Assignment 1 - DANA 4840

Manraj Singh

Winter 2025

Question 1: Steps in k-means Clustering

Answer:

K-means clustering involves the following steps:

1. Choose the number of clusters (k).
2. Initialize k cluster centroids randomly.
3. Assign each data point to the nearest centroid.
4. Compute new centroids as the mean of the points assigned to each cluster.
5. Repeat steps 3 and 4 until centroids no longer change or the maximum iterations are reached.

Question 2: Elbow Plot Explanation

Answer:

An elbow plot helps determine the optimal number of clusters in K-means clustering by plotting the within sum of squared distances against the number of clusters. The elbow point represents the ideal k where adding more clusters does not significantly reduce inertia.

Question 3: K-means Clustering on hcvdat0.csv

(a) Handling Missing Values

We will first identify columns with missing values and visualize their distribution using histograms before handling them.

```
library(tidyverse)
```

```
## Warning: package 'purrr' was built under R version 4.4.1
```

```
## Warning: package 'lubridate' was built under R version 4.4.1
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 4.4.1
```

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(ggplot2)

set.seed(123)

setwd('/Users/manraj/Downloads/Assignment 1 attached files Feb 12, 2025 233 PM/')

df <- read.csv("hcvdat0.csv")

head(df)
```

```
##   PatientID      Category Age Sex  ALB  ALP  ALT  AST  BIL   CHE CHOL CREA  GGT
## 1         1 0=Blood Donor  32   m 38.5 52.5  7.7 22.1  7.5  6.93 3.23  106 12.1
## 2         2 0=Blood Donor  32   m 38.5 70.3 18.0 24.7  3.9 11.17 4.80   74 15.6
## 3         3 0=Blood Donor  32   m 46.9 74.7 36.2 52.6  6.1  8.84 5.20   86 33.2
## 4         4 0=Blood Donor  32   m 43.2 52.0 30.6 22.6 18.9  7.33 4.74   80 33.8
## 5         5 0=Blood Donor  32   m 39.2 74.1 32.6 24.8  9.6  9.15 4.32   76 29.9
## 6         6 0=Blood Donor  32   m 41.6 43.3 18.5 19.7 12.3  9.92 6.05  111 91.0
##   PROT
## 1 69.0
## 2 76.5
## 3 79.3
## 4 75.7
## 5 68.7
## 6 74.0
```

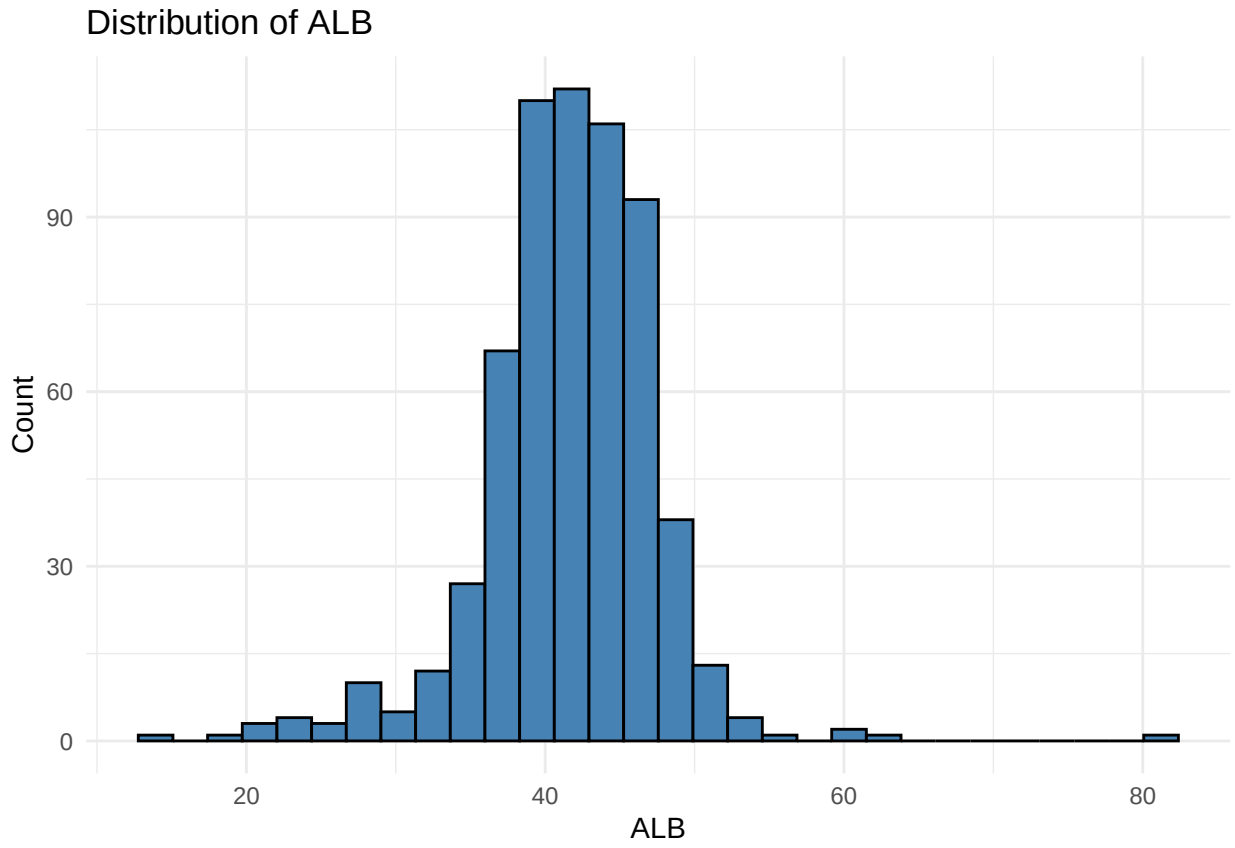
```
missing_counts <- colSums(is.na(df))
missing_cols <- names(missing_counts[missing_counts > 0])
print(missing_counts[missing_counts > 0])
```

```
##  ALB  ALP  ALT CHOL PROT
##    1   18    1   10    1
```

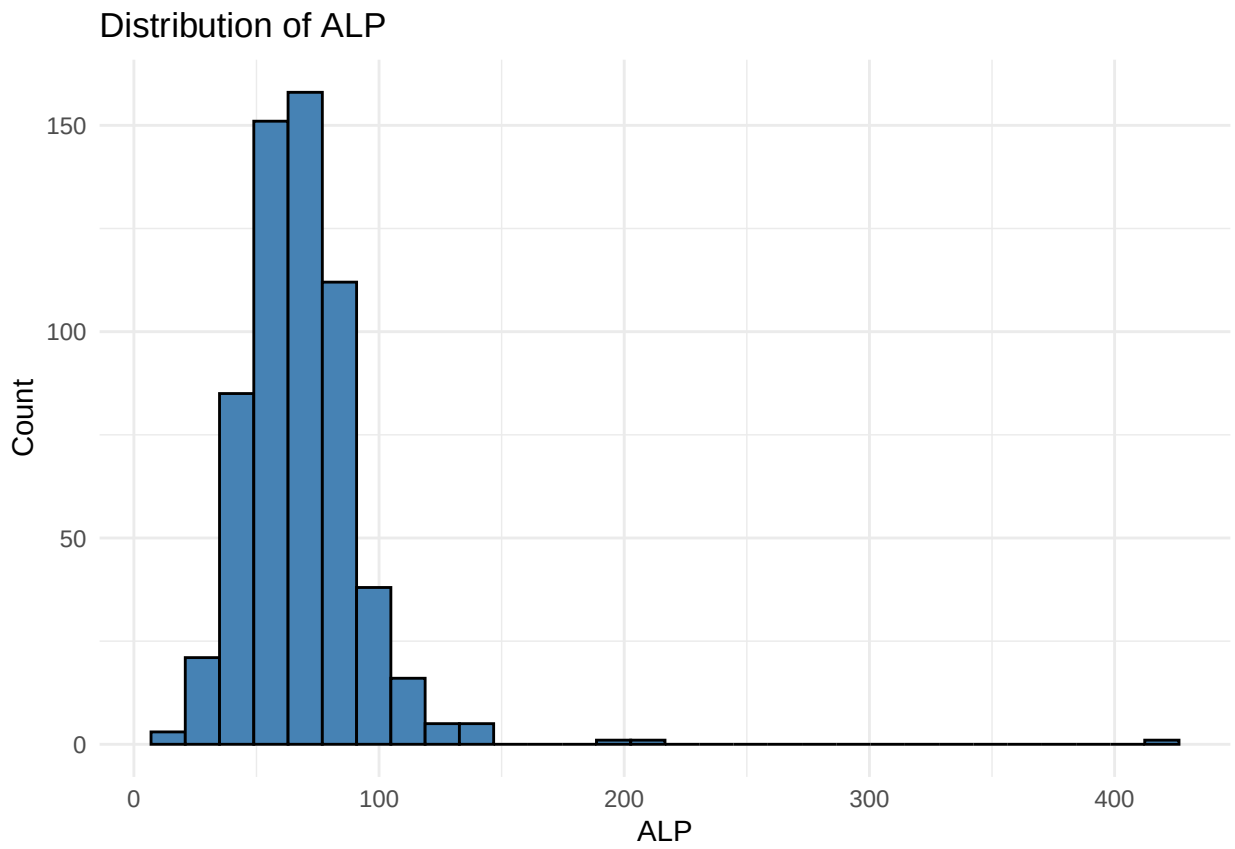
```r
for (col in missing_cols) {
  p <- ggplot(df, aes_string(x = col)) +
    geom_histogram(fill = "steelblue", color = "black", na.rm = TRUE) +  # Auto bin width
    theme_minimal() +
    labs(title = paste("Distribution of", col), x = col, y = "Count")

  print(p)
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

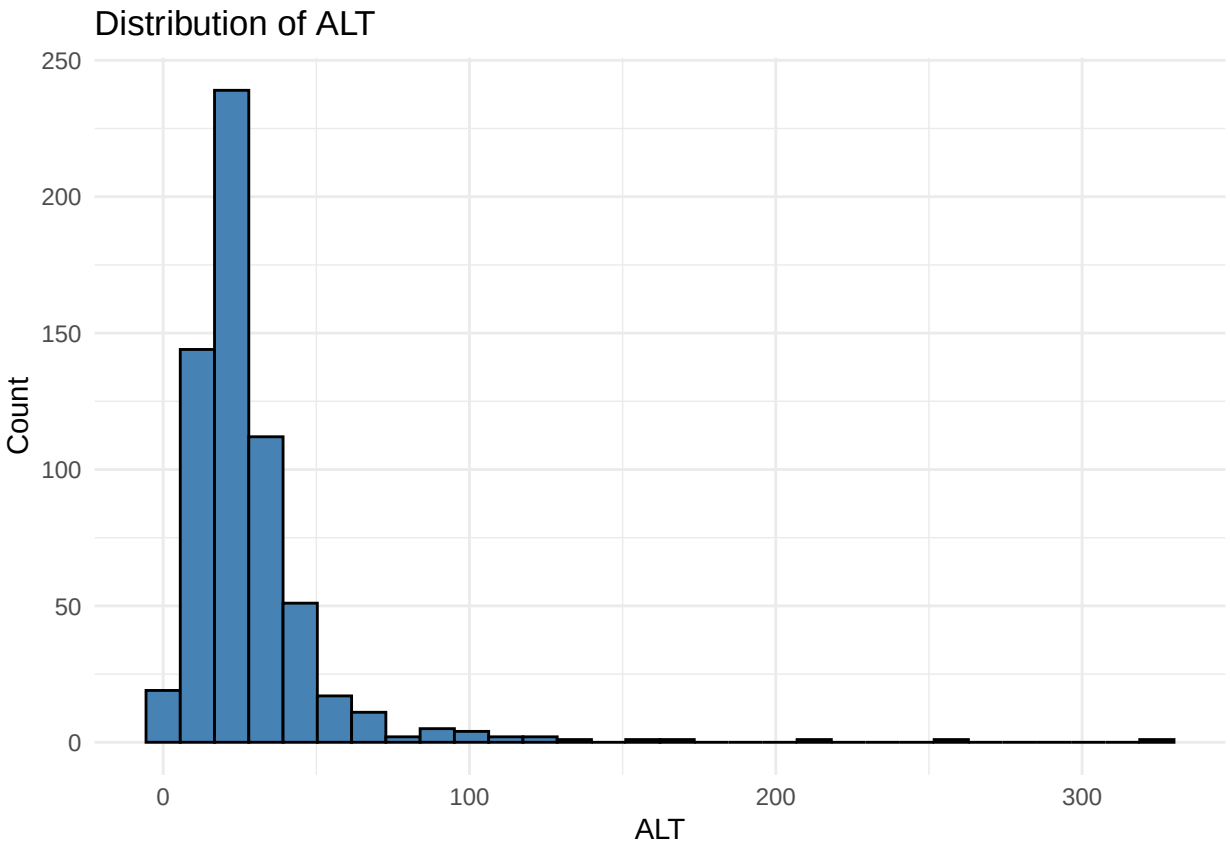
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



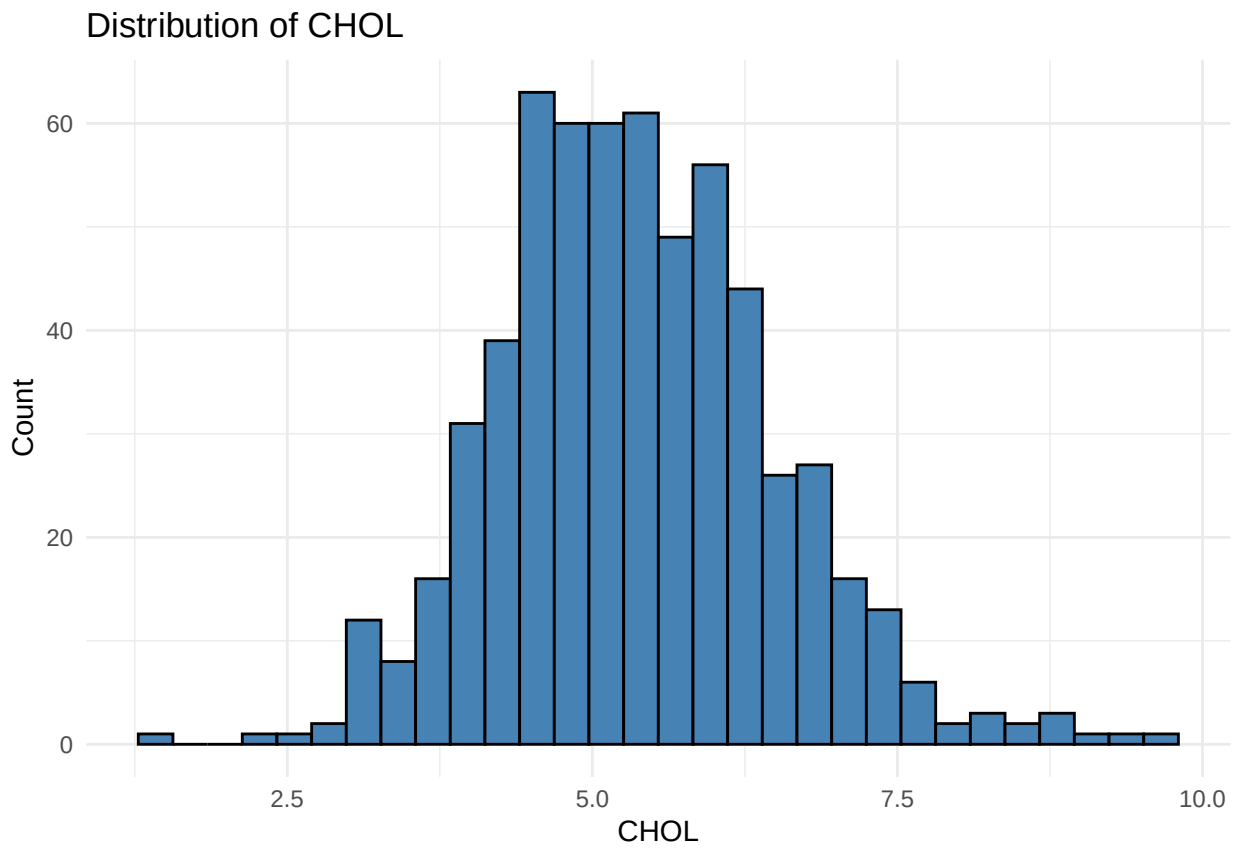

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



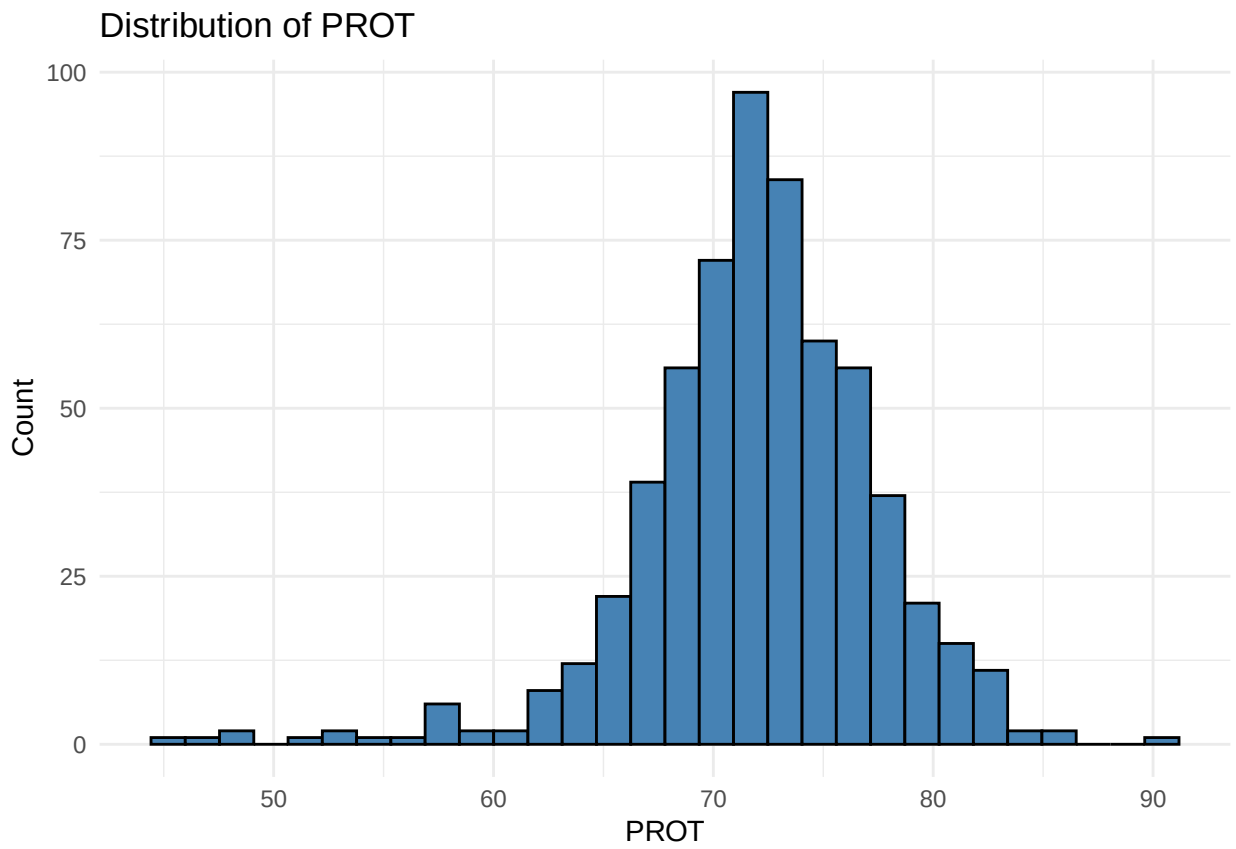

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

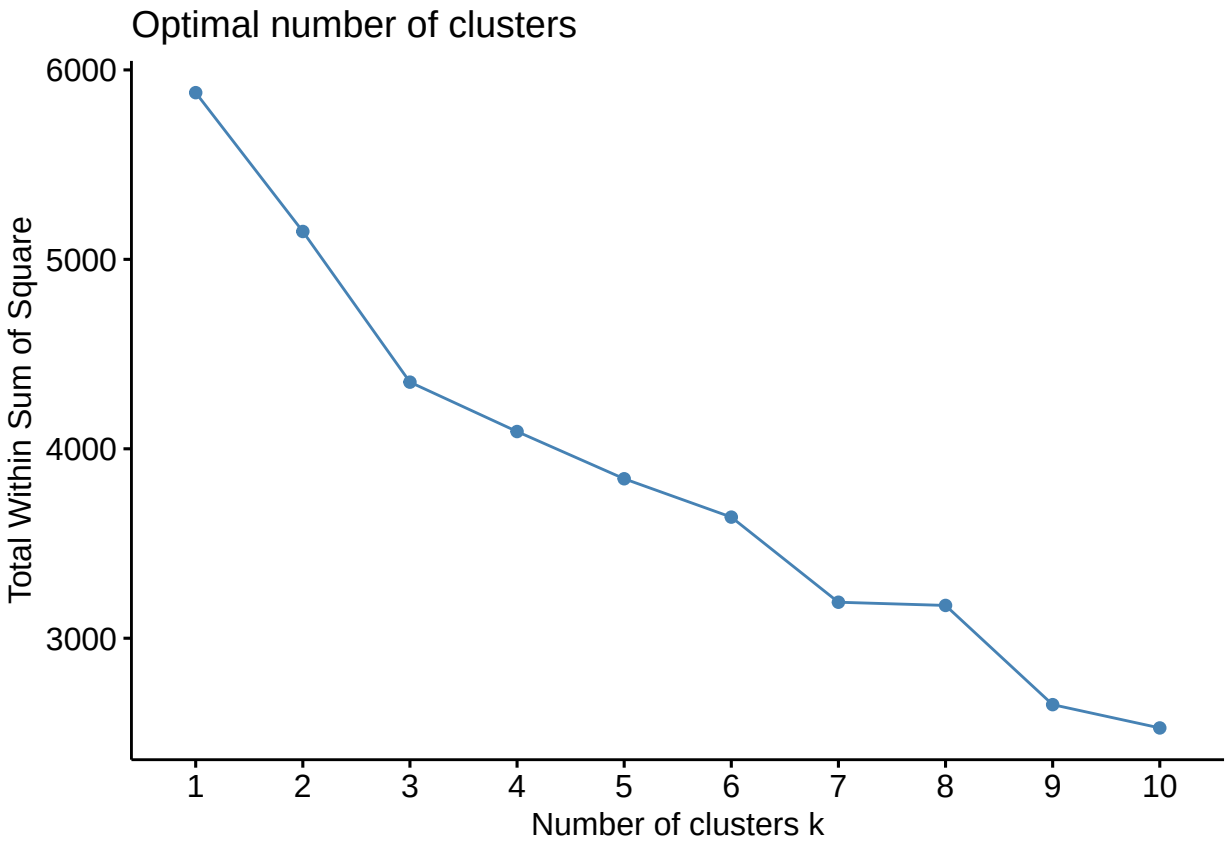


All the distributions of the missing columns appear to be normally distributed except with some extreme values we can impute by taking the IQR and then from that IQR we can using the mean to impute them. But since in 3.b it is asked to remove the missing values I did not implement this solution.

(b) Removing Missing Values and Finding Optimal Clusters

We will remove rows with missing values to ensure clean clustering.

```
df_clean <- na.omit(df)

df_numeric <- df_clean[, 5:14]

df_scaled <- scale(df_numeric)

fviz_nbclust(df_scaled, kmeans, method = "wss")
```



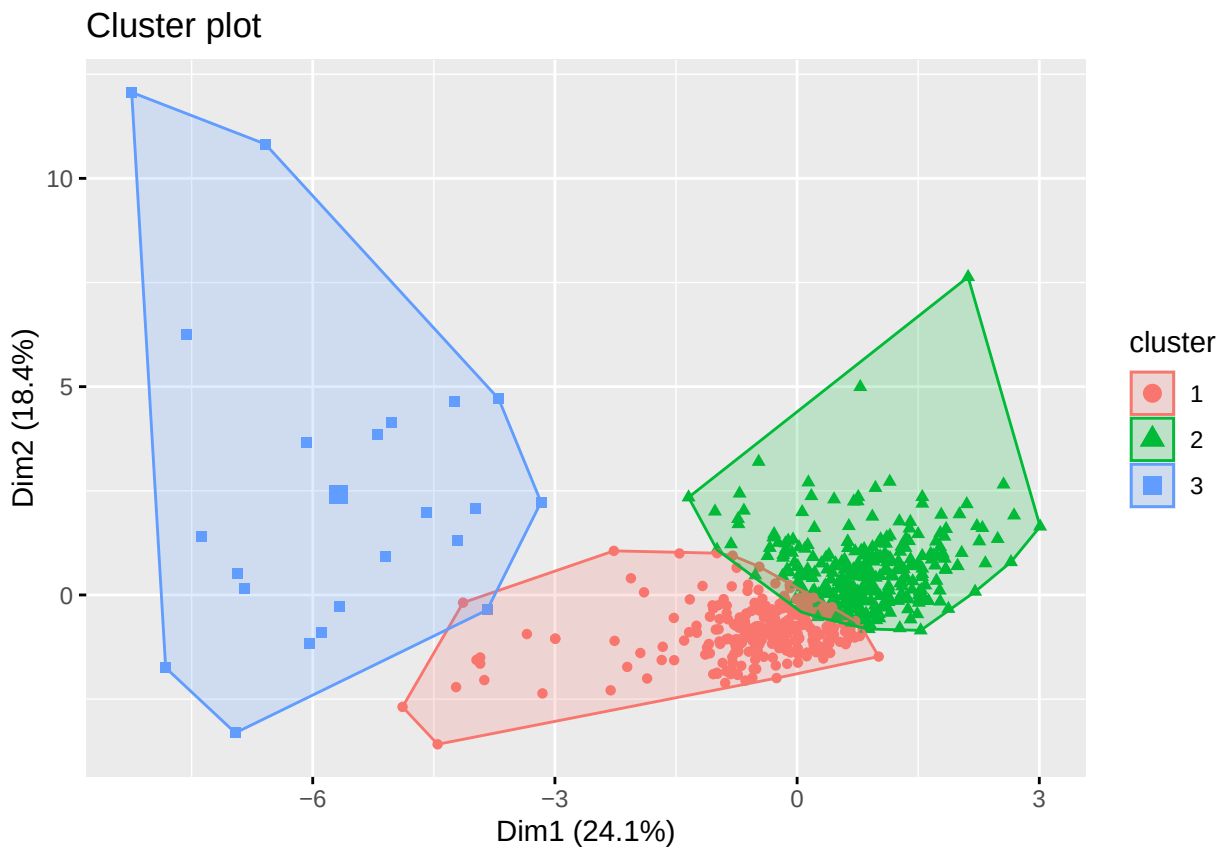

```r
set.seed(123)
k <- 3
km_res <- kmeans(df_scaled, centers = k, nstart = 25)

df_numeric <- as.data.frame(df_scaled)
df_numeric$cluster <- as.factor(km_res$cluster)

df_character <- df_clean %>% select_if(is.character)

df_clean <- cbind(df_character, df_numeric)
```

From the elbow plot we can see that the optimal number of clusters is 3 so we use k = 3 to find the clusters.

(c) K-means Clustering and Plot

We apply K-means clustering with the optimal k found above.

```r
set.seed(123)  # Ensures cluster reproducibility
fviz_cluster(km_res, data = df_scaled, geom = "point")
```



K-means is performing well, we see that the clusters are fairly distinguishable with little overlapping.

(d) Compare Means Across Clusters

```
aggr_means <- df_clean %>% group_by(cluster) %>%
  summarise(across(c(ALB, ALP, ALT, AST), mean, na.rm = TRUE))
```

```
## Warning: There was 1 warning in `summarise()`.
## i In argument: `across(c(ALB, ALP, ALT, AST), mean, na.rm = TRUE)`.
## i In group 1: `cluster = 1`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```r
print(aggr_means)
```

```
## # A tibble: 3 x 5
##   cluster    ALB    ALP     ALT     AST
##   <fct>    <dbl>  <dbl>   <dbl>   <dbl>
## 1 1       -0.355 -0.278 -0.287  -0.175
## 2 2        0.449  0.113  0.250  -0.0503
## 3 3       -1.77   1.85   0.0870  2.82
```

The means are quite different from plotting the means across the clusters. AST is the most similar means for cluster 1 and 2, but other columns have good distribution of means. This means that AST is almost the same for cluster 1 and 2.

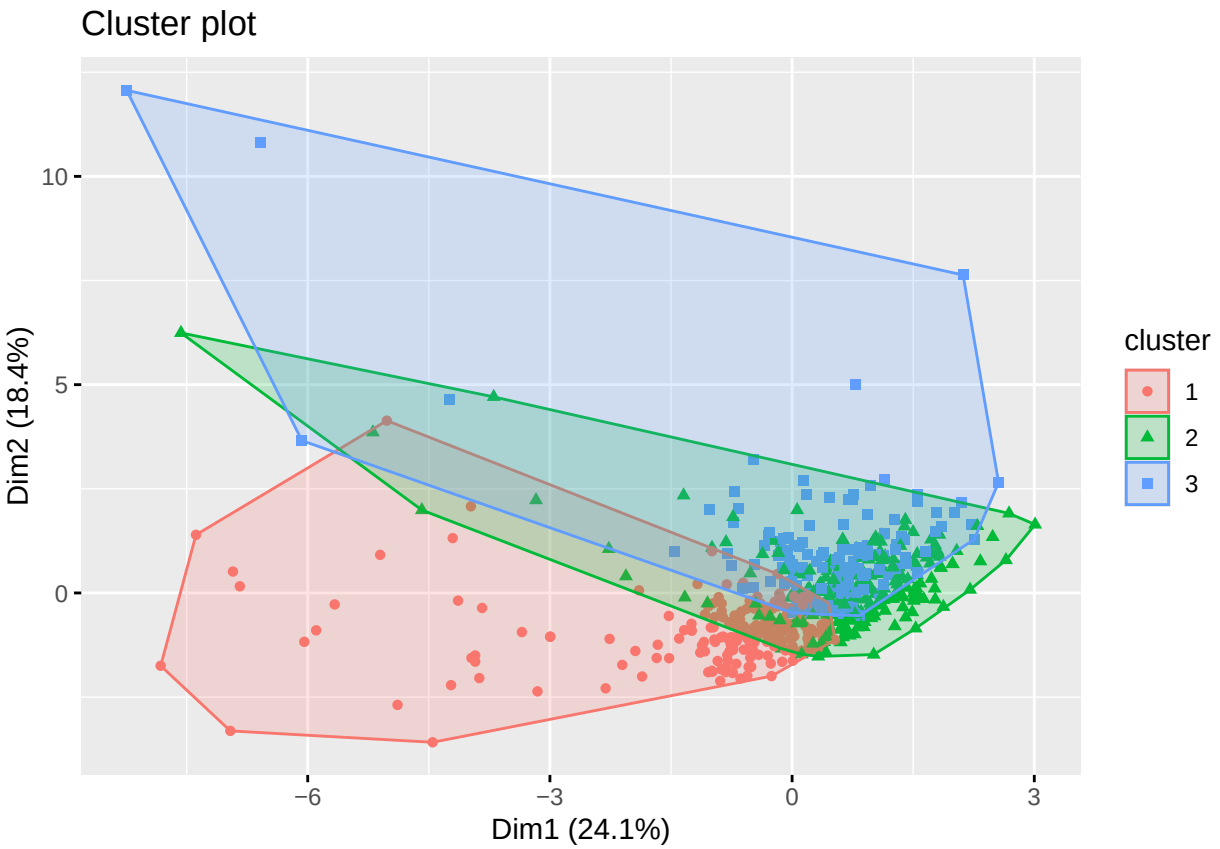
Question 4: K-medoids Clustering

(a) Apply K-medoids Clustering

```r
set.seed(123)  # Ensures reproducibility
km_res_medoid <- pam(df_scaled, k)
df_clean$medoid_cluster <- as.factor(km_res_medoid$clustering)
```

(b) K-medoids Plot

```r
fviz_cluster(km_res_medoid, data = df_scaled, geom = "point")
```



Using K-medoids the clusters overlap a lot, it is not distinguishable most prominent overlaps are cluster 2 and cluster 3.

(c) Comparison of K-means and K-medoids

```
cluster_table <- table(df_clean$cluster, df_clean$medoid_cluster)
cluster_table
```

```
##
##       1   2   3
##   1 200  49  19
##   2   3 183 113
##   3  13   5   4
```

Comparison: K-means uses centroids, while K-medoids selects actual data points as centers, making it more robust to outliers. But in this data set we see that K-means do a much better task than K-medoids, beacase the data is sparse and since K-means uses centroids (mean of the clusters) the clusters are much better as they are able to move the centroids. But since K-medoids choose actual data points it make a lot of over laps.

Preferred Approach: For this dataset K-means is a better approach.