WSABI Summer Research Lab **Summer 2025**

Lecture 19: Clustering

Instructor: Jonathan Pipping *Author: JP*

19.1 NBA Player Archetypes

19.1.1 Problem Setup

Question: Can we group NBA players into offensive roles or archetypes based on how they play?

Data: We have NBA synergy data from 2012-2022, which contains information about the context in which players take shots. Each row contains the following information:

- i: index of the i^{th} player in our dataset
- x_i: feature vector containing information about the proportion of times a player shoots from a particular play type (e.g. post-up, catch and shoot, etc.)

Grouping: How do we use this data to group players into offensive roles? To do this, we introduce the field of **unsupervised learning**.

19.2 Unsupervised Learning

19.2.1 The Supervised vs. Unsupervised Paradigm

In supervised learning, we have a response variable y and aim to learn a function f that maps inputs $\boldsymbol{X}$ to outputs y. The ultimate goal is to predict y for new observations, and we can express this as a function approximation problem:

$$y = f(\boldsymbol{X})$$

Examples of supervised learning include regression (predicting continuous values) and classification (predicting categorical labels). In contrast, unsupervised learning operates **without** a response variable y. With only $\boldsymbol{X}$ available, our objective is to understand the hidden structure of the data.

19.2.2 Types of Unsupervised Learning

Unsupervised learning encompasses several key tasks, each with distinct objectives:

- **Clustering**: Group similar data points together based on their features. The goal is to discover natural groupings in the data without prior knowledge of group labels.
- **Dimensionality Reduction**: Reduce the number of features while preserving important information. This helps with visualization, computational efficiency, and removing noise.

- **Association Rule Learning**: Discover relationships between variables in large datasets, often used in market basket analysis to find items that frequently co-occur.

- **Anomaly Detection**: Identify unusual data points that differ significantly from the majority of the data, useful for fraud detection and quality control.

- **Feature Learning**: Automatically discover useful representations of the data, such as learning meaningful features from raw input data.

Each of these tasks serves different purposes in data analysis and can be applied to various domains depending on the specific goals of the analysis.

19.2.3 The Challenge of Evaluation

A fundamental challenge in unsupervised learning is **evaluation**: without a response variable, we lack a natural measure of success like out-of-sample error. What strategies can we use to evaluate the quality of our unsupervised learning methods?

- **Internal validation**: Measure cluster quality using properties of the data itself, such as within-cluster variation, between-cluster separation, or silhouette scores. These metrics don't require external labels but may not always align with human intuition about what constitutes "good" clusters.

- **External validation**: Compare clustering results against known ground truth labels when available. This includes metrics like adjusted Rand index, normalized mutual information, or purity scores. While ideal, ground truth is rarely available in truly unsupervised scenarios.

- **Stability analysis**: Assess how robust the clustering is to small changes in the data or algorithm parameters. Stable clusters that persist across different subsets or parameter settings are generally more trustworthy.

- **Domain expertise**: Evaluate clusters based on their interpretability and alignment with domain knowledge. Clusters that make sense to subject matter experts are often more valuable than those that optimize mathematical criteria alone.

- **Downstream task performance**: Test how well the clustering improves performance on related supervised tasks, such as using cluster assignments as features in a classification problem.

The choice of evaluation strategy depends on the specific application and available information. In practice, a combination of these approaches often provides the most comprehensive assessment of clustering quality.

19.2.4 Clustering as Unsupervised Learning

Clustering is the task of grouping similar objects together while keeping dissimilar objects separate. Formally, we seek to partition $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ into k non-empty, disjoint subsets $C_1, C_2, \ldots, C_k$ such that points within the same cluster are *similar* and points in different clusters are *dissimilar*. In short, clustering aims to discover grouping structure in the data without any prior knowledge of what that structure should be. We'll formalize this in the next section.

19.3 The Clustering Problem

19.3.1 Formal Definition

Definition 19.1 (Clustering Problem)**.** *Given a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ where each $\boldsymbol{x}_i \in \mathbb{R}^p$, a clustering algorithm aims to partition $\mathcal{D}$ into k non-empty, disjoint subsets $C_1, C_2, \ldots, C_k$ such that:*

- *$\bigcup_{j=1}^{k} C_j = \mathcal{D}$ (every observation is assigned to exactly one cluster)*
- *$C_i \cap C_j = \emptyset$ for $i \neq j$ (clusters are disjoint)*
- *Observations within the same cluster are similar with respect to some distance metric*
- *Observations in different clusters are dissimilar with respect to some distance metric*

19.3.2 Similarity and Distance Metrics

To quantify the similarity between observations, we formally define a **distance metric**.

Definition 19.2 (Distance Metric)**.** *A function $d : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}_{\geq 0}$ is a distance metric if it satisfies:*

- *$d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ (non-negativity)*
- *$d(\boldsymbol{x}, \boldsymbol{y}) = 0$ if and only if $\boldsymbol{x} = \boldsymbol{y}$ (identity of indiscernibles)*
- *$d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ (symmetry)*
- *$d(\boldsymbol{x}, \boldsymbol{z}) \leq d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z})$ (triangle inequality)*

Common distance metrics include:

Definition 19.3 (Euclidean Distance)**.** *The Euclidean distance between two points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$ is:*

$$d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2 = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

Definition 19.4 (Manhattan Distance)**.** *The Manhattan distance between two points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$ is:*

$$d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_1 = \sum_{i=1}^{p} |x_i - y_i|$$

Definition 19.5 (Angular Distance)**.** *The angular distance between two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$ is:*

$$d(\boldsymbol{x}, \boldsymbol{y}) = \arccos\left(\frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2}\right)$$

19.4 K-means Clustering

19.4.1 The K-means Objective

K-means is the most widely used clustering algorithm. It aims to minimize the within-cluster sum of squares.

Definition 19.6 (K-means Objective)**.** *Given a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ and a desired number of clusters k, the k-means objective is:*

$$\min_{C_1,\ldots,C_k} \sum_{j=1}^{k} \sum_{\boldsymbol{x}_i \in C_j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2$$

where $\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{\boldsymbol{x}_i \in C_j} \boldsymbol{x}_i$ is the centroid of cluster C_j.

An equivalent formulation directly optimizes the centroids:

$$\min_{\{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k\}} \sum_{i=1}^{n} \min_{j=1,\ldots,k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2$$

19.4.2 The Lloyd–Forgy Algorithm

Theorem 19.7 (Lloyd–Forgy Algorithm Convergence)**.** *The Lloyd–Forgy algorithm for k-means clustering converges to a local optimum in a finite number of iterations.*

Algorithm 1 Lloyd–Forgy Algorithm for K-means

1: Choose k and initialize centroids $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$
2: **repeat**
3: **Assignment step**: $c_i \leftarrow \arg\min_j \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2$ for all i
4: **Update step**: $\boldsymbol{\mu}_j \leftarrow \frac{1}{n_j} \sum_{i:c_i=j} \boldsymbol{x}_i$ for all j
5: **until** assignments no longer change

Sketch of Convergence. The algorithm monotonically decreases the objective function in each iteration:

Assignment step: For each point $\boldsymbol{x}_i$, assigning it to the nearest centroid $\boldsymbol{\mu}_j$ minimizes $\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2$. Since each point is reassigned to minimize its contribution to the objective, the total sum of squares cannot increase.

Update step: For each cluster C_j, the centroid $\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{\boldsymbol{x}_i \in C_j} \boldsymbol{x}_i$ is the minimizer of $\sum_{\boldsymbol{x}_i \in C_j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2$ with respect to $\boldsymbol{\mu}_j$. This follows from the fact that the mean minimizes the sum of squared deviations.

Since the objective function is bounded below by 0 and decreases monotonically in each iteration, the algorithm must converge to a local optimum. The convergence is finite because there are only finitely many possible cluster assignments. □

19.4.3 Geometric Interpretation

The k-means algorithm has a beautiful geometric interpretation through Voronoi diagrams. We define a Voronoi diagram as follows:

Definition 19.8 (Voronoi Diagram)**.** *Given a set of centroids* $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$*, the Voronoi cell for centroid* $\boldsymbol{\mu}_j$ *is the set of all points* $\boldsymbol{x}$ *such that:*

$$\|\boldsymbol{x} - \boldsymbol{\mu}_j\| \leq \|\boldsymbol{x} - \boldsymbol{\mu}_i\| \quad \text{for all } i \neq j$$

Proposition 19.9 (K-means and Voronoi Diagrams)**.** *At convergence, k-means produces a Voronoi tessellation of the space, where each cluster corresponds to a Voronoi cell centered at its centroid.*

This geometric interpretation explains several key properties of k-means:

- **Linear decision boundaries**: The boundary between any two clusters is a hyperplane perpendicular to the line segment connecting their centroids
- **Convex clusters**: Each Voronoi cell is convex, meaning any line segment between two points in the same cluster lies entirely within that cluster
- **Spherical bias**: The algorithm naturally favors spherical clusters of similar size

A visualization of the k-means algorithm is shown in Figure 19.1. The algorithm iteratively updates both the cluster assignments (creating Voronoi cells) and the centroids (moving the cell centers), eventually converging to a stable partition.

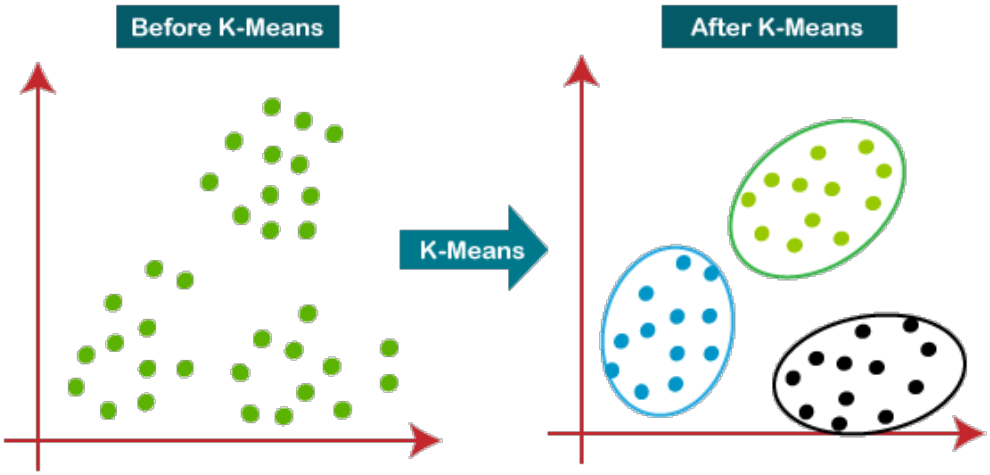

Figure 19.1: K-means clustering on a 2D dataset.

19.4.4 Initialization Strategies

The choice of initial centroids significantly affects the final result, as k-means converges to a local optimum, not necessarily the global optimum of the objective function. Fortunately, K-means++ is a simple initialization strategy that achieves an $O(\log k)$ approximation to the optimal k-means objective in expectation.

Definition 19.10 (K-means++ Initialization)**.** *K-means++ initializes centroids as follows:*

1. *Choose the first centroid uniformly at random from the dataset*

2. *For each subsequent centroid, choose a point with probability proportional to the squared distance to the nearest existing centroid, namely*

$$\frac{\min_j \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2}{\sum_{j=1}^{k} \min_j \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2}$$

This means that as the distance from the nearest centroid increases, the probability of choosing that point as a new centroid increases.

19.4.5 Choosing the Number of Clusters

The number of clusters k is a critical hyperparameter that significantly affects clustering quality. There is no universally correct way to choose k, but one common heuristic is the elbow method.

Definition 19.11 (Elbow Method)**.** *Plot the within-cluster sum of squares $WSS(k)$ against k and choose the value where the rate of decrease slows down, forming an "elbow" in the curve. This point represents a trade-off between reducing within-cluster variation and avoiding overfitting.*

The elbow method is simple and intuitive, but it can be subjective since the "elbow" is not always clearly defined. In practice, it's often used in combination with domain knowledge and other validation techniques. An example of the elbow method is shown in Figure 19.2. From this figure, we can see that the elbow is at $k = 3$ or $k = 4$.

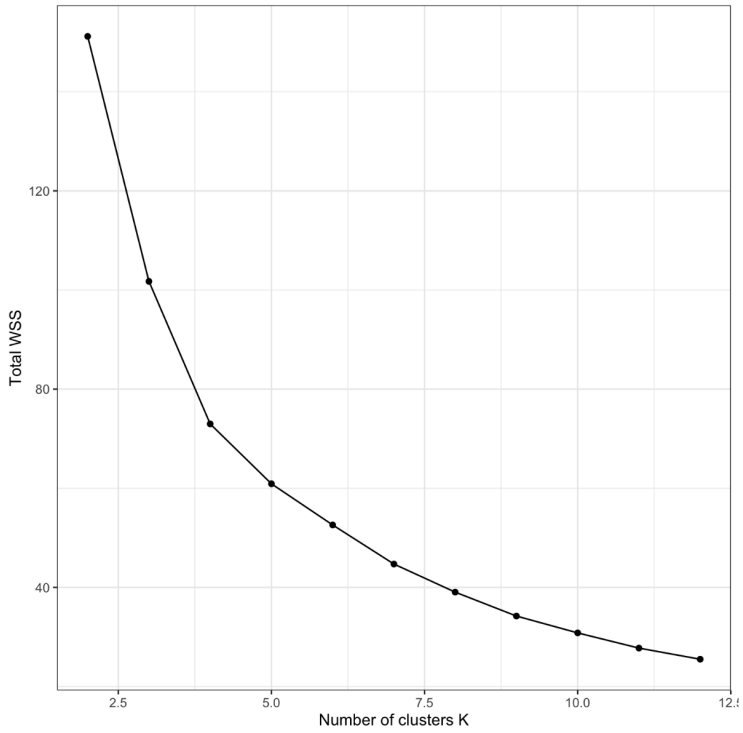

Figure 19.2: Elbow method for choosing the number of clusters. Here, the elbow is at $k = 3$ or $k = 4$.

19.4.6 Properties of K-means

Proposition 19.12 (K-means Properties)**.** *K-means clustering has the following properties:*

- ***Convex clusters****: Each cluster forms a convex region in* $\mathbb{R}^p$
- ***Linear boundaries****: Decision boundaries between clusters are linear*
- ***Hard assignments****: Each point belongs to exactly one cluster*
- ***Spherical clusters****: Assumes clusters are roughly spherical and of similar size*

19.5 Hierarchical Clustering

19.5.1 The Hierarchical Approach

Unlike k-means, hierarchical clustering doesn't require specifying the number of clusters in advance. Instead, it creates a complete hierarchy of clusterings, or a dendrogram. We define a dendrogram below and plot an example in Figure 19.3.

Definition 19.13 (Dendrogram)**.** *A dendrogram is a tree-like diagram that shows the hierarchical relationship between clusters. The height at which two clusters merge represents their dissimilarity.*

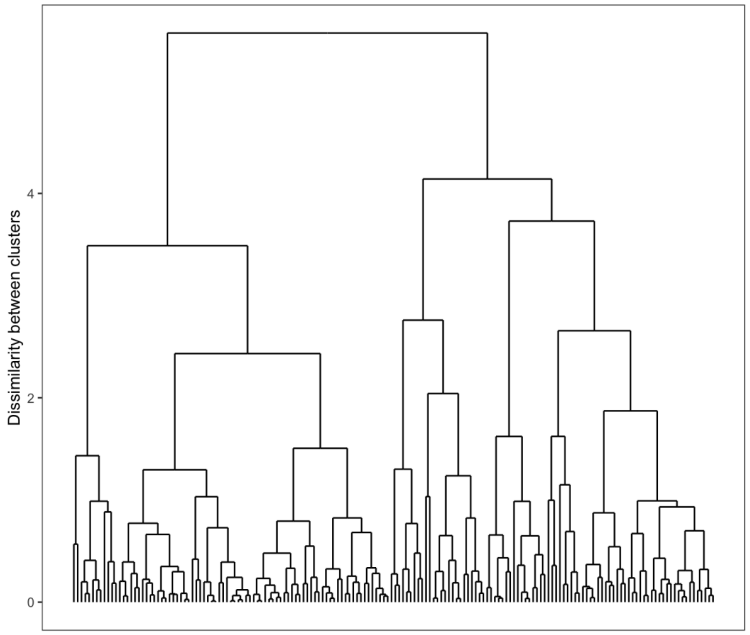

Figure 19.3: An example dendogram, showing the hierarchical relationship between clusters. The height at which two clusters merge represents their dissimilarity.

19.5.2 Agglomerative Clustering

The agglomerative hierarchical clustering algorithm is defined as follows.

Algorithm 2 Agglomerative Hierarchical Clustering

1: Start with each point in its own cluster: $C_i = \{\boldsymbol{x}_i\}$ for $i = 1, \ldots, n$
2: **repeat**
3: Find the pair of clusters (C_i, C_j) with minimum dissimilarity $d(C_i, C_j)$
4: Merge C_i and C_j into a new cluster $C_{new} = C_i \cup C_j$
5: Remove C_i and C_j from the set of clusters
6: **until** all points are in a single cluster

19.5.3 Linkage Methods

The key decision in hierarchical clustering is how to measure dissimilarity between clusters. We define several common linkage methods below, each of which relies on a distance metric $d(\cdot, \cdot)$ like we defined in 19.3.2.

Definition 19.14 (Single Linkage)**.** *The single linkage dissimilarity between clusters A and B is:*

$$d_{SL}(A, B) = \min_{\boldsymbol{x} \in A, \boldsymbol{y} \in B} d(\boldsymbol{x}, \boldsymbol{y})$$

Definition 19.15 (Complete Linkage)**.** *The complete linkage dissimilarity between clusters A and B is:*

$$d_{CL}(A, B) = \max_{\boldsymbol{x} \in A, \boldsymbol{y} \in B} d(\boldsymbol{x}, \boldsymbol{y})$$

Definition 19.16 (Average Linkage)**.** *The average linkage dissimilarity between clusters A and B is:*

$$d_{AL}(A, B) = \frac{1}{|A| \cdot |B|} \sum_{\boldsymbol{x} \in A} \sum_{\boldsymbol{y} \in B} d(\boldsymbol{x}, \boldsymbol{y})$$

Definition 19.17 (Ward's Method)**.** *Ward's method merges the pair of clusters that minimizes the increase in total within-cluster sum of squares:*

$$d_{Ward}(A, B) = \frac{|A| \cdot |B|}{|A| + |B|} \|\boldsymbol{\mu}_A - \boldsymbol{\mu}_B\|^2$$

where $\boldsymbol{\mu}_A$ and $\boldsymbol{\mu}_B$ are the centroids of clusters A and B.

19.5.4 Properties of Linkage Methods

Proposition 19.18 (Linkage Method Properties)**.** *Different linkage methods have distinct characteristics:*

- ***Single linkage*** *tends to produce elongated clusters and is sensitive to noise*
- ***Complete linkage*** *produces compact, spherical clusters and is robust to noise*
- ***Average linkage*** *provides a balance between single and complete linkage*
- ***Ward's method*** *minimizes within-cluster variance, similar to k-means*

19.5.5 Cutting the Dendrogram

After constructing the complete dendrogram, we need to determine where to "cut" it to obtain a final clustering. Cutting a dendrogram involves choosing a horizontal line (dissimilarity threshold) that determines how many clusters we get. Points that are connected below this line belong to the same cluster.

There are two main approaches to determining where to cut a dendrogram:

1. **Specify the number of clusters** k: Choose a desired number of clusters, then cut at the height that produces exactly k clusters.
2. **Look for large vertical jumps**: Identify heights where there are substantial increases in dissimilarity (large vertical gaps in the dendrogram), indicating natural cluster boundaries. Then cut at that height.

Algorithm 3 Cutting a Dendrogram

```
Construct the complete dendrogram using agglomerative clustering
Approach 1: Choose the desired number of clusters k
Approach 2: Identify large vertical jumps in the dendrogram
Find the height h where the dendrogram has exactly k branches (Approach 1) or where there's a large jump (Approach 2)
Cut the dendrogram at height h
Assign each leaf node to the cluster corresponding to its branch
```

19.6 Gaussian Mixture Models

19.6.1 The Probabilistic Approach

Model-based clustering assumes the data is generated from a mixture of probability distributions. We begin by defining a finite mixture model.

Definition 19.19 (Finite Mixture Model)**.** *A finite mixture model assumes the data follows a mixture of K probability distributions:*

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x})$$

where $\pi_k \geq 0$ are mixing proportions with $\sum_{k=1}^{K} \pi_k = 1$, and f_k are component densities.

19.6.2 Gaussian Mixture Models

Definition 19.20 (Gaussian Mixture Model)**.** *A Gaussian Mixture Model (GMM) assumes that each component distribution is a multivariate normal:*

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the multivariate normal density with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$.

The parameters of a Gaussian Mixture Model are the following:

- π_k: mixing proportions (probabilities that sum to 1)
- $\boldsymbol{\mu}_k$: mean vector of the k-th component
- $\boldsymbol{\Sigma}_k$: covariance matrix of the k-th component

19.6.3 Parameter Estimation via the EM Algorithm

We estimate the parameters of a GMM using the Expectation-Maximization (EM) algorithm, which is a general algorithm for maximum likelihood estimation in the presence of latent variables.

Theorem 19.21 (EM Algorithm for GMM)**.** *The EM algorithm for a GMM alternates between the following two steps:*

1. ***E-step**: Compute posterior probabilities $z_{ik} = P(\text{cluster} = k \mid \boldsymbol{x}_i)$*
2. ***M-step**: Update parameters to maximize expected complete log-likelihood*

A step-by-step description of the EM algorithm for a GMM is shown in Algorithm 4.

Algorithm 4 EM Algorithm for GMM

Initialize $\{\pi_k^{(0)}, \boldsymbol{\mu}_k^{(0)}, \boldsymbol{\Sigma}_k^{(0)}\}$
repeat
  E-step: $z_{ik}^{(t)} = \frac{\pi_k^{(t)} \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}{\sum_\ell \pi_\ell^{(t)} \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_\ell^{(t)}, \boldsymbol{\Sigma}_\ell^{(t)})}$
  M-step:

$$\pi_k^{(t+1)} = \frac{1}{n} \sum_i z_{ik}^{(t)}$$

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_i z_{ik}^{(t)} \boldsymbol{x}_i}{\sum_i z_{ik}^{(t)}}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_i z_{ik}^{(t)} (\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(t+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(t+1)})^T}{\sum_i z_{ik}^{(t)}}$$

until log-likelihood converges

19.6.4 Soft vs. Hard Assignment

A key difference between GMM and other clustering methods is the type of cluster assignments they produce.

Definition 19.22 (Soft Assignment)**.** *In GMM, each point $\boldsymbol{x}_i$ has a probability z_{ik} of belonging to cluster k, where $\sum_{k=1}^K z_{ik} = 1$. This means each point has some probability of belonging to each cluster.*

Definition 19.23 (Hard Assignment)**.** *A hard assignment assigns each point to exactly one cluster. For GMM, this is done by choosing the cluster with the highest probability:*

$$c_i = \arg\max_k z_{ik}$$

K-means and hierarchical clustering naturally produce hard assignments, where each point belongs to exactly one cluster. GMM produces soft assignments by default, but can be converted to hard assignments using the rule above. Soft assignments are useful when points might legitimately belong to multiple clusters or when we want to quantify uncertainty in cluster membership.

19.6.5 Model Selection

Choosing the number of components K is crucial for GMM. Unlike k-means, GMM provides principled ways to select K using information criteria that balance model fit with complexity. We define these below.

Definition 19.24 (Bayesian Information Criterion)**.** *The BIC for a GMM with K components is:*

$$BIC(K) = -2 \log L_K + p_K \log n$$

where L_K is the maximized likelihood and p_K is the number of parameters. Lower BIC values indicate better models in terms of the trade-off between fit and complexity.

Definition 19.25 (Akaike Information Criterion)**.** *The AIC for a GMM with K components is:*

$$AIC(K) = -2 \log L_K + 2p_K$$

where L_K is the maximized likelihood and p_K is the number of parameters. Lower AIC values indicate better models in terms of the trade-off between fit and complexity.

Both BIC and AIC penalize model complexity to avoid overfitting. The first term ($-2 \log L_K$) measures how well the model fits the data, while the second term penalizes the number of parameters. BIC penalizes more heavily for large sample sizes, while AIC is less conservative. In practice, we fit GMMs with different values of K and choose the one that minimizes the chosen criterion.

For a GMM with K components in p dimensions, the number of parameters is:

$$p_K = K - 1 + Kp + K\frac{p(p+1)}{2}$$

where the terms represent mixing proportions, means, and covariance matrices respectively.

19.7 Evaluation and Model Selection

19.7.1 Internal Validation Metrics

Since clustering lacks a response variable, we often rely on internal validation metrics to evaluate the quality of a clustering. We define several common internal validation metrics below.

Definition 19.26 (Silhouette Coefficient)**.** *The silhouette coefficient for point i is:*

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

where $a(i)$ is the average distance from point i to other points in the same cluster, and $b(i)$ is the minimum average distance from point i to points in other clusters.

The silhouette coefficient ranges from -1 to 1. Values close to 1 indicate points are well-matched to their clusters, values close to 0 suggest points are on the border between clusters, and values close to -1 indicate points may be assigned to the wrong cluster.

Definition 19.27 (Calinski-Harabasz Index)**.** *The Calinski-Harabasz index is:*

$$CH = \frac{tr(B_k)}{tr(W_k)} \cdot \frac{n-k}{k-1}$$

where B_k is the between-cluster scatter matrix and W_k is the within-cluster scatter matrix.

The Calinski-Harabasz index measures the ratio of between-cluster dispersion to within-cluster dispersion. Higher values indicate better-defined clusters. This metric is particularly useful for comparing different numbers of clusters.

19.7.2 External Validation

When ground truth is available, external metrics can be used to evaluate the quality of a clustering. We define several common external validation metrics below.

Definition 19.28 (Adjusted Rand Index)**.** *The Adjusted Rand Index (ARI) measures similarity between two clusterings:*

$$ARI = \frac{RI - \textit{Expected } RI}{\max(RI) - \textit{Expected } RI}$$

where RI is the Rand Index.

The ARI ranges from -1 to 1, where 1 indicates perfect agreement between clusterings, 0 indicates random agreement, and negative values indicate worse-than-random agreement. ARI is particularly useful because it accounts for chance agreement.

Definition 19.29 (Normalized Mutual Information)**.** *The Normalized Mutual Information (NMI) is:*

$$NMI = \frac{2 \cdot I(C, C')}{H(C) + H(C')}$$

where $I(C, C')$ is the mutual information between clusterings C and C', and $H(C)$ is the entropy of clustering C.

The NMI ranges from 0 to 1, where 1 indicates perfect agreement and 0 indicates no mutual information between the clusterings. NMI is symmetric and invariant to cluster label permutations.

19.8 Back to NBA Player Clustering

19.8.1 Applying K-means

First, we use `flexclust::kcca` in `R` to perform k-means clustering. We use an elbow plot to determine the number of clusters, which we see in Figure 19.4. From this figure, we see that the elbow is at $k = 3$, so we begin by fitting 3 clusters. The code for this is shown below.

```
# load flexclust
library(flexclust)
# fit model
k_means = flexclust::kcca(training_data, k = 3)
```

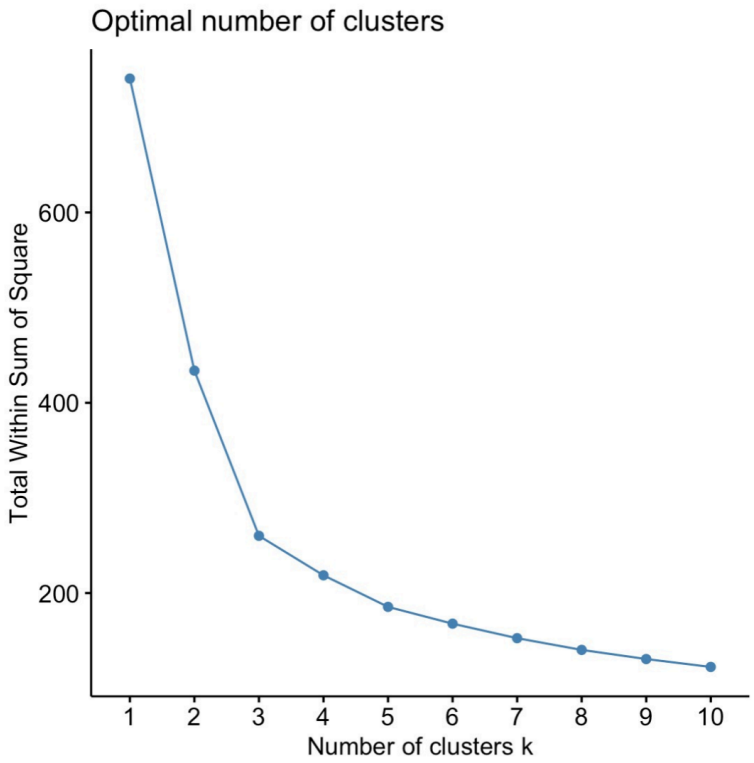

Figure 19.4: Elbow plot for NBA player clustering. The elbow is at $k = 3$.

19.8.2 Interpreting the Clusters

Once we fit the model, we plot the results in Figure 19.5 and observe the following clusters:

- **Cluster 1**: Ball-handlers, who take the majority of their shots off the dribble or spotting up off the ball

- **Cluster 2**: Bigs, who take most of their shots as the roller in a pick-and-roll, cutting to the basket, or off an offensive rebound

- **Cluster 3**: Wing players, who spot up on almost all of their shots

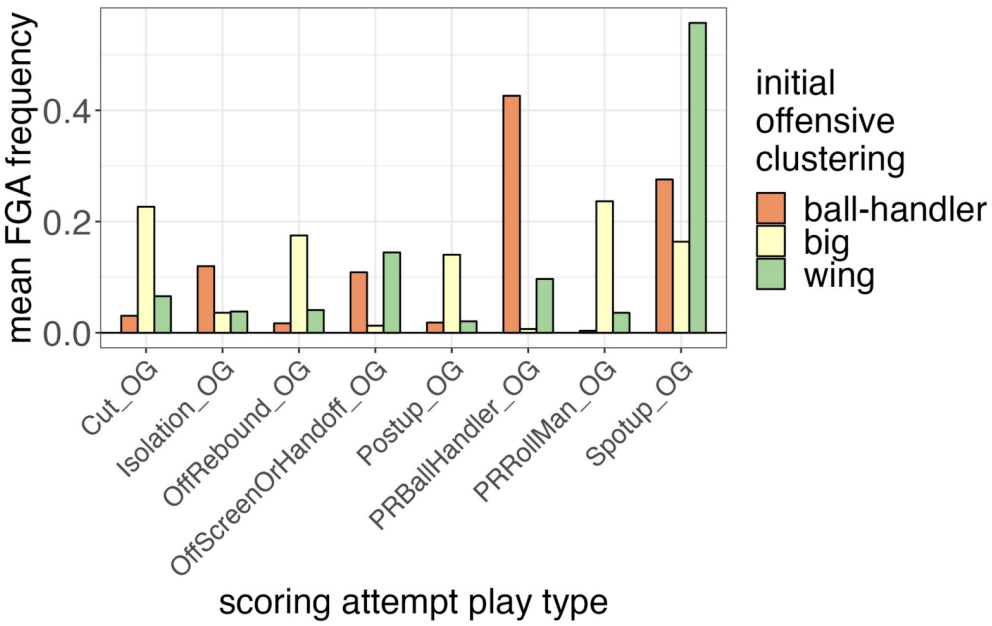

Figure 19.5: K-means clustering of NBA players. It's clear that our three clusters can be described using traditional offensive roles: ball-handlers, bigs, and wings.

To validate our clustering, we examine the mix of shooting play types for players from each cluster in Figure 19.6. These assignments are not perfect, but they are consistent with our intuition from the previous figure and align well with traditional basketball roles.

19.9 Summary and Practical Considerations

19.9.1 Algorithm Comparison

Method	Cluster Shape	Number of Clusters	Assignment Type	Computational Cost
K-means	Spherical	Pre-specified	Hard	Low
Hierarchical	Flexible	Any	Hard	Medium
GMM	Elliptical	Chosen by model selection	Soft	Medium

Table 19.1: Comparison of three main clustering methods

19.9.2 Key Takeaways

- Different clustering paradigms make different assumptions about data structure
- No single method is universally best: choose the one that best fits the data and our goals
- Domain knowledge is crucial for feature engineering and result interpretation
- Always visualize results, and consider validating using multiple metrics

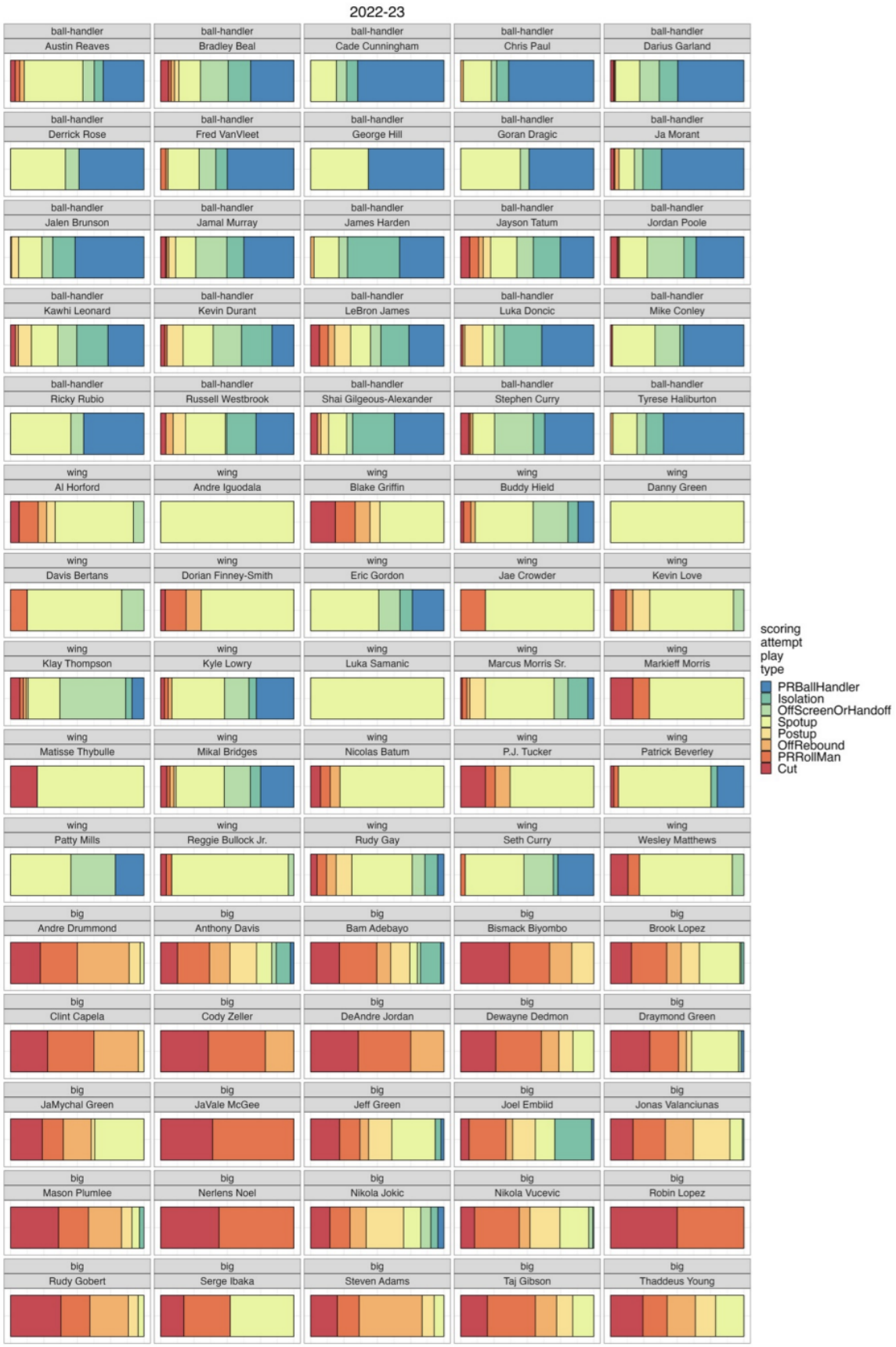

Figure 19.6: Mix of shooting play types for players from each cluster.