```r
# 21bds0215
# chinmay paliwal
# Module 2
# Load required library
library(dplyr)

# Load the dataset
data <- read.csv("/Users/chinmaypaliwal/Desktop/happiness.csv")

# Step 1: Remove duplicate rows
data <- distinct(data)

# Step 2: Replace missing values
# Numeric columns: Replace NA with mean
numeric_cols <- sapply(data, is.numeric)
data[, numeric_cols] <- lapply(data[, numeric_cols], function(x) {
  ifelse(is.na(x), mean(x, na.rm = TRUE), x)
})

# Categorical columns: Replace NA with mode
categorical_cols <- sapply(data, is.character)
for (col in names(data)[categorical_cols]) {
  mode_value <- names(sort(table(data[[col]]), decreasing = TRUE))[1]
  data[[col]][is.na(data[[col]])] <- mode_value
}

# Step 3: Standardize categorical values (convert to lowercase)
data[, categorical_cols] <- lapply(data[, categorical_cols], tolower)

# Save the cleaned dataset
write.csv(data, "cleaned_happiness.csv", row.names = FALSE)




# Module 3
# Load required libraries
library(dplyr)
library(forecast)

# Load the dataset
data <- read.csv("/Users/chinmaypaliwal/Desktop/happiness.csv")

# Step 1: Aggregate data by year
# Replace 'prestige' with the column to analyze over time
time_series_data <- data %>%
  group_by(year) %>%
  summarise(avg_happiness = mean(prestige, na.rm = TRUE))  # Replace 'prestige' with
relevant column

# Step 2: Create a time series object
# Replace 'avg_happiness' with the column you're analyzing
happiness_ts <- ts(time_series_data$avg_happiness, start = min(time_series_data$year),
frequency = 1)

# Step 3: Open a larger plotting device
x11()  # Opens a new graphics window (works on Windows/Linux)

# Adjust margins before plotting
par(mar = c(5, 5, 4, 2))

# Plot the time series
plot(happiness_ts, main = "Happiness Over Time", xlab = "Year", ylab = "Average
Happiness", col = "blue")

# Step 4: Decompose the time series
decomposed <- decompose(happiness_ts)
x11()  # Open a new graphics window for decomposition plot
par(mar = c(5, 5, 4, 2))  # Adjust margins
plot(decomposed)

# Step 5: Apply ARIMA model for forecasting
fit <- auto.arima(happiness_ts)
summary(fit)

# Step 6: Forecast the next 5 years
forecasted <- forecast(fit, h = 5)

# Open a new plotting window for forecast plot
x11()
par(mar = c(5, 5, 4, 2))  # Adjust margins
plot(forecasted)

# Optional: Save the plots to files
png("happiness_time_series_plot.png", width = 800, height = 600)
plot(happiness_ts, main = "Happiness Over Time", xlab = "Year", ylab = "Average
Happiness", col = "blue")
dev.off()

png("happiness_forecast_plot.png", width = 800, height = 600)
plot(forecasted, main = "Happiness Forecast", xlab = "Year", ylab = "Predicted Average
Happiness")
dev.off()
```

```r
# Save the aggregated time series data
write.csv(time_series_data, "aggregated_time_series_data.csv", row.names = FALSE)




# Module 4
# Load required libraries
library(dplyr)
library(ggplot2)
library(moments)  # For skewness and kurtosis

# Load the dataset
data <- read.csv("/Users/chinmaypaliwal/Desktop/happiness.csv")

# Step 1: Separate numeric and categorical columns
numeric_cols <- sapply(data, is.numeric)
categorical_cols <- sapply(data, is.character)

# Step 2: 1-D Statistical Analysis
# Numeric columns: Calculate descriptive statistics
numeric_summary <- data %>%
  select(which(numeric_cols)) %>%
  summarise_all(list(
    mean = ~mean(., na.rm = TRUE),
    median = ~median(., na.rm = TRUE),
    sd = ~sd(., na.rm = TRUE),
    var = ~var(., na.rm = TRUE),
    min = ~min(., na.rm = TRUE),
    max = ~max(., na.rm = TRUE),
    skewness = ~skewness(., na.rm = TRUE),
    kurtosis = ~kurtosis(., na.rm = TRUE)
  ))

print("1-D Descriptive Statistics for Numeric Columns:")
print(numeric_summary)

# Categorical columns: Frequency distributions
print("1-D Frequency Distributions for Categorical Columns:")
for (col in names(data)[categorical_cols]) {
  print(paste("Column:", col))
  print(table(data[[col]]))
}

# Step 3: 2-D Statistical Analysis
# Pairwise correlation matrix for numeric columns
print("2-D Correlation Matrix for Numeric Columns:")
cor_matrix <- cor(data[, numeric_cols], use = "complete.obs")
print(cor_matrix)

# Categorical vs numeric analysis: Boxplots
for (cat_col in names(data)[categorical_cols]) {
  for (num_col in names(data)[numeric_cols]) {
    print(ggplot(data, aes_string(x = cat_col, y = num_col)) +
            geom_boxplot() +
            ggtitle(paste("Boxplot of", num_col, "by", cat_col)) +
            theme_minimal())
  }
}

# Scatterplot matrix for numeric columns
print("Scatterplot Matrix for Numeric Columns:")
pairs(data[, numeric_cols], main = "Scatterplot Matrix")

# Save results to a CSV file
write.csv(numeric_summary, "1D_numeric_summary.csv", row.names = FALSE)
write.csv(cor_matrix, "2D_correlation_matrix.csv")




#Module 5
# Load required libraries
library(dplyr)
library(ggplot2)

# Load the dataset
data <- read.csv("/Users/chinmaypaliwal/Desktop/happiness.csv")

# Step 1: Preprocess the data
# Remove non-numeric columns (if any), keeping only relevant columns for clustering
numeric_data <- data %>%
  select_if(is.numeric) %>%
  na.omit()  # Remove rows with missing values

# Step 2: Scale the data (important for K-means)
scaled_data <- scale(numeric_data)

# Step 3: Determine the optimal number of clusters using the Elbow method
wss <- sapply(1:10, function(k) {
  kmeans(scaled_data, centers = k, nstart = 10)$tot.withinss
})
```

```
# Plot the Elbow curve to visualize the optimal k
plot(1:10, wss, type = "b", pch = 19, frame = FALSE,
     xlab = "Number of Clusters", ylab = "Within-Cluster Sum of Squares",
     main = "Elbow Method for Optimal K")

# Step 4: Fit the K-means model (assuming the elbow point suggests k = 3, but adjust as
needed)
optimal_k <- 3  # Replace this with your optimal number based on the Elbow method plot
kmeans_result <- kmeans(scaled_data, centers = optimal_k, nstart = 10)

# Step 5: View K-means results
print("Cluster Centers:")
print(kmeans_result$centers)

# Add cluster labels to the original data
data$cluster <- as.factor(kmeans_result$cluster)

# Step 6: Visualize the clusters (assuming you have 2 main dimensions to plot)
# Here we use the first two principal components for visualization
pca_result <- prcomp(scaled_data)
pca_data <- as.data.frame(pca_result$x)

# Plot the first two principal components with cluster labels
ggplot(pca_data, aes(x = PC1, y = PC2, color = data$cluster)) +
  geom_point(size = 3) +
  labs(title = "K-means Clustering Results", x = "Principal Component 1", y = "Principal
Component 2") +
  theme_minimal()

# Optional: Save the clustering results
write.csv(data, "happiness_with_clusters.csv", row.names = FALSE)




# knn cluster
# Load required libraries
library(dplyr)
library(class)  # For KNN
library(caret)  # For train/test split
library(ggplot2)

# Load the dataset
data <- read.csv("/Users/chinmaypaliwal/Desktop/happiness.csv")

# Step 1: Preprocess the data
# Remove non-numeric columns (if any), keeping only relevant columns for KNN
numeric_data <- data %>%
  select_if(is.numeric) %>%
  na.omit()  # Remove rows with missing values

# Step 2: Scale the data (important for KNN)
scaled_data <- scale(numeric_data)

# Step 3: Create a response variable (class labels)
# We assume a variable (e.g., 'prestige') is used as the class for KNN classification.
# This step requires a categorical or binary response column.

# For this example, we will create a binary response:
# (e.g., if 'prestige' > median(prestige) then class 1, else class 0).
response <- ifelse(data$prestige > median(data$prestige, na.rm = TRUE), 1, 0)
response <- as.factor(response)  # Convert to factor for classification

# Add the response variable to the dataset
data <- cbind(numeric_data, response)

# Step 4: Split the data into training and testing sets (70% train, 30% test)
set.seed(123)  # For reproducibility
trainIndex <- createDataPartition(data$response, p = 0.7, list = FALSE)
train_data <- data[trainIndex, ]
test_data <- data[-trainIndex, ]

# Step 5: Apply KNN model
# Select K (e.g., K = 3 here)
k_value <- 3
knn_pred <- knn(train = train_data[, -ncol(train_data)],  # Exclude response column
                test = test_data[, -ncol(test_data)],  # Exclude response column
                cl = train_data$response,  # Class labels
                k = k_value)

# Step 6: Evaluate the KNN model
confusion_matrix <- confusionMatrix(knn_pred, test_data$response)
print(confusion_matrix)

# Step 7: Visualize the results (using the first 2 principal components for simplicity)
# PCA for dimensionality reduction (since KNN works on feature space)
pca_result <- prcomp(scaled_data)
pca_data <- as.data.frame(pca_result$x)

# Combine PCA with response labels for visualization
pca_data$response <- response

# Plot the first two principal components with KNN class labels
```

```
ggplot(pca_data, aes(x = PC1, y = PC2, color = response)) +
  geom_point(size = 3) +
  labs(title = "KNN Classification Results", x = "Principal Component 1", y = "Principal
Component 2") +
  theme_minimal()

# Optional: Save the results
write.csv(data, "happiness_knn_results.csv", row.names = FALSE)




# hirerchial clustering
# Load required libraries
library(dplyr)
library(ggplot2)

# Load the dataset
data <- read.csv("/Users/chinmaypaliwal/Desktop/happiness.csv")

# Step 1: Preprocess the data
# Remove non-numeric columns (if any), keeping only relevant columns for clustering
numeric_data <- data %>%
  select_if(is.numeric) %>%
  na.omit()  # Remove rows with missing values

# Step 2: Scale the data (important for clustering)
scaled_data <- scale(numeric_data)

# Step 3: Compute the distance matrix using Euclidean distance
dist_matrix <- dist(scaled_data, method = "euclidean")

# Step 4: Perform hierarchical clustering using complete linkage method
hc <- hclust(dist_matrix, method = "complete")

# Step 5: Plot the dendrogram
plot(hc, main = "Dendrogram of Hierarchical Clustering", xlab = "Samples", ylab =
"Distance", hang = -1)

# Step 6: Cut the dendrogram to form clusters (choose number of clusters, e.g., 3)
k <- 3  # Set the number of clusters (this can be adjusted based on dendrogram)
clusters <- cutree(hc, k)

# Step 7: Add cluster labels to the dataset
data$cluster <- as.factor(clusters)
```

```r
# Step 8: Visualize the clusters (using first two principal components for visualization)
pca_result <- prcomp(scaled_data)
pca_data <- as.data.frame(pca_result$x)

# Add cluster labels to PCA data for visualization
pca_data$cluster <- as.factor(clusters)

# Plot the first two principal components with cluster labels
ggplot(pca_data, aes(x = PC1, y = PC2, color = cluster)) +
  geom_point(size = 3) +
  labs(title = "Hierarchical Clustering Results", x = "Principal Component 1", y = "Principal
Component 2") +
  theme_minimal()

# Optional: Save the clustering results to a CSV file
write.csv(data, "happiness_with_clusters_hierarchical.csv", row.names = FALSE)




# Module 6
# Load required libraries
library(dplyr)
library(ggplot2)

# Load the dataset
data <- read.csv("/Users/chinmaypaliwal/Desktop/happiness.csv")

# Step 1: Preprocess the data
# Select only numeric columns and remove missing values
numeric_data <- data %>%
  select_if(is.numeric) %>%
  na.omit()  # Remove rows with missing values

# Step 2: Scale the data (important for PCA)
scaled_data <- scale(numeric_data)

# Step 3: Perform PCA
pca_result <- prcomp(scaled_data, center = TRUE, scale. = TRUE)

# Step 4: Summary of PCA
# The summary provides the proportion of variance explained by each principal component
summary(pca_result)

# Step 5: Plot the explained variance (Scree plot)
# This plot shows the proportion of variance explained by each principal component
screeplot(pca_result, main = "Scree Plot", col = "blue", pch = 19)
```

```
# Step 6: Biplot of the first two principal components
# This plot shows how the samples and variables are distributed in the first two principal
components
biplot(pca_result, main = "PCA Biplot")

# Step 7: Create a data frame with the principal components
pca_data <- as.data.frame(pca_result$x)

# Step 8: Visualize the first two principal components in a scatter plot
ggplot(pca_data, aes(x = PC1, y = PC2)) +
  geom_point(size = 3, color = "darkblue") +
  labs(title = "PCA: First Two Principal Components", x = "Principal Component 1", y =
"Principal Component 2") +
  theme_minimal()

# Optional: Save the PCA results to a CSV file
write.csv(pca_data, "happiness_pca_results.csv", row.names = FALSE)
```