K-Means

STAT-S 610: Introduction to Statistical Computing (Fall 2024)

Instructor
Dr. Julia Fukuyama
Team Members
Gandhar Ravindra Pansare
Swarn Gaba
Vedika Halwasiya

Introduction

Project Overview

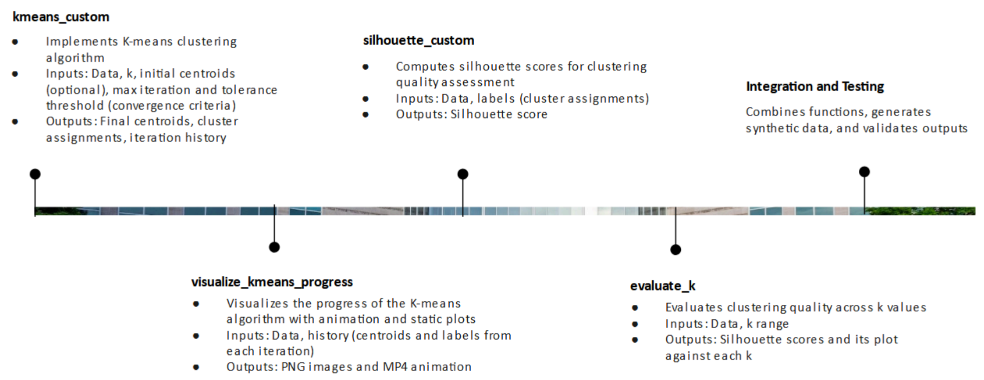
K-means clustering is one of the most popular unsupervised machine learning algorithms used for grouping data into clusters based on similarity. The goal of this project was to design a robust and modular implementation of the K-means algorithm from scratch without relying on existing libraries. Additionally, the project involved visualizing the iterative clustering progress and evaluating clustering quality to determine the optimal number of clusters (k).

Objectives

- **Implement the K-means algorithm**: Develop a modular, custom implementation that is easy to understand, debug, and reproduce
- **Visualize algorithm progress**: Create scatter plots and animations showing the movement of centroids and changes in cluster assignments over iterations
- **Evaluate clustering quality**: Use silhouette scores to assess cluster quality and guide users in choosing the optimal k

To ensure that the code is functioning correctly, we will validate it through a combination of theoretical checks, empirical testing, and visual inspection.

Design and Implementation


kmeans_custom
• Implements K-means clustering algorithm
• Inputs: Data, k, initial centroids (optional), max iteration and tolerance threshold (convergence criteria)
• Outputs: Final centroids, cluster assignments, iteration history
silhouette_custom
• Computes silhouette scores for clustering quality assessment
• Inputs: Data, labels (cluster assignments)
• Outputs: Silhouette score
Integration and Testing
Combines functions, generates synthetic data, and validates outputs
visualize_kmeans_progress
• Visualizes the progress of the K-means algorithm with animation and static plots
• Inputs: Data, history (centroids and labels from each iteration)
• Outputs: PNG images and MP4 animation
evaluate_k
• Evaluates clustering quality across k values
• Inputs: Data, k range
• Outputs: Silhouette scores and its plot against each k

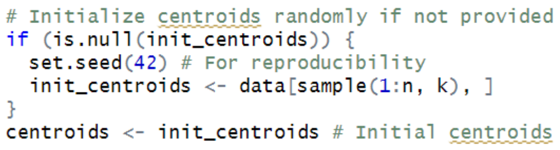
Core Function

The `kmeans_custom` function is the primary implementation of the K-means algorithm, designed to cluster data points into k groups based on their similarity. It is built to ensure reproducibility, computational efficiency, and adherence to the algorithm's theoretical framework. The function executes the key steps of K-means: centroid initialization, assignment of points to clusters, centroid updates, and convergence checks.

Centroid Initialization

Centroids are initialized either randomly from the dataset or through user-defined values. If no centroids are provided, the function randomly selects k points from the dataset as initial centroids. A fixed seed (`set.seed(42)`) ensures reproducibility, making the function results consistent across runs. This step is crucial because:

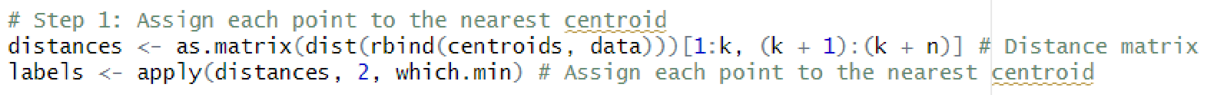
- Proper initialization avoids empty clusters or poorly distributed starting points, which could impact the clustering results.
- Reproducibility is essential for debugging, testing, and ensuring deterministic behavior.



```
# Initialize centroids randomly if not provided
if (is.null(init_centroids)) {
  set.seed(42) # For reproducibility
  init_centroids <- data[sample(1:n, k), ]
}
centroids <- init_centroids # Initial centroids
```

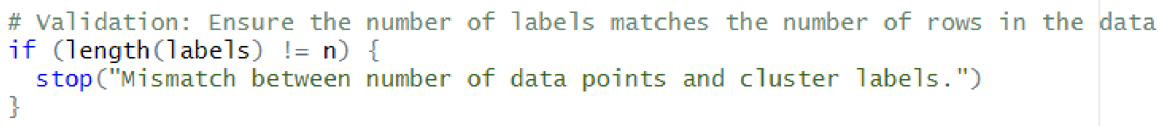
Assignment Step

Each data point is assigned to the nearest centroid based on **Euclidean distance**. This distance metric is widely used for clustering because it is intuitive and effectively captures similarity in multi-dimensional space. To calculate distances efficiently, a distance matrix is constructed where each row represents a centroid and each column represents a data point. The nearest centroid for each point is then identified using the `which.min` function.



```
# Step 1: Assign each point to the nearest centroid
distances <- as.matrix(dist(rbind(centroids, data)))[1:k, (k + 1):(k + n)] # Distance matrix
labels <- apply(distances, 2, which.min) # Assign each point to the nearest centroid
```

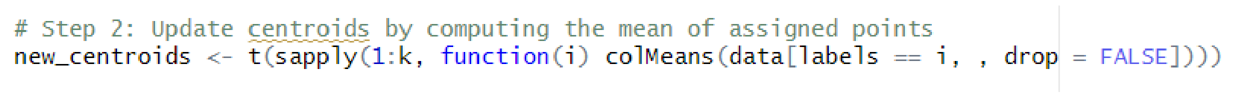
To ensure correctness, the function includes a validation step, verifying that the number of cluster labels matches the number of data points. This prevents logical errors in subsequent steps.



```
# Validation: Ensure the number of labels matches the number of rows in the data
if (length(labels) != n) {
  stop("Mismatch between number of data points and cluster labels.")
}
```

Centroid Update

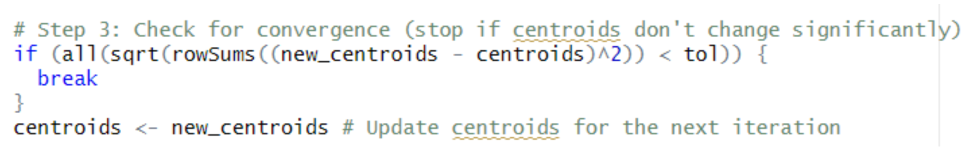
After assigning points to clusters, the centroids are updated as the mean of all points assigned to each cluster. This approach minimizes the variance within clusters, aligning with the primary goal of K-means clustering. The function handles cases where no points are assigned to a cluster by leaving the centroid unchanged. This ensures stability and prevents computational errors.



```r
# Step 2: Update centroids by computing the mean of assigned points
new_centroids <- t(sapply(1:k, function(i) colMeans(data[labels == i, , drop = FALSE])))
```

Convergence Check

To determine when to stop iterating, the function uses a tolerance threshold (`tol`), which specifies the minimum allowable movement for centroids between iterations. If the Euclidean distance between old and new centroids is smaller than the threshold for all centroids, the algorithm halts. This ensures that the clustering process does not continue unnecessarily once the centroids stabilize.



```r
# Step 3: Check for convergence (stop if centroids don't change significantly)
if (all(sqrt(rowSums((new_centroids - centroids)^2)) < tol)) {
  break
}
centroids <- new_centroids # Update centroids for the next iteration
```

This design choice balances computational efficiency and clustering accuracy, avoiding redundant iterations while ensuring convergence.

Visualization

The `visualize_kmeans_progress` function provides an intuitive representation of the K-means clustering process by generating both static and animated visualizations. These visualizations highlight how centroids and cluster assignments evolve over iterations, enhancing the interpretability of the algorithm.

Static Visualization

At each iteration, the function generates scatter plots displaying: data points, color-coded by cluster and centroids, visually distinct to highlight their movements. These plots provide a snapshot of the clustering state at every iteration.

Animated Visualization

The function combines static plots into an animation, illustrating: centroid movements across iterations and dynamic changes in cluster assignments. This allows users to observe the clustering process in real time, offering a comprehensive view of how clusters stabilize.

Evaluation Metric

Silhouette scores provide a quantitative measure of clustering quality:

- **Intra-cluster distance (a)**: How close points are within the same cluster
- **Inter-cluster distance (b)**: How far a point is from the nearest cluster
- The silhouette score for a point is:

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

Evidence of Functionality

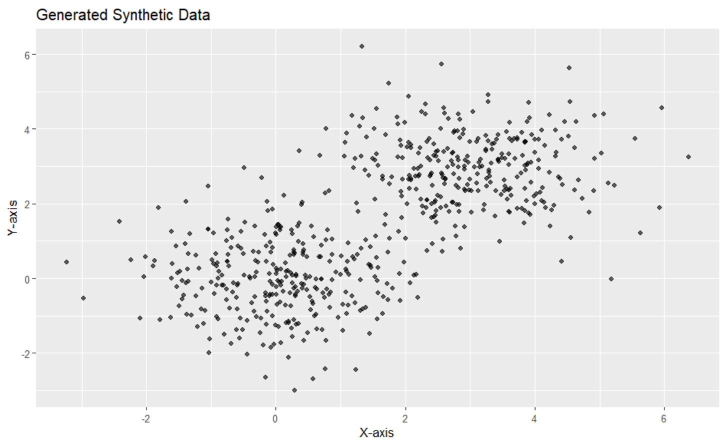
The functionality of the K-means implementation was tested extensively to ensure correctness, interpretability, and usability. This section outlines the process of validating the code with synthetic data, visualizing the clustering progression, and evaluating cluster quality across different values of k. The results demonstrate that the `kmeans_custom` function and its associated visualizations and evaluations work as intended, meeting the objectives of this project.

Synthetic Data

To test the implementation, synthetic data was generated using two multivariate normal distributions:

- **Cluster 1**: Centered at (0,0) with a diagonal covariance matrix
- **Cluster 2**: Centered at (3,3) with the same covariance structure

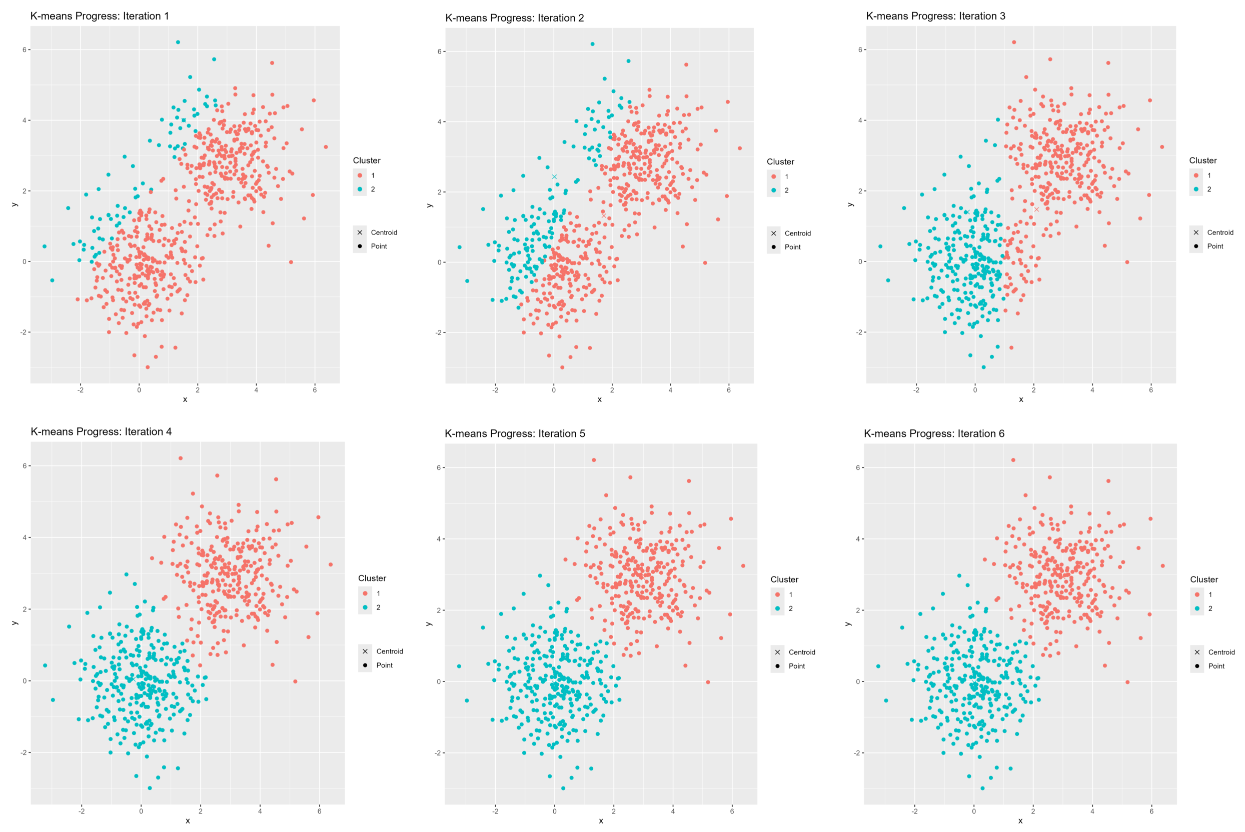
The data consists of 600 points (300 points per cluster), forming two distinct groups. This structure provides a straightforward case to validate the correctness of the K-means algorithm and assess its ability to separate well-defined clusters.



Scatterplot showing the dataset

Clustering Results and Visualization

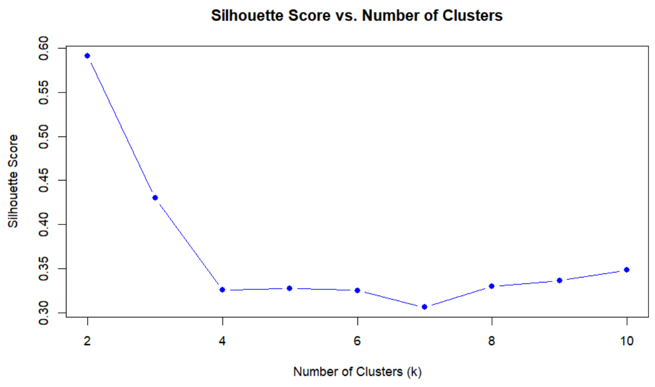
The `kmeans_custom` function was applied to the synthetic dataset with $k = 2$, accurately identifying the two clusters and their centroids. Then, the clustering process was visualized using `visualize_kmeans_progress`, which produced scatter plots displaying intermediate states, showing centroid movement and cluster assignment changes and an MP4 video illustrating the full progression of the algorithm.



Scatter plots displaying the intermediate steps, showing centroid movement and cluster assignment changes

Evaluation of Cluster Quality

To evaluate clustering quality, silhouette scores were computed for k values ranging from 2 to 10. Silhouette scores measure how well data points fit within their assigned clusters while being distinct from other clusters. Higher scores indicate better-defined clusters. The `evaluate_k` function was used to compute and plot silhouette scores for each k. The results showed that the optimal k for this dataset was 2, which aligns with the known structure of the synthetic data.



Line plot showing silhouette scores versus k; peak is at $k = 2$, indicating optimal number of clusters

Testing and Validation

We used the `testthat` package for systematic testing and automation and created unit tests to confirm that individual functions behave as expected.

Implementation of K-means algorithm:

- Validate the output structure: The result should include centroids, labels, and history
- Ensure that the number of centroids in the output equals k
- Check that the length of labels matches the number of data points
- Confirm that centroids stabilize after convergence

Visualizing the algorithm's process:

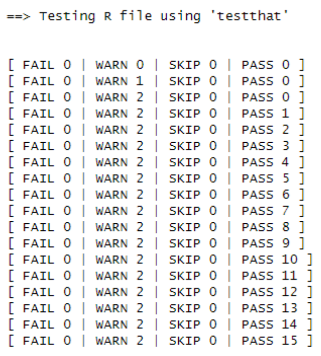
- Confirm that the function generates:
  - PNG files for each iteration
  - An MP4 animation summarizing the clustering process

Evaluation Metric - Silhouette Score:

- Check that the silhouette matrix has n rows (one per point) and 3 columns (a, b, and s)
- Validate that s(i) values are within the range [−1,1]

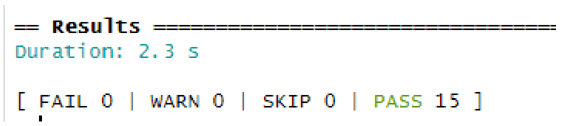
Evaluation of Cluster Quality:

- Ensure that the length of the output matches the length of k_range
- Check that the silhouette scores decrease or plateau as k increases beyond the optimal value (indicative of overfitting)



```
==> Testing R file using 'testthat'

[ FAIL 0 | WARN 0 | SKIP 0 | PASS 0 ]
[ FAIL 0 | WARN 1 | SKIP 0 | PASS 0 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 0 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 1 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 2 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 3 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 4 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 5 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 6 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 7 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 8 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 9 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 10 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 11 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 12 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 13 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 14 ]
[ FAIL 0 | WARN 2 | SKIP 0 | PASS 15 ]
```



```
══ Results ═════════════════════════════════
Duration: 2.3 s

[ FAIL 0 | WARN 0 | SKIP 0 | PASS 15 ]
```

Conclusion

This project successfully implemented the K-means clustering algorithm, meeting the objectives of creating a modular and reproducible function while enhancing its interpretability through effective visualizations. The `kmeans_custom` function accurately clustered synthetic data into distinct groups, demonstrating the algorithm's ability to iteratively optimize centroids and assign data points to appropriate clusters.

The visualizations created with the `visualize_kmeans_progress` function provided a clear and intuitive understanding of the clustering process, showing how centroids moved and clusters stabilized over iterations. Additionally, the evaluation of clustering quality using silhouette scores confirmed the effectiveness of the implementation, particularly in identifying the optimal number of clusters.

Through testing and validation, this project ensured the reliability and usability of the K-means functions. The combination of accurate results, detailed visualizations, and quality evaluation showcases the algorithm's capability to handle clustering tasks and makes it a strong tool for practical applications. This work not only demonstrates a solid understanding of the K-means algorithm but also lays a foundation for applying it to real-world datasets in future projects.

GitHub Link - https://github.iu.edu/sgaba/Kmeans