# STAT 302 Fall 2024 - Final Project

Casey Logan

2024-12-13

# K means clustering

In this project, an unsupervised learning technique called **k-means clustering** will be implemented. This method is used to partition a dataset into $k$ distinct clusters based on the similarity of the data points. The goal of k-means is to minimize the variance within each cluster, so that points within the same cluster are as similar as possible.

The k-means algorithm works as follows:

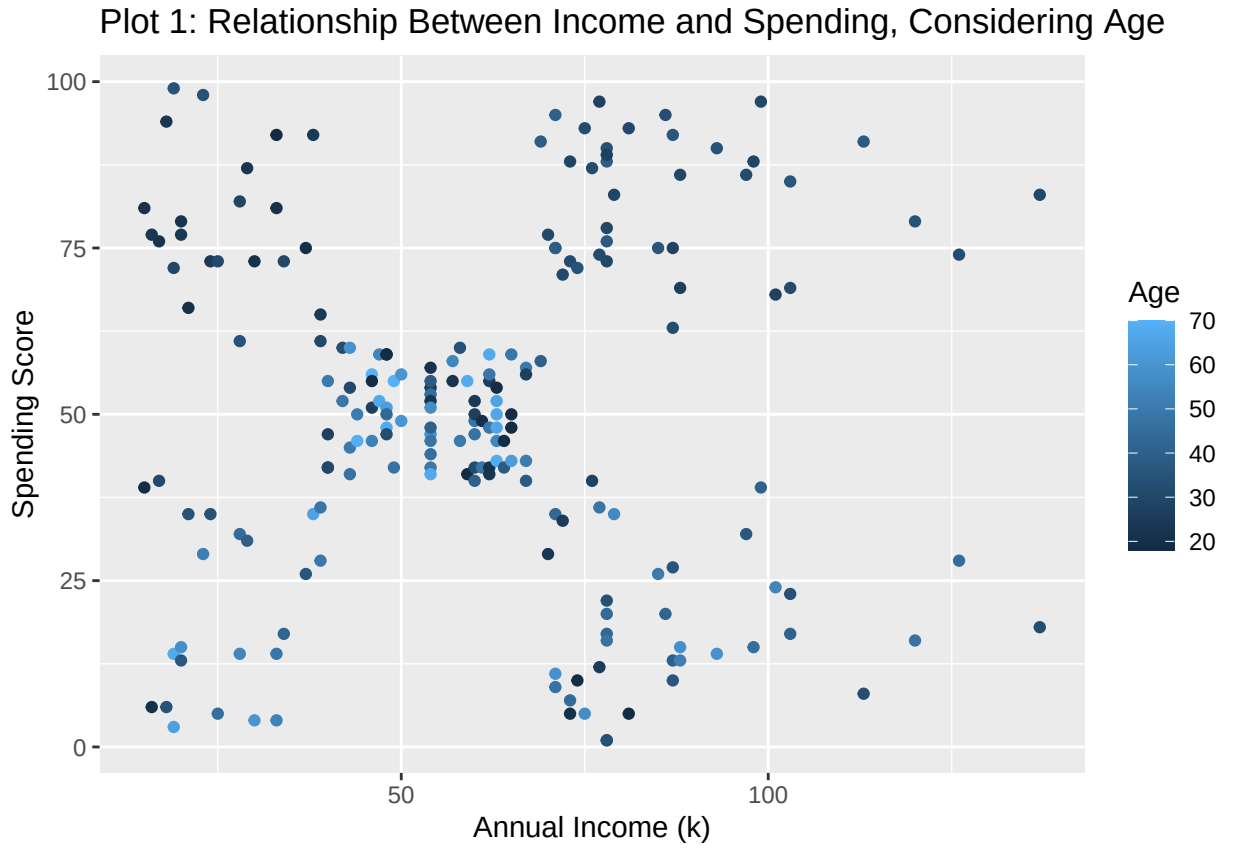
- Choose $k$ initial centroids (typically randomly selected points from the dataset).
- Assign each data point to the closest centroid, forming KK clusters.
- Recompute the centroids of the clusters by calculating the mean of all points assigned to each cluster.
- Repeat the assignment and update steps until the centroids no longer change (or change very little), indicating convergence.

# Exploring the data

The data set we will use to implement K means clustering is `"Mall_Customers.csv"`. This data set contains information regarding the gender, age, spending habits (represented as a score), and annual income of 200 people. For this algorithm, we will only be using age, annual income, and spending score to create the clusters.

```
# Import the data

mall_customers <- read.csv("Mall_Customers.csv")
colnames(mall_customers)[4] <- "Annual_Income"
colnames(mall_customers)[5] <- "Spending_Score"
```

```
# Exploring the data set

library(ggplot2)

ggplot(data=mall_customers, mapping=aes(x=Annual_Income, y=Spending_Score, color=Age)) +
  geom_point() +
  labs(x="Annual Income (k)",
       y="Spending Score",
       title="Plot 1: Relationship Between Income and Spending, Considering Age")
```

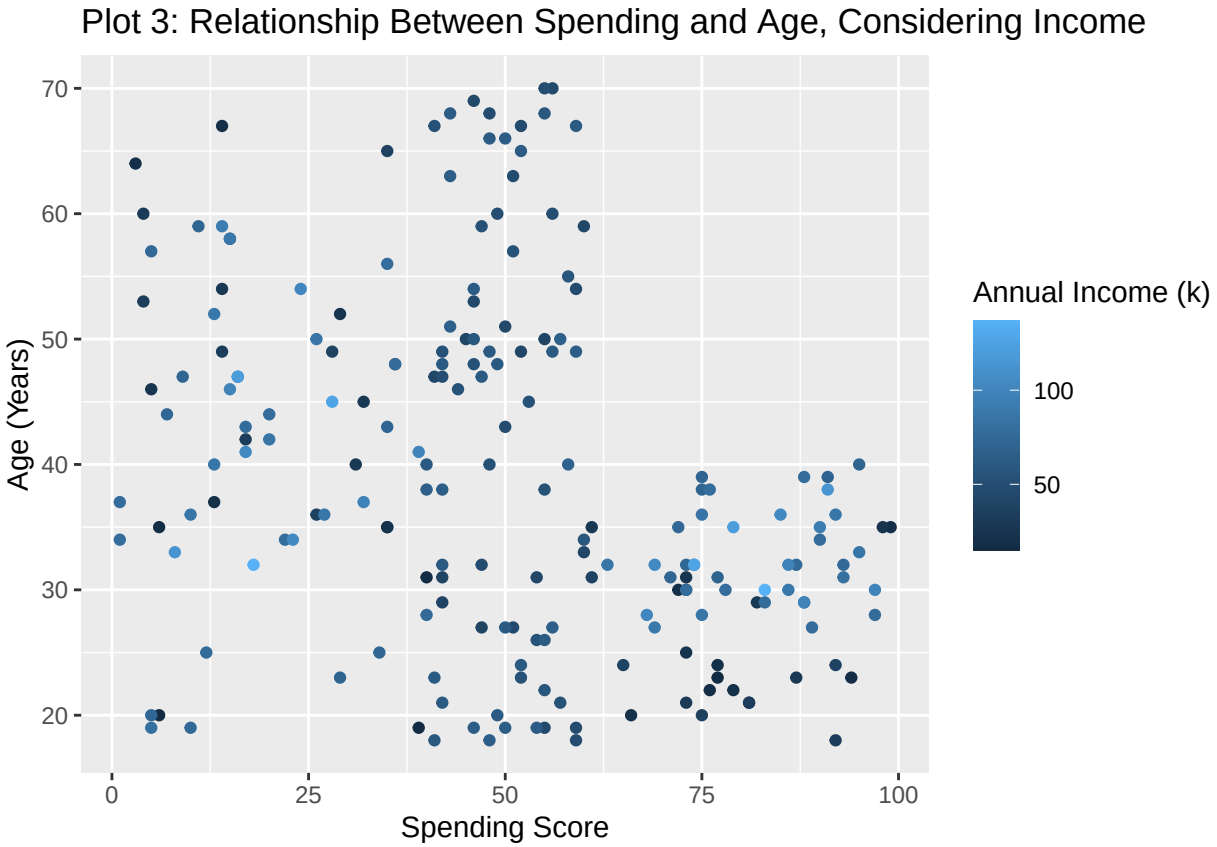



This first plot showing Income vs Spending shows that there is likely a relationship between the amount that people spend and their income. The plot shows an x pattern around the graph, grouped by extremes - low income with either particularly low spending score or particularly high spending score, high income with a similar pattern, and finally a group in the middle - moderate incomes and moderate spending scores. There seems to be a fair amount of variability in terms of age, however larger spending scores seem to be mostly associated with younger ages.

```
ggplot(data=mall_customers, mapping=aes(x=Annual_Income, y=Age, color=Spending_Score)) +
  geom_point() +
  labs(x="Annual Income (k)",
       y="Age (Years)",
       color="Spending Score",
       title="Plot 2: Relationship Between Income and Age, Considering Spending")
```


This second plot showing Income vs. Age does not provide much information about a relationship between income and age, as there is quite a bit of spread throughout the plot. The spread does not extend to the top right of the graph, implying that people with higher incomes do not have higher ages, and in fact there is a slight decrease in the maximum income as age increases. Additionally, there seems to be a potential trend where people with a younger age and lower income spend more, and a bit more generally people with a younger age seem to spend more

```
ggplot(data=mall_customers, mapping=aes(x=Spending_Score, y=Age, color=Annual_Income)) +
  geom_point() +
  labs(x="Spending Score",
       y="Age (Years)",
       color="Annual Income (k)",
       title="Plot 3: Relationship Between Spending and Age, Considering Income")
```




The trend described above is partially supported by this third plot showing Spending Score vs. Age. This plot shows spending scores greater than 65 are only found among people younger than 40. The plot also shows a large amount of people, and people of many ages, with spending scores centered in the middle around 50. This group appears to have generally lower incomes.

## Implementing the algorithm

```
# Functions

find_closest_centroid <- function(centroids, k) {
  # Convert the data to a matrix for ease of future manipulation
  matrix_data <- as.matrix(standard_mall)
  matrix_centroids <- as.matrix(centroids)

  # Calculate euclidean distance of each point from each centroid
  # Assign the point to the centroid with the smallest distance
  distances <- sapply(1:k,
                      FUN=function(k) {sqrt(rowSums((sweep(standard_mall,
                                            MARGIN=2,
                                            matrix_centroids[k, ]))^2))})
  return(apply(distances,
               MARGIN=1,
               FUN=which.min))
}


update_centroids <- function(cluster_assignment, current_centroids, k) {
  # Recalculate each centroid based on assigned clusters
  for (cluster in 1:k) {
    # Find all the points assigned to the centroid
    points <- standard_mall[cluster_assignment == cluster, ]

    # Calculate column means to create new centroid
    current_centroids[cluster, ] <- colMeans(points)
  }
  return(centroids)
}


converge_centroids <- function(centroids, k) {
  # Assign points to clusters and update centroids until centroids no longer change
  new_centroids <- centroids + 1
  while (sum(new_centroids != centroids) != 0) {
    # Assign points to centroid and calculate updated centroid
    assigned_groups <- find_closest_centroid(centroids, k=k)
    new_centroids <- update_centroids(assigned_groups, centroids, k)
    centroids <- new_centroids
  }
  return(centroids)
}


calculate_wss <- function(centroids, k) {
  final_clusters <- find_closest_centroid(centroids, k=k)
  total_sum <- 0
  for (cluster in 1:k) {
    # Find points for this cluster
    points <- standard_mall[final_clusters == cluster,
```

```
                            c("Age","Annual_Income", "Spending_Score")]

    total_sum <- total_sum + sum(rowSums( (sweep(points,
                                                 MARGIN=2,
                                                 colMeans(points))^2) ))
  }
  return(total_sum)
}
```

The functions above are used to implement a k-means clustering algorithm.

1. `find_closest_centroid` is used to assign points to the closest centroid to them, in terms of Euclidean distance.
2. `update_centroids` is used to re-calculate the centroid of a cluster by calculating the mean of its points.
3. `converge_centroids` is used to repeat the process of assigning points to clusters and re-calculating centroids until the values of the centroids no longer change.
4. `calculate_wss` is used to calculate the Within-Cluster-Sum of Squared Errors (WSS) for a specified number of clusters, k. The formula we are using to calculate the WSS:

$$\text{WSS} = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \left(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_k\right)^\top \left(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_k\right)$$

## Completing the Algorithm with 3 clusters

```
# Standardizing the data for better analysis
standard_mall <- data.frame(CustomerID=mall_customers[, "CustomerID"],
                            Age=NA,
                            Annual_Income=NA,
                            Spending_Score=NA)

for (col in c("Age", "Annual_Income", "Spending_Score")) {
  min <- min(mall_customers[, col])
  max <- max(mall_customers[, col])
  standard_mall[, col] <- (mall_customers[, col] - min) / (max - min)
}

standard_mall <- standard_mall[, c("Age", "Annual_Income", "Spending_Score")]


k <- 3

centroid_points <- sample(x=nrow(standard_mall), size=k)
centroids <- rbind(matrix(ncol=3, nrow=0), as.matrix(standard_mall[centroid_points, ]))
k_centroids <- converge_centroids(centroids, k)
wss <- calculate_wss(k_centroids, k)
wss
```

```
## [1] 26.55656
```

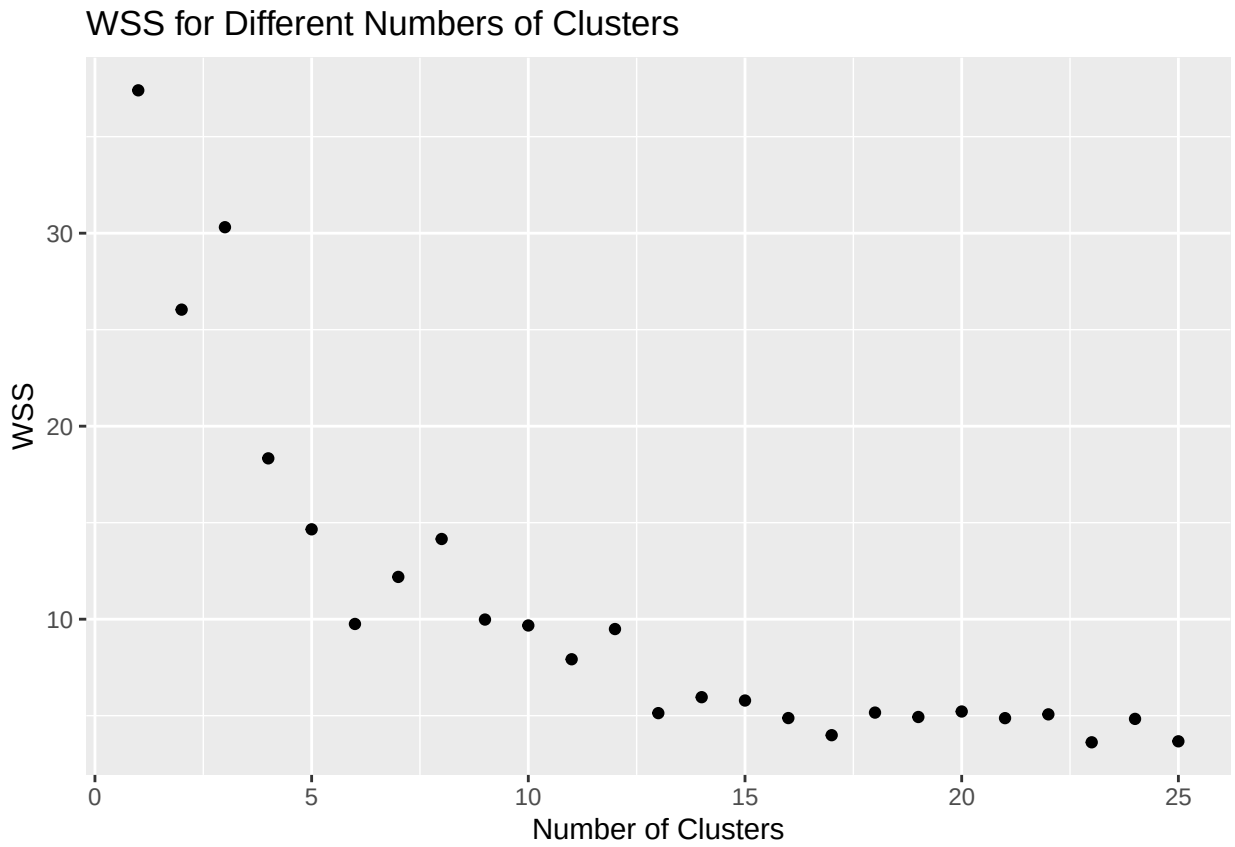
The WSS when $k = 3$ is approximately 26.5566. This value is used as a measurement to quantify how far points are from their respective centroids when we have 3 clusters in the dataset.

## Completing the algorithm with k clusters

```
# Standardizing the data for better analysis
standard_mall <- data.frame(CustomerID=mall_customers[, "CustomerID"],
                            Age=NA,
                            Annual_Income=NA,
                            Spending_Score=NA)

for (col in c("Age", "Annual_Income", "Spending_Score")) {
  min <- min(mall_customers[, col])
  max <- max(mall_customers[, col])
  standard_mall[, col] <- (mall_customers[, col] - min) / (max - min)
}

standard_mall <- standard_mall[, c("Age", "Annual_Income", "Spending_Score")]


k_wss <- numeric(length=25)
for (k in 1:25) {
  centroid_points <- sample(x=nrow(standard_mall), size=k)
  centroids <- rbind(matrix(ncol=3, nrow=0), as.matrix(standard_mall[centroid_points, ]))
  k_centroids <- converge_centroids(centroids, k)
  k_wss[k] <- calculate_wss(k_centroids, k)
}
```

The above process is used to find the Within-Cluster-Sum of Squared Errors (WSS) for values of k between 1 and 25. This information will be used to determine the most appropriate number of clusters to calculate, where the WSS is minimized but the clusters are not over fit:

```
library(ggplot2)
wss_data <- data.frame(k=1:25, wss=k_wss)
ggplot(wss_data, mapping=aes(x=k, y=wss)) +
  geom_point() +
  labs(x="Number of Clusters",
       y="WSS",
       title="WSS for Different Numbers of Clusters")
```




The concern with using too few clusters is that data points that do not share similarities will be grouped into the same cluster simply due to lack of enough clusters. For example, if you only use 1 cluster you will not be able to gain meaningful insights since all points are in the same cluster. This concern is shown as a high WSS as distances between points and the centroids are very large.

The concern with using too many clusters is that you will not be able to gain any meaningful information from cluster characteristics, since clusters will only be composed of data points that are close to identical. If, for example, you have 100 data points and 100 clusters, then each point will be in its own cluster. In this case you cannot discover anything new that you did not already know before.

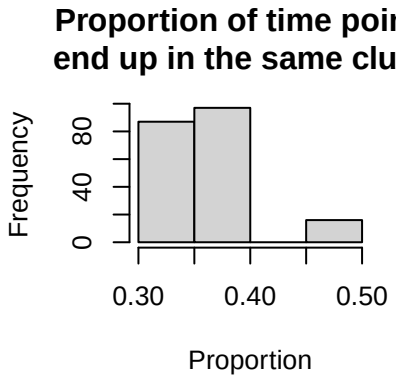
The plot above shows that for 1-4 clusters, the WSS is significantly decreasing as the number of clusters increases. For 8-25 clusters, the WSS is still decreasing, however it is decreasing by smaller amounts each time. This may imply that we are over-fitting the clusters. For 5-7 clusters, you see an in between. The WSS is still decreasing by a significant amount, but this decrease is starting to slow down. For the purposes of the rest of this project, we will use 5 clusters, as it represents an appropriate middle ground between a low enough WSS, without being over fit.

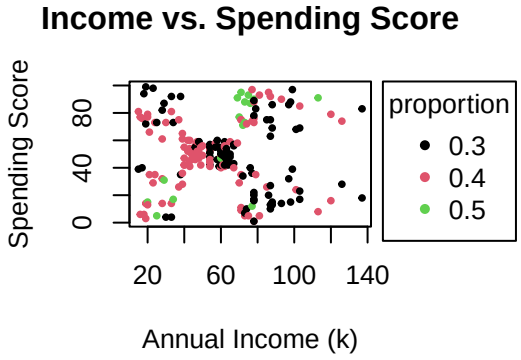
## Comparing seeds as starting points

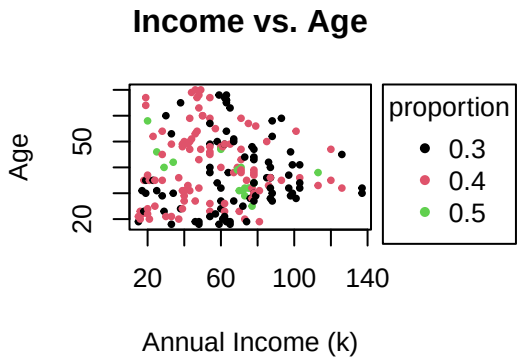
```
seeds <- c(0823, 1202, 1213, 0428, 1129, 0106, 1013, 0108, 0111, 0401)
k <- 5

seed_point_assignment <- data.frame(point=1:200)

for (i in 1:10) {
  seed <- seeds[i]
  set.seed(seed)

  centroid_points <- sample(x=nrow(standard_mall), size=k)
  centroids <- rbind(matrix(ncol=3, nrow=0), as.matrix(standard_mall[centroid_points, ]))

  current_centroids <- converge_centroids(centroids, k)
  point_assignment <- find_closest_centroid(current_centroids, k)

  seed_point_assignment[[as.character(seed)]] <- point_assignment
}

# Calculate the proportion of times a point ends up in the same cluster
mall_customers$proportion <- apply(seed_point_assignment,
                                   MARGIN=1,
                                   FUN=function(row)
                                    {sum(row == as.numeric(names(sort(-table(row)))[1]))/ 10
})
```

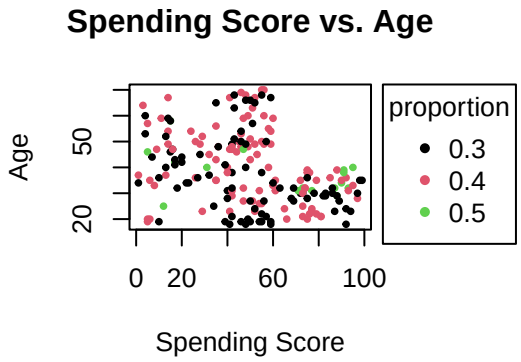
```
# Plots

par(mfrow=c(2, 2), mar = c(5, 4, 4, 8), xpd = TRUE)

hist(mall_customers$proportion, breaks=3,
     xlab="Proportion",
     main="Proportion of time points \n end up in the same cluster")

plot(x=mall_customers$Annual_Income,
     y=mall_customers$Spending_Score,
     col=as.factor(mall_customers$proportion),
     cex=0.7,
     pch=20,
     xlab="Annual Income (k)",
     ylab="Spending Score",
     main="Income vs. Spending Score")

legend(x = "topright",
       inset = c(-0.6, 0),
       legend = levels(as.factor(mall_customers$proportion)),
       col = unique(as.factor(mall_customers$proportion)),
       pch=20,
       title="proportion")

plot(x=mall_customers$Annual_Income,
     y=mall_customers$Age,
```

```
     col=as.factor(mall_customers$proportion),
     pch=20,
     cex=0.7,
     xlab="Annual Income (k)",
     ylab="Age",
     main="Income vs. Age")

legend(x = "topright",
       inset = c(-0.6, 0),
       legend = levels(as.factor(mall_customers$proportion)),
       col = unique(as.factor(mall_customers$proportion)),
       pch=20,
       title="proportion")

plot(x=mall_customers$Spending_Score,
     y=mall_customers$Age,
     col=as.factor(mall_customers$proportion),
     pch=20,
     cex=0.7,
     xlab="Spending Score",
     ylab="Age",
     main="Spending Score vs. Age")

legend(x = "topright",
       inset = c(-0.6, 0),
       legend = levels(as.factor(mall_customers$proportion)),
       col = unique(as.factor(mall_customers$proportion)),
       pch=20,
       title="proportion")
```




The histogram above shows the proportion of times that points end up in the same cluster. Most points end up in the same cluster about $30-40\%$ of the time, with few points ending up in the same cluster about $50\%$ of the time. The average amount of times that a point ends up in the same cluster is 0.3645. If this value was 1, then every point would end up in the same cluster, no matter the starting centroids. If this value is 0, then points never end up in the same cluster, no matter the starting centroids. With an average of 0.3645, points only end up in the same cluster about $\frac{1}{3}$ of the time.

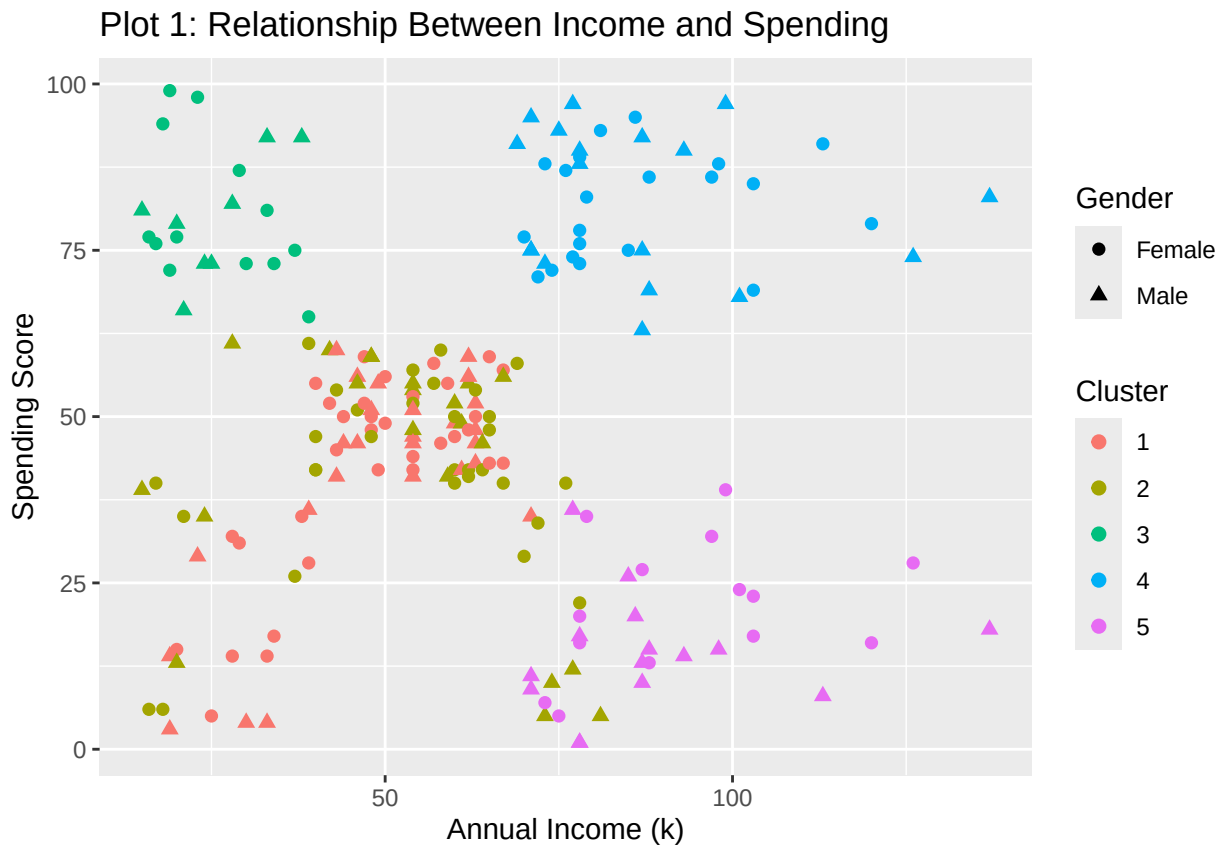
One big reason for this could be because the naming of the clusters may be different based on starting points. For example, consider cluster 1 with the seed 123 containing 5 points. Using seed 456, there is a cluster containing those same five points, but is instead named cluster 5. In this case, it is difficult to determine if points are ending up in the same cluster due to issues with naming convention.

Another potential reason could be that certain points lie directly in between two clusters, so they are likely to switch between clusters frequently for different seeds. To see if this is true, the above plots can be analyzed to see what kind of points have various proportions of ending up in the same cluster. Based on the above three scatter plots, there is no clear pattern. In the `Income vs. Spending Score` plot, only points with both low spending score and income, or high spending score and income, have a proportion of 0.5. This could mean that points matching these criteria are more likely to end up in the same cluster, however we can not say this for certain. Looking at the above histogram showing the proportion of times points end up in the same cluster, we can see that not many points have a proportion of 0.5. So while it is possible that points with these income and spending score characteristics are more likely to have a proportion of 0.5, we do not have enough information to say for certain.

Finally, another other potential reasons for why these proportions are not higher, would be that with a value of $k = 5$, 5 clusters is not enough to appropriately group the data. Since we calculated the WSS above and saw at 5 clusters, the decrease in WSS starts to diminish, we have some information to suggest that this is not the case. However, since we are dealing with random variability it is still possible.

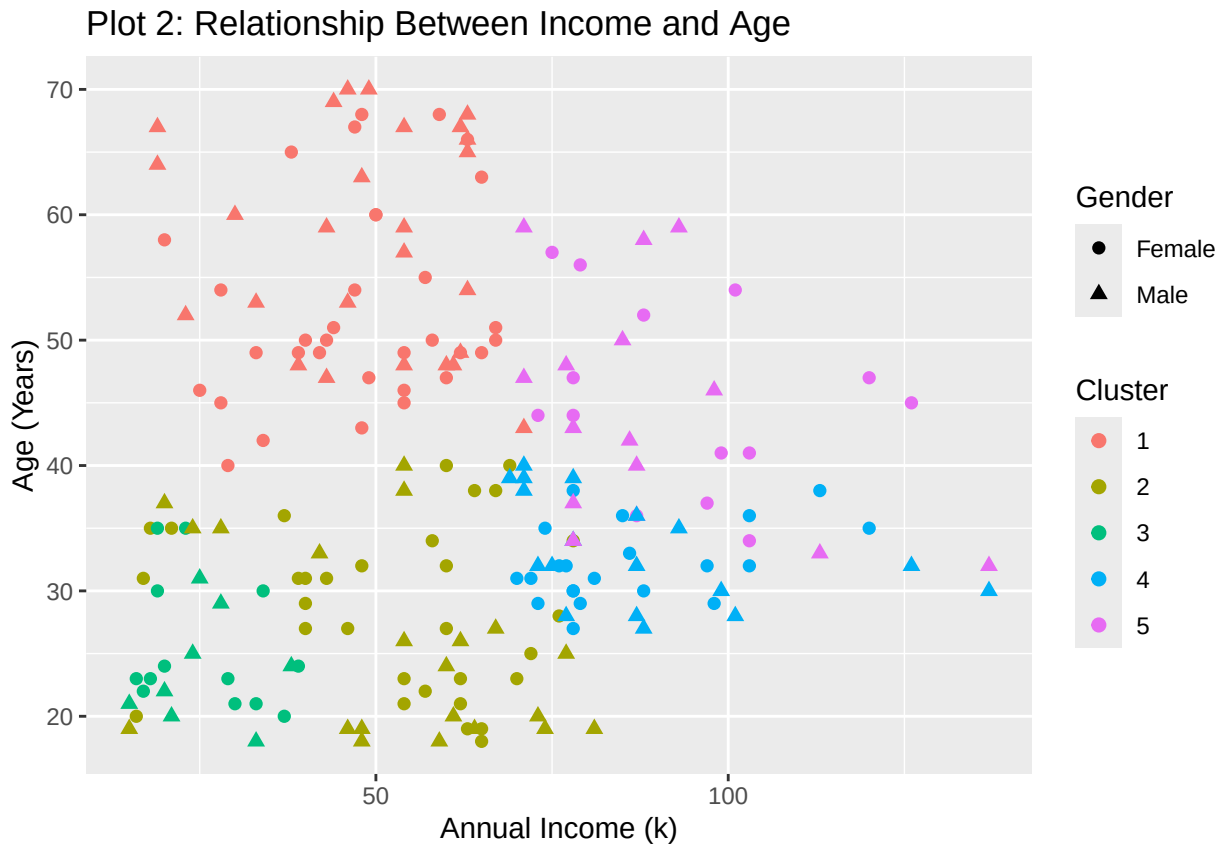
## Analyzing the final clusters

```
set.seed(0401)
k <- 5

centroid_points <- sample(x=nrow(standard_mall), size=k)
centroids <- rbind(matrix(ncol=3, nrow=0), as.matrix(standard_mall[centroid_points, ]))
final_centroids <- converge_centroids(centroids, k)
final_cluster_assignment <- find_closest_centroid(final_centroids, k)

mall_customers[, "cluster"] <- final_cluster_assignment

ggplot(data=mall_customers, mapping=aes(x=Annual_Income,
                                        y=Spending_Score,
                                        color=as.factor(cluster),
                                        shape=Gender)) +
  geom_point(size=2) +
  labs(x="Annual Income (k)",
       y="Spending Score",
       color="Cluster",
       title="Plot 1: Relationship Between Income and Spending")
```

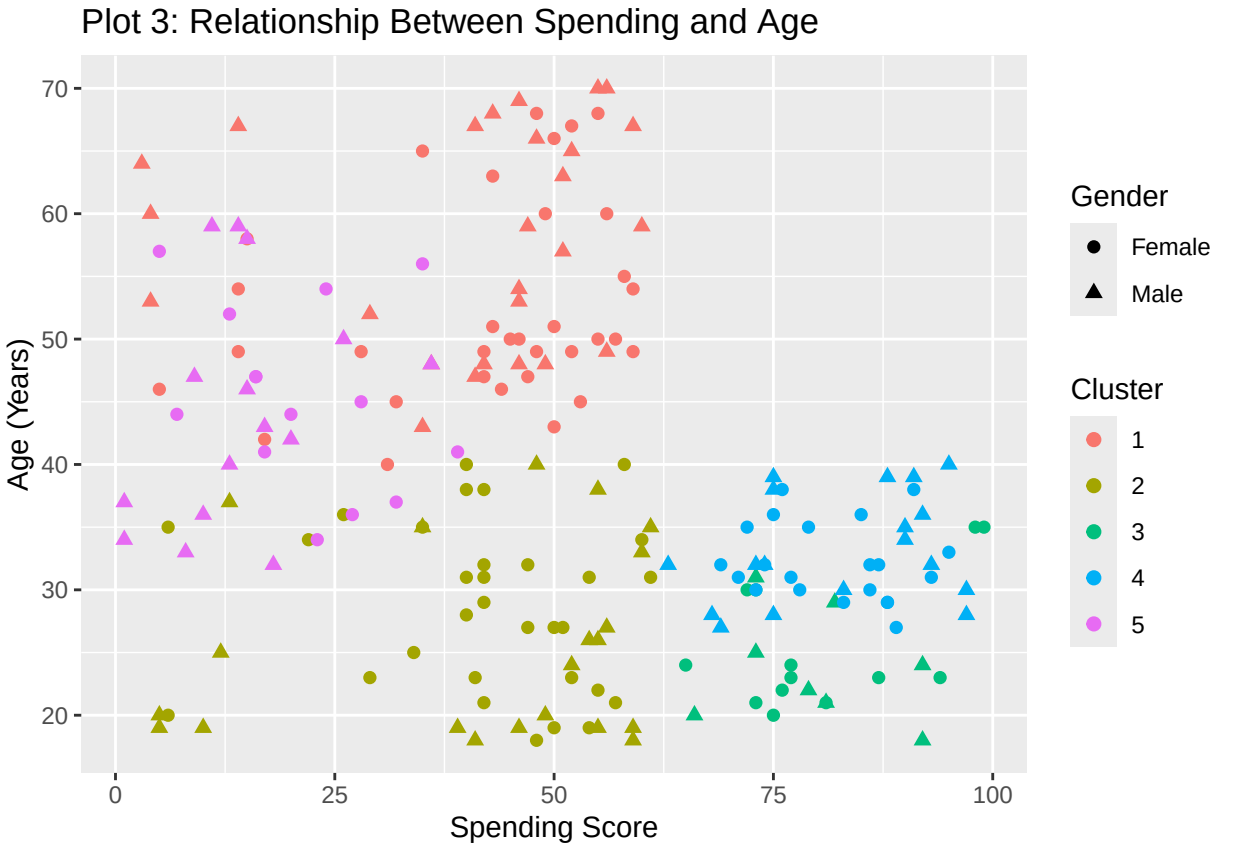



```
ggplot(data=mall_customers, mapping=aes(x=Annual_Income,
                                        y=Age,
```

```
                                      color=as.factor(cluster),
                                      shape=Gender)) +
  geom_point(size=2) +
  labs(x="Annual Income (k)",
       y="Age (Years)",
       color="Cluster",
       title="Plot 2: Relationship Between Income and Age")
```




```
ggplot(data=mall_customers, mapping=aes(x=Spending_Score,
                                        y=Age,
                                        color=as.factor(cluster),
                                        shape=Gender)) +
  geom_point(size=2) +
  labs(x="Spending Score",
       y="Age (Years)",
       color="Cluster",
       title="Plot 3: Relationship Between Spending and Age")
```




The above plots show that the data has some clear clusters.

- Cluster 1: Low-middle income, low-middle spending score, middle-high age.
- Cluster 2: Low-middle (mostly middle) income and spending score, young age.
- Cluster 3: Low income, high spending score, young age.
- Cluster 4: High income, high spending score, young age.
- Cluster 5: High income, low spending score, middle-high age (mostly middle).

Clusters 3 & 4 tend to be very distinct from the other clusters (as seen in plots 1 & 2), with slight overlap in terms of age and spending score. This can be seen in the third plot.

Clusters 1 & 5 tend to be mostly distinct from both each other and clusters 4 & 5. Cluster 2 provides a lot of overlap between clusters 1 & 5. These clusters (1, 2, & 5) appear to be the most distinct in terms of the relationship between income and age, as seen in plot 2.

A person with a high income and a high spending score is associated with young age. Alternatively, if they have a high income and *low* spending score, their expected age will be higher.

A person with a low income and high spending score is mostly associated with a young age. If they have a low income *and* low spending score, it becomes more difficult to tell what age they have due to the overlap of clusters 1 & 2.

Overall, spending scores in combination with income are useful in identifying clear clusters, with age providing another useful factor in distinguishing between clusters.

When identifying the composition of the clusters according to gender, which in this data set is categorized by `Female` or `Male`, there is not any clear differentiation. The distribution of both `Female` and `Male` data points throughout each plot is fairly even and appears to be unpredictable based on the information available.