

# STAT 610 project: K-Means Clustering

Jinglong Wang, Yijia Li, Mujia Chen

December 9 2024

<https://github.com/jinglong-w-dotcom/stat610project>

##Idea: We have  $n$  points (vectors),  $x_1, \dots, x_n$  in  $\mathbb{R}^p$ , and divide these points into non-empty  $K$  clusters (groups),  $C_1, \dots, C_K$ . Our goal is to minimize the within-cluster variation to make the points within each cluster as close to each other as possible. We define K-means clustering formula as below:

$$\min_{C_1, \dots, C_K} \left\{ \sum_{k=1}^K \frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \right\}$$

##Algorithm

Basing on the K-means principle, we design the following 5 functions:

- `compute_centers`: Calculate cluster centroids. The  $k$ -th cluster center  $\bar{x}_k$  is the average of all the points in the  $k$ -th cluster.

$$\bar{x}_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

- `WCSS`: Calculate the within-cluster sum of squares to evaluate clustering compactness (where using Euclidean distance to define the "closest"  $\sum_{j=1}^n (x_{ij} - x_{i'j})^2$ ). An iteration reduces out total WCSS, as shown below,

$$\begin{aligned} \sum_{k=1}^K \frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 &= \sum_{k=1}^K 2 \sum_{i \in C_k} (x_i - \bar{x}_k)^2 \text{ step 2(a)} \\ &= 2 \sum_{i=1}^n (x_i - \bar{x}_{k(i)})^2 \text{ break the cluster, } k(i) \text{ means } x_i \text{ is in the } k(i)\text{-th cluster} \\ &\geq 2 \sum_{i=1}^n (x_i - \bar{x}_{kc(i)})^2 \text{ step 2(b), } x_i \text{ assigned to its closest cluster} \\ &= \sum_{k=1}^K 2 \sum_{i \in C'_k} (x_i - \bar{x}_k)^2 \end{aligned}$$

- `k_means`: Perform K-Means Clustering

1. Initialize clusters. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.

2. Iterate until the cluster assignments stop changing:

(a) For each of the K clusters, compute the cluster center, using function `compute_centers`.

(b) Assign each point to the cluster whose center is closest. We use function `WCSS` to evaluate the distribution.

- `K_means_progresses`: Visualize clustering progress
- `Kmeans_quality_measure`: Compare results for different K value

`##compute_centers`

Input:

- *X* is the n x m matrix where each row represents a point.
- *labels* is the n x 1 column vector, representing current cluster assignments for each point.
- *K* is the number of cluster.

Output:

- *centers* is the k x m matrix of centroids.

```
# Compute centers for each cluster
compute_centers <- function(X, labels, K) {
  centers <- matrix(NA, nrow = K, ncol = ncol(X)) #initialize a NA matrix
  n <- nrow(X) # number of data points

  for (k in 1:K) {
    # Get the points in cluster k
    cluster_points <- X[labels == k, , drop = FALSE]
    if (nrow(cluster_points) > 0) {
      # Compute the mean for each feature (column)
      centers[k, ] <- colMeans(cluster_points, na.rm = TRUE)
    } else {
      # Reinitialize center for the empty cluster (randomly choose a point from the sample)
      centers[k, ] <- X[sample(1:n, 1), ]
    }
  }
  return(centers)
}
```

`##WCSS(Within-Cluster Sum of Squares)`

Input:

- *X* is the n x m matrix where each row represents a point.
- *labels* is the n x 1 column vector, representing current cluster assignments for each point.
- *centers* is the k x m matrix of centroids.

Output:

- *result* is the value of Within-Cluster Sum of Squares

```
# Function to compute Within-Cluster Sum of Squares,
#i.e. double the sum of squared distances from each point
#to the center of its assigned cluster
WCSS <- function(X, labels, centers) {
  n <- nrow(X) # number of data points
  squared_dist <- numeric(n) # to store result

  for (i in 1:n) {
    # Get the index of the assigned cluster for point i
    cluster_index <- labels[i]

    # Get the center of the assigned cluster
    center <- centers[cluster_index, ]

    # Compute the distance from point i to its assigned cluster center
    squared_dist[i] <- sum((X[i, ] - center)^2)
  }

  result <- 2 * sum(squared_dist)

  return(result)
}
```

##k\_means

Input:

- *X* is the  $n \times m$  matrix where each row represents a point.
- *K* is the number of cluster.
- *initial\_clusters* is the initial cluster assignments.
- *max\_iters* is maximum iterations allowed.
- *tol* is convergence tolerance.
- *optimal\_simu* is the number of simulation runs to find the best result.

Output:

- *centers* and *clusters*: final cluster assignments.
- *optimal\_centers* and *\$optimal\_clusters\$*: the final cluster assignments in different simulations.
- *list\_cluster* and *list\_centers*: the each cluster assignments of k-means process.
- *Within - Cluster\_Sum\_ofSquares*: the WCSS of each cluster assignments in k-means process.

```
library(animation)
```

```
## Warning: package 'animation' was built under R version 4.4.2
```

```

k_means <- function(X, K, initial_clusters = NULL, max_iters = 100, tol = 1e-6,
                    optimal_simu = 100) {
  # Number of samples
  n <- nrow(X)
  if(n < K){
    stop("The number of cluster centers is greater than the number of data points.")
  }

  if(nrow(X) == 0 || ncol(X) == 0){
    stop("Input dataset is empty.")
  }

  if(any(is.na(X))){
    stop("Input dataset contains missing values.")
  }

  if(!is.numeric(X)){
    stop("Input dataset contains non-numeric values.")
  }

  total_wcss <- rep(NA, optimal_simu)
  total_labels <- list()
  total_centers <- list()

  #The initial cluster can affect the final result. Running the algorithm multiple times
#from different random initial configurations, we can select the optimal result, but this
#optimal result does not mean minimal.
  for (simulation in 1:optimal_simu) {
    total_labels[[simulation]] <- list()
    total_centers[[simulation]] <- list()

    # Initialize centers
    if (is.null(initial_clusters)) {
      # Randomly assign a cluster number to each observation
      initial_clusters <- sample(1:K, size = n, replace = TRUE)
      centers <- compute_centers(X, initial_clusters, K)
    } else {
      centers <- compute_centers(X, initial_clusters, K)
    }

    for (i in 1:max_iters) {
      # Compute distances from each point to the center of its cluster
      distances <- as.matrix(dist(rbind(X, centers)))[1:n, (n+1):(n+K), drop = FALSE]

      # Assign each point to the nearest center
      labels <- apply(distances, 1, which.min)

      #store the data of labels and center points
      total_labels[[simulation]][[i]] <- labels
      total_centers[[simulation]][[i]] <- centers

      #Recompute centers
      new_centers <- compute_centers(X, labels, K)
    }
  }
}

```

```

    # Check for convergence (if centers do not change much)
    if (sum((new_centers - centers)^2) < tol) {
      break
    }

    centers <- new_centers
    #new_wcss <- WCSS(X, labels, centers)
  }

  total_wcss[simulation] <- WCSS(X, labels, centers)
}

index <- which.min(total_wcss)

list_cluster <- total_labels[[index]]
list_centers <- total_centers[[index]]

optimal_clusters <- lapply(total_labels, function(sublist) tail(sublist, 1)[[1]])
optimal_centers <- lapply(total_centers, function(sublist) tail(sublist, 1)[[1]])

clusters <- tail(total_labels[[index]], 1)[[1]]
centers <- tail(total_centers[[index]], 1)[[1]]

return(list(centers = centers, clusters = clusters,
            optimal_centers = optimal_centers, optimal_clusters = optimal_clusters,
            total_wcss = total_wcss,
            list_cluster = list_cluster, list_centers = list_centers,
            "Within-Cluster Sum of Squares" = total_wcss[index]))
}

```

##K\_means\_progresses

Input:

- *klist* is the clustering progress data from *k\_means* function.
- *X* is the *n* x *m* matrix where each row represents a point.

Output:

- a animated GIF

```

K_means_progresses <- function(klist, X){
  total_labels <- klist[["list_cluster"]]
  total_centers <- klist[["list_centers"]]
  library(animation)
  saveGIF({
    # Plot cluster assignments
    for (i in 1:length(total_labels)) {
      current_wcss <- round(WCSS(X, total_labels[[i]], total_centers[[i]]), 2)
      plot(X, col = total_labels[[i]], pch = 16, xlab = 'Feature 1', ylab = 'Feature 2',
           main = paste("Within-cluster variation is", current_wcss))
      legend("topright", legend = paste("Iterations:", i), bty = "n", cex = 1.2, col = "black")
    }
  })
}

```

```

    points(total_centers[[i]], col = 'red', pch = 3, cex = 2) # Plot centers
    Sys.sleep(0.5) # Pause for visualization
  }
}, movie.name = "kmeans_progress.gif")
}

```

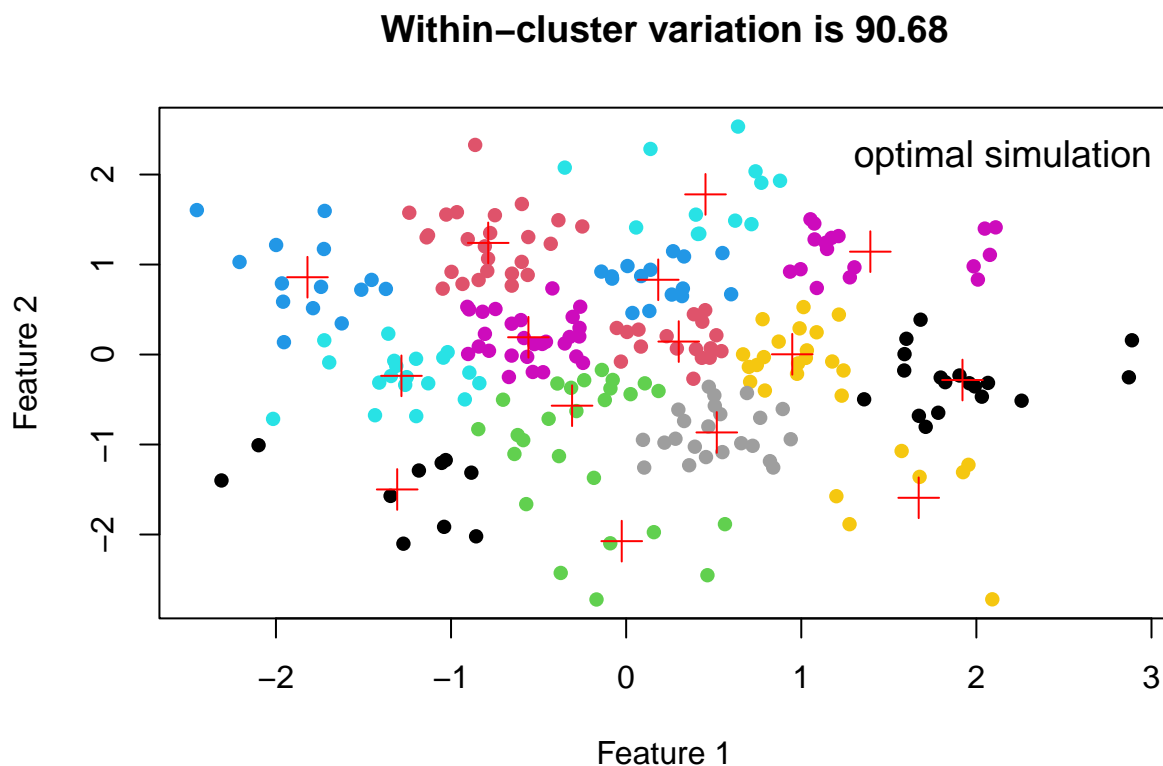
- plot a optimal result of k-means cluster

```

set.seed(2)
x <- matrix(rnorm(500), ncol = 2)

km.out <- k_means(X = x, K=15 , max_iters = 1000)
par(mfcol = c(1, 1))
plot(x, col = km.out[["clusters"]], pch = 16, xlab = 'Feature 1', ylab = 'Feature 2',
     main = paste("Within-cluster variation is", round(km.out$`Within-Cluster Sum of Squares`, 2)))
legend("topright", legend = paste("optimal simulation"), bty = "n", cex = 1.2, col = "black")
points(km.out[["centers"]], col = 'red', pch = 3, cex = 2) # Plot centers

```



- Create a animation plots to show the cluster centers and cluster assignment.

```

K_means_progresses(km.out, x)

```

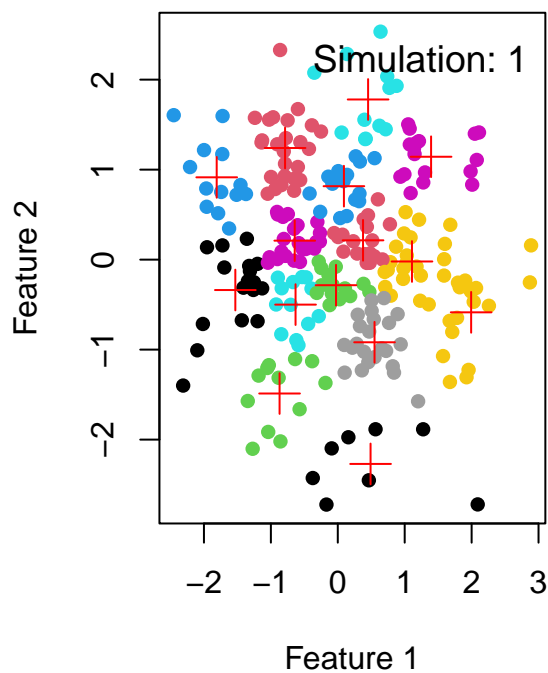
## Output at: kmeans\_progress.gif

```
## [1] TRUE
```

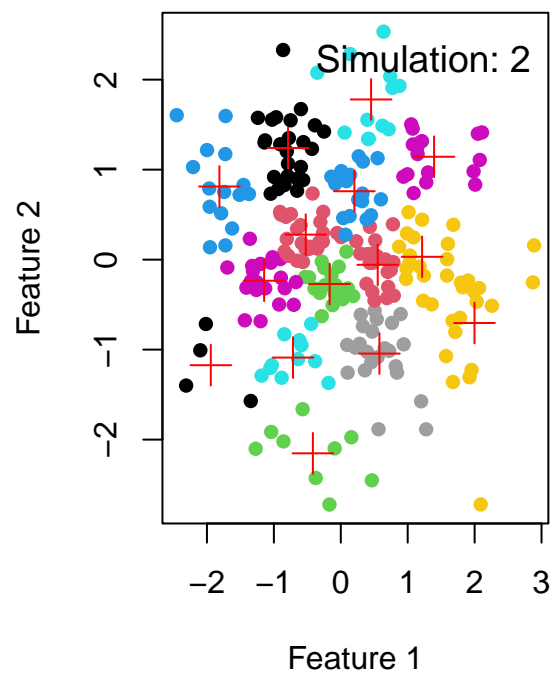
- Show the different results obtained from the four times simulation.
- Draw the histogram of WCSS obtained from 100 simulations. The WCSS's distribution is close to normal distribution because of CLT.

```
#Since the initial position affects the final result, multiple simulations are performed  
#to get the optimal result  
par(mfcol = c(1, 2))  
for (i in 1:4) {  
  current_wcss <- round(km.out[["total_wcss"]][[i]],2)  
  plot(x, col = km.out[["optimal_clusters"]][[i]], pch = 16, xlab = 'Feature 1', ylab = 'Feature 2',  
       main = paste("Within-cluster variation is", current_wcss))  
  legend("topright", legend = paste("Simulation:", i), bty = "n", cex = 1.2, col = "black")  
  points(km.out[["optimal_centers"]][[i]], col = 'red', pch = 3, cex = 2) # Plot centers  
}
```

**Within-cluster variation is 97.86**

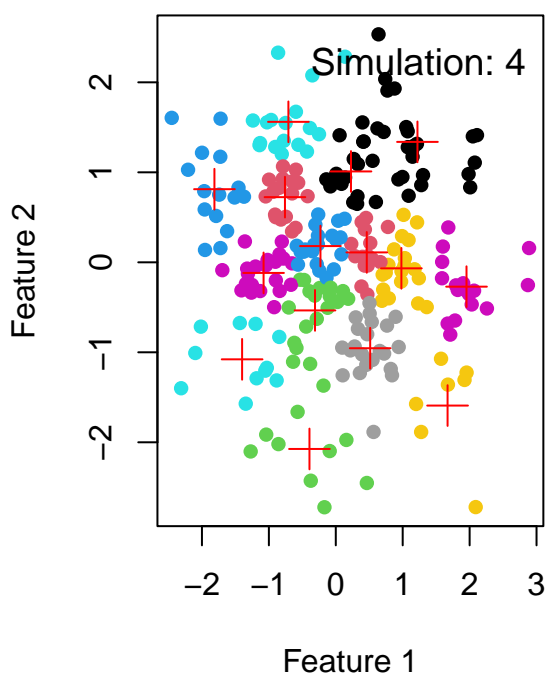
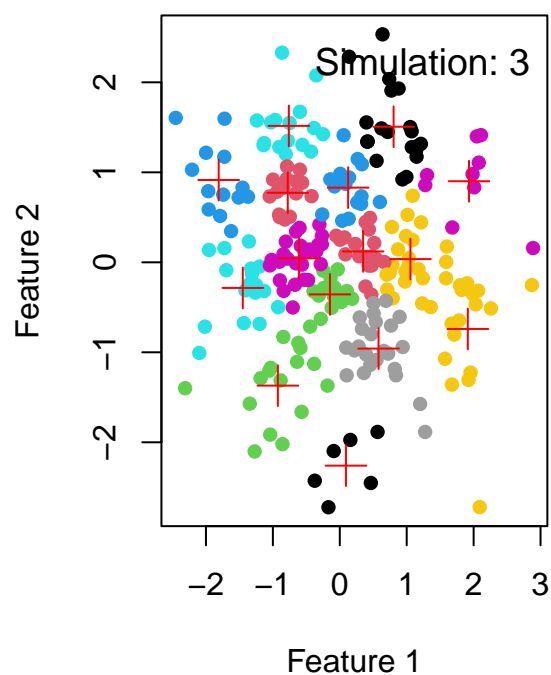


**Within-cluster variation is 97.63**



Within-cluster variation is 99.26

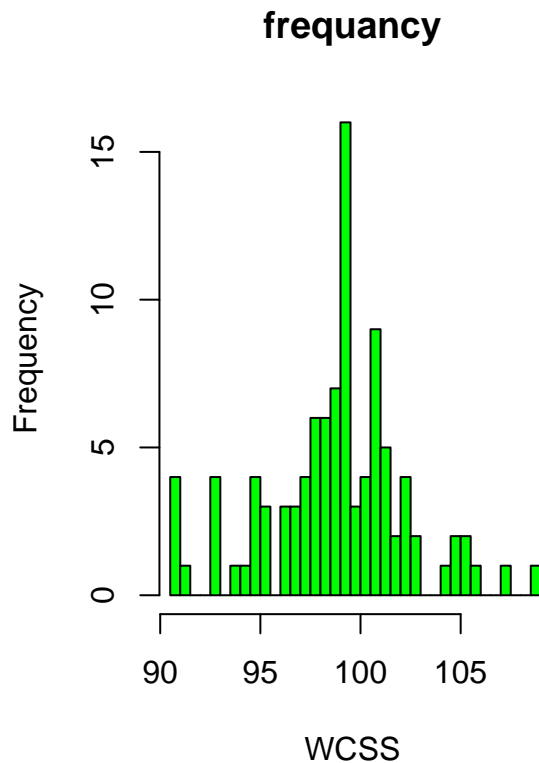
Within-cluster variation is 97.64



```
hist(km.out[["total_wcss"]], breaks = 50, col = "green", xlab = "WCSS",
     xlim = c(min(km.out[["total_wcss"]]), max(km.out[["total_wcss"]])),
     main = "frequancy", freq = TRUE)
summary(km.out[["total_wcss"]])
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  90.68   97.13   99.15   98.79  100.77  108.51
```





##Kmeans\_quality\_measure

- Taking a range of values of k and computing WCSS for each k value
- Plot the optimal WCSS results for each value of k
- Plot k value against WCSS

Input:

- $K_{range}$  is the range of values for K to test.
- $X$  is the  $n \times m$  matrix where each row represents a point.

Output:

- a plot of k value with WCSS

```
Kmeans_quality_measure <- function(K_range, X){
  m <- rep(NA, K_range) #to record the sum of WCK in difference K value
  par(mfcol = c(1, 3))
  for (a in 1:K_range) {
    result <- k_means(X, K = a)
    m[a] <- round(result$`Within-Cluster Sum of Squares`, 2)

    plot(X, col = result[["clusters"]], pch = 16, xlab = 'Feature 1', ylab = 'Feature 2',
```

```

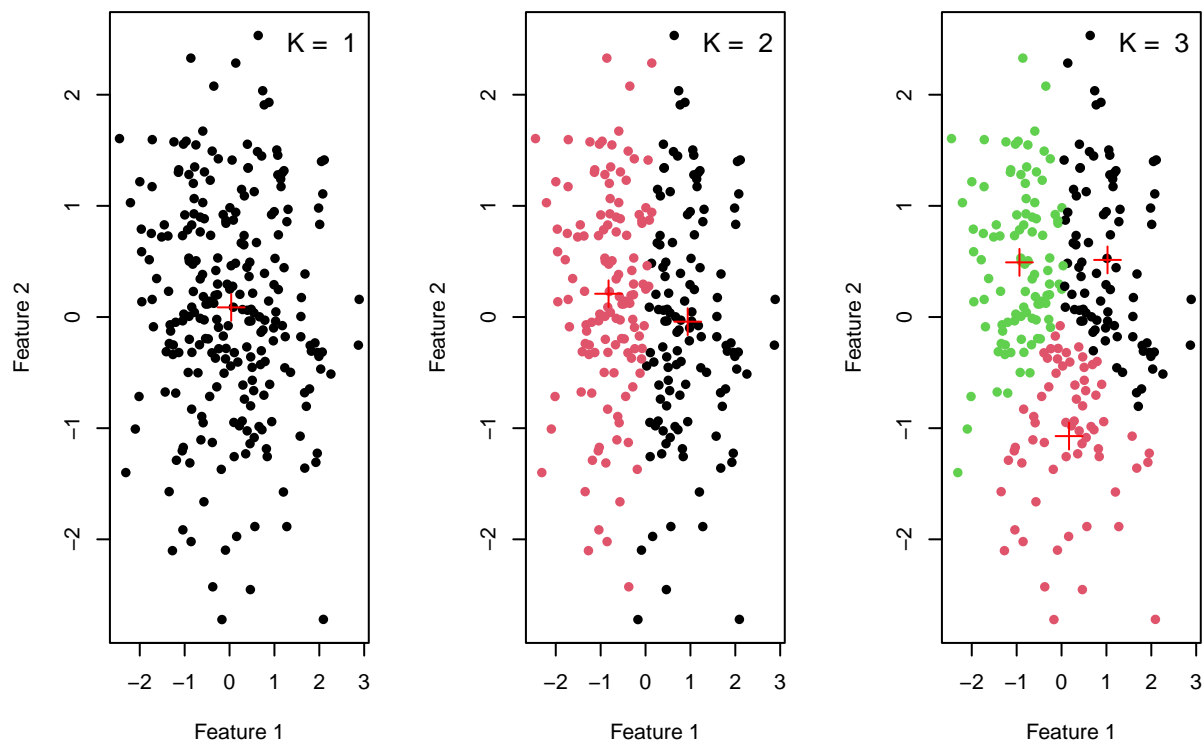
    main = paste("Within-cluster variation is", round(result$`Within-Cluster Sum of Squares`, 2))
    legend("topright", legend = paste("K = ", a), bty = "n", cex = 1.2, col = "black")
    points(result[["centers"]], col = 'red', pch = 3, cex = 2) # Plot centers
  }

  k_value <- 1:K_range
  par(mfcol = c(1, 1))
  plot(k_value, m, type = "b", col = "blue", pch = 16, xlab = "value of K",
       ylab = "Within-Cluster Sum of Squares")
  return(m)
}

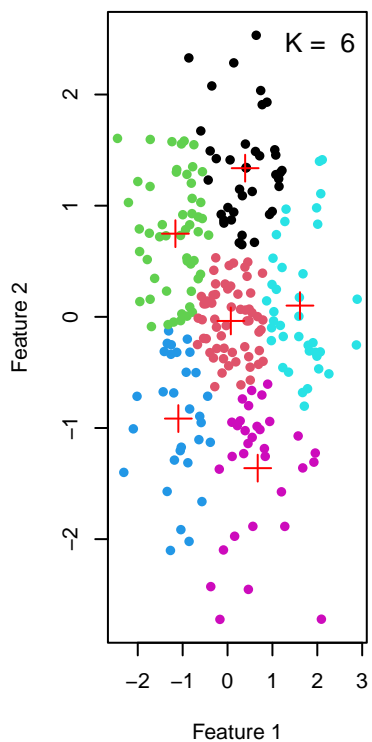
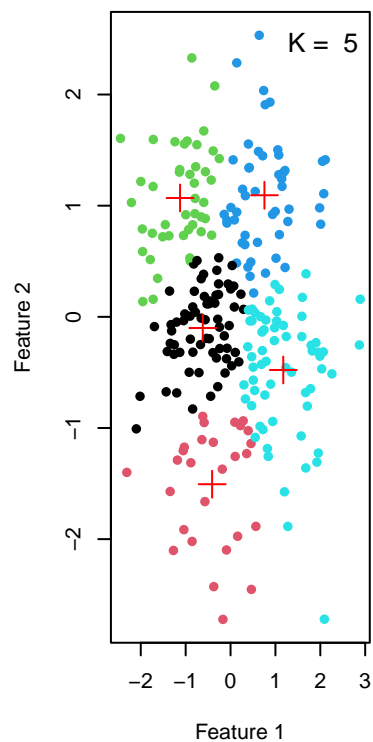
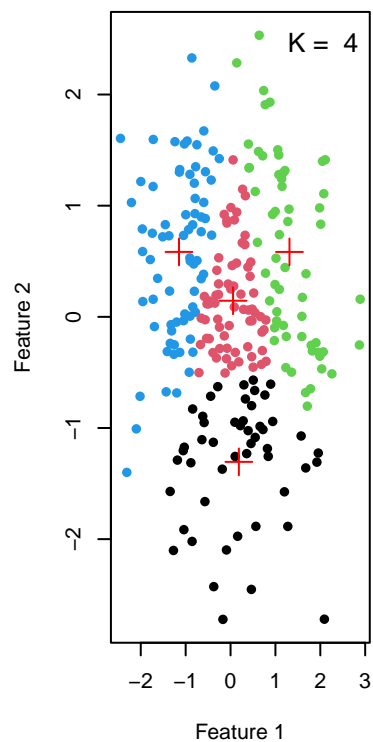
Kmeans_quality_measure(9, x)

```

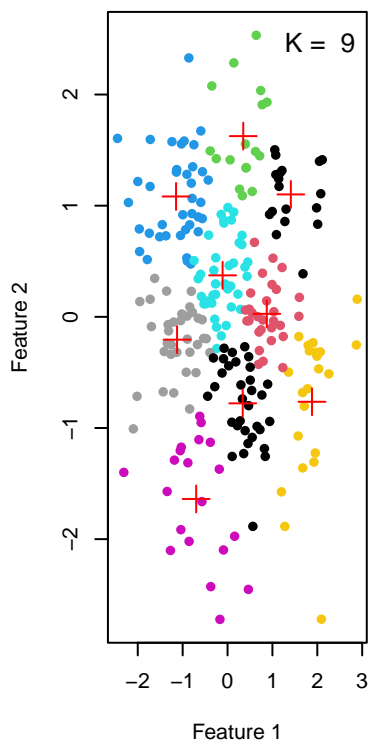
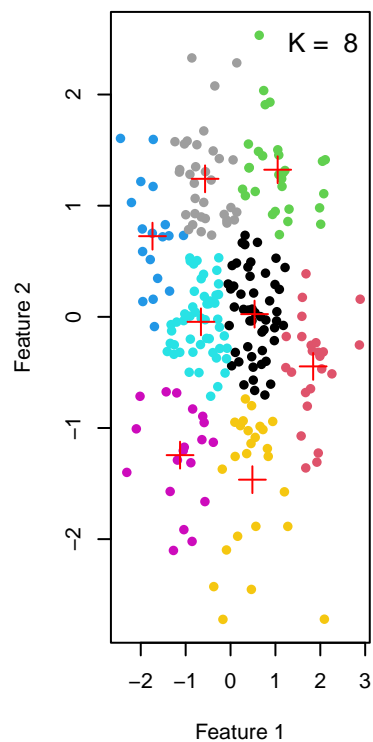
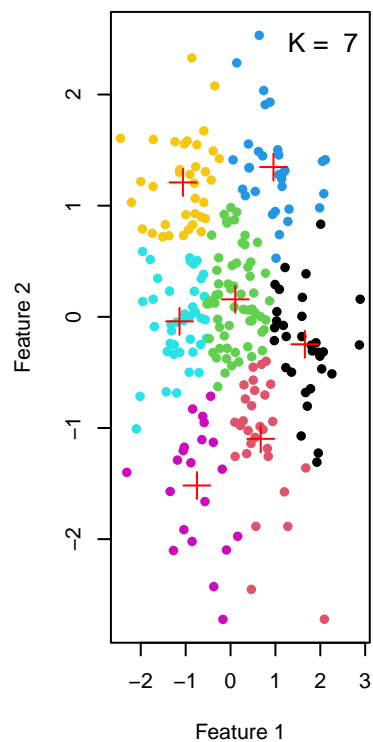
Within-cluster variation is 1063. Within-cluster variation is 666. Within-cluster variation is 466.

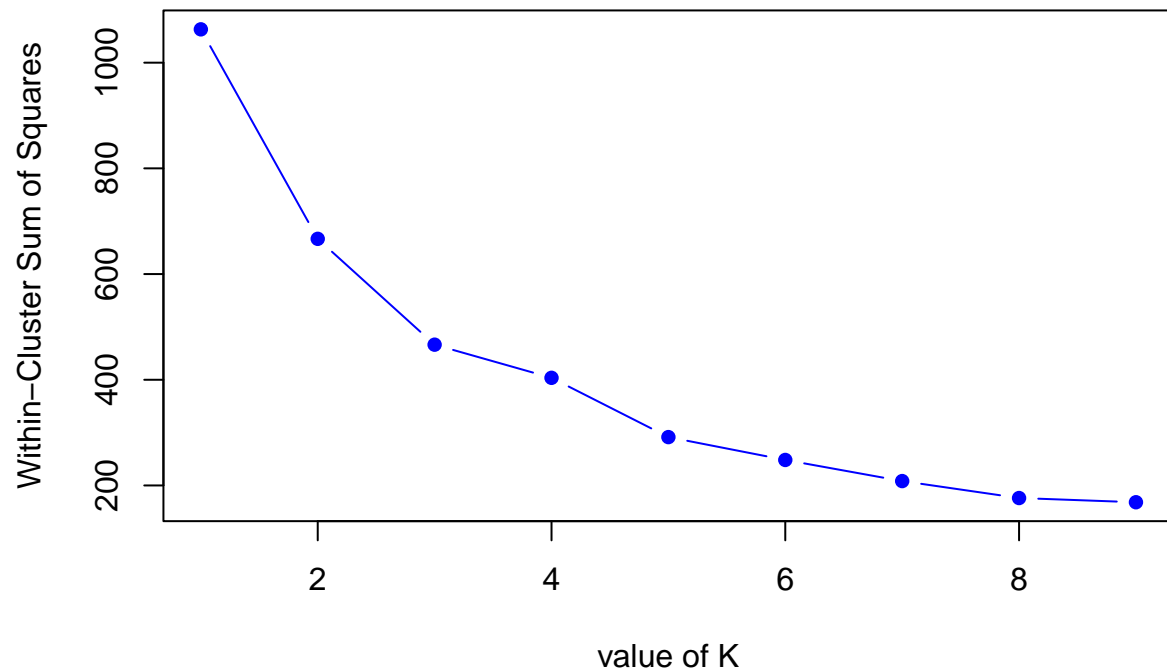


Within-cluster variation is 403. Within-cluster variation is 291. Within-cluster variation is 248.



Within-cluster variation is 208. Within-cluster variation is 176. Within-cluster variation is 168.





```
## [1] 1063.04 666.64 466.37 403.66 291.49 248.26 208.28 176.22 168.24
```

conclusion: As the k value increases, WCSS becomes smaller and smaller.

```
##Test
```

```
#getwd()
library(testthat)
test_dir('.')
```

```
## v | F W S OK | Context
## / |      0 | kmeans
```

```
## \ |      18 | kmeans
## v |      18 | kmeans [10.1s]
##
```

```
## == Results =====
## Duration: 10.1 s
##
## [ FAIL 0 | WARN 0 | SKIP 0 | PASS 18 ]
```