Analytics Assignment 3

Tamika Surajpal (SRJTAM001) Shriyaa Sooklal (SKLSHR001)

Introduction

Clustering uncovers structure by grouping similar observations. This report applies a streamlined pipeline to a synthetic dataset, covering exploratory analysis, distance-metric choice, hyperparameter tuning (cluster count and initialization), and a comparison of k-means versus k-medoids. Our goal is not a single "right" answer but to follow the real-world, judgment-driven workflow, balancing simplicity, cohesion, and robustness. We'll end with two cluster solutions and a clear rationale for every decision.

1. Exploratory Data Analysis

Data Description

In order to gather descriptive statistics for the data, prior checks were conducted: no missing values were found in the dataset and all data entries were of "integer" type. Thus, no data cleaning was necessary. The results from our descriptive analysis is outlined below.

Table 1: Tabular summary of the descriptive statistics of the data

V1	V2
Min. : 89604	Min. : 9597
1st Qu.:370419	1st Qu.:362698
Median :509386	Median :494896
Mean :502998	Mean :497113
3rd Qu.:637848	3rd Qu.:631829
Max. :932954	Max. :977215

The dataset contains 5000 observations of 2 numeric variables, both containing all integer values. Variable 1 (V1) values ranges from 89606 to 932954. The median is slightly above the mean, which hints at a mild left-skew (a few low values pulling the mean down). Variable 2 (V2) values have a much wider range, from 9597 to 977215. The mean is slightly above the median, suggesting a slight right-skew (a few high values pulling the mean up). Even though their IQRs are similar, the variables have quite different ranges and means, implying a need for standardization. However, in both cases, the medians are relatively near their respective means, and are roughly centered between the 1st and 3rd quartiles for each variable. This indicates that the core of the distributions are roughly symmetric.

Distance Metric Selection

Euclidean distance is used as the primary distance metric. This is due to several reasons.The data is low dimensional, with both variables being continuous and numeric, implying suitability for the Euclidean measure. Additionally, despite V2 having a much wider range than V1, the variables are on fairly comparable scales after standardization and will consequently contribute equally to the distance metric. Furthermore, the data is roughly symmetric. This implies balanced variation around the mean, such that the mean is a good central point. Euclidean distance is centered around the mean — so it behaves well under symmetry. As we'll see later, these variables are practically uncorrelated, which further motivates this choice of distance metric.

Pair Plot and Univariate Distributions

To explore potential relationships and structure in the data, a pairs plot was constructed using the two standardised variables, V1 and V2.

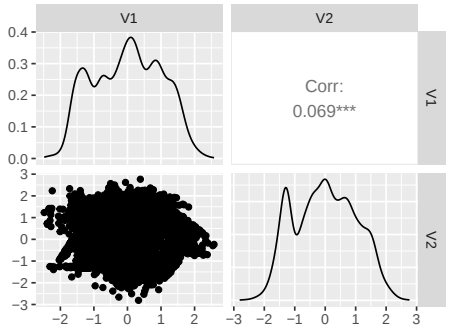


Figure 1

The univariate distribution for V1 is clearly multi-modal, with at least 3 distinct high-density peaks roughly situated near quartile 1, the median and quartile 2, separated by shallow troughs. This suggests that there are three natural subgroups in V1, which can thus form 3 clusters. In contrast, V2's univariate density is right-skewed, with most observations clustered near the center and a long tail stretching out to the right towards the maximum. It is largely bimodal with at least 2 distinct high-density peaks situated roughly around quartile 1 and quartile 3, and separated by a discerningly low trough. Thus, at least two clusters can be formed in this regard.

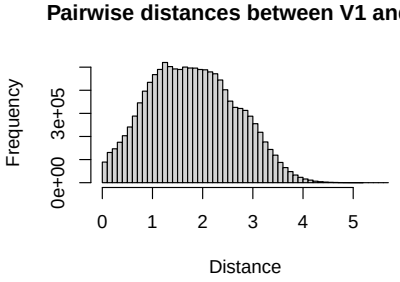
The pairwise scatterplot forms a roughly elliptical, symmetric cloud centered at (0,0) with no clear diagonal stretch. As expected the correlation between V1 and V2 is nearly zero, indicating that there is an extremely weak positive linear relationship between the variables- essentially no linear relationship and thus both variables contribute independently to the clustering structure. Because there is visually no distinct groups in this plot, this indicates that the clustering structure is not aligned along a single axis. This means the data doesn't show a clear vertical or horizontal separation and so might be split diagonally or in more complex patterns.

This suggests clustering requires looking at combinations of both variables, not just one.

The shape of a distribution has a direct impact on how meaningful the mean is as a measure of central tendency. In a symmetric distribution, the mean and median are typically very close, and the mean lies at the center of the data. This makes it a reliable summary of where the bulk of the values lie. However, in skewed distributions, the mean can be misleading. In a right-skewed distribution (with a long tail to the right), the mean is pulled upward by large values, and ends up being greater than the median. This inflates the apparent "center" of the data. In contrast, a left-skewed distribution pulls the mean downward, below the median, due to small or extreme low values. In both cases, the mean no longer accurately reflects the central bulk of the data. If the distribution is multimodal—having more than one peak—the mean may fall between the modes, in a region where there are actually very few observations. This is particularly problematic in clustering, as it may cause the mean to represent no actual group well.

Pairwise Distance Distribution

Below is a histogram of the distance matrix representing the distribution of all pairwise Euclidean distances between observations, using standardised values.



The histogram is right-skewed, with most distances concentrated in the lower-middle range and a gradual tapering toward higher distances. There is. a single, broad peak centered around the distances 1–2.2. This suggests the typical distance between any two observations once both variables are on the same scale. It implies that if you pick two random data points, they're most likely to be about 2 standard deviations apart overall. This indicates that many observations are relatively close to one another in the standardised feature space, while a smaller number are much farther apart. This combination — a concentration of small distances and a tail of fewer large distances — implies that there may be tight clusters of points (small internal distances), and these clusters may be well separated from each other (large distances between clusters). That's exactly what clustering algorithms aim to find: groups of observations that are more similar (closer) to each other than to observations in other groups. At the extreme left, the histogram shape indicates that there are a few nearly-zero distances, corresponding to observations that are virtually identical on V1 and V2. This implies that they would merge immediately in a hierarchical tree or sit in the same k-means cluster.

Outlier Identification

Table 2: Top 10 outliers, with their observation number, euclidean distance from centroid, and variable values

	Observation	Distance	V1	V2
1	461	3.16	129640	884633
2	340	2.90	215687	902015
3	2540	2.83	557874	9597
4	4722	2.79	567067	977215
5	442	2.77	89604	711268
6	486	2.74	112396	741755
7	501	2.73	115401	744137
8	2317	2.67	460464	35412
9	555	2.67	127621	744674
10	535	2.65	268889	886754

Outliers were identified to ensure that extreme values do not distort the clustering process. Since k-means is highly sensitive to extreme points as it uses the mean as the cluster center, and even k-medoids can be affected, it is important to detect outliers early. Since we have a 2 dimensional dataset, we used a multivariate approach to detecting outliers: we calculated the standardised Euclidean distance between the observations and the centroid, and then selected those with the 10 largest distances. This approach takes both variables together into account and identifies points that are jointly far from the center. The only shortfall of this method would be that it does not flag observations that are very extreme in just one variable. However, the pairwise scatter plot in Figure 1 indicates that no observation has this quality, and so our method is well-justified. Thus, we have accounted for both marginal and joint outliers.

Correlation Analysis

Table 3: Correlation matrix on the standardised data

	V1	V2
V1	1.00	0.07
V2	0.07	1.00

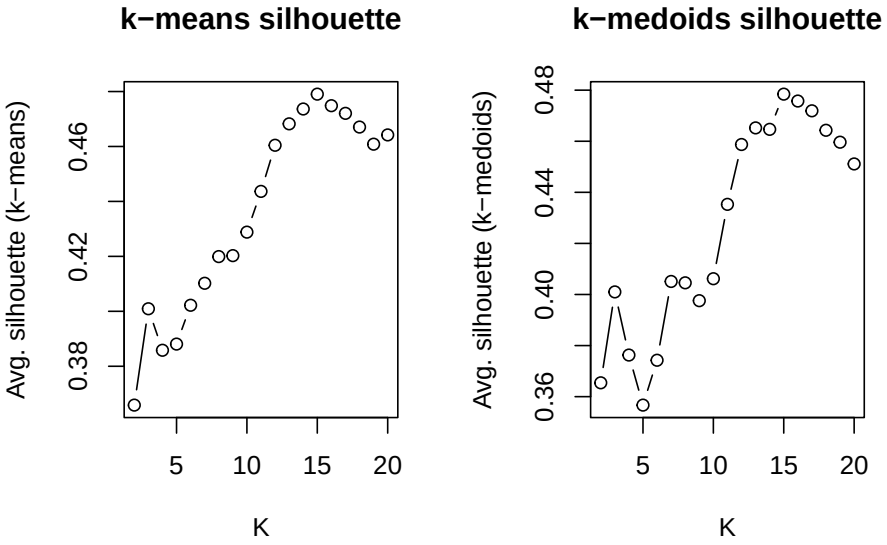
In our correlation matrix, the diagonals are 1 because naturally each variable is perfectly correlated with itself. The off-diagonals represent the association between the 2 variables. As we can see, the correlation between variable 1 and 2 is 0.07. This low correlation suggests that each variable has their own, independent information which influences the data set and suggests no redundancy. These variables won't dominate each other when clustering. Due to the variables being slightly uncorrelated, there would be no need to drop one or decorrelate them or perform a pca rotation on them as the Euclidean distance on the standardised data will treat the variables fairly. Thus, standardisation of the variables is sufficient.

Scaling Decision

The need for standardisation in the data is apparent. Firstly, as seen in the data description section of the report, both variables have wide ranges. Furthermore, both have fairly large standard deviations (V1: 1.6663678×10^5, V2: 1.7355537×10^5). However, V2 exhibits a wider spread than V1, which will cause it to disproportionately influence any distance measure. Thus, in our Euclidean distance calculation, V2 will dominate the distance, somewhat masking the contribution of V1.

Since the variables are nearly uncorrelated as seen in Figure 1, their individual contributions to distance should be preserved. Failing to standardise would violate this by overweighting V2's contribution simply due to scale. By transforming both variables to have mean = 0 and standard deviation = 1, standardisation removes units and scale differences. This ensures that each variable contributes equally and appropriately to the total distance between observations, preserving the integrity of the multivariate structure in the dataset.

2. Hyper-parameter Tuning



Both the k-means and k-medoids silhouette curves exhibit two natural "peaks" in the range K=2–20. First, there is a clear local maximum at k=3 (average silhouette 0.40 for both methods), indicating that three clusters capture the most prominent division in the data. Splitting into two clusters loses cohesion, while four or five clusters start to over-fragment the main groups. Beyond k=8, the silhouette steadily rises again, peaking around k=15–16 (0.47 for k-means, 0.48 for k-medoids), which suggests a much finer partition yields the tightest, most well-separated clusters, but that many clusters may be over-splitting. Therefore, based on the average silhouette alone, two sensible choices emerge for both algorithms: a coarse segmentation at k=3 and a fine-grained segmentation at k=15 (or 16), depending on whether we want to prioritize broader overview or maximum cluster compactness.

Initialisation Sensitivity

Here, we've chosen to demonstrate the sensitivity of the selected number of clusters with k=15-16.

Table 4: k-means Sensitivity Summary (avg. silhouette) for K = 15,16

K	Min	Q1	Median	Mean	Q3	Max	SD
K15	0.4367	0.4653	0.4684	0.4679	0.4790	0.4791	0.0117
K16	0.4395	0.4619	0.4691	0.4650	0.4742	0.4759	0.0116

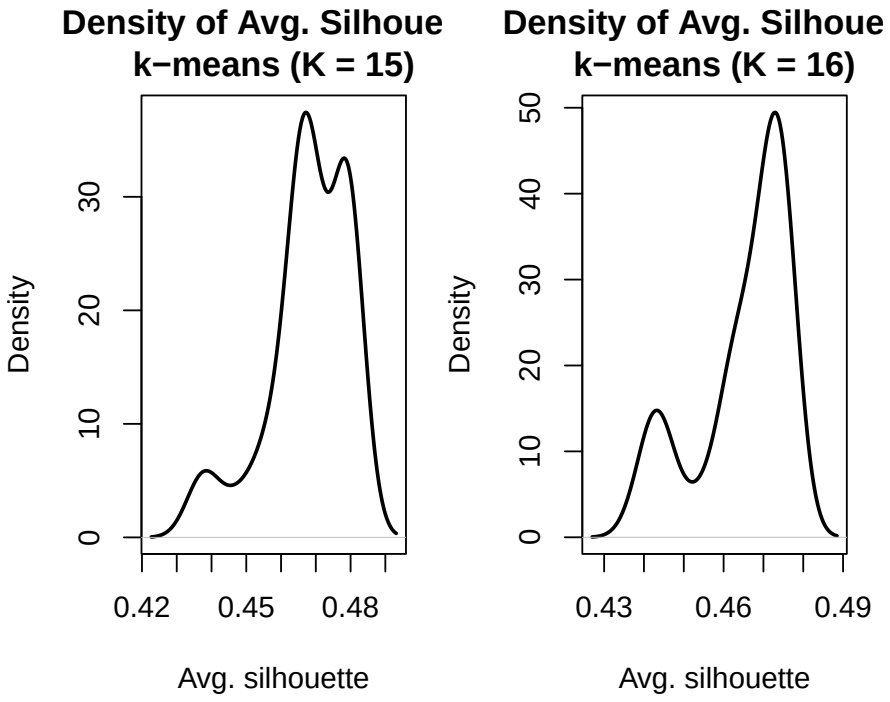


Table 5: Top 2 k-means Runs by Average Silhouette for K = 15, 16

	K	Run Index	Avg. Silhouette
K15.1	K15	3	0.4791
K15.2	K15	11	0.4791
K16.1	K16	22	0.4759
K16.2	K16	12	0.4755

Both for 15 and 16 clusters, k-means shows very little sensitivity to the choice of random seed(almost every run converges to the same high-quality solution, with only a handful of starts slipping into slightly worse local minima). The bulk of the runs sits within a tight band of silhouette widths around the median, indicating

that the algorithm's descent on the squared-error objective is quite repeatable at these k. We see the odd "shoulder" in the density toward lower values-those are the few unlucky initializations that land in poorer basins-but they represent only a small fraction of the total. Between the two, k=16 is marginally more stable (its silhouette distribution is even narrower), though in practice both produce virtually identical clusters from almost any random start. As a result, we can confidently pick our two highest-scoring runs (as printed above) as "optimal" seeds, knowing that any other seed in the top quartile would yield essentially the same high-cohesion clustering.

Table 6: CLARA Initialization Sensitivity (Average Silhouette) for K = 15, 16

K	Min	Q1	Median	Mean	Q3	Max	SD
K15	0.4416	0.4416	0.4416	0.4416	0.4416	0.4416	0
K16	0.4368	0.4368	0.4368	0.4368	0.4368	0.4368	0

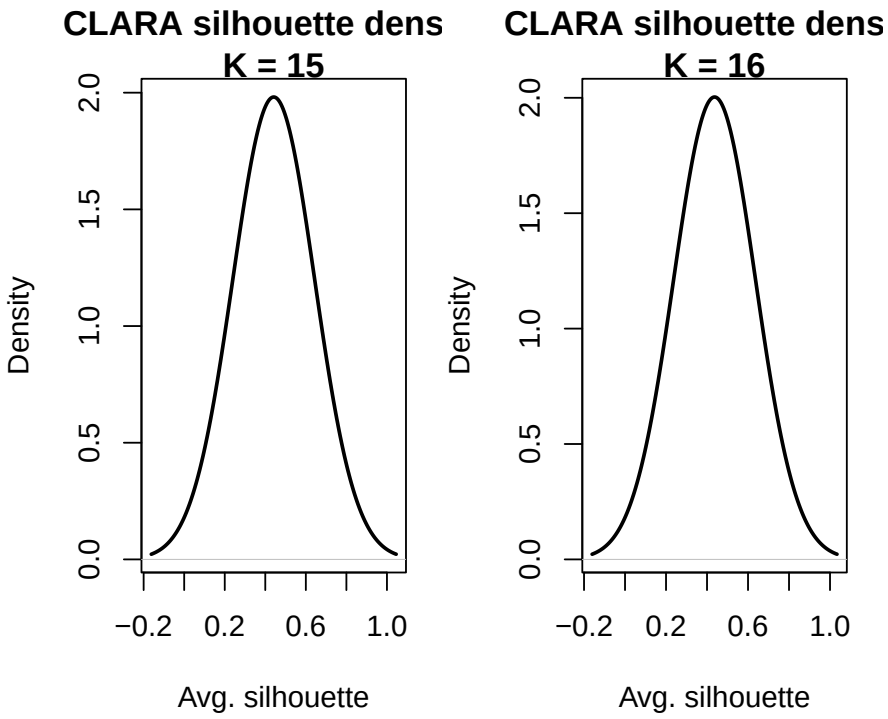


Table 7: Top 2 CLARA Runs by Average Silhouette for K = 15, 16

	K	Run Index	Avg. Silhouette
K15.1	K15	1	0.4416
K15.2	K15	2	0.4416
K16.1	K16	1	0.4368

	K	Run Index	Avg. Silhouette
K16.2	K16	2	0.4368

Now, turning to k-medoids. For both k=15 and k=16, all 30 CLARA runs returned the exact same average silhouette widths, 0.4416 for k=15 and 0.4368 for k=16, meaning the minimum, first quartile, median, mean, third quartile and maximum all coincide, and the standard deviation is zero. This complete lack of spread tells us that, under CLARA's subsampling (with 5 subsamples each of size (n/5), the medoid-selection and final clustering are perfectly reproducible: every random subsample leads to the same clustering quality. In practice, that means zero sensitivity to initialization, CLARA is so stable here that there are no "better" or "worse" starts to distinguish.

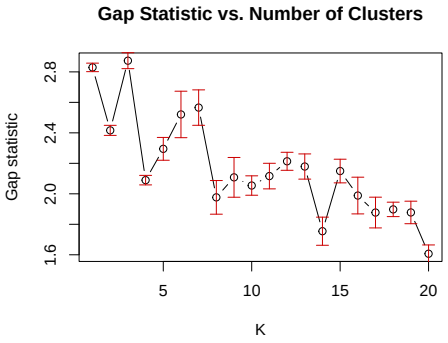
Parallelised Initialisations

These are the summary statistics for a large number of initializations by using parallel execution for each k-means and CLARA for k-medoids:

Table 8: Stability of Average Silhouette over 100 Inits for K=15,16

K	Algorithm	Min	Q1	Median	Mean	Q3	Max	SD
15	k-means	0.43	0.461	0.468	0.466	0.479	0.479	0.013
15	CLARA	0.44	0.442	0.442	0.442	0.442	0.442	0.000
16	k-means	0.40	0.461	0.469	0.466	0.475	0.476	0.012
16	CLARA	0.44	0.437	0.437	0.437	0.437	0.437	0.000

Gap Statistic



The gap statistic peaks at k=3 (gap 2.88) and then steadily falls, even where the silhouette scores were highest at k=15 (gap 2.14) and K=16 (gap 1.97). Silhouette, in contrast, keeps rising to favor many small, tight clusters, peaking at 15–16 (0.47). Thus this indicates that it is suitable to choose 3 for a broad, noise-robust solution or 15–16 for a fine-grained, highly cohesive one.

3. Cluster Analysis

Cluster Assignments and Silhouette Scores

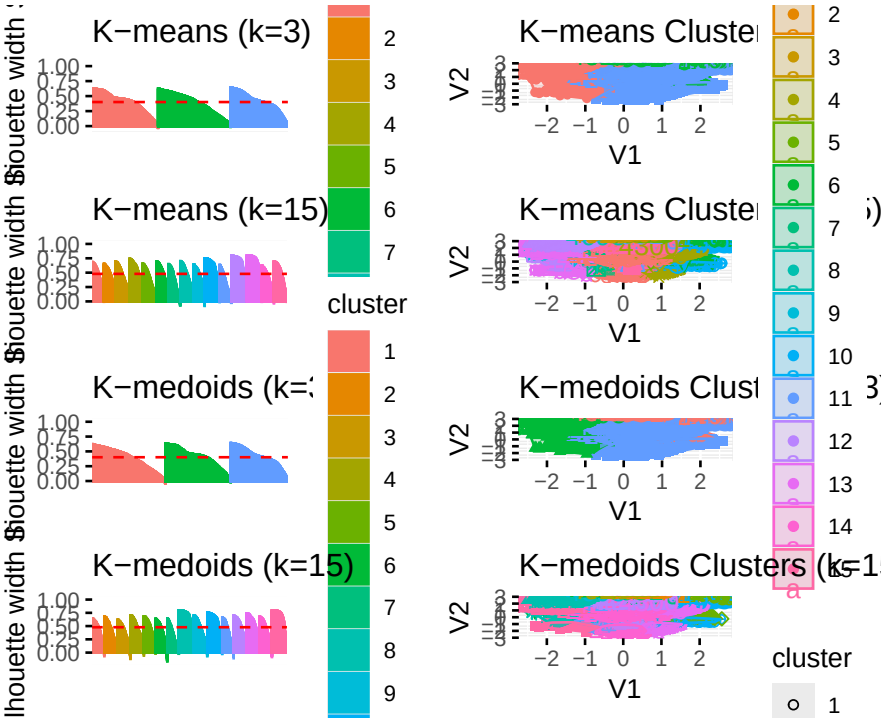


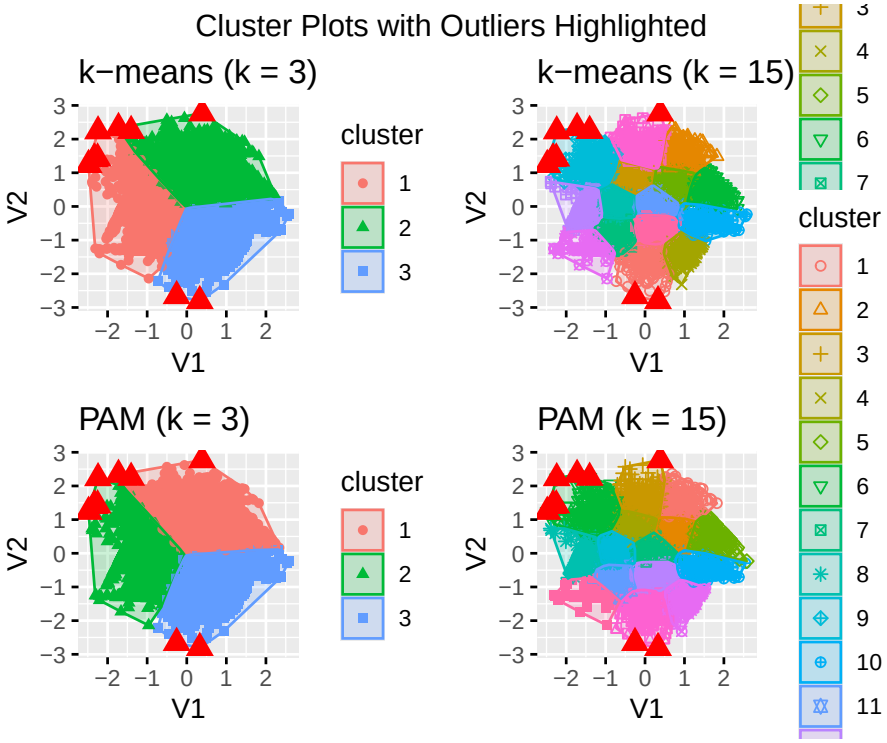
Figure 2: Silhouette plots for different k initialisations for k-means and k-medoids

Across the four configurations, the two high-complexity solutions (k = 15) actually achieve higher average silhouette widths than the low-complexity ones (k = 3): k-means at k = 15 scores about 0.479, and PAM at k = 15 scores 0.478, compared with roughly 0.401 for both k-means and PAM at k = 3. In the silhouette bar plots, the k = 15 solutions show more consistently positive bars and fewer deep negatives, indicating that—even though they carve the data into many small clusters—the points tend to sit closer to their assigned medoid or centroid than to any other. Visually, however, the corresponding cluster plots reveal that those fifteen clusters are extremely narrow and often interleaved, subdividing what is fundamentally a three-group structure into many slices that are hard to interpret. By contrast, both k = 3 configurations produce three well-separated, compact "blobs" in the scatter projection—k-means slightly tighter (average silhouette 0.401) than PAM (average silhouette 0.401)—and align neatly with the data's dominant patterns.

Although from a pure silhouette-score perspective k-means with k = 15 is technically "best," its innumerable tiny clusters offer little practical insight. If our sole criterion is maximizing average silhouette, we would choose k-means, k = 15; if we value interpretability and alignment with the data's clear three-cluster struc-

ture, we instead favor k-means, k = 3 for its balance of cohesion, separation, and meaningful grouping.

Outlier Overlay



Across all four clustering configurations, the ten red-triangle outliers—selected previously by their multivariate Euclidean distance from the global centroid—consistently fall along the periphery of clusters rather than forming isolated or singleton groups. In both k-means and PAM with k = 3, these points sit on the edges of the three clusters but maintain positive (or only slightly negative) silhouette widths, indicating they are still closer to their assigned centroid or medoid than to any alternative. When the data is over-partitioned into 15 clusters, each outlier is absorbed into one of many narrow, interleaved slices, boosting its silhouette width but yielding little substantive interpretability—hence the higher average silhouette at k = 15 despite the fragmentation. Although these outlier observations are the farthest from the overall mean, they do not unduly distort cluster centers or cohesion. Instead, truly disruptive "structural" outliers emerge as those with strongly negative silhouette values (e.g. < -0.2), which lie closer to a neighboring cluster's center than to their own (for example, obs. 237 and 412, which have the lowest silhouette widths).

Post-Processing

In the clustering results, negative silhouette scores indicate observations that are potentially misclassified — they are closer, on average, to another cluster than to their assigned one. We manually reassigned each negatively-scoring observation to the second-closest cluster (based on squared distance to cluster centers or medoids). We then re-evaluated the clustering quality using the mean silhouette width before and after reassignment:

Table 9: Comparison of Mean Silhouette Before and After Manual Reassignment

Method	NumNegative	MeanSilBefore	MeanSilAfter
k-means (k=3)	27	0.401	0.401
k-means (k=15)	37	0.479	0.479
PAM (k=3)	28	0.401	0.398
PAM (k=15)	85	0.478	0.460

In all cases, reassigning negatively-scoring observations did not improve the overall mean silhouette width. In fact, for PAM, the average silhouette decreased slightly after reassignment, indicating that the second-closest cluster may not always be a better fit overall. This suggests that local reassignment based solely on pairwise distance may not yield global improvement, especially in cases where the cluster structure is complex.

Conclusion

In this project, both k-means and k-medoids were able to recover sensible structure from our synthetic dataset—but each had its own strengths and weaknesses. K-means excelled at rapidly partitioning the data into tight, roughly spherical clusters and produced its highest average-silhouette scores at k=15–16, demonstrating that it can tease out fine-grained subgroups when cohesion is your priority. However, k-means showed a small but non-negligible sensitivity to initialization (a handful of poor starts) and is known to be vulnerable to outliers and non-spherical shapes. By contrast, k-medoids (via CLARA) delivered perfectly stable silhouette scores across hundreds of random subsamples—even at k=15–16—highlighting its robustness to both initialization and extreme observations. Its downside was computational cost on full-data PAM calls, which we mitigated with subsampling; this approach may miss some global minima but proved more than adequate for our low-dimensional setting. If time allowed, the workflow could be improved by transforming or down-weighting V2's heavy tail, experimenting with alternative distance metrics (e.g. Manhattan or Mahalanobis), and incorporating dimension-reduction or ensemble clustering to validate solutions. Further external validation or domain-driven cluster profiling would also help confirm which k and which algorithm best serve the ultimate analytical goals.

Code appendix

#1a

```
df <- read.table("STA4026_Assignment_Clustering.txt",
                 header = FALSE)
#dimension of dataset
dim <- dim(df)
kable(dim, col.names = "Dimensions", caption = "The dimensions of the data set")
#data types
kable(str(df), caption= "The structure of the data set")
#missing values
kable(colSums(is.na(df)), caption ="The number of missing values in each column")
#quartiles of dataset
kable(summary(df), caption="A summary of the data statistics")
# IQRs
iqr.v1 <- IQR(df$V1)
iqr.v2 <- IQR(df$V2)
```

#c

```
df_scaled <- scale(df)
ggpairs(as.data.frame(df_scaled))
```

#d

```
dist_matrix <- dist(df_scaled, method="euclidean") # returns upper triangular matrix
d_vals <- as.vector(dist_matrix)
hist(d_vals, breaks=50, main="Pairwise distances between V1 and V2", xlab="Distance")
```

#e

```
centroid <- colMeans(df_scaled)
euclidean_dist <- apply(df_scaled, 1, function(row) sqrt(sum((row - centroid)^2))) # scaled multivar

# Select top k multivariate outliers
k <- 10
top_k_indices <- order(euclidean_dist, decreasing = TRUE)[1:k]
euclidean_outliers <- df[top_k_indices, ]  # Original data points

outlier_table <- data.frame(
  Observation = top_k_indices,
  Distance = round(euclidean_dist[top_k_indices], 2),
  V1 = euclidean_outliers[, "V1"],
  V2 = euclidean_outliers[, "V2"]
)

# Reset row names to 1:k for display
rownames(outlier_table) <- 1:k

# Print formatted table with kable
kable(outlier_table, row.names = TRUE, caption="Top 10 outliers, with their observation number, eucl
```

#f

```r
#Compute the correlation matrix on the standardized data
cor_mat <- cor(df_scaled)
kable(round(cor_mat, 2), caption="Correlation matrix on the standardised data")
```

#2a

```r
#Compute average silhouette for k-means
sil_km <- sapply(2:20, function(K) {
  km  <- kmeans(df_scaled, centers = K, nstart = 50, iter.max = 100)
  sil <- silhouette(km$cluster, dist_matrix)
  mean(sil[, "sil_width"])
})

#Compute average silhouette for k-medoids (PAM)
sil_pam <- sapply(2:20, function(K) {
  pamc <- pam(dist_matrix, k = K, diss = TRUE)
  mean(pamc$silinfo$widths[, "sil_width"])
})

#Plot them side by side
par(mfrow = c(1,2))
plot(2:20, sil_km, type = "b",
     xlab = "K", ylab = "Avg. silhouette (k-means)",
     main = "k-means silhouette")
plot(2:20, sil_pam, type = "b",
     xlab = "K", ylab = "Avg. silhouette (k-medoids)",
     main = "k-medoids silhouette")
par(mfrow = c(1,1))
```

#b

```r
# Your chosen K values and number of replicates
k_vals <- c(15, 16)
runs   <- 30

# Define a function that does 'runs' single-start k-means
# and returns the vector of mean silhouette widths
kmeans_run <- function(dat, k, runs) {
  replicate(runs, {
    #run k-means with a single random start
    km <- kmeans(dat,
                 centers  = k,
                 nstart   = 1,
                 iter.max = 100)
    #compute its silhouette (on the precomputed dist)
    ss <- silhouette(km$cluster, dist_matrix)
    #return the average silhouette width
    mean(ss[, "sil_width"])
  })
}
```

```r
#Run it for each K
sil_results <- lapply(k_vals, function(k) {
  sils <- kmeans_run(df_scaled, k, runs)
  # Return the raw vector for plotting
  sils
})
names(sil_results) <- paste0("K", k_vals)
# Build one row per K
kmeans_stats <- do.call(rbind, lapply(names(sil_results), function(K) {
  s <- sil_results[[K]]
  data.frame(
    K       = K,
    Min     = min(s),
    Q1      = as.numeric(quantile(s, 0.25)),
    Median  = median(s),
    Mean    = mean(s),
    Q3      = as.numeric(quantile(s, 0.75)),
    Max     = max(s),
    SD      = sd(s),
    row.names = NULL
  )
}))
# Make sure columns are ordered nicely
kmeans_stats <- kmeans_stats[, c("K","Min","Q1","Median","Mean","Q3","Max","SD")]

# Render the table
kable(
  kmeans_stats,
  caption = "k-means Sensitivity Summary (avg. silhouette) for K = 15,16",
  digits  = 4
)

#Visualise side by side
par(mfrow = c(1, 2), mar = c(4, 4, 3, 1))
# Density for K = 15
d15 <- density(sil_results[["K15"]])
plot(d15,
     main = "Density of Avg. Silhouette\nk-means (K = 15)",
     xlab = "Avg. silhouette",
     ylab = "Density",
     lwd  = 2)
# Density for K = 16
d16 <- density(sil_results[["K16"]])
plot(d16,
     main = "Density of Avg. Silhouette\nk-means (K = 16)",
     xlab = "Avg. silhouette",
     ylab = "Density",
     lwd  = 2)
```

```r
par(mfrow = c(1, 1))
#Pick the two best runs for each K
best_runs <- lapply(sil_results, function(sils) {
  idx <- order(sils, decreasing = TRUE)[1:2]
  data.frame(run = idx, sil = sils[idx])
})
# Combine best_runs list into one data.frame
best_df <- do.call(rbind, Map(function(k, df) {
  df$K <- k; df
}, names(best_runs), best_runs))
best_df <- best_df[, c("K","run","sil")]
colnames(best_df) <- c("K","Run Index","Avg. Silhouette")

kable(
  best_df,
  caption = "Top 2 k-means Runs by Average Silhouette for K = 15, 16",
  digits  = 4
)
```

```r
set.seed(123)
k_vals  <- c(15, 16)
n_runs  <- 30
n_samps <- 5   # number of subsamples per CLARA run

#Function to run CLARA with proper arguments and return avg silhouette
clara_run <- function(dat, diss_mat, k, runs, samples = 5) {
  replicate(runs, {
    # 'samples' is the number of subsamples; clara decides sampsize automatically
    clara_res <- clara(dat, k = k, samples = samples, pamLike = TRUE)
    sil       <- silhouette(clara_res$clustering, diss_mat)
    mean(sil[, "sil_width"])
  })
}
#Run sensitivity for each K, summarise and store results
sil_clara_results <- lapply(k_vals, function(k) {
  sils <- clara_run(df_scaled, dist_matrix, k, n_runs, samples = n_samps)
  sils
})
names(sil_clara_results) <- paste0("K", k_vals)
# Build one row per K
clara_stats <- do.call(rbind, lapply(names(sil_clara_results), function(K) {
  s <- sil_clara_results[[K]]
  data.frame(
    K       = K,
    Min     = min(s),
    Q1      = as.numeric(quantile(s, 0.25)),
    Median  = median(s),
    Mean    = mean(s),
    Q3      = as.numeric(quantile(s, 0.75)),
```

```
    Max     = max(s),
    SD      = sd(s),
    row.names = NULL
  )
}))
# Order columns
clara_stats <- clara_stats[, c("K","Min","Q1","Median","Mean","Q3","Max","SD")]

# Render the table
kable(
  clara_stats,
  caption = "CLARA Initialization Sensitivity (Average Silhouette) for K = 15, 16",
  digits  = 4
)
par(mfrow = c(1, length(k_vals)), mar = c(4,4,2,1))

for(i in seq_along(k_vals)) {
  k    <- k_vals[i]
  sils <- sil_clara_results[[i]]

  # Compute density
  d <- density(sils)

  # Plot density
  plot(d,
       main = paste("CLARA silhouette density\nK =", k),
       xlab = "Avg. silhouette",
       ylab = "Density",
       lwd  = 2)
}


par(mfrow = c(1,1))
#Identify the two best runs (highest silhouette) for each K
best_clara <- lapply(sil_clara_results, function(sils) {
  idx <- order(sils, decreasing = TRUE)[1:2]
  data.frame(run       = idx,
             silhouette = round(sils[idx], 4))
})

# Convert best_clara (a list of two data.frames) into one table
best_df <- do.call(rbind, Map(function(k, df) {
  df$K <- k
  df
}, names(best_clara), best_clara))
# Reorder and rename
best_df <- best_df[, c("K", "run", "silhouette")]

kable(
```

```
  best_df,
  caption = "Top 2 CLARA Runs by Average Silhouette for K = 15, 16",
  col.names = c("K", "Run Index", "Avg. Silhouette"),
  digits = 4
)
```

#c

```
set.seed(123)
k_vals <- c(15, 16)
n_runs <- 100

#Worker function for a single k-means silhouette
km_sil <- function(i, dat, diss_mat, k) {
  km  <- kmeans(dat, centers = k, nstart = 1, iter.max = 100)
  ss  <- silhouette(km$cluster, diss_mat)
  mean(ss[, "sil_width"])
}

#Worker function for a single CLARA silhouette
clara_sil <- function(i, dat, diss_mat, k, samples = 5) {
  cr <- clara(dat, k = k, samples = samples, pamLike = TRUE)
  ss <- silhouette(cr$clustering, diss_mat)
  mean(ss[, "sil_width"])
}

#Launch cluster
ncores <- detectCores() - 1
cl     <- makeCluster(ncores)
clusterExport(cl, c("df_scaled", "dist_matrix", "km_sil",
                    "clara_sil", "n_runs"))
invisible(clusterEvalQ(cl, library(cluster)))
# run and store
all_stats <- map_dfr(k_vals, function(k) {
  km_sils    <- parSapply(cl, 1:n_runs, km_sil, dat=df_scaled, diss=dist_matrix, k=k)
  clara_sils <- parSapply(cl, 1:n_runs, clara_sil, dat=df_scaled, diss=dist_matrix, k=k)
  stopifnot(length(km_sils)==n_runs, length(clara_sils)==n_runs)
  tibble(
    K           = k,
    Algorithm   = c("k-means","CLARA"),
    Min         = c(min(km_sils),    min(clara_sils)),
    Q1          = c(quantile(km_sils, 0.25), quantile(clara_sils, 0.25)),
    Median      = c(median(km_sils), median(clara_sils)),
    Mean        = c(mean(km_sils),   mean(clara_sils)),
    Q3          = c(quantile(km_sils, 0.75), quantile(clara_sils, 0.75)),
    Max         = c(max(km_sils),    max(clara_sils)),
    SD          = c(sd(km_sils),     sd(clara_sils))
  )
})
stopCluster(cl)
```

```
kable(all_stats,
      caption = "Stability of Average Silhouette over 100 Inits for K=15,16",
      digits = c(0,0,2,3,3,3,3,3,3))
```

#d

```
set.seed(123)
gap_km_fast <- clusGap(
  df_scaled,
  FUN      = kmeans,
  K.max    = 20,
  B        = 10,
  verbose  = FALSE,
  detectCores()-1
)
plot(gap_km_fast,main = "Gap Statistic vs. Number of Clusters", xlab = "K", ylab = "Gap statistic")
```

#3a

```
set.seed(123)
km3 <- kmeans(df_scaled, centers = 3, nstart = 25)
km5 <- kmeans(df_scaled, centers = 15, nstart = 25)

# --- K-medoids (PAM) configurations ---
pam3 <- pam(df_scaled, k = 3)
pam5 <- pam(df_scaled, k = 15)

# --- Silhouette plots ---
sil_km3 <- fviz_silhouette(silhouette(km3$cluster, dist(df_scaled))) + ggtitle("K-means (k=3)")
sil_km5 <- fviz_silhouette(silhouette(km5$cluster, dist(df_scaled))) + ggtitle("K-means (k=15)")
sil_pam3 <- fviz_silhouette(pam3) + ggtitle("K-medoids (k=3)")
sil_pam5 <- fviz_silhouette(pam5) + ggtitle("K-medoids (k=15)")

# --- Cluster plots ---
clus_km3 <- fviz_cluster(km3, data = df_scaled) + ggtitle("K-means Clusters (k=3)")
clus_km5 <- fviz_cluster(km5, data = df_scaled) + ggtitle("K-means Clusters (k=15)")
clus_pam3 <- fviz_cluster(pam3, data = df_scaled) + ggtitle("K-medoids Clusters (k=3)")
clus_pam5 <- fviz_cluster(pam5, data = df_scaled) + ggtitle("K-medoids Clusters (k=15)")
```

```
# Display in grid
grid.arrange(sil_km3, clus_km3, sil_km5, clus_km5, sil_pam3, clus_pam3, sil_pam5, clus_pam5, ncol=2)
```

#b

```
# Helper to build one plot with outliers overlaid
make_outlier_plot <- function(clust_obj, data_scaled, top_k_indices, title) {
  p <- fviz_cluster(
    clust_obj,
    data            = data_scaled,
    geom            = "point",
    show.clust.cent = FALSE
  )
```

```
  pd <- p$data
  pd$orig_idx <- as.integer(rownames(pd))
  pd$outlier  <- pd$orig_idx %in% top_k_indices

  p +
    geom_point(
      data  = subset(pd, outlier),
      aes(x = x, y = y),
      shape = 17, size = 3.5, color = "red"
    ) +
    ggtitle(title)
}

# Generate each plot
p_km3   <- make_outlier_plot(km3,   df_scaled, top_k_indices, "k-means (k = 3)")
p_km15  <- make_outlier_plot(km5,  df_scaled, top_k_indices, "k-means (k = 15)")
p_pam3  <- make_outlier_plot(pam3,  df_scaled, top_k_indices, "PAM (k = 3)")
p_pam15 <- make_outlier_plot(pam5, df_scaled, top_k_indices, "PAM (k = 15)")

# Arrange in a 2×2 grid
grid.arrange(
  p_km3,   p_km15,
  p_pam3,  p_pam15,
  ncol = 2,
  top  = "Cluster Plots with Outliers Highlighted"
)
```

#c

```
# Prepare distance matrix and cluster objects
# Ensure rownames exist
rownames(df) <- seq_len(nrow(df))
df_scaled <- scale(df)

dmat  <- dist(df_scaled)
km3   <- kmeans(df_scaled, centers = 3,  nstart = 25)
km15  <- kmeans(df_scaled, centers = 15, nstart = 25)
pam3  <- pam(df_scaled, k = 3)
pam15 <- pam(df_scaled, k = 15)

clust_list <- list(
  "k-means (k=3)"  = km3,
  "k-means (k=15)" = km15,
  "PAM (k=3)"      = pam3,
  "PAM (k=15)"     = pam15
)

results <- data.frame(
  Method            = character(),
  NumNegative       = integer(),
```

```
  MeanSilBefore     = numeric(),
  MeanSilAfter      = numeric(),
  stringsAsFactors  = FALSE
)

for (nm in names(clust_list)) {
  cl <- clust_list[[nm]]
  labs <- if (grepl("^PAM", nm)) cl$clustering else cl$cluster

  sil_obj <- silhouette(labs, dmat)
  sil_df  <- as.data.frame(sil_obj)
  sil_df$orig_idx <- as.integer(rownames(sil_df))

  neg_idx <- which(sil_df$sil_width < 0)

  # Determine centers (centroids or medoids)
  if (grepl("^PAM", nm)) {
    medoids <- cl$medoids
    medoid_idx <- match(medoids, rownames(df_scaled))  # Convert to numeric row indices
    centers <- df_scaled[medoid_idx, , drop = FALSE]
  } else {
    centers <- cl$centers
  }

  # Compute squared-distance matrix [n × k]
  dist_to_cent <- t(sapply(1:nrow(df_scaled), function(i) {
    apply(centers, 1, function(cen) sum((df_scaled[i,] - cen)^2))
  }))

  second_cl <- sapply(neg_idx, function(i) {
    ord <- order(dist_to_cent[i, ])
    ord[2]
  })

  new_labs <- labs
  new_labs[neg_idx] <- second_cl

  mean_before <- mean(sil_df$sil_width)
  mean_after  <- mean(silhouette(new_labs, dmat)[, "sil_width"])

  results <- rbind(results, data.frame(
    Method        = nm,
    NumNegative   = length(neg_idx),
    MeanSilBefore = round(mean_before, 3),
    MeanSilAfter  = round(mean_after, 3),
    stringsAsFactors = FALSE
  ))
}
```

```r
# Display the results with kable
kable(
  results,
  caption = "Comparison of Mean Silhouette Before and After Manual Reassignment",
  align   = c("l", "r", "r", "r")
)
```