Color by cluster

4. Cluster Profiles:

- Box plots by cluster
- Understand cluster characteristics
- Compare feature distributions

5. Heatmaps:

- Show cluster centroids
- Feature values as colors
- Easy to compare clusters

Remember:

- Visual validation is essential
- Numbers don't tell full story
- Interpretability matters

Domain Knowledge: The Most Important Factor

Why Domain Knowledge Matters:

- Statistical methods are tools, not truth
- Clusters must be interpretable
- Context determines utility
- Actionability is key

Questions to Ask:

- Do clusters make sense?
- Can we interpret them?
- Are they actionable?
- Do they align with existing knowledge?
- Will stakeholders understand?

Example: Customer Segmentation
Statistical methods suggest $k = 7$
Domain expert insight:

- $k = 7$ too many segments to manage
- Marketing can handle 3-4 segments
- Some statistical clusters very similar
- Business case for $k = 4$

Decision: Choose $k = 4$

Balance statistical evidence with practical constraints

Let statistics guide you, but let domain knowledge decide

Complete Example: Determining Optimal k

Workflow:

1. Standardize features
2. Apply elbow method: suggests $k = 3$ or $k = 4$
3. Silhouette analysis: maximum at $k = 3$
4. Gap statistic: peak at $k = 3$
5. Visual inspection: $k = 3$ shows clear separation
6. Domain knowledge: 3 segments align with customer types
7. **Decision:** $k = 3$

Final Validation:

- Cluster sizes: 35%, 40%, 25% (reasonable)
- Silhouette width: 0.52 (good)
- Visual separation: clear
- Interpretability: high
- Business value: clear action items

Report:

- Document all methods used
- Show convergence of evidence
- Explain final choice
- Profile clusters
- Provide recommendations

Best Practices Checklist

Before Clustering:

- ☐ Explore data visually
- ☐ Handle missing values
- ☐ Remove or note outliers
- ☐ Standardize features
- ☐ Consider dimensionality reduction
- ☐ Choose appropriate distance metric

During Clustering:

- ☐ Try multiple algorithms
- ☐ Use multiple initializations
- ☐ Test range of k values
- ☐ Record all parameters

After Clustering:

- ☐ Apply multiple validation methods
- ☐ Visualize results
- ☐ Profile each cluster
- ☐ Check stability
- ☐ Interpret with domain knowledge
- ☐ Document decisions
- ☐ Validate on new data if possible

Reporting:

- ☐ Describe preprocessing steps
- ☐ Explain method choice
- ☐ Show validation results
- ☐ Provide cluster profiles
- ☐ Give actionable insights

Method Comparison Matrix

Aspect	k-Means	k-Medoids	Hierarchical	DBSCAN*
Specify k	Yes	Yes	No	No
Scalability	High (10^6)	Medium (10^4)	Low (10^3)	Medium
Cluster shape	Spherical	Flexible	Depends on linkage	Arbitrary
Outlier robust	No	Yes	Depends	Yes
Deterministic	No	No	Yes	Yes
Distance metric	Euclidean	Any	Any	Any
Complexity	$O(nkdi)$	$O(k(n-k)^2 i)$	$O(n^2 \log n)$	$O(n \log n)$
Interpretability	High	High	High	Medium

*DBSCAN is a density-based method (preview only - not covered in detail)

Recommendation: Start with k-Means, try hierarchical for small data, use PAM if outliers present

When to Use Which Method

Use k-Means when:

- Large dataset ($n > 10{,}000$)
- Know approximate k
- Expect spherical clusters
- No significant outliers
- Speed is important

Use k-Medoids (PAM) when:

- Moderate dataset ($n < 10{,}000$)
- Outliers present
- Need interpretable centers
- Non-Euclidean distance needed
- Can afford computation time

Use Hierarchical when:

- Small dataset ($n < 5{,}000$)
- Don't know k
- Want complete hierarchy
- Need dendrogram
- Exploratory analysis

Consider Alternatives when:

- Arbitrary cluster shapes (DBSCAN)
- Overlapping clusters (Gaussian Mixtures)
- Soft assignments (Fuzzy C-Means)
- Very high dimensions (Spectral Clustering)

Beyond Basic Clustering

Advanced Topics (Preview):

1. **DBSCAN**
   - Density-based
   - No need to specify k
   - Finds arbitrary shapes
   - Robust to outliers
2. **Gaussian Mixture Models**
   - Model-based approach
   - Soft cluster assignments
   - Probabilistic framework
   - EM algorithm
3. **Spectral Clustering**
   - Uses eigenvalues of similarity matrix
   - Can find non-convex clusters
   - Good for image segmentation
4. **Fuzzy C-Means**
   - Points can belong to multiple clusters
   - Membership degrees [0,1]
   - Useful for overlapping groups

These methods address limitations of k-Means and hierarchical clustering

Many clustering methods exist; choose based on data characteristics and goals

Section 4 Summary

Key Concepts Covered:

- k-selection challenge
- Elbow method
- Silhouette analysis
- Gap statistic
- Cluster validation metrics
- Visual validation
- Domain knowledge importance
- Common pitfalls
- Best practices
- Method comparison

Important Takeaways:

- Use multiple validation methods
- No single "correct" k
- Balance statistics with domain knowledge
- Always visualize results
- Standardize your data
- Try multiple algorithms
- Document your process
- Validate, validate, validate

Complete Algorithm Comparison

	k-Means	k-Medoids	Single Linkage	Ward's
Centers	Means	Medoids	N/A	N/A
k required	Yes	Yes	No	No
Outliers	Sensitive	Robust	Sensitive	Moderate
Shape	Spherical	Flexible	Elongated	Spherical
Speed	Fast	Slow	Medium	Medium
Best for	Large data, speed	Outliers, interpret	Chains	Compact

General Recommendation:

- Start with k-Means (default choice)
- Try hierarchical for exploration (especially if $n < 5{,}000$)
- Use PAM if outliers are a concern
- Always validate results multiple ways

Course Summary: What We Learned

Hour 1: k-Means

- Unsupervised learning
- k-Means algorithm
- WCSS objective
- Initialization problem
- Advantages and limitations

Hour 2: k-Medoids and Distances

- PAM algorithm
- Robustness to outliers
- Distance metrics
- Euclidean, Manhattan, Minkowski
- Standardization necessity

Hour 3: Hierarchical

- Agglomerative clustering
- Dendrograms
- Linkage methods
- Single, Complete, Average, Ward's
- Method comparison

Hour 4: Optimal k

- Elbow method
- Silhouette analysis
- Gap statistic
- Validation metrics
- Best practices

Further Resources

Books:

- Hastie, Tibshirani, Friedman: *Elements of Statistical Learning*
- James et al.: *Introduction to Statistical Learning*
- Kaufman, Rousseeuw: *Finding Groups in Data*

R Packages:

- `cluster`: PAM, silhouette, gap statistic
- `factoextra`: Visualization
- `fpc`: Cluster validation
- `dendextend`: Dendrogram manipulation
- `NbClust`: 30 clustering indices

Python Packages:

- `sklearn.cluster`: All major algorithms
- `scipy.cluster`: Hierarchical clustering
- `yellowbrick`: Visualization

Online Resources:

- scikit-learn documentation
- Stack Overflow
- Cross Validated (stats.stackexchange.com)
- R-bloggers

Practice Datasets:

- UCI Machine Learning Repository
- Kaggle datasets
- R built-in datasets (iris, mtcars, USArrests)

Next Steps for Practice

Immediate Practice:

1. Analyze Iris dataset
   - Apply k-Means, PAM, hierarchical
   - Compare results
   - Determine optimal k
2. Work with USArrests data
   - Don't forget to standardize
   - Try different linkage methods
   - Interpret clusters
3. Generate synthetic data
   - Create non-spherical clusters
   - See where k-Means fails
   - Compare with other methods

Project Ideas:

- Customer segmentation
- Image compression
- Document clustering
- Gene expression analysis
- Anomaly detection

Advanced Topics to Explore:

- DBSCAN and density-based methods
- Gaussian Mixture Models
- Spectral clustering
- Subspace clustering
- Ensemble clustering
- Deep learning for clustering