# Lab File

## Fundamentals of Data Science Lab

### Session: Jan - May 2025

Programme: BTech. CS - Data Science

Sem: 4
Batch: 5

**Submitted By:**
Name: Kshitij Chandrakar
SAP ID: 500124827

**Submitted To:**
Dr. Sachi Chaudhary

# Experiment 6
# Excercise1

April 21, 2025

[1]:
```
library(ggplot2)
library(dplyr)
library(cluster)
```

```
Attaching package: 'dplyr'


The following objects are masked from 'package:stats':

    filter, lag


The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

[2]:
```
set.seed(1)
```

[3]:
```
mall_data <- read.csv("/home/asus/content/Notes/Semester 4/FDN Lab/Experiments/
↪Experiment 6/Mall_Customers.csv")
```

[4]:
```
head(mall_data)
summary(mall_data)
```

A data.frame: 6 × 5

| | CustomerID <int> | Gender <chr> | Age <int> | Annual.Income..k.. <int> | Spending.Score..1.100. <int> |
|---|---|---|---|---|---|
| 1 | 1 | Male | 19 | 15 | 39 |
| 2 | 2 | Male | 21 | 15 | 81 |
| 3 | 3 | Female | 20 | 16 | 6 |
| 4 | 4 | Female | 23 | 16 | 77 |
| 5 | 5 | Female | 31 | 17 | 40 |
| 6 | 6 | Female | 22 | 17 | 76 |

```
   CustomerID          Gender               Age         Annual.Income..k..
 Min.   :  1.00   Length:200         Min.   :18.00   Min.   : 15.00
```

```
1st Qu.: 50.75   Class :character   1st Qu.:28.75   1st Qu.: 41.50
Median :100.50   Mode  :character   Median :36.00   Median : 61.50
Mean   :100.50                      Mean   :38.85   Mean   : 60.56
3rd Qu.:150.25                      3rd Qu.:49.00   3rd Qu.: 78.00
Max.   :200.00                      Max.   :70.00   Max.   :137.00
Spending.Score..1.100.
Min.   : 1.00
1st Qu.:34.75
Median :50.00
Mean   :50.20
3rd Qu.:73.00
Max.   :99.00
```

[5]:
```
colnames(mall_data) <- c("CustomerID", "Gender", "Age", "AnnualIncome",␣
↪"SpendingScore")
```

[6]:
```
ggplot(mall_data, aes(x = AnnualIncome, y = SpendingScore)) +
  geom_point(aes(color = Gender), alpha = 0.7) +
  labs(title = "Annual Income vs Spending Score by Gender",
       x = "Annual Income (k$)",
       y = "Spending Score (1-100)") +
  theme_minimal()
```

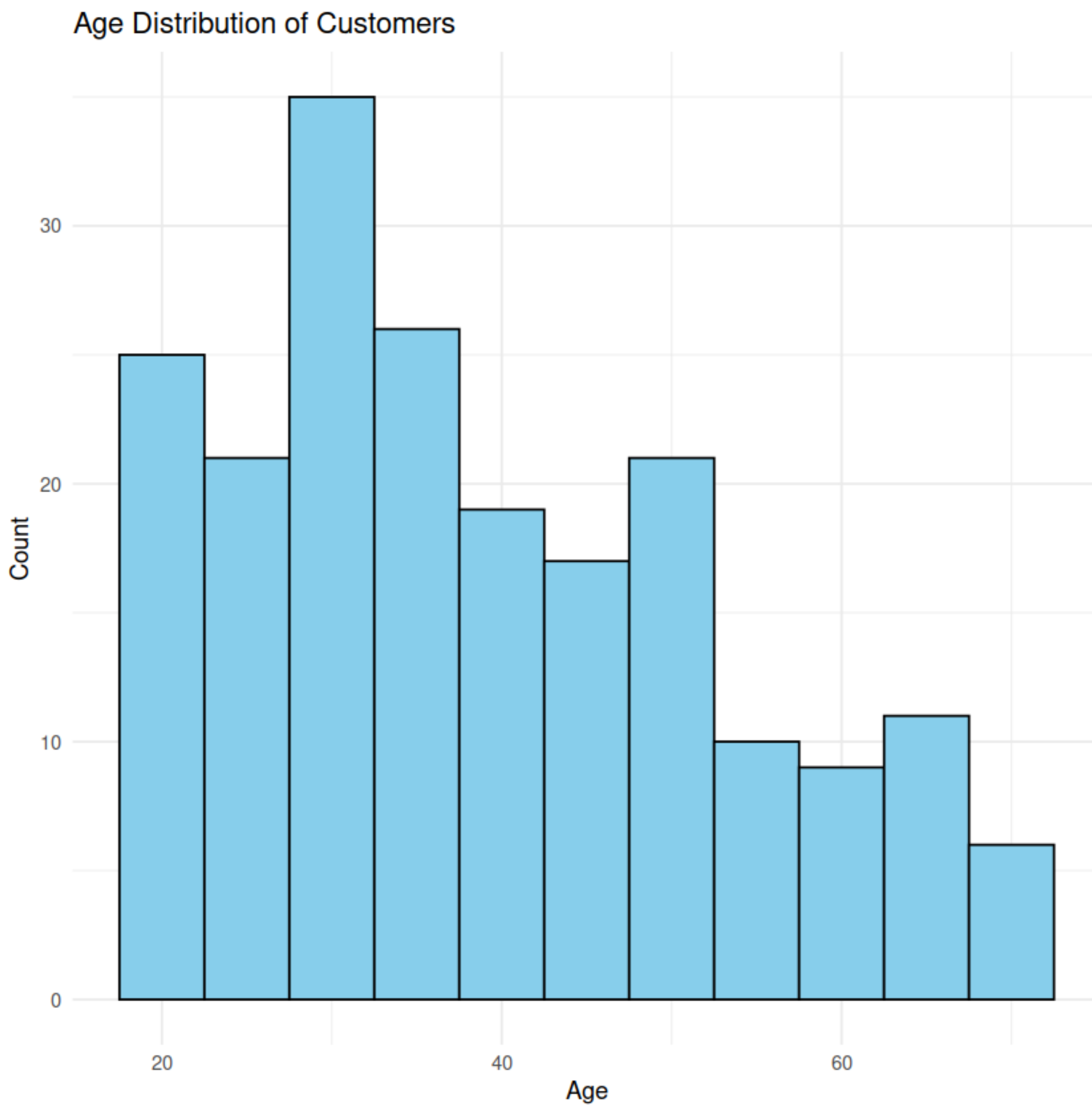


2


[7]:
```
ggplot(mall_data, aes(x = Age)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +
  labs(title = "Age Distribution of Customers",
       x = "Age",
       y = "Count") +
  theme_minimal()
```



3




[8]:
```
ggplot(mall_data, aes(x = Gender, y = SpendingScore, fill = Gender)) +
  geom_boxplot() +
  labs(title = "Spending Score Distribution by Gender",
       y = "Spending Score (1-100)") +
  theme_minimal()
# ===================================
# Part 2
# ===================================
```


[9]:

```
data_for_clustering <- mall_data[, c("AnnualIncome", "SpendingScore")]
```

[10]:

```
wss <- function(k) {
  kmeans(data_for_clustering, k, nstart = 10)$tot.withinss
}
```

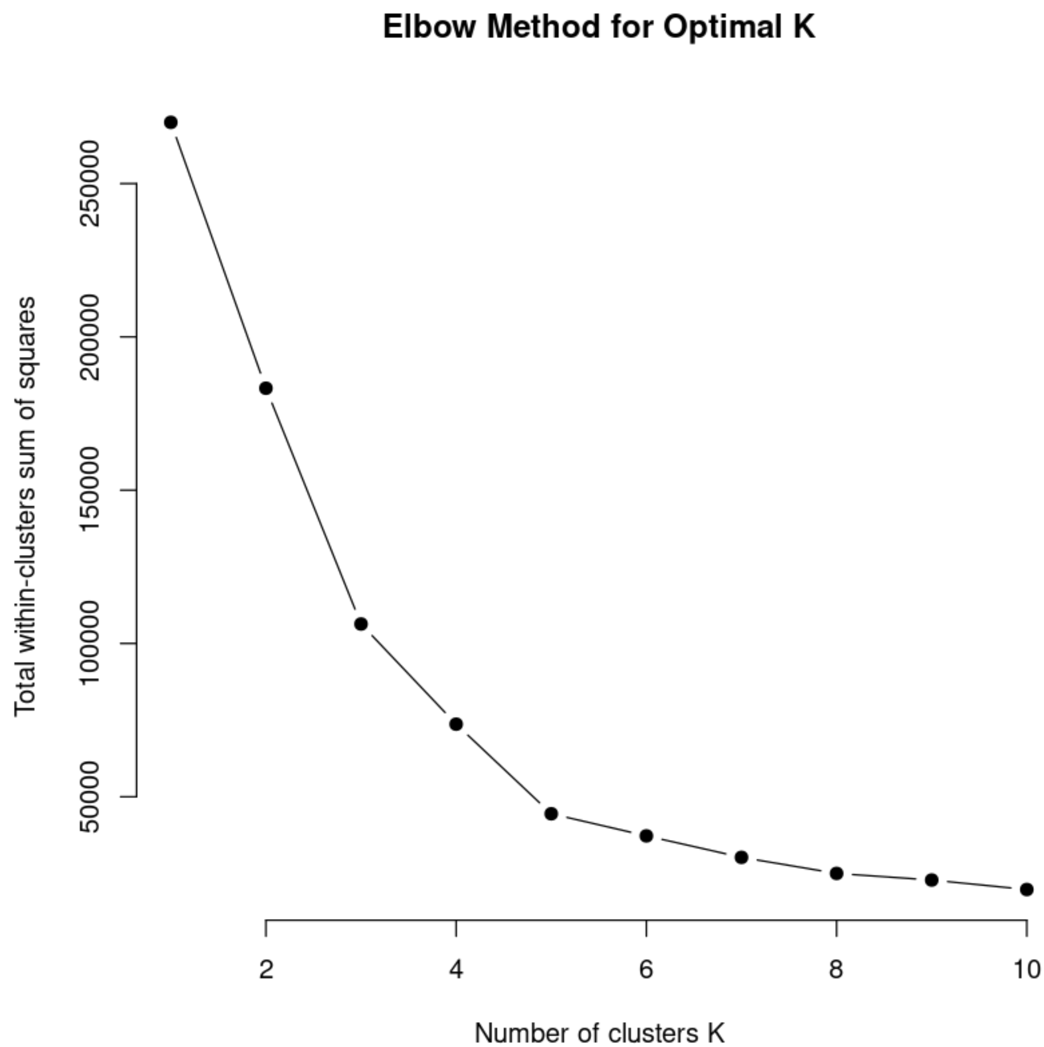
[11]:

```
k_values <- 1:10
wss_values <- sapply(k_values, wss)
```

[12]:

```
plot(k_values, wss_values,
     type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters K",
     ylab = "Total within-clusters sum of squares",
```

```r
    main = "Elbow Method for Optimal K")
```



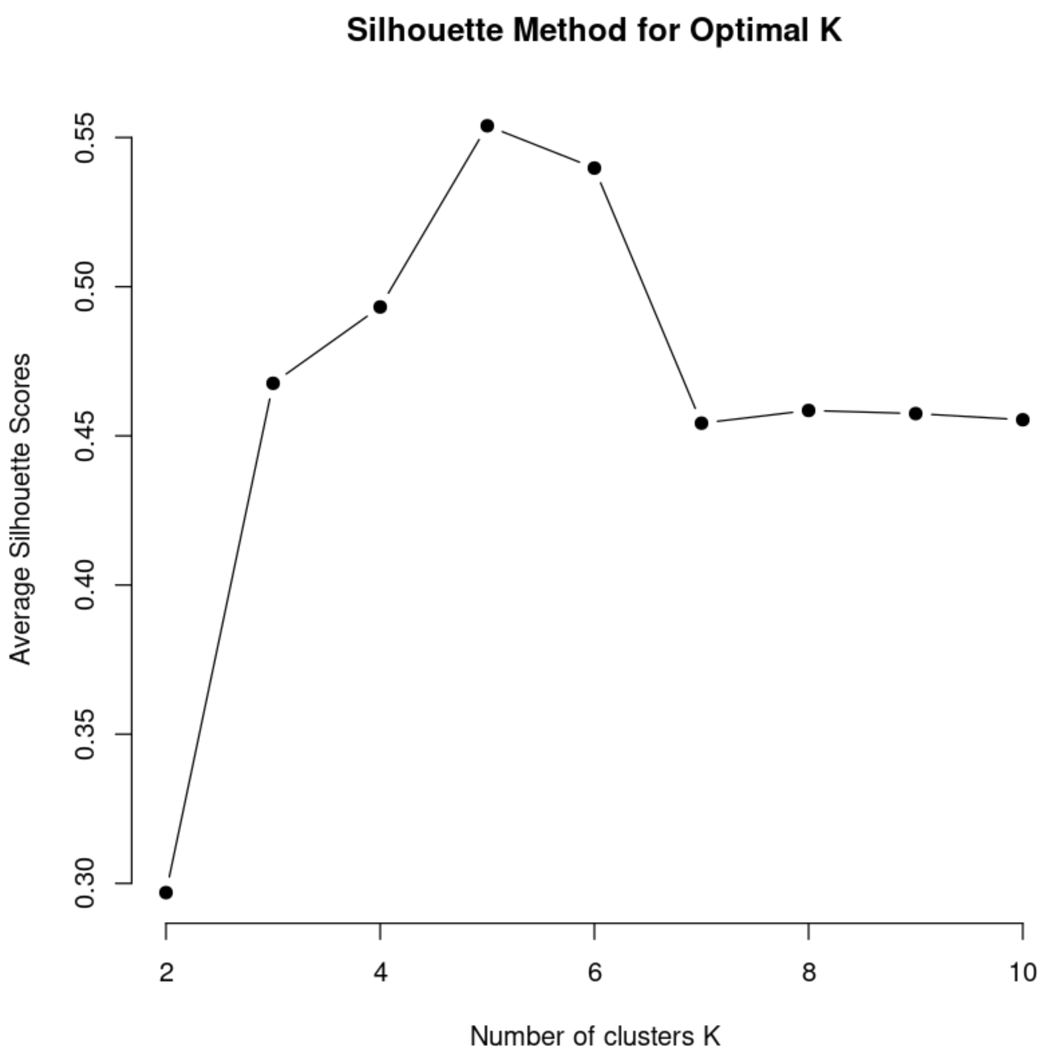

[13]:
```r
avg_sil <- function(k) {
  km.res <- kmeans(data_for_clustering, centers = k, nstart = 25)
  ss <- silhouette(km.res$cluster, dist(data_for_clustering))
  mean(ss[, 3])
}
```

[14]:
```r
k_values <- 2:10
avg_sil_values <- sapply(k_values, avg_sil)
```



6

[15]:
```r
plot(k_values, avg_sil_values,
     type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters K",
     ylab = "Average Silhouette Scores",
     main = "Silhouette Method for Optimal K")
```



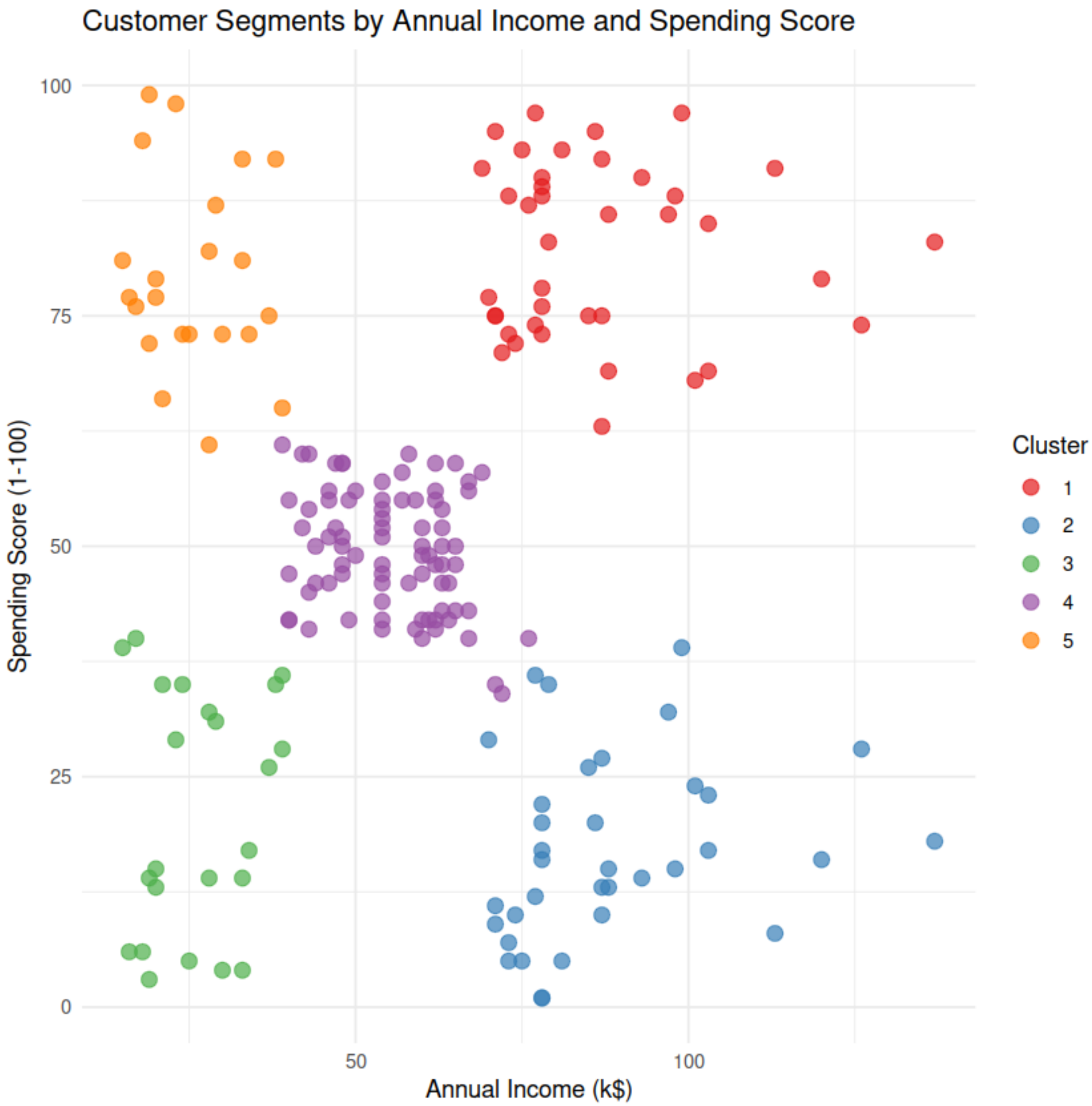

[16]:
```r
final_k <- 5
```

[17]:
```r
kmeans_result <- kmeans(data_for_clustering, centers = final_k, nstart = 25)
mall_data$Cluster <- as.factor(kmeans_result$cluster)
```



7

[18]:
```
ggplot(mall_data, aes(x = AnnualIncome, y = SpendingScore, color = Cluster)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_color_brewer(palette = "Set1") +
  labs(title = "Customer Segments by Annual Income and Spending Score",
       x = "Annual Income (k$)",
       y = "Spending Score (1-100)") +
  theme_minimal()
```




[19]:
```
cluster_stats <- mall_data %>%
  group_by(Cluster) %>%
  summarise(
    Count = n(),
    Avg_Age = mean(Age),
```

8

```
  Avg_Income = mean(AnnualIncome),
  Avg_Spending = mean(SpendingScore),
  Female_Pct = sum(Gender == "Female") / n() * 100
)
```

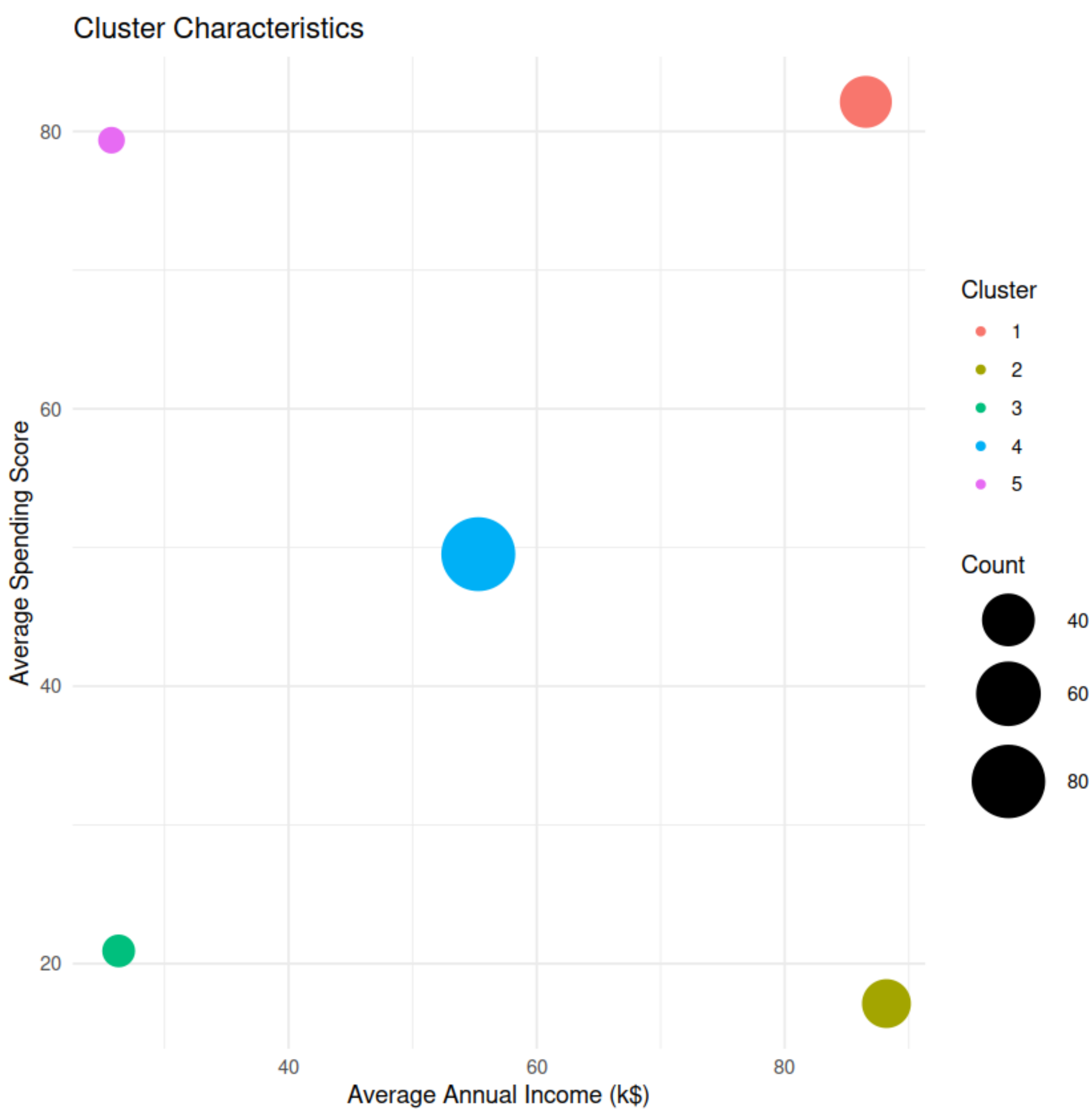
[20]:
```
# Printing Stats
print(cluster_stats)
```

```
# A tibble: 5 × 6
  Cluster Count Avg_Age Avg_Income Avg_Spending Female_Pct
  <fct>   <int>   <dbl>
<dbl>        <dbl>
<dbl>
1 1          39    32.7       86.5         82.1       53.8
2 2          35    41.1       88.2         17.1       45.7
3 3          23    45.2       26.3         20.9       60.9
4 4          81    42.7       55.3         49.5       59.3
5 5          22    25.3       25.7         79.4       59.1
```

[21]:
```
ggplot(cluster_stats, aes(x = Avg_Income, y = Avg_Spending, size = Count, color␣
 ↪= Cluster)) +
  geom_point() +
  scale_size(range = c(5, 15)) +
  labs(title = "Cluster Characteristics",
       x = "Average Annual Income (k$)",
       y = "Average Spending Score") +
  theme_minimal()
```



9


Cluster Characteristics
Average Spending Score
Average Annual Income (k$)
80
60
40
20
40
60
80
Cluster
1
2
3
4
5
Count
40
60
80



10

# Excercise2

April 21, 2025

[8]:
```r
library(tidyverse)
library(cluster)
```

[9]:
```r
set.seed(1)
```

[10]:
```r
happiness <- read_csv("/home/asus/content/Notes/Semester 4/FDN Lab/Experiments/
↪Experiment 6/archive(11)/2019.csv")
```

```
Rows: 156 Columns: 9
  Column specification



Delimiter: ","
chr (1): Country
dbl (8): Overall rank, Score, GDP per capita, Social support, Healthy
life e…

  Use `spec()` to retrieve the full column specification for this
data.
  Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

[11]:
```r
head(happiness)
summary(happiness)
```

A tibble: 6 × 9

| Overall rank <dbl> | Country <chr> | Score <dbl> | GDP per capita <dbl> | Social support <dbl> | Healthy life expectancy <dbl> |
|---|---|---|---|---|---|
| 1 | Finland | 7.769 | 1.340 | 1.587 | 0.986 |
| 2 | Denmark | 7.600 | 1.383 | 1.573 | 0.996 |
| 3 | Norway | 7.554 | 1.488 | 1.582 | 1.028 |
| 4 | Iceland | 7.494 | 1.380 | 1.624 | 1.026 |
| 5 | Netherlands | 7.488 | 1.396 | 1.522 | 0.999 |
| 6 | Switzerland | 7.480 | 1.452 | 1.526 | 1.052 |

```
  Overall rank       Country              Score        GDP per capita
 Min.   :  1.00    Length:156         Min.   :2.853   Min.   :0.0000
 1st Qu.: 39.75    Class :character   1st Qu.:4.545   1st Qu.:0.6028
 Median : 78.50    Mode  :character   Median :5.380   Median :0.9600
```

```
 Mean   : 78.50                      Mean   :5.407   Mean   :0.9051
 3rd Qu.:117.25                      3rd Qu.:6.184   3rd Qu.:1.2325
 Max.   :156.00                      Max.   :7.769   Max.   :1.6840
 Social support  Healthy life expectancy Freedom to make life choices
 Min.   :0.000   Min.   :0.0000          Min.   :0.0000
 1st Qu.:1.056   1st Qu.:0.5477          1st Qu.:0.3080
 Median :1.272   Median :0.7890          Median :0.4170
 Mean   :1.209   Mean   :0.7252          Mean   :0.3926
 3rd Qu.:1.452   3rd Qu.:0.8818          3rd Qu.:0.5072
 Max.   :1.624   Max.   :1.1410          Max.   :0.6310
   Generosity     Perceptions of corruption
 Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.1087   1st Qu.:0.0470
 Median :0.1775   Median :0.0855
 Mean   :0.1848   Mean   :0.1106
 3rd Qu.:0.2482   3rd Qu.:0.1412
 Max.   :0.5660   Max.   :0.4530
```

[12]:
```
# Filtering out the text cols
features <- c("Overall rank", "Score", "GDP per capita", "Social support",␣
↪"Healthy life expectancy", "Freedom to make life choices", "Generosity",␣
↪"Perceptions of corruption")

happiness <- happiness %>%
  select(all_of(features)) %>%
  na.omit()
```

[13]:
```
features <- c("Overall rank", "Score", "GDP per capita", "Social support",␣
↪"Healthy life expectancy", "Freedom to make life choices", "Generosity",␣
↪"Perceptions of corruption")

happiness_country <- happiness %>%
  select(all_of(features))
```

[14]:
```
#SCAAAAAAAAAAAAAAAALIng
happiness_scaled <- scale(happiness)
```

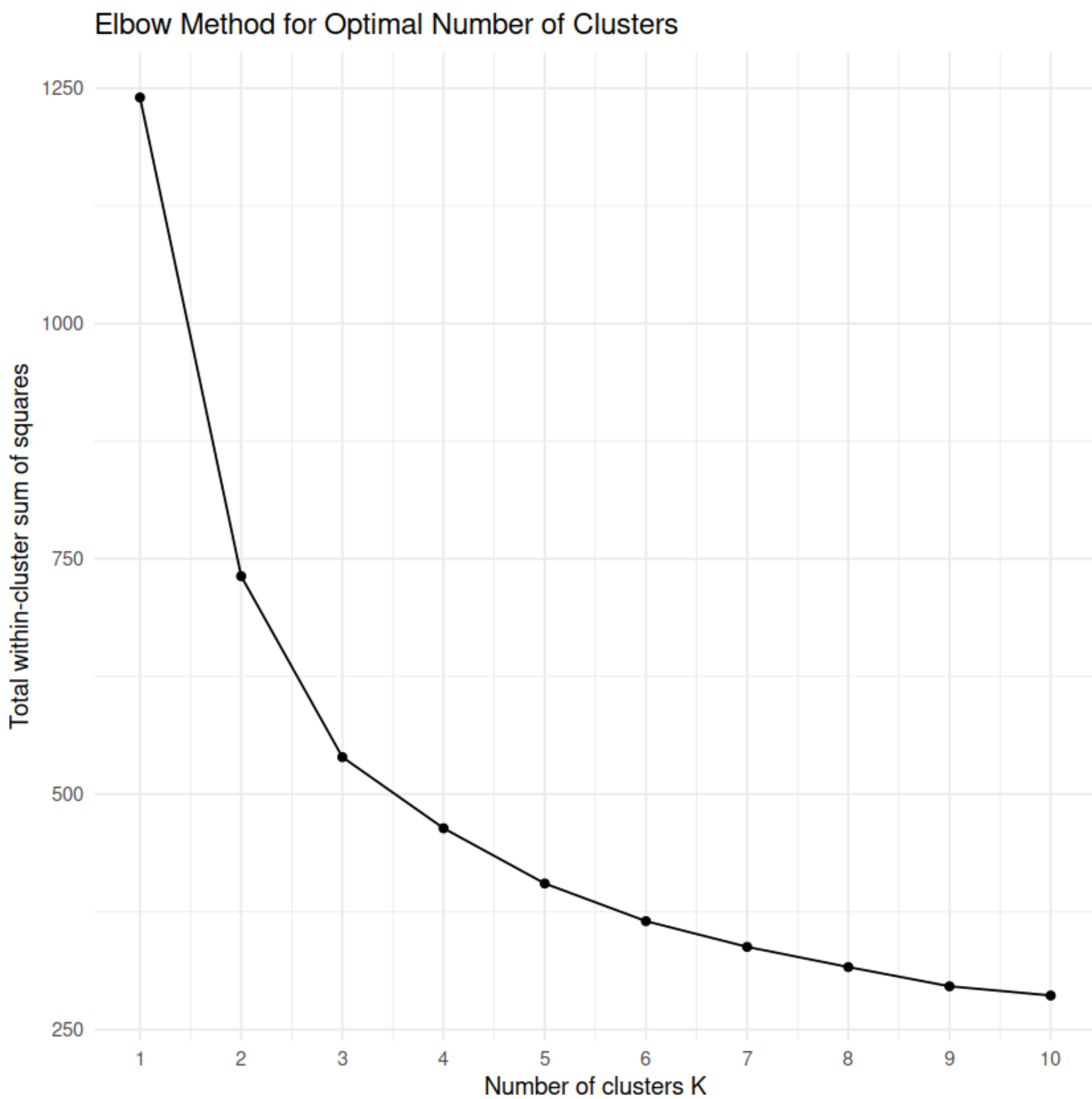
[15]:
```
# total within-cluster sum of squares
wss <- function(k) {
  kmeans(happiness_scaled, k, nstart = 10)$tot.withinss
}
```

[16]:
```
k_values <- 1:10
wss_values <- map_dbl(k_values, wss)
```

[17]:
```
ggplot(data.frame(k = k_values, wss = wss_values), aes(k, wss)) +
  geom_line() + geom_point() +
```

12

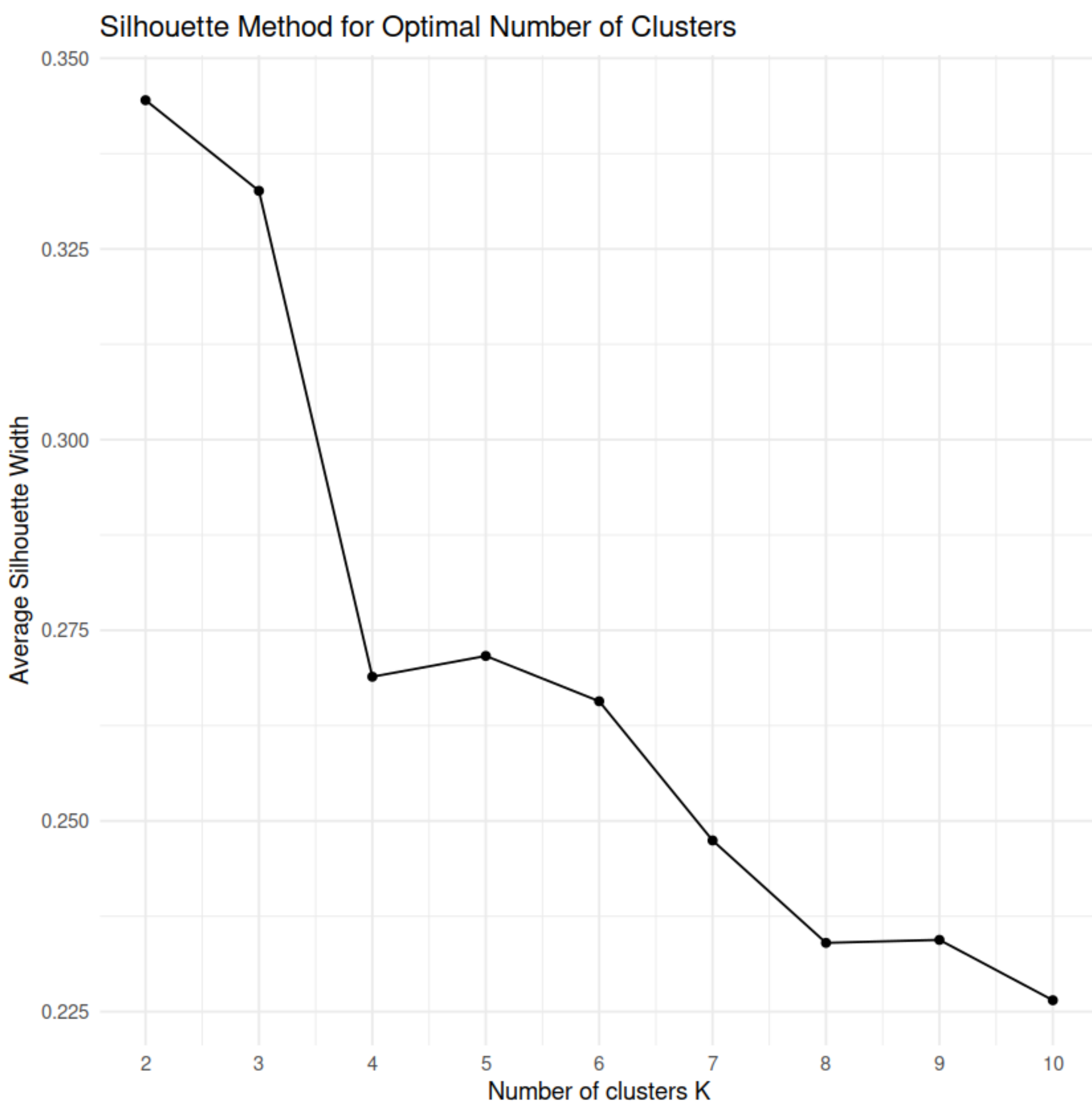
```
  scale_x_continuous(breaks = k_values) +
  labs(title = "Elbow Method for Optimal Number of Clusters",
       x = "Number of clusters K",
       y = "Total within-cluster sum of squares") +
  theme_minimal()
```




[18]:
```
avg_sil <- function(k) {
  km.res <- kmeans(happiness_scaled, centers = k, nstart = 25)
  ss <- silhouette(km.res$cluster, dist(happiness_scaled))
  mean(ss[, 3])
}
```

[19]:
```
k_values <- 2:10
avg_sil_values <- map_dbl(k_values, avg_sil)
```

[20]:
```
ggplot(data.frame(k = k_values, silhouette = avg_sil_values), aes(k,
↪silhouette)) +
  geom_line() + geom_point() +
  scale_x_continuous(breaks = k_values) +
  labs(title = "Silhouette Method for Optimal Number of Clusters",
       x = "Number of clusters K",
       y = "Average Silhouette Width") +
  theme_minimal()
```





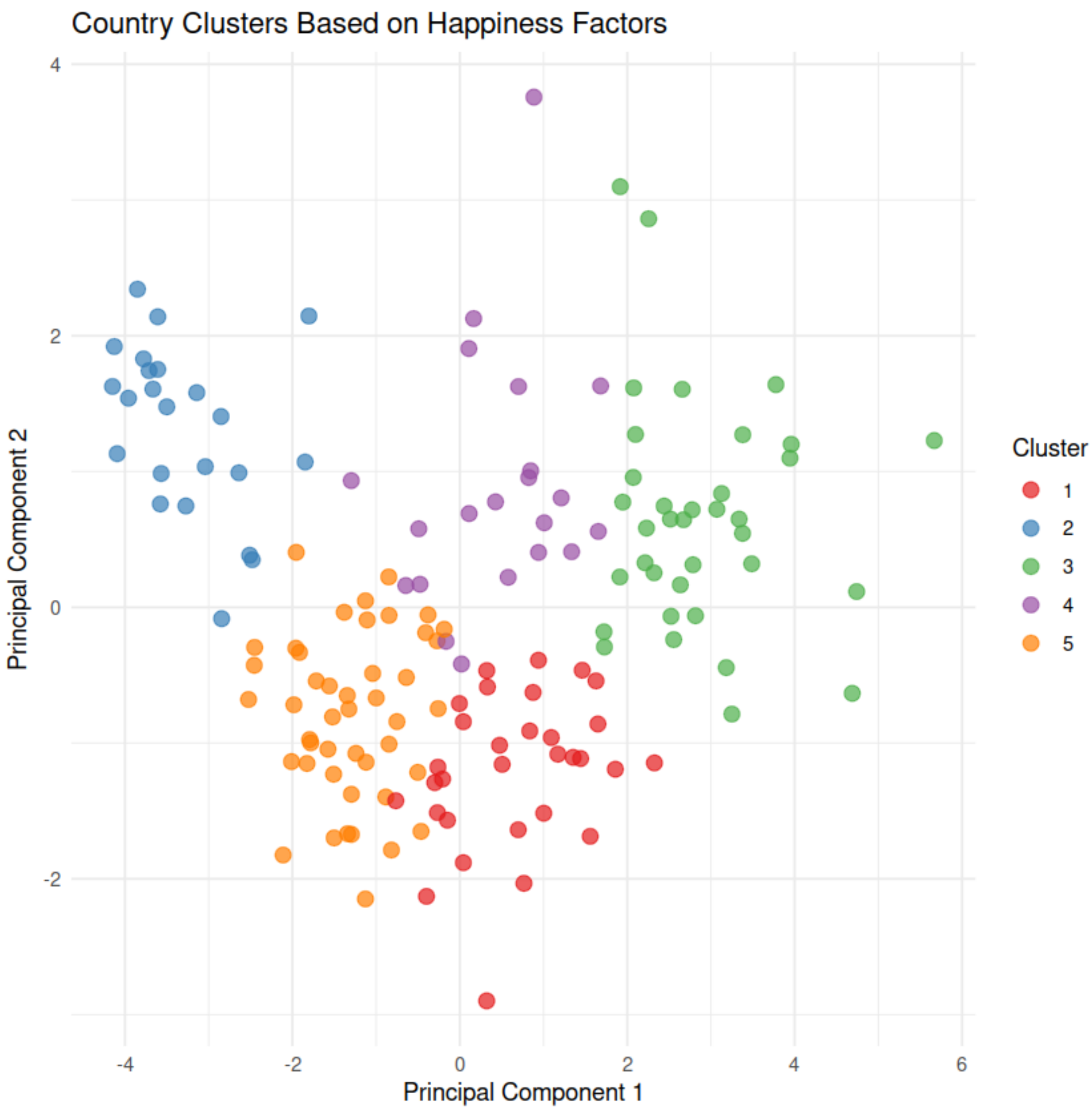

[21]:
```
optimal_k <- 5
```

[22]:
```r
kmeans_result <- kmeans(happiness_scaled, centers = optimal_k, nstart = 25)
```

[23]:
```r
# Add cluster assignments to original data
happiness$Cluster <- as.factor(kmeans_result$cluster)
```

[24]:
```r
# Perform PCA for visualization
pca_result <- prcomp(happiness_scaled, scale. = TRUE)
pca_df <- as.data.frame(pca_result$x[, 1:2])
pca_df$Cluster <- happiness$Cluster
```

[25]:
```r
# Plot clusters in PCA space
ggplot(pca_df, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_color_brewer(palette = "Set1") +
  labs(title = "Country Clusters Based on Happiness Factors",
       x = "Principal Component 1",
       y = "Principal Component 2") +
  theme_minimal()
```




[26]:

```
# Prepare data for parallel coordinates plot
cluster_means <- happiness %>%
  group_by(Cluster) %>%
  summarise(across(where(is.numeric), mean))
```

[27]:

```
cluster_means_long <- cluster_means %>%
  pivot_longer(cols = -Cluster, names_to = "Feature", values_to = "Mean_Value")
```

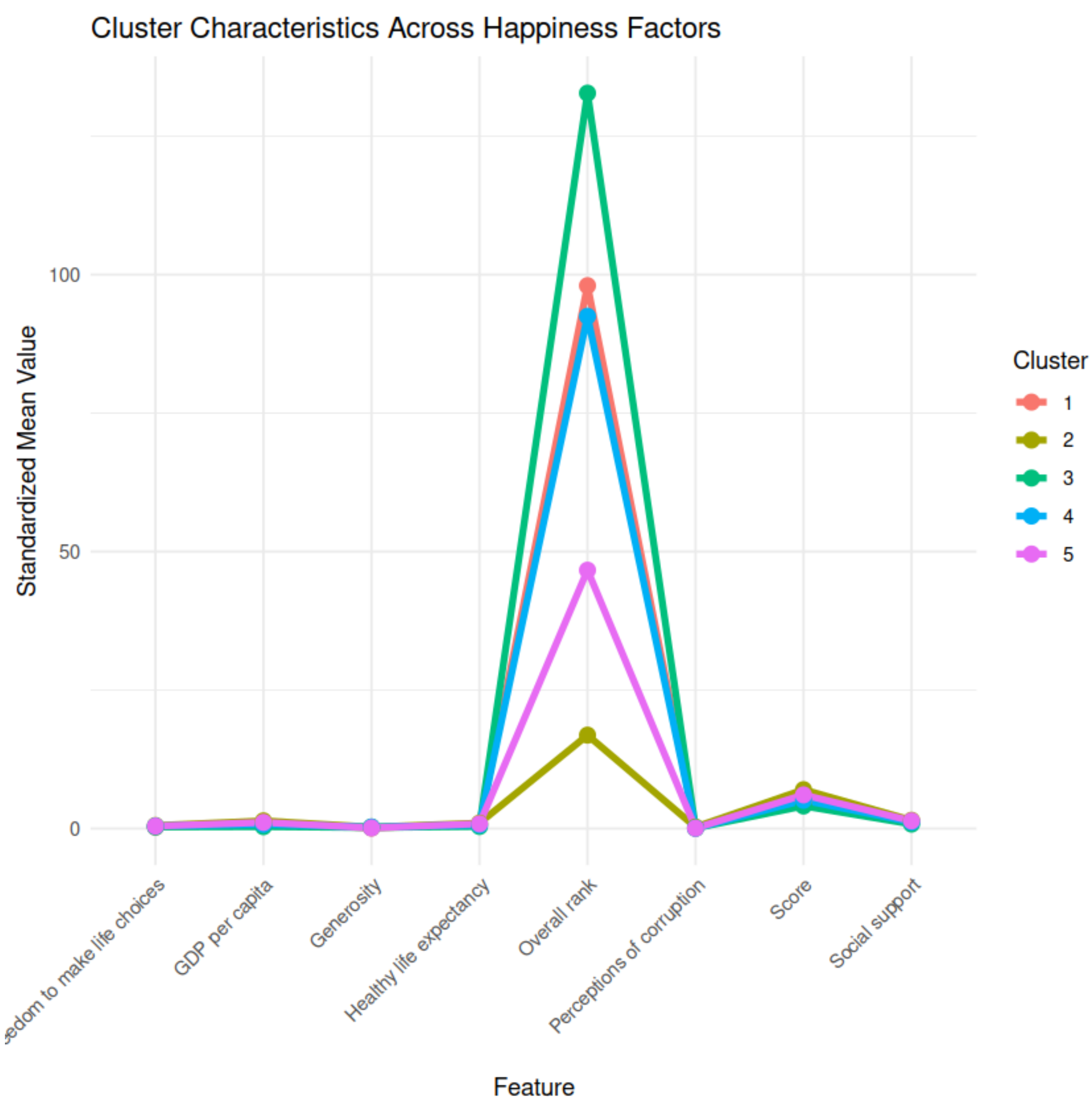
[28]:

```
ggplot(cluster_means_long, aes(x = Feature, y = Mean_Value, group = Cluster,␣
↪color = Cluster)) +
  geom_line(size = 1.5) +
  geom_point(size = 3) +
  labs(title = "Cluster Characteristics Across Happiness Factors",
```

16

```
    y = "Standardized Mean Value") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
Warning message:
"Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
 Please use `linewidth` instead."
```




[29]:
```
# Calculate and display cluster characteristics
cluster_profiles <- happiness %>%
  group_by(Cluster) %>%
  summarise(
    Count = n(),
```

17

```
    Avg_GDP = mean(`GDP per capita`, na.rm = TRUE),
    Avg_Social = mean(`Social support`, na.rm = TRUE),
    Avg_Health = mean(`Healthy life expectancy`, na.rm = TRUE),
    Avg_Freedom = mean(`Freedom to make life choices`, na.rm = TRUE),
    Avg_Generosity = mean(Generosity, na.rm = TRUE),
    Avg_Corruption = mean(`Perceptions of corruption`, na.rm = TRUE)
  )
```

[30]:

```
# Print cluster profiles
print(cluster_profiles)
```

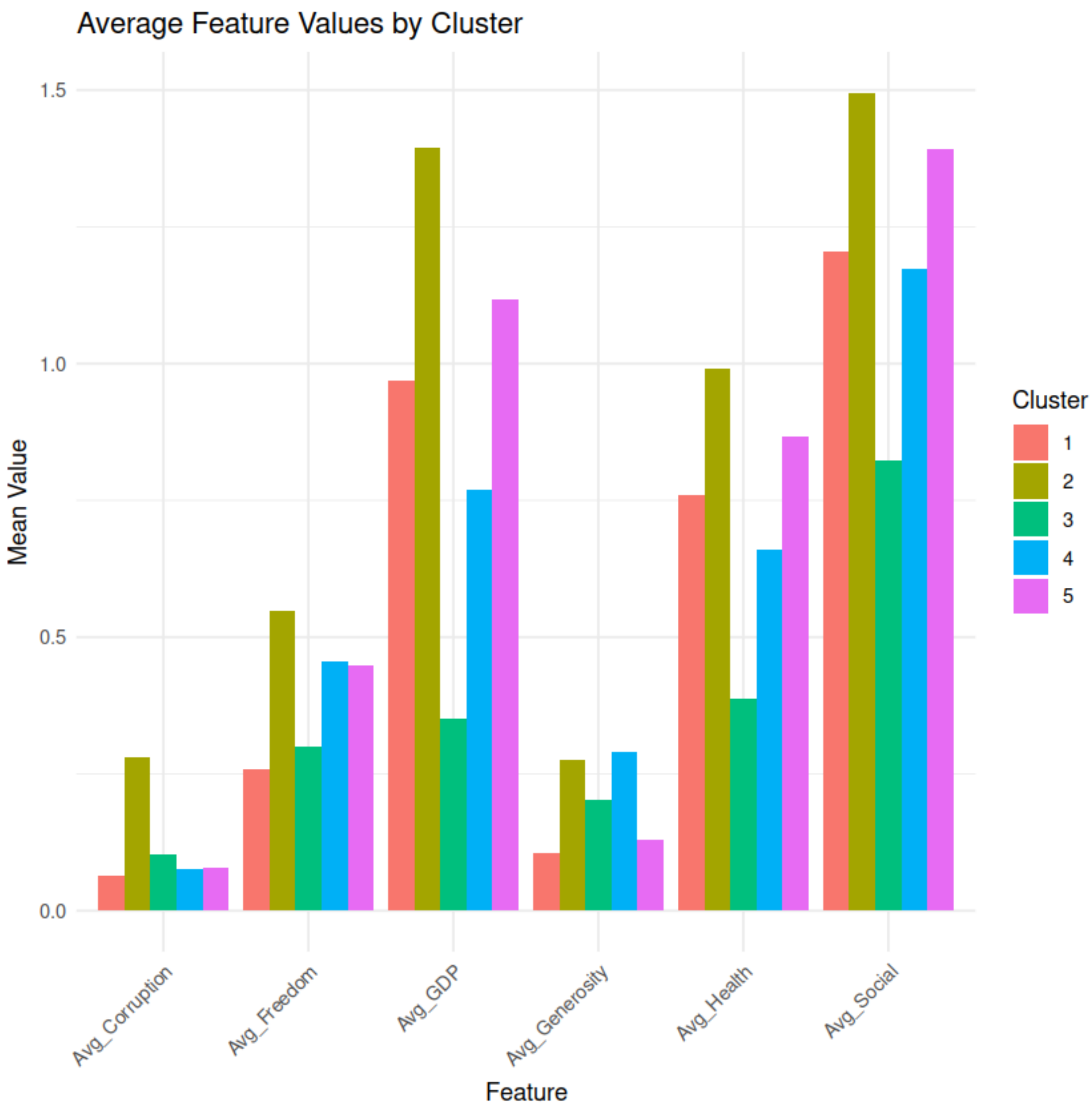
```
# A tibble: 5 × 8
  Cluster Count Avg_GDP Avg_Social Avg_Health Avg_Freedom Avg_Generosity
  <fct>   <int>   <dbl>
<dbl>      <dbl>
<dbl>          <dbl>
1 1          31   0.968      1.20       0.759       0.259
0.106
2 2          23   1.40       1.49       0.990       0.548
0.275
3 3          36   0.351      0.824      0.387       0.301
0.203
4 4          21   0.771      1.17       0.661       0.457
0.290
5 5          45   1.12       1.39       0.867       0.449
0.130
#   1 more variable: Avg_Corruption <dbl>
```

[31]:

```
cluster_profiles_long <- cluster_profiles %>%
  select(-Count) %>%
  pivot_longer(cols = -Cluster, names_to = "Feature", values_to = "Mean_Value")
```

[32]:

```
ggplot(cluster_profiles_long, aes(x = Feature, y = Mean_Value, fill = as.
↪factor(Cluster))) +
  geom_col(position = "dodge") +
  labs(title = "Average Feature Values by Cluster",
       y = "Mean Value",
       fill = "Cluster") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```


Average Feature Values by Cluster
1.5
1.0
0.5
0.0
Mean Value
Cluster
1
2
3
4
5
Avg_Corruption
Avg_Freedom
Avg_GDP
Avg_Generosity
Avg_Health
Avg_Social
Feature

# Excercise3

April 21, 2025

[1]:
```
library(tidyverse)
library(cluster)
library(gridExtra)
```

```
  Attaching core tidyverse packages

tidyverse 2.0.0
  dplyr     1.1.4      readr     2.1.5
  forcats   1.0.0      stringr   1.5.1
  ggplot2   3.5.2      tibble    3.2.1
  lubridate 1.9.4      tidyr     1.3.1
  purrr     1.0.4
  Conflicts

tidyverse_conflicts()
  dplyr::filter() masks stats::filter()
  dplyr::lag()    masks stats::lag()
  Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts to
become errors

Attaching package: ‘gridExtra’


The following object is masked from ‘package:dplyr’:

    combine
```

[2]:
```
set.seed(1)
```

[3]:
```
happiness <- read_csv("/home/asus/content/Notes/Semester 4/FDN Lab/Experiments/
↪Experiment 6/archive(11)/2019.csv")
```

```
Rows: 156 Columns: 9
```

Column specification

```
Delimiter: ","
chr (1): Country
dbl (8): Overall rank, Score, GDP per capita, Social support, Healthy
life e…

 Use `spec()` to retrieve the full column specification for this
data.
 Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

[4]:
```
numeric_data <- happiness %>%
  select(
  "Overall rank", "Score", "GDP per capita", "Social support", "Healthy life␣
↪expectancy", "Freedom to make life choices", "Generosity", "Perceptions of␣
↪corruption"
    ) %>%
  scale()
```

[5]:
```
rownames(numeric_data) <- happiness$`Country`
```

[6]:
```
wcss <- map_dbl(1:10, ~ kmeans(numeric_data, ., nstart = 25)$tot.withinss)
```

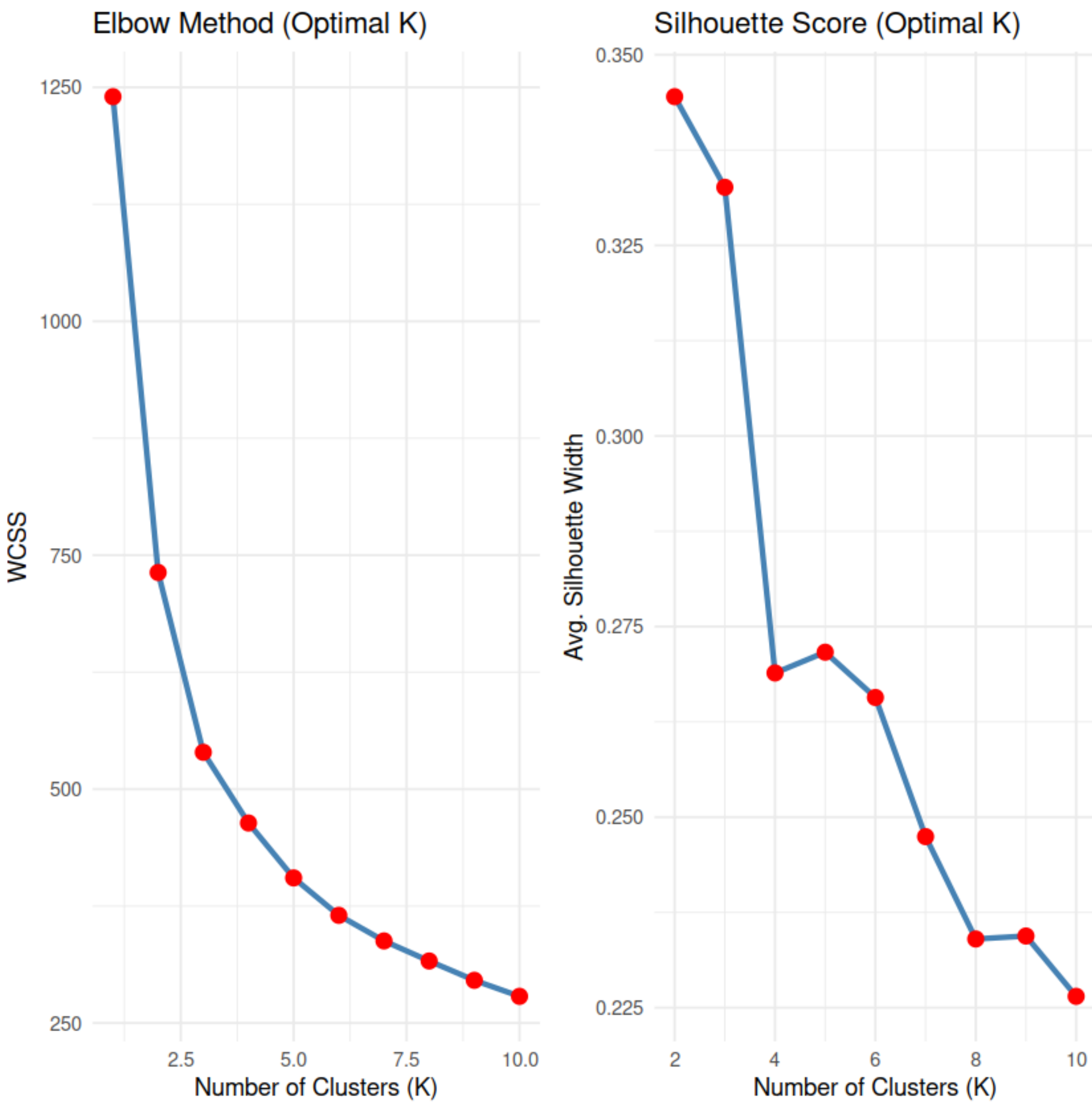
[7]:
```
avg_sil <- map_dbl(2:10, ~ {
  km <- kmeans(numeric_data, ., nstart = 25)
  silhouette_score <- silhouette(km$cluster, dist(numeric_data))
  mean(silhouette_score[, 3])
})
```

[8]:
```
elbow_plot <- ggplot(data.frame(K = 1:10, WCSS = wcss), aes(K, WCSS)) +
  geom_line(color = "steelblue", size = 1.2) +
  geom_point(color = "red", size = 3) +
  labs(title = "Elbow Method (Optimal K)", x = "Number of Clusters (K)", y =␣
↪"WCSS") +
  theme_minimal()

silhouette_plot <- ggplot(data.frame(K = 2:10, Silhouette = avg_sil), aes(K,␣
↪Silhouette)) +
  geom_line(color = "steelblue", size = 1.2) +
  geom_point(color = "red", size = 3) +
  labs(title = "Silhouette Score (Optimal K)", x = "Number of Clusters (K)", y␣
↪= "Avg. Silhouette Width") +
  theme_minimal()
```

```
grid.arrange(elbow_plot, silhouette_plot, ncol = 2)
```

```
Warning message:
"Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
 Please use `linewidth` instead."
```




[9]:
```
k2 <- kmeans(numeric_data, centers = 2, nstart = 25)
k3 <- kmeans(numeric_data, centers = 3, nstart = 25)
k4 <- kmeans(numeric_data, centers = 4, nstart = 25)
k5 <- kmeans(numeric_data, centers = 5, nstart = 25)
```

[10]:
```
happiness$Cluster_K2 <- as.factor(k2$cluster)
happiness$Cluster_K3 <- as.factor(k3$cluster)
```

```
happiness$Cluster_K4 <- as.factor(k4$cluster)
happiness$Cluster_K5 <- as.factor(k5$cluster)
```

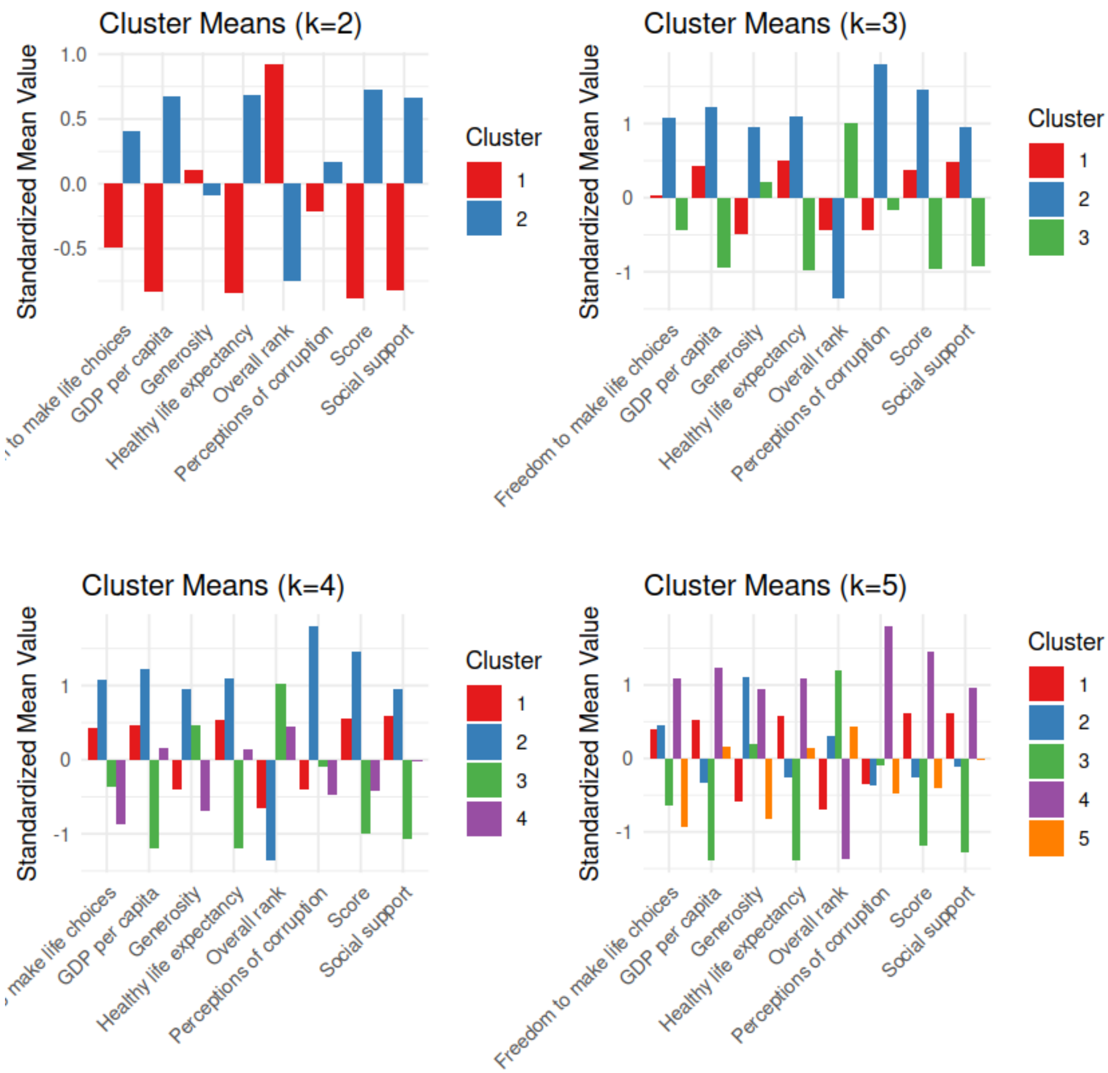
[11]:
```
plot_cluster_means <- function(km_result, title) {
  centers <- as.data.frame(km_result$centers)
  centers$Cluster <- factor(rownames(centers))

  centers_long <- centers %>%
    pivot_longer(cols = -Cluster, names_to = "Feature", values_to = 
 ↪"Mean_Value")

  ggplot(centers_long, aes(x = Feature, y = Mean_Value, fill = Cluster)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(title = title, y = "Standardized Mean Value", x = "") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    scale_fill_brewer(palette = "Set1")
}
```

[12]:
```
p2 <- plot_cluster_means(k2, "Cluster Means (k=2)")
p3 <- plot_cluster_means(k3, "Cluster Means (k=3)")
p4 <- plot_cluster_means(k4, "Cluster Means (k=4)")
p5 <- plot_cluster_means(k5, "Cluster Means (k=5)")

grid.arrange(p2, p3, p4, p5, ncol = 2)
```


Cluster Means (k=2)
Standardized Mean Value
1.0
0.5
0.0
-0.5
GDP per capita
Generosity
Healthy life expectancy
Overall rank
Perceptions of corruption
Score
Social support
Cluster
1
2
Cluster Means (k=3)
Standardized Mean Value
1
0
-1
Freedom to make life choices
GDP per capita
Generosity
Healthy life expectancy
Overall rank
Perceptions of corruption
Score
Social support
Cluster
1
2
3
Cluster Means (k=4)
Standardized Mean Value
1
0
-1
GDP per capita
Generosity
Healthy life expectancy
Overall rank
Perceptions of corruption
Score
Social support
Cluster
1
2
3
4
Cluster Means (k=5)
Standardized Mean Value
1
0
-1
Freedom to make life choices
GDP per capita
Generosity
Healthy life expectancy
Overall rank
Perceptions of corruption
Score
Social support
Cluster
1
2
3
4
5

# Experiment 7

April 21, 2025

[ ]:
```
# Load required libraries
set.seed(1)
library(tidyverse)
library(caret)
library(glmnet)
library(mlbench)
library(randomForest)
```

[ ]:
```
# Load Pima Indians Diabetes dataset
data("PimaIndiansDiabetes2")
df <- PimaIndiansDiabetes2
```

[ ]:
```
# Check structure & missing values
glimpse(df)
summary(df)
df <- na.omit(df)
```

[ ]:
```
preProc <- preProcess(df[, -9], method = c("center", "scale"))
df_scaled <- predict(preProc, df)
```

[ ]:
```
df_scaled <- df_scaled %>% mutate(bmi_age_ratio = mass / age)
```

[ ]:
```
cor_matrix <- cor(df_scaled %>% select(-diabetes))
corrplot(cor_matrix, method = "color", type = "upper", tl.cex = 0.7)
```

[ ]:
```
ctrl <- rfeControl(functions = rfFuncs, method = "cv", number = 10)
rfe_result <- rfe(
  x = df_scaled %>% select(-diabetes),
  y = df_scaled$diabetes,
  sizes = 1:8, # Test subsets of 1 to 8 features
  rfeControl = ctrl
)

# Top selected features
print(rfe_result)
plot(rfe_result, type = c("g", "o"))
```

[ ]:
```
x <- model.matrix(diabetes ~ ., df_scaled)[, -1] # Exclude intercept
y <- ifelse(df_scaled$diabetes == "pos", 1, 0)

# Fit LASSO
cv_lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")
plot(cv_lasso)

# Coefficients at optimal lambda
coef(cv_lasso, s = "lambda.min")
```

[ ]:
```
trainIndex <- createDataPartition(df_scaled$diabetes, p = 0.8, list = FALSE)
train <- df_scaled[trainIndex, ]
test <- df_scaled[-trainIndex, ]
```

[ ]:
```
model_all <- train(
  diabetes ~ .,
  data = train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

pred_all <- predict(model_all, test)
confusionMatrix(pred_all, test$diabetes)
```

[ ]:
```
model_selected <- train(
  diabetes ~ glucose + mass + bmi_age_ratio,
  data = train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)

# Predictions
pred_selected <- predict(model_selected, test)
confusionMatrix(pred_selected, test$diabetes)
```

[ ]:
```

```

# Experiment 8

April 21, 2025

[1]:
```
library(tidyverse)
library(dplyr)
```

```
 Attaching core tidyverse packages

tidyverse 2.0.0
 dplyr     1.1.4     readr     2.1.5
 forcats   1.0.0     stringr   1.5.1
 ggplot2   3.5.2     tibble    3.2.1
 lubridate 1.9.4     tidyr     1.3.1
 purrr     1.0.4
 Conflicts

tidyverse_conflicts()
 dplyr::filter() masks stats::filter()
 dplyr::lag()    masks stats::lag()
 Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts to
become errors
```

[2]:
```
# Basic way to load a CSV
train <- read.csv("/home/asus/content/Notes/Semester 4/FDN Lab/Experiments/
↪Experiment 8/titanic/train.csv")

test <- read.csv("/home/asus/content/Notes/Semester 4/FDN Lab/Experiments/
↪Experiment 8/titanic/test.csv")
```

[3]:
```
# Removing Unessacry Cols
train <- train %>% select(-one_of("Cabin", "Ticket", "Name", "Embarked"))
test <- test %>% select(-one_of("Cabin", "Ticket", "Name", "Embarked"))
```

[4]:
```
train <- train %>% fill(everything(), .direction = "down")
test <- test %>% fill(everything(), .direction = "down")
```

[5]:
```
X_train <- train %>% select(-Survived)
Y_train <- train %>% select(Survived)
```

[6]:
```
train_df <- as_tibble(train) %>%
  mutate(Survived = train)
```

[7]:
```
# Train the model
logit_model <- glm(Survived ~ .,
                   data = train,
                   family = binomial)
```

[8]:
```
predictions <- predict(logit_model, newdata = train, type = "response")
```

[9]:
```
predicted_classes <- ifelse(predictions > 0.5, 1, 0)
```

[10]:
```
ground_truth <- train$Survived
```

[11]:
```
conf_matrix <- table(Predicted = predicted_classes, Actual = ground_truth)
```

[12]:
```
print(conf_matrix)
# Actual
# Predicted   0   1
#           0 472 110
#           1  77 232
```

```
         Actual
Predicted   0   1
        0 472 110
        1  77 232
```

[13]:
```
accuracy <- sum(diag(conf_matrix))/sum(conf_matrix)
precision <- conf_matrix[2,2]/sum(conf_matrix[2,])
recall <- conf_matrix[2,2]/sum(conf_matrix[,2])
f1_score <- 2 * (precision * recall) / (precision + recall)
```

[14]:
```
summary(logit_model)
print(conf_matrix)
cat("\nAccuracy:", round(accuracy, 3))
cat("\nPrecision:", round(precision, 3))
cat("\nRecall/Sensitivity:", round(recall, 3))
cat("\nF1 Score:", round(f1_score, 3))
cat("\nSpecificity:", round(conf_matrix[1,1]/sum(conf_matrix[,1]), 3))
```

```
Call:
glm(formula = Survived ~ ., family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6513  -0.6196  -0.4077   0.6269   2.6737
```
28

```
Coefficients:
```

```
              Estimate Std. Error z value Pr(>|z|)    
(Intercept)  4.6705255  0.5301556   8.810  < 2e-16 ***
PassengerId  0.0000850  0.0003468   0.245  0.80639    
Pclass      -1.0457412  0.1374434  -7.609 2.77e-14 ***
Sexmale     -2.8025118  0.2007943 -13.957  < 2e-16 ***
Age         -0.0338376  0.0067970  -4.978 6.42e-07 ***
SibSp       -0.3422950  0.1094887  -3.126  0.00177 ** 
Parch       -0.1195347  0.1171594  -1.020  0.30760    
Fare         0.0031898  0.0023918   1.334  0.18233    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1186.66  on 890  degrees of freedom
Residual deviance:  790.18  on 883  degrees of freedom
AIC: 806.18

Number of Fisher Scoring iterations: 5


         Actual
Predicted   0   1
        0 472 110
        1  77 232

Accuracy: 0.79
Precision: 0.751
Recall/Sensitivity: 0.678
F1 Score: 0.713
Specificity: 0.86
```