Summarized

Arpan Sapkota

2023-07-25

Summarized

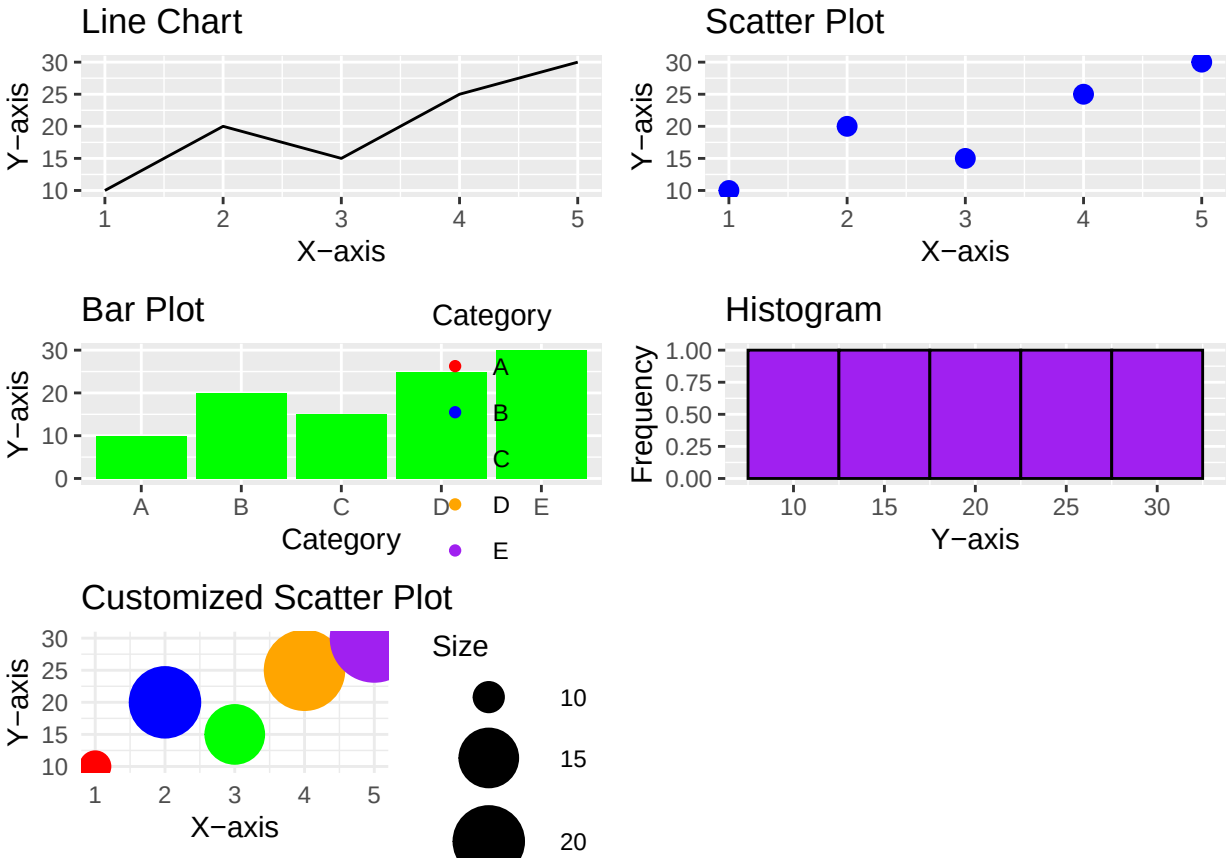
1. R Script and R Studio Basics:
    - Use R Script in R Studio to write and execute R code.
    - Knit R script to HTML or PDF for documentation.
2. Data Manipulation in R:
    - Create data frames and vectors using R.
    - Perform basic operations like addition, subtraction, multiplication, and division in R.
    - Handle missing values and convert data types.
    - Use subsetting and indexing to access specific elements in R objects.

```r
# Create vectors and data frame
my_vector <- c(1, 2, 3, 4, 5)
my_data <- data.frame(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 28, 22, 27),
  Score = c(90, 85, 92, 78, 88)
)

# Perform basic operations
result_add <- my_vector + 5
result_sub <- my_vector - 2
result_mul <- my_vector * 3
result_div <- my_vector / 2

# Handle missing values and convert data types
my_vector_with_na <- c(1, 2, NA, 4, NA, 6)
clean_vector <- na.omit(my_vector_with_na)

numeric_vector <- c(1, 2, 3, 4, 5)
character_vector <- as.character(numeric_vector)

# Use subsetting and indexing
element_3 <- my_vector[3]

names_column <- my_data$Name
row_3 <- my_data[my_data$ID == 3, ]
```

3. Data Visualization with ggplot2:

- Create various types of plots (line chart, scatter plot, bar plot, histogram) using ggplot2.
- Customize plot aesthetics (colors, sizes, labels) in ggplot2.

```r
# Load necessary libraries
library(ggplot2)

# Sample data
df <- data.frame(
  x = c(1, 2, 3, 4, 5),
  y = c(10, 20, 15, 25, 30),
  category = c("A", "B", "C", "D", "E")
)

# Line chart
line_chart <- ggplot(df, aes(x, y)) +
  geom_line() +
  labs(title = "Line Chart", x = "X-axis", y = "Y-axis")

# Scatter plot
scatter_plot <- ggplot(df, aes(x, y)) +
  geom_point(size = 3, color = "blue") +
  labs(title = "Scatter Plot", x = "X-axis", y = "Y-axis")

# Bar plot
bar_plot <- ggplot(df, aes(x = category, y = y)) +
  geom_bar(stat = "identity", fill = "green") +
  labs(title = "Bar Plot", x = "Category", y = "Y-axis")

# Histogram
histogram_plot <- ggplot(df, aes(y)) +
  geom_histogram(binwidth = 5, fill = "purple", color = "black") +
  labs(title = "Histogram", x = "Y-axis", y = "Frequency")

# Customize plot aesthetics
custom_plot <- ggplot(df, aes(x, y, color = category, size = y)) +
  geom_point() +
  labs(title = "Customized Scatter Plot", x = "X-axis", y = "Y-axis") +
  theme_minimal() +
  scale_color_manual(values = c("red", "blue", "green", "orange", "purple")) +
  scale_size(range = c(5, 15)) +
  guides(color = guide_legend(title = "Category"), size = guide_legend(title = "Size"))

# Plot all the graphs
gridExtra::grid.arrange(line_chart, scatter_plot, bar_plot, histogram_plot, custom_plot, ncol = 2)
```



4. Data Analysis and Statistics:

- Perform descriptive statistics (mean, median, standard deviation) in R.
- Perform hypothesis tests and goodness-of-fit tests to check for normality and equality of variances.
- Perform independent sample t-tests and interpret the results.

```r
# Sample data
group1 <- c(10, 15, 12, 18, 20)
group2 <- c(22, 25, 28, 30, 35)

# Perform descriptive statistics
mean_group1 <- mean(group1)
median_group1 <- median(group1)
sd_group1 <- sd(group1)

mean_group2 <- mean(group2)
median_group2 <- median(group2)
sd_group2 <- sd(group2)

# Perform hypothesis test for normality (Shapiro-Wilk test)
normality_test_group1 <- shapiro.test(group1)
normality_test_group2 <- shapiro.test(group2)

# Perform hypothesis test for equality of variances (Levene's test)
variance_test <- var.test(group1, group2)
```

```r
# Perform independent sample t-test
t_test <- t.test(group1, group2)

# Print results
cat("Group 1 Descriptive Statistics:\n")
```

```
## Group 1 Descriptive Statistics:
```

```r
cat("Mean:", mean_group1, "  Median:", median_group1, "  Standard Deviation:", sd_group1, "\n\n")
```

```
## Mean: 15   Median: 15   Standard Deviation: 4.123106
```

```r
cat("Group 2 Descriptive Statistics:\n")
```

```
## Group 2 Descriptive Statistics:
```

```r
cat("Mean:", mean_group2, "  Median:", median_group2, "  Standard Deviation:", sd_group2, "\n\n")
```

```
## Mean: 28   Median: 28   Standard Deviation: 4.949747
```

```r
cat("Normality Test for Group 1:\n")
```

```
## Normality Test for Group 1:
```

```r
print(normality_test_group1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  group1
## W = 0.96356, p-value = 0.8325
```

```r
cat("Normality Test for Group 2:\n")
```

```
## Normality Test for Group 2:
```

```r
print(normality_test_group2)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  group2
## W = 0.98944, p-value = 0.9777
```

```r
cat("Equality of Variances Test:\n")
```

```
## Equality of Variances Test:
```

```
print(variance_test)
```

```
##
##  F test to compare two variances
##
## data:  group1 and group2
## F = 0.69388, num df = 4, denom df = 4, p-value = 0.7319
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.07224482 6.66436768
## sample estimates:
## ratio of variances
##          0.6938776
```

```
cat("Independent Sample t-test:\n")
```

```
## Independent Sample t-test:
```

```
print(t_test)
```

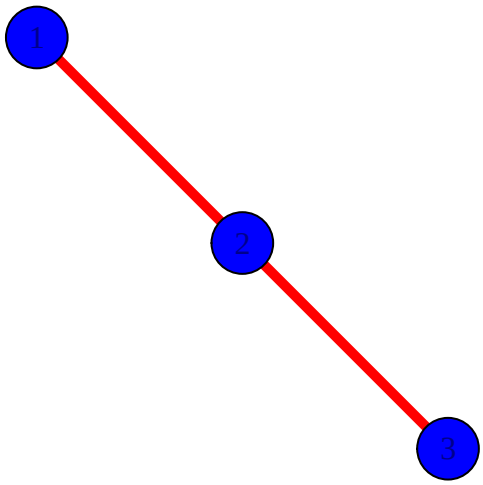
```
##
##  Welch Two Sample t-test
##
## data:  group1 and group2
## t = -4.5124, df = 7.747, p-value = 0.002136
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -19.681492  -6.318508
## sample estimates:
## mean of x mean of y
##        15        28
```

5. Social Network Analysis (SNA) using igraph:
    - Create graphs and plot them using the igraph package in R.
    - Compute graph metrics like degree, closeness, and betweenness centrality.

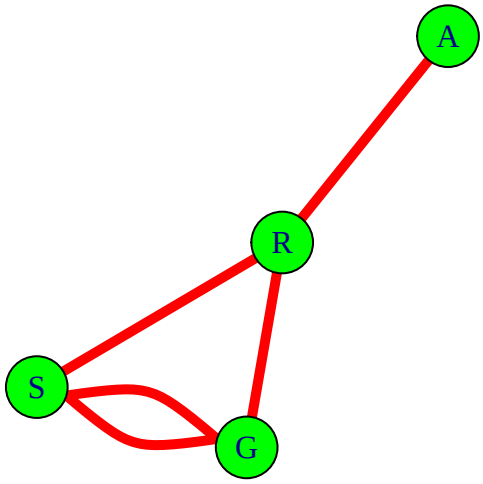
```
# Load the igraph package
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```r
# Create a graph object with edges (1, 2) and (2, 3)
g <- graph(edges=c(1, 2, 2, 3), directed=FALSE)

# Plot the graph
plot(g, vertex.color="blue", vertex.size=30, edge.color="red", edge.width=5)
```




```r
# Define another graph with elements "S", "R", "R", "G", "G", "S", "S", "G", "A", "R"
gl <- graph(c("S", "R", "R", "G", "G", "S", "S", "G", "A", "R"), directed=FALSE)

# Plot the graph with customized aesthetics
plot(gl, vertex.color="green", vertex.size=30, edge.color="red", edge.width=5)
```




```
# Compute graph metrics
degree_gl <- degree(gl) # Degree centrality
closeness_gl <- closeness(gl) # Closeness centrality
betweenness_gl <- betweenness(gl) # Betweenness centrality

# Print graph metrics
cat("Degree Centrality:\n")
```

```
## Degree Centrality:
```

```
print(degree_gl)
```

```
## S R G A
## 3 3 3 1
```

```
cat("Closeness Centrality:\n")
```

```
## Closeness Centrality:
```

```
print(closeness_gl)
```

```
##         S         R         G         A
## 0.2500000 0.3333333 0.2500000 0.2000000
```

```r
cat("Betweenness Centrality:\n")
```

```
## Betweenness Centrality:
```

```r
print(betweenness_gl)
```

```
## S R G A
## 0 2 0 0
```

6. Text Mining with tm and wordcloud:
    - Convert Twitter data into a data frame and create a corpus using the tm package.
    - Pre-process the text data for text mining.
    - Create a term document matrix and analyze term frequencies.
    - Create word clouds for visualization.

```r
# Load required libraries
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```r
# Create sample text data
data <- c("This is a sample text.",
          "Text mining is interesting.",
          "Word clouds are cool!")

# Step 1: Convert the data into a data frame
data_df <- data.frame(text = data)

# Step 2: Create a corpus using the "text" column in the data frame
text_corpus <- Corpus(VectorSource(data_df$text))

# Step 3: Pre-process the text data for text mining
text_corpus <- tm_map(text_corpus, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(text_corpus, content_transformer(tolower)):
## transformation drops documents
```

```r
text_corpus <- tm_map(text_corpus, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(text_corpus, removeNumbers): transformation
## drops documents
```

```r
text_corpus <- tm_map(text_corpus, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(text_corpus, removePunctuation): transformation
## drops documents
```

```r
text_corpus <- tm_map(text_corpus, removeWords, stopwords("en"))
```

```
## Warning in tm_map.SimpleCorpus(text_corpus, removeWords, stopwords("en")):
## transformation drops documents
```

```r
text_corpus <- tm_map(text_corpus, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(text_corpus, stripWhitespace): transformation
## drops documents
```

```r
# Step 4: Create a term document matrix and analyze term frequencies
term_matrix <- DocumentTermMatrix(text_corpus)
term_frequencies <- rowSums(as.matrix(term_matrix))

# Step 5: Create word clouds for visualization
wordcloud(words = names(term_frequencies), freq = term_frequencies, scale = c(5, 0.5), random.order = F
```

32

```
# Optional: If you want to view the corpus after pre-processing
inspect(text_corpus)
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 3
##
## [1]  sample text            text mining interesting word clouds cool
```

7. Machine Learning - Supervised Models:
    - Split data into training and test datasets for model building and evaluation.
    - Fit supervised models like linear regression, KNN regression, logistic regression, and Naive Bayes classification.
    - Evaluate models using confusion matrix, sensitivity, and specificity.

```
# Load required packages
library(class)
```

```
##
## Attaching package: 'class'

## The following object is masked from 'package:igraph':
##
##     knn
```

```r
# Generate random data
set.seed(123) # Set random seed for reproducibility

# Random data with 100 observations
data <- data.frame(
  age = sample(18:99, 100, replace = TRUE),
  sex = sample(c("male", "female"), 100, replace = TRUE),
  education = sample(c("No education", "Primary", "Secondary", "Beyond secondary"), 100, replace = TRUE
  socioeconomic_status = sample(c("Low", "Middle", "High"), 100, replace = TRUE),
  body_mass_index = runif(100, 14, 38)
)

# Step 2: Split data into training and test datasets
set.seed(123) # Set random seed for reproducibility
train_index <- sample(1:nrow(data), 0.8 * nrow(data)) # Index of training data rows
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

# Step 3: Fit supervised models

# Linear Regression
lm_model <- lm(body_mass_index ~ age + sex + education + socioeconomic_status, data = train_data)

# KNN Regression
# Remove rows with missing values in body_mass_index from training dataset
train_data <- train_data[!is.na(train_data$body_mass_index),]

#' knn_model <- knn(train_data[, -which(names(train_data) == "body_mass_index")],
#' test_data[, -which(names(test_data) == "body_mass_index")],
#' train_data$body_mass_index, k = 5)

# Logistic Regression
# logit_model <- glm(sex ~ age + education + socioeconomic_status, data = train_data, family = binomial

# Naive Bayes Classification
library(e1071)
nb_model <- naiveBayes(sex ~ age + education + socioeconomic_status, data = train_data)


# Step 4: Evaluate models

# For Logistic Regression and Naive Bayes Classification
#predicted_logit <- predict(logit_model, newdata = test_data, type = "response")
#predicted_nb <- predict(nb_model, newdata = test_data, type = "raw")

# Confusion matrix for Logistic Regression
#confusion_matrix_logit <- table(Actual = test_data$sex, Predicted = round(predicted_logit))
#confusion_matrix_logit

# Confusion matrix for Naive Bayes
#confusion_matrix_nb <- table(Actual = test_data$sex, Predicted = predicted_nb)
#confusion_matrix_nb
```

```
# Sensitivity and Specificity for Logistic Regression
#sensitivity_logit <- confusion_matrix_logit[2, 2] / sum(confusion_matrix_logit[2, ])
#specificity_logit <- confusion_matrix_logit[1, 1] / sum(confusion_matrix_logit[1, ])

# Sensitivity and Specificity for Naive Bayes
#sensitivity_nb <- confusion_matrix_nb[2, 2] / sum(confusion_matrix_nb[2, ])
#specificity_nb <- confusion_matrix_nb[1, 1] / sum(confusion_matrix_nb[1, ])
```

8. Dimensionality Reduction and Clustering:
    - Perform Principal Component Analysis (PCA) and hierarchical clustering.
    - Evaluate and interpret PCA results using Kaiser's criteria and scree plot.
    - Perform k-means clustering.

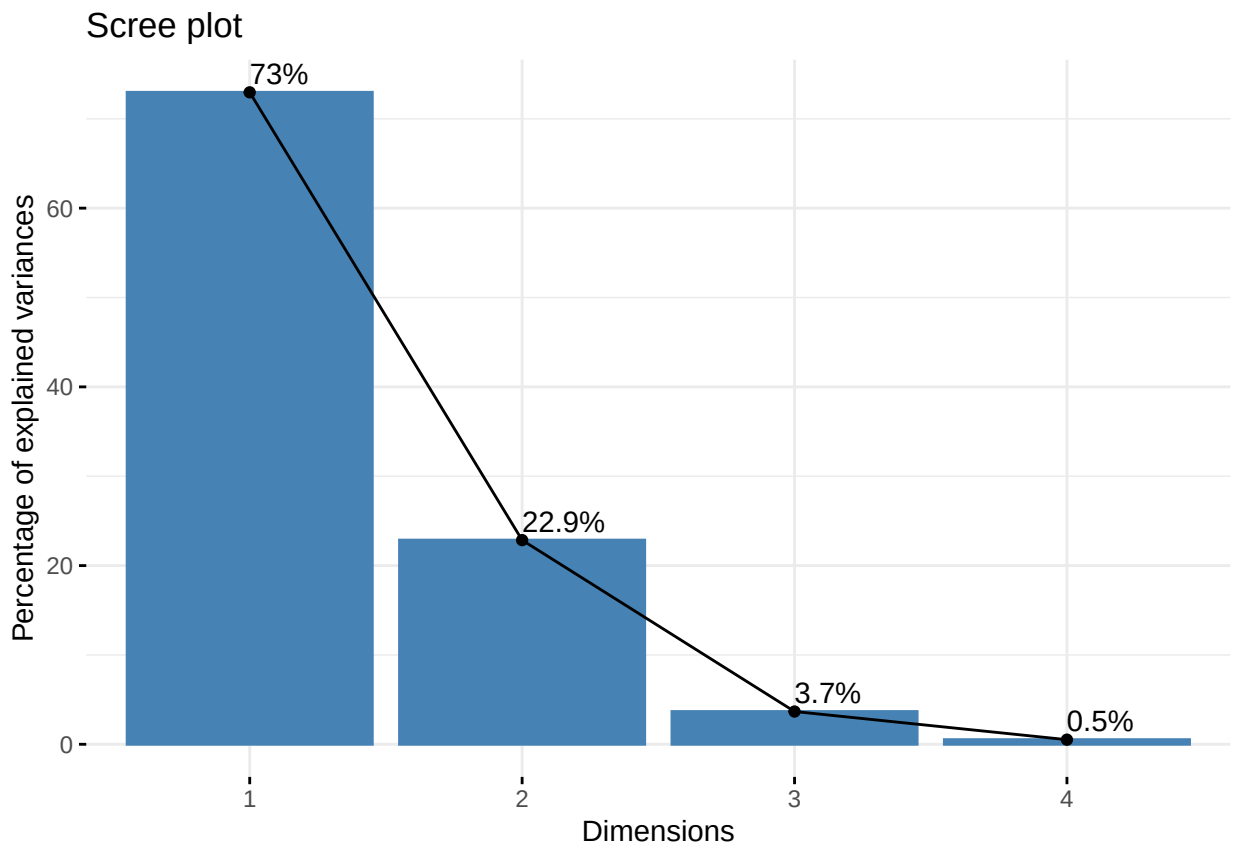
```
# Load required packages
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:igraph':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

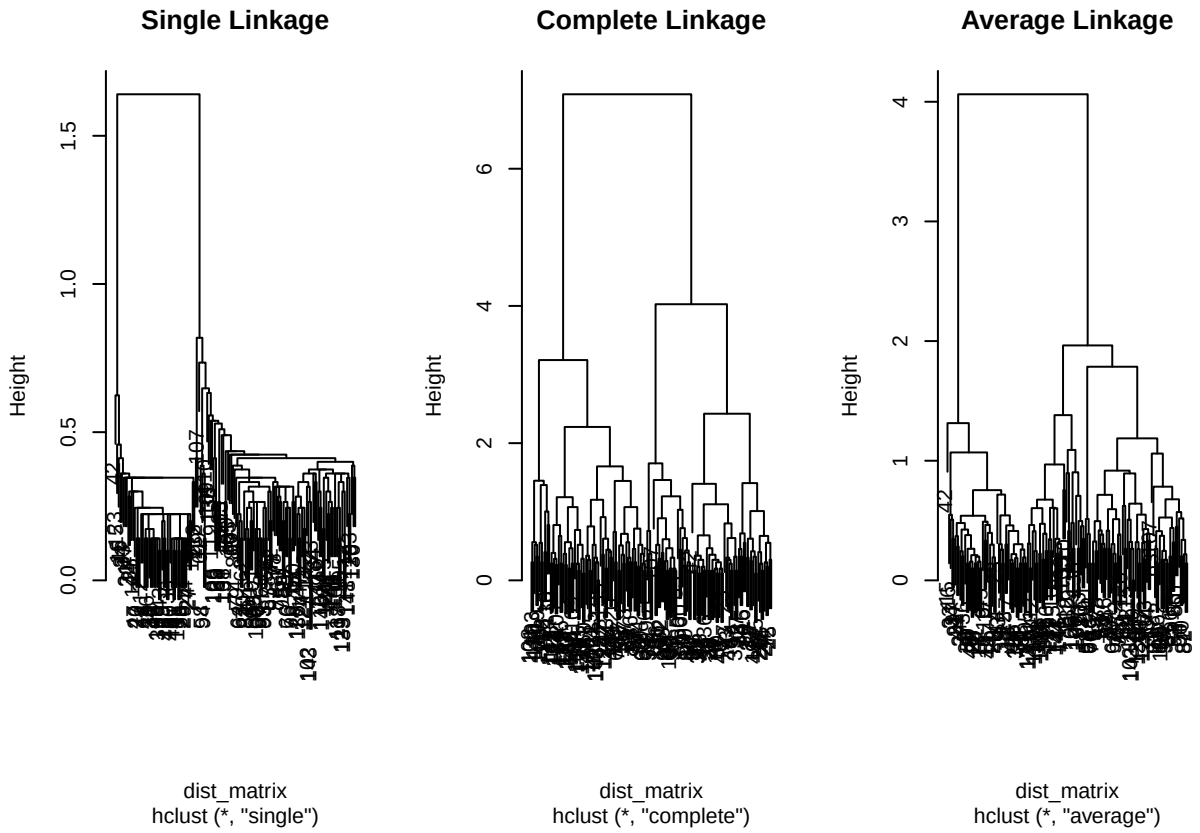
```
library(factoextra)
```

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

```
library(cluster)
library(ggplot2)

# Assuming you have already loaded the dataset, here we'll use the iris dataset for demonstration purpo

# Step 1: Perform Principal Component Analysis (PCA)

# Fit PCA model
pca_model <- prcomp(iris[, 1:4], scale. = TRUE)

# Evaluate PCA results using Kaiser's criteria
pca_eigenvalues <- pca_model$sdev^2
total_variance <- sum(pca_eigenvalues)
variance_explained <- pca_eigenvalues / total_variance * 100

# Scree plot
fviz_eig(pca_model, addlabels = TRUE)
```




```r
# Step 2: Perform Hierarchical Clustering

# Compute dissimilarity distance
dist_matrix <- dist(iris[, 1:4], method = "euclidean")

# Fit hierarchical clustering models
hclust_single <- hclust(dist_matrix, method = "single")
hclust_complete <- hclust(dist_matrix, method = "complete")
hclust_average <- hclust(dist_matrix, method = "average")

# Plot dendograms
par(mfrow = c(1, 3))
plot(hclust_single, main = "Single Linkage")
plot(hclust_complete, main = "Complete Linkage")
plot(hclust_average, main = "Average Linkage")
```

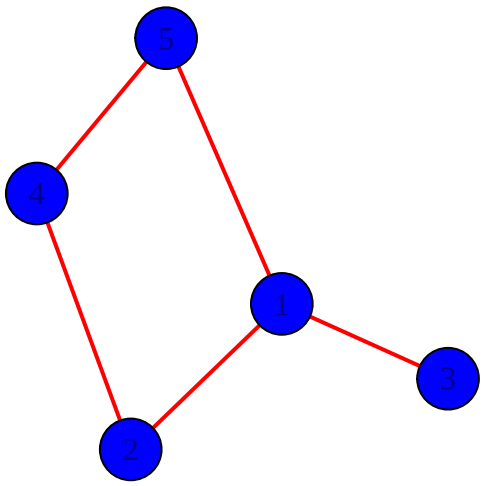



```r
# Step 3: Perform k-means Clustering

# Fit k-means clustering model with k=3 clusters
kmeans_model <- kmeans(iris[, 1:4], centers = 3)

# Plot k-means clusters
ggplot(data = iris, aes(x = Sepal.Length, y = Petal.Length, color = factor(kmeans_model$cluster))) +
  geom_point() +
  labs(title = "K-means Clustering", x = "Sepal Length", y = "Petal Length")
```

9. Graph Analysis:

   - Perform SNA using igraph with adjacency matrix.
   - Plot undirected SNA graphs and interpret results.

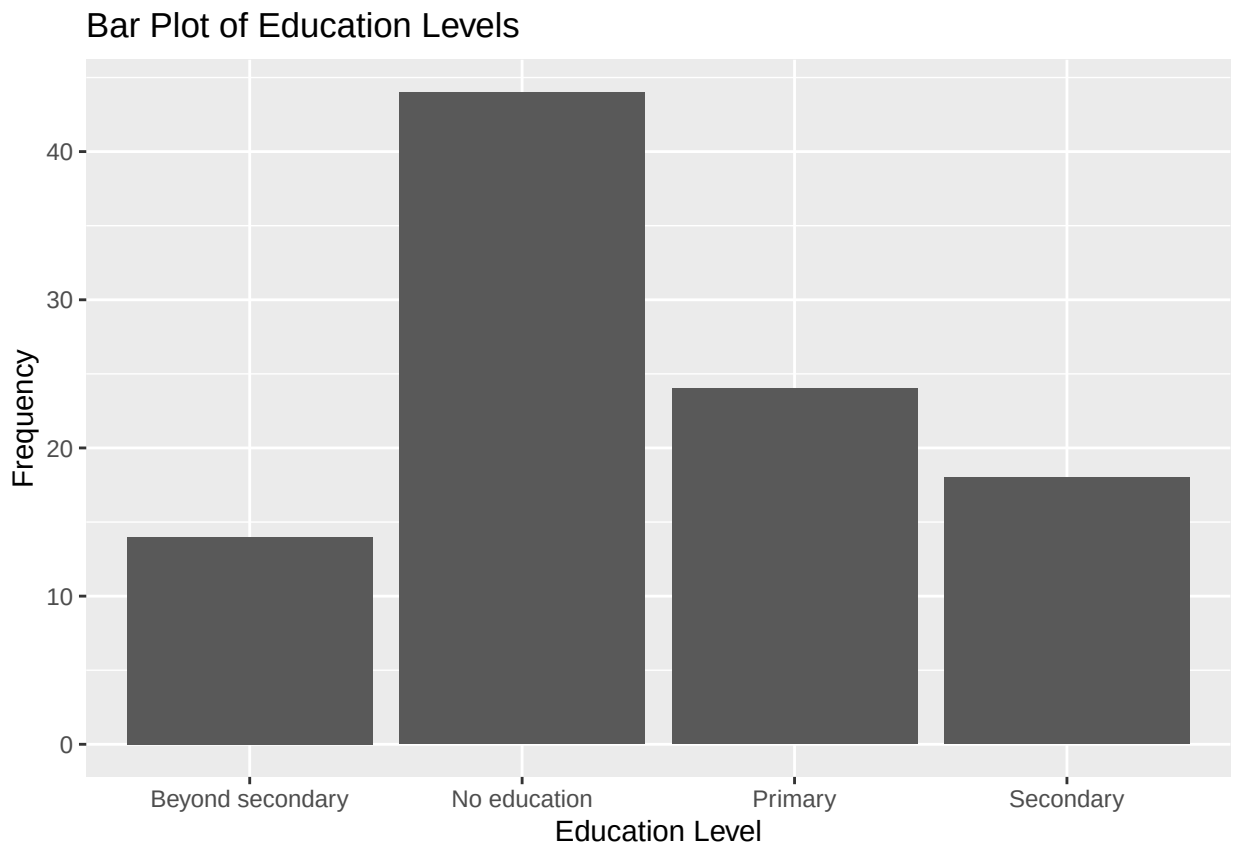
```
# Load required packages
library(igraph)

# Assuming you have already loaded or created the adjacency matrix, here we'll use a sample adjacency m
# Replace the sample_adj_matrix with your actual adjacency matrix

# Sample adjacency matrix
sample_adj_matrix <- matrix(c(0, 1, 1, 0, 1,
                              1, 0, 0, 1, 0,
                              1, 0, 0, 0, 0,
                              0, 1, 0, 0, 1,
                              1, 0, 0, 1, 0), nrow = 5, byrow = TRUE)

# Step 1: Create a graph object using the adjacency matrix
graph <- graph.adjacency(sample_adj_matrix, mode = "undirected", weighted = TRUE)

# Step 2: Plot the undirected SNA graph and interpret the results
plot(graph, vertex.color = "blue", vertex.size = 30, edge.color = "red", edge.width = 2, main = "Undire
```



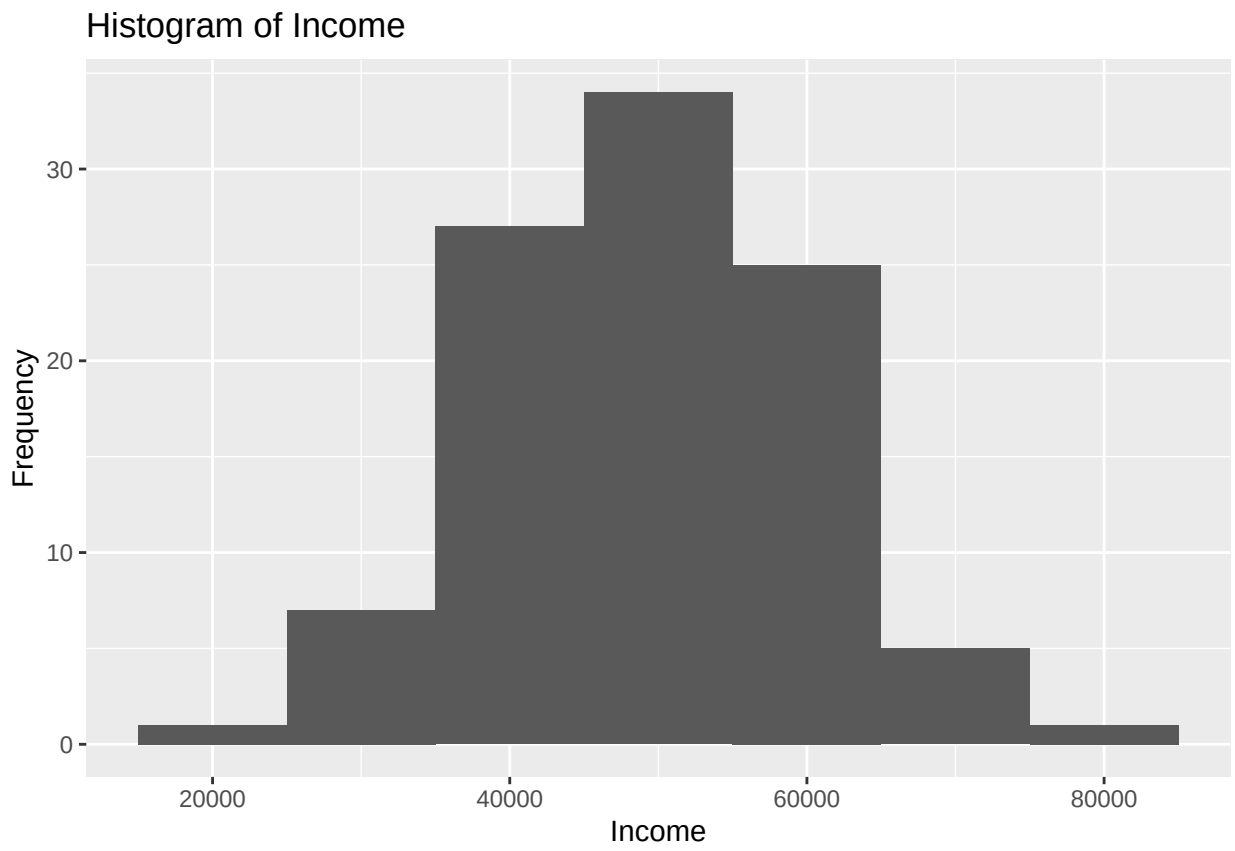
10. Data Visualization and Interpretation:

- Visualize data using different types of plots (bar plot, histogram, line plot, pie chart).
- Interpret plots and analyze results to draw meaningful conclusions.

```
# Load required packages
library(ggplot2)

# Assuming you have a dataset named "data" with variables "age", "sex", "education", "income", and "reg

# Example data (replace this with your actual dataset)
set.seed(123) # Set random seed for reproducibility
age <- sample(18:99, 100, replace = TRUE)
sex <- sample(c("male", "female"), 100, replace = TRUE)
education <- sample(c("No education", "Primary", "Secondary", "Beyond secondary"), 100, replace = TRUE)
income <- rnorm(100, mean = 50000, sd = 10000)
region <- sample(c("North", "South", "East", "West"), 100, replace = TRUE)

data <- data.frame(age, sex, education, income, region)

# Bar plot: Visualize the frequency distribution of education levels
ggplot(data, aes(x = education)) +
  geom_bar() +
  labs(title = "Bar Plot of Education Levels", x = "Education Level", y = "Frequency")
```

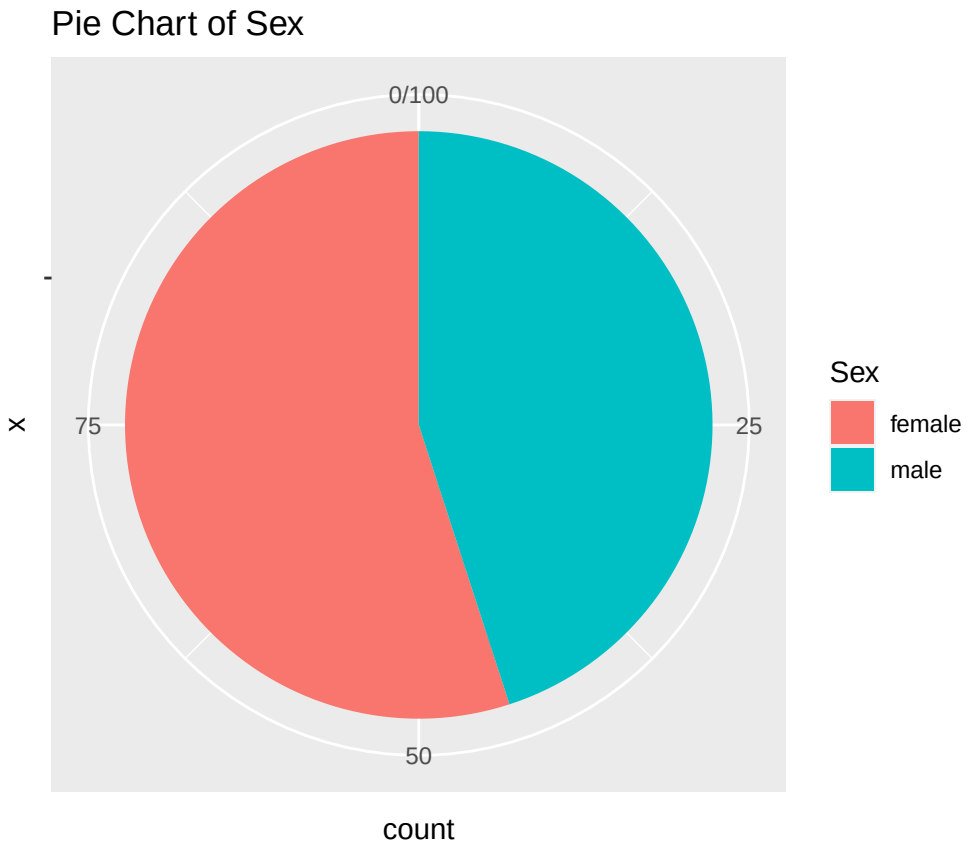



```
# Histogram: Visualize the distribution of income
ggplot(data, aes(x = income)) +
  geom_histogram(binwidth = 10000) +
  labs(title = "Histogram of Income", x = "Income", y = "Frequency")
```




```r
# Line plot: Visualize the trend of income by age
ggplot(data, aes(x = age, y = income, group = 1)) +
  geom_line() +
  labs(title = "Line Plot of Income by Age", x = "Age", y = "Income")
```


```
# Pie chart: Visualize the distribution of sex
ggplot(data, aes(x = "", fill = sex)) +
  geom_bar(width = 1) +
  coord_polar("y", start = 0) +
  labs(title = "Pie Chart of Sex", fill = "Sex")
```




```
# Interpretation:
# 1. The bar plot shows the frequency distribution of education levels, with most people having "Second
# 2. The histogram displays the income distribution, and it seems to follow a normal distribution cente
# 3. The line plot indicates that income tends to increase with age.
# 4. The pie chart depicts the distribution of sex, with roughly equal representation of males and fema
```