Experiment 5: Data Manipulation & Visualization in R

Objective:
Visualize data using R packages and apply data manipulation using `dplyr`, `data.table`, `reshape2`, `tidyr`, etc.

Code:

```
library(dplyr)
library(data.table)
library(reshape2)
library(tidyr)
library(ggplot2)

data <- as.data.table(mtcars)
data$model <- rownames(mtcars)

summary_tbl <- data %>%
  group_by(cyl) %>%
  summarise(avg_mpg = mean(mpg), avg_hp = mean(hp))

melted_data <- melt(data, id.vars = "model", measure.vars = c("mpg", "hp"))
wide_data <- pivot_wider(melted_data, names_from = variable, values_from =
value)

ggplot(data, aes(x = factor(cyl), y = mpg, fill = factor(cyl))) +
  geom_boxplot() +
  labs(title = "MPG Distribution by Cylinder Count", x = "Cylinders", y =
"Miles per Gallon") +
  theme_minimal()
```

Coverage:

- Data summarization
- Reshaping and tidying
- Visualization with `ggplot2`

Experiment 6: Visualizing LIME Explanations in R

Objective:
Visualize the importance of LIME for interpreting black-box models.

Code:

```
library(lime)
library(caret)
library(ggplot2)

data(iris)
iris$Species <- as.factor(iris$Species)

model <- train(Species ~ ., data = iris, method = "rf")
explainer <- lime(iris[, -5], model)
```

```
samples <- iris[1:5, -5]

explanation <- explain(samples, explainer, n_labels = 1, n_features = 3)
plot_features(explanation)
```

Coverage:

- Model interpretation using LIME
- Feature importance visualization

Experiment 7: Linear & Logistic Regression in R

Objective:
Build and evaluate linear and logistic regression models.

Code:

```
library(caret)
library(ggplot2)

# Linear Regression
data("mtcars")
lm_model <- train(mpg ~ hp + wt, data = mtcars, method = "lm")
lm_pred <- predict(lm_model, mtcars)

ggplot(mtcars, aes(x = mpg, y = lm_pred)) +
  geom_point(color = "blue") +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red")
+
  labs(title = "Linear Regression: Actual vs Predicted MPG", x = "Actual
MPG", y = "Predicted MPG") +
  theme_minimal()

# Logistic Regression
data("iris")
iris$binary_species <- as.factor(ifelse(iris$Species == "setosa", "setosa",
"other"))
log_model <- train(binary_species ~ Sepal.Length + Petal.Length, data =
iris, method = "glm", family = "binomial")
log_pred <- predict(log_model, iris)

conf_mat <- confusionMatrix(log_pred, iris$binary_species)
print(conf_mat)
```

Coverage:

- Regression modeling
- Prediction and evaluation
- Confusion matrix analysis

Experiment 8: Association Rule Mining in R

Objective:
Demonstrate association rule mining using Apriori algorithm.

Code:

```
library(arules)
library(arulesViz)

data("Groceries")
rules <- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.5,
minlen = 2))

inspect(head(rules, 5))
plot(rules, method = "graph", engine = "htmlwidget")
```

Coverage:

- Frequent itemset mining
- Rule generation and visualization

Experiment 9: Text Mining & Sentiment Analysis (R Code)

```
# Install & load required packages (run once)
# install.packages(c("tm", "SnowballC", "syuzhet", "wordcloud",
"RColorBrewer"))

library(tm)
library(SnowballC)
library(syuzhet)
library(wordcloud)
library(RColorBrewer)

# Step 1: Input text data
text_data <- c(
  "I love this product! It works wonderfully.",
  "This is the worst experience I've ever had.",
  "Not bad, could be better.",
  "Absolutely fantastic! Highly recommend it.",
  "I am very disappointed with the service."
)

# Step 2: Create and preprocess corpus
corpus <- Corpus(VectorSource(text_data))
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, stemDocument)

# Step 3: Text mining – term-document matrix & word cloud
tdm <- TermDocumentMatrix(corpus)
word_freqs <- sort(rowSums(as.matrix(tdm)), decreasing = TRUE)
```

```
wordcloud(names(word_freqs), word_freqs, max.words = 80, colors =
brewer.pal(8, "Dark2"))

# Step 4: Sentiment analysis
sentiments <- get_nrc_sentiment(text_data)
sentiment_scores <- colSums(sentiments)

# Step 5: Visualization & output
barplot(sentiment_scores, las = 2, col = rainbow(10),
        main = "Sentiment Distribution", ylab = "Frequency")
print(sentiment_scores)
```

🎯 What this experiment demonstrates

- **Text Preprocessing:** Cleaning, stemming, and stopword removal.
- **Text Mining:** Creation of Term-Document Matrix and Word Cloud.
- **Sentiment Analysis:** Using NRC lexicon (positive, negative, joy, fear, etc.).
- **Visualization:** Word cloud and sentiment bar chart.
