**Practical 6 Output**

**# Install required packages (if not installed)**

**> install.packages("tidyverse")**

**Updating loaded packages**

**> install.packages("tidyverse")**

**package ‘tidyverse’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpGcaZiU\downloaded\_packages**

**> install.packages("ggplot2")**

**package ‘ggplot2’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpGcaZiU\downloaded\_packages**

**> install.packages("forecast")**

**package ‘forecast’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpGcaZiU\downloaded\_packages**

**> install.packages("tseries")**

**package ‘tseries’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpGcaZiU\downloaded\_packages**

**# Load libraries**

**> library(tidyverse)**

**── 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 3.5.1 ✔ tibble 3.2.1**

**✔ lubridate 1.9.4 ✔ tidyr 1.3.1**

**✔ purrr 1.0.2**

**ℹ Use the conflicted package to force all conflicts to become errors**

**> library(ggplot2)**

**> library(forecast)**

**> library(tseries)**

**# Load COVID-19 dataset**

**> covid\_data <- read.csv("https://raw.githubusercontent.com/datasets/covid-19/main/data/countries-aggregated.csv")**

**# Filter data for a specific country (e.g., India)**

**> india\_data <- covid\_data %>% filter(Country == "India") %>% select(Date, Confirmed)**

**# Convert date column to Date format**

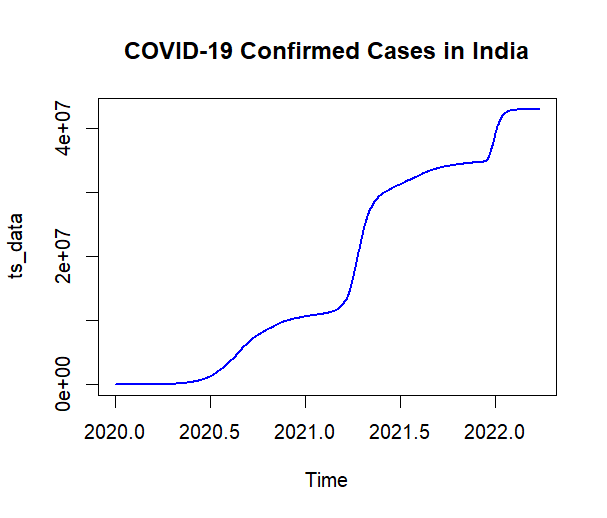
**> india\_data$Date <- as.Date(india\_data$Date)**

**# Convert Confirmed cases into a time series object**

**> ts\_data <- ts(india\_data$Confirmed, start = c(2020, 1), frequency = 365)**

**# Plot the time series**

**> plot.ts(ts\_data, main = "COVID-19 Confirmed Cases in India", col = "blue", lwd = 2)**

****

**# Check stationarity using Augmented Dickey-Fuller (ADF) test**

**> adf.test(ts\_data)**

**Augmented Dickey-Fuller Test**

**data: ts\_data**

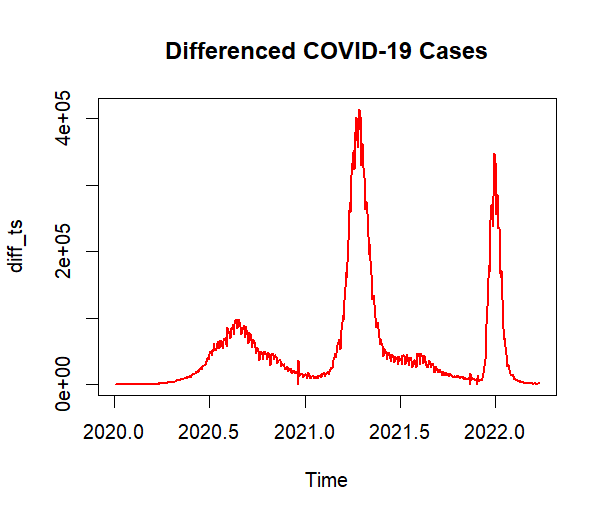
**Dickey-Fuller = -2.2419, Lag order = 9, p-value = 0.4759**

**alternative hypothesis: stationary**

**# Differencing if needed**

**> diff\_ts <- diff(ts\_data)**

**> plot.ts(diff\_ts, main = "Differenced COVID-19 Cases", col = "red", lwd = 2)**

****

**# Fit ARIMA model**

**> model <- auto.arima(ts\_data)**

**> summary(model)**

**Series: ts\_data**

**ARIMA(4,2,3)**

**Coefficients:**

**ar1 ar2 ar3 ar4 ma1 ma2 ma3**

**0.6945 -0.6612 1.0296 -0.1835 -0.4440 0.5651 -0.7384**

**s.e. 0.0555 0.0286 0.0322 0.0444 0.0438 0.0283 0.0302**

**sigma^2 = 51379636: log likelihood = -8378.55**

**AIC=16773.09 AICc=16773.27 BIC=16810.71**

**Training set error measures:**

**ME RMSE MAE MPE MAPE MASE ACF1**

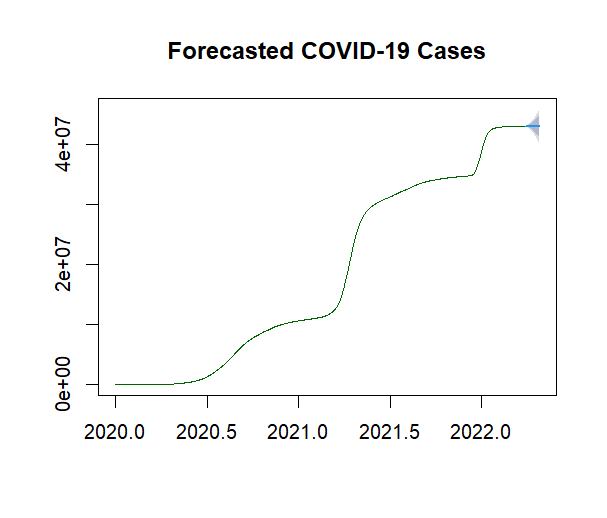
**Training set 0.4354916 7128.32 3833.045 0.07766895 1.426657 0.00015572 0.00473392**

**# Forecast next 30 days**

**> forecasted\_cases <- forecast(model, h = 30)**

**# Plot forecasted values**

**> plot(forecasted\_cases, main = "Forecasted COVID-19 Cases", col = "darkgreen")**

****

**Practical 7 Output**

**# Install required packages (if not installed)**

**> install.packages("arules")**

**> install.packages("arules")**

**package ‘arules’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpGcVA4t\downloaded\_packages**

**> install.packages("arulesViz")**

**package ‘arulesViz’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpGcVA4t\downloaded\_packages**

**# Load libraries**

**> library(arules)**

**> library(arulesViz)**

**# Load dataset (Groceries dataset from arules package)**

**> data("Groceries")**

**# Inspect the dataset**

**> summary(Groceries)**

**transactions as itemMatrix in sparse format with**

**9835 rows (elements/itemsets/transactions) and**

**169 columns (items) and a density of 0.02609146**

**most frequent items:**

**whole milk other vegetables rolls/buns soda yogurt (Other)**

**2513 1903 1809 1715 1372 34055**

**element (itemset/transaction) length distribution:**

**sizes**

**1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18**

**2159 1643 1299 1005 855 645 545 438 350 246 182 117 78 77 55 46 29 14**

**19 20 21 22 23 24 26 27 28 29**

**14 9 11 4 6 1 1 1 1 3**

**Min. 1st Qu. Median Mean 3rd Qu. Max.**

**1.000 2.000 3.000 4.409 6.000 32.000**

**includes extended item information - examples:**

**labels level2 level1**

**1 frankfurter sausage meat and sausage**

**2 sausage sausage meat and sausage**

**3 liver loaf sausage meat and sausage**

**# Apply Apriori algorithm with support=0.01 and confidence=0.3**

**> rules <- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.3, target="rules"))**

**Apriori**

**Parameter specification:**

**Confidence minval smax arem aval originalSupport maxtime support minlen maxlen**

**0.3 0.1 1 none FALSE TRUE 5 0.01 1 10**

**target ext**

**rules TRUE**

**Algorithmic control:**

**filter tree heap memopt load sort verbose**

**0.1 TRUE TRUE FALSE TRUE 2 TRUE**

**Absolute minimum support count: 98**

**set item appearances ...[0 item(s)] done [0.00s].**

**set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].**

**sorting and recoding items ... [88 item(s)] done [0.00s].**

**creating transaction tree ... done [0.00s].**

**checking subsets of size 1 2 3 4 done [0.00s].**

**writing ... [125 rule(s)] done [0.00s].**

**creating S4 object ... done [0.00s].**

**# Inspect top 5 association rules**

**> inspect(head(rules, 5))**

**lhs rhs support confidence coverage lift count**

**[1] {hard cheese} => {whole milk} 0.01006609 0.4107884 0.02450432 1.607682 99**

**[2] {butter milk} => {other vegetables} 0.01037112 0.3709091 0.02796136 1.916916 102**

**[3] {butter milk} => {whole milk} 0.01159126 0.4145455 0.02796136 1.622385 114**

**[4] {ham} => {whole milk} 0.01148958 0.4414062 0.02602949 1.727509 113**

**[5] {sliced cheese} => {whole milk} 0.01077783 0.4398340 0.02450432 1.721356 106**

**# Visualize association rules**

**> plot(rules, method = "graph", control = list(type = "items"))**

**Available control parameters (with default values):**

**layout = stress**

**circular = FALSE**

**ggraphdots = NULL**

**edges = <environment>**

**nodes = <environment>**

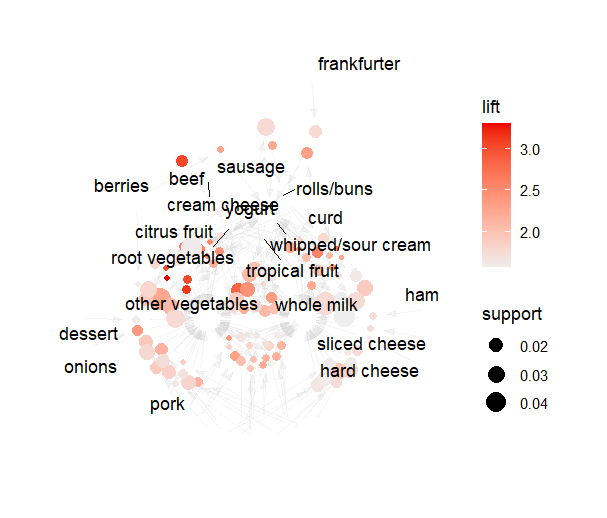
**nodetext = <environment>**

**colors = c("#EE0000FF", "#EEEEEEFF")**

**engine = ggplot2**

**max = 100**

**verbose = FALSE**

****

**Practical 8 Output**

**# Install required packages (if not installed)**

**> install.packages("igraph")**

**> install.packages("igraph")**

**package ‘igraph’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpAnGoLl\downloaded\_packages**

**> install.packages("ggraph")**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpAnGoLl\downloaded\_packages**

**> install.packages("tidyverse")**

**package ‘tidyverse’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpAnGoLl\downloaded\_packages**

**# Load libraries**

**> library(igraph)**

**> library(ggraph)**

**Loading required package: ggplot2**

**> library(tidyverse)**

**── Attaching core tidyverse packages ──────────────────────────────────────────────────────────────── tidyverse 2.0.0 ──**

**✔ dplyr 1.1.4 ✔ readr 2.1.5**

**✔ forcats 1.0.0 ✔ stringr 1.5.1**

**✔ lubridate 1.9.4 ✔ tibble 3.2.1**

**✔ purrr 1.0.2 ✔ tidyr 1.3.1**

**ℹ Use the conflicted package to force all conflicts to become errors**

**# Create a sample dataset (edges representing connections between nodes)**

**> edges <- data.frame(**

**+ from = c("A", "A", "B", "C", "C", "D", "E", "F"),**

**+ to = c("B", "C", "D", "D", "E", "F", "G", "G")**

**+ )**

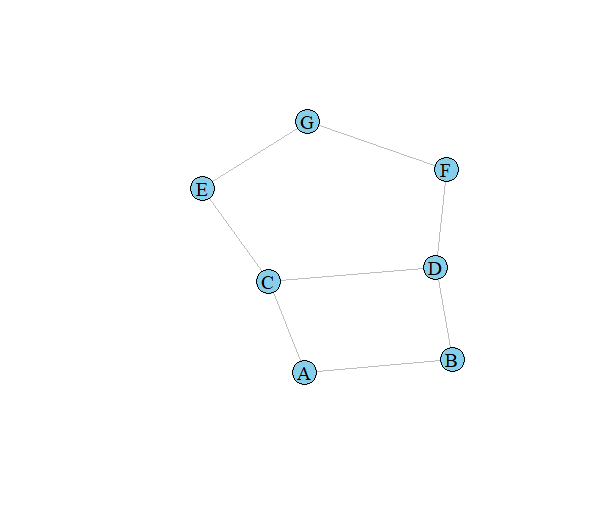
**# Convert the edge list into a graph object**

**> network\_graph <- graph\_from\_data\_frame(edges, directed = FALSE)**

**# Plot the basic network**

**> plot(network\_graph, vertex.size = 20, vertex.color = "skyblue",**

**+ vertex.label.color = "black", edge.color = "gray")**

****

**# Calculate centrality measures**

**> degree\_centrality <- degree(network\_graph)**

**> betweenness\_centrality <- betweenness(network\_graph)**

**# Print centrality measures**

**> print(degree\_centrality)**

**A B C D E F G**

**2 2 3 3 2 2 2**

**> print(betweenness\_centrality)**

**A B C D E F G**

**1 1 5 5 2 2 1**

**# Advanced visualization using ggraph**

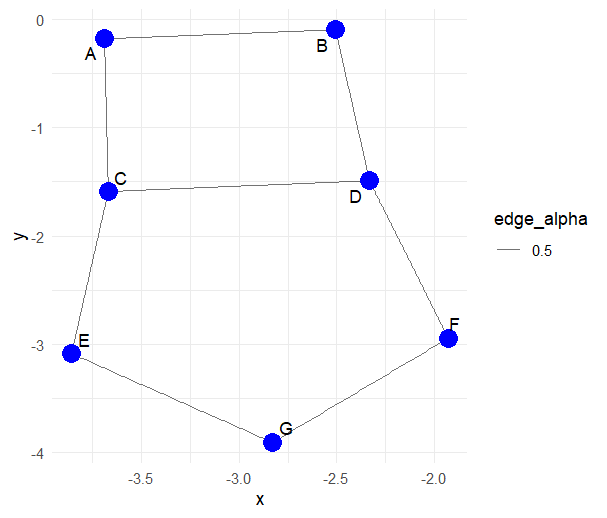
**> ggraph(network\_graph, layout = "fr") +**

**+ geom\_edge\_link(aes(edge\_alpha = 0.5)) +**

**+ geom\_node\_point(color = "blue", size = 5) +**

**+ geom\_node\_text(aes(label = name), repel = TRUE) +**

**+ theme\_minimal()**

****

**Practical 9 Output**

**#Install required packages (if not installed)**

**> install.packages("tm")**

**package ‘tm’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpAz386G\downloaded\_packages**

**> install.packages("SnowballC")**

**package ‘SnowballC’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpAz386G\downloaded\_packages**

**> install.packages("wordcloud")**

**package ‘wordcloud’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpAz386G\downloaded\_packages**

**> install.packages("ggplot2")**

**> install.packages("ggplot2")**

**package ‘ggplot2’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\Rtmp2lZWiC\downloaded\_packages**

**# Load libraries**

**> library(tm)**

**> library(SnowballC)**

**> library(wordcloud)**

**> library(ggplot2)**

**# Sample text dataset**

**> text\_data <- c("Text mining is a powerful technique to analyze unstructured data.",**

**+ "Machine learning algorithms enhance text analysis.",**

**+ "Data preprocessing improves text mining outcomes.")**

**# Convert text data into a corpus**

**> corpus <- Corpus(VectorSource(text\_data))**

**# Text preprocessing: Convert to lowercase, remove punctuation, numbers, and stopwords**

**> 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, stripWhitespace)**

**# Stemming (reducing words to root form)**

**> corpus <- tm\_map(corpus, stemDocument)**

**# Create a Term-Document Matrix (TDM)**

**> tdm <- TermDocumentMatrix(corpus)**

**> tdm\_matrix <- as.matrix(tdm)**

**> word\_freq <- sort(rowSums(tdm\_matrix), decreasing = TRUE)**

**# Convert to data frame for visualization**

**> word\_freq\_df <- data.frame(word = names(word\_freq), freq = word\_freq)**

**# Generate a Word Cloud**

**> wordcloud(words = word\_freq\_df$word, freq = word\_freq\_df$freq, min.freq = 1,**

**+ max.words = 50, random.order = FALSE, colors = brewer.pal(8, "Dark2"))**

****

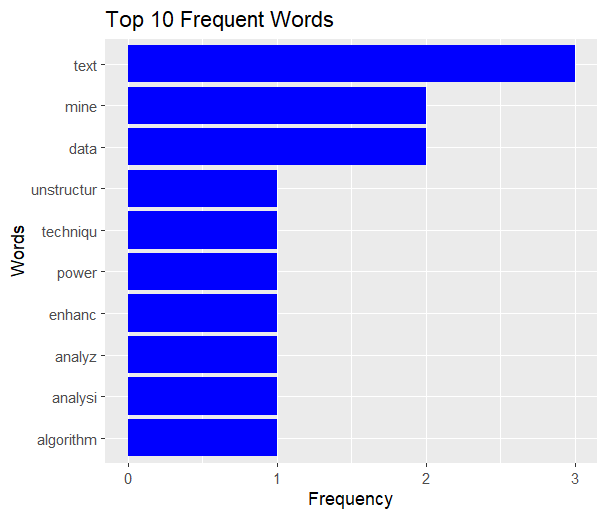
**# Bar plot of top words**

**> ggplot(word\_freq\_df[1:10,], aes(x = reorder(word, freq), y = freq)) +**

**+ geom\_bar(stat = "identity", fill = "blue") +**

**+ coord\_flip() +**

**+ labs(title = "Top 10 Frequent Words", x = "Words", y = "Frequency")**

****

**Practical 10 Output**

**#Install required packages (if not installed)**

**> install.packages("ggplot2")**

**package ‘ggplot2’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpcvOfAY\downloaded\_packages**

**> install.packages("dplyr")**

**package ‘dplyr’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpcvOfAY\downloaded\_packages**

**> install.packages("outliers")**

**package ‘outliers’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpcvOfAY\downloaded\_packages**

**> install.packages("anomalize")**

**package ‘pcaPP’ successfully unpacked and MD5 sums checked**

**package ‘DEoptimR’ successfully unpacked and MD5 sums checked**

**package ‘curl’ successfully unpacked and MD5 sums checked**

**package ‘pyinit’ successfully unpacked and MD5 sums checked**

**package ‘rrcov’ successfully unpacked and MD5 sums checked**

**package ‘robustbase’ successfully unpacked and MD5 sums checked**

**package ‘furrr’ successfully unpacked and MD5 sums checked**

**package ‘warp’ successfully unpacked and MD5 sums checked**

**package ‘RcppRoll’ successfully unpacked and MD5 sums checked**

**package ‘httr2’ successfully unpacked and MD5 sums checked**

**package ‘PerformanceAnalytics’ successfully unpacked and MD5 sums checked**

**package ‘RobStatTM’ successfully unpacked and MD5 sums checked**

**package ‘rsample’ successfully unpacked and MD5 sums checked**

**package ‘padr’ successfully unpacked and MD5 sums checked**

**package ‘slider’ successfully unpacked and MD5 sums checked**

**package ‘anytime’ successfully unpacked and MD5 sums checked**

**package ‘tsfeatures’ successfully unpacked and MD5 sums checked**

**package ‘tidyquant’ successfully unpacked and MD5 sums checked**

**package ‘assertthat’ successfully unpacked and MD5 sums checked**

**package ‘timetk’ successfully unpacked and MD5 sums checked**

**package ‘sweep’ successfully unpacked and MD5 sums checked**

**package ‘tibbletime’ successfully unpacked and MD5 sums checked**

**package ‘anomalize’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpcvOfAY\downloaded\_packages**

**# Load libraries**

**> library(ggplot2)**

**> library(dplyr)**

**> library(outliers)**

**> library(anomalize)**

**# Generate a sample dataset with anomalies**

**> set.seed(123)**

**> data <- data.frame(Value = c(rnorm(98, mean = 50, sd = 10), 150, 160)) # Two anomalies added**

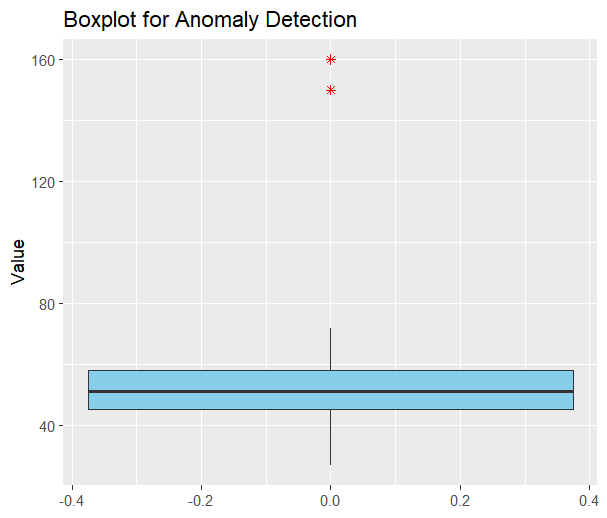
**# Boxplot to visualize anomalies**

**> ggplot(data, aes(y = Value)) +**

**+ geom\_boxplot(fill = "skyblue", outlier.color = "red", outlier.shape = 8) +**

**+ ggtitle("Boxplot for Anomaly Detection") +**

**+ ylab("Value")**

****

**# Detecting anomalies using Interquartile Range (IQR)**

**> Q1 <- quantile(data$Value, 0.25)**

**> Q3 <- quantile(data$Value, 0.75)**

**> IQR\_value <- Q3 - Q1**

**> lower\_bound <- Q1 - 1.5 \* IQR\_value**

**> upper\_bound <- Q3 + 1.5 \* IQR\_value**

**# Identifying outliers**

**> data$Anomaly <- ifelse(data$Value < lower\_bound | data$Value > upper\_bound, "Anomaly", "Normal")**

**> print(data)**

**Value Anomaly**

**1 44.39524 Normal**

**2 47.69823 Normal**

**3 65.58708 Normal**

**4 50.70508 Normal**

**5 51.29288 Normal**

**6 67.15065 Normal**

**7 54.60916 Normal**

**8 37.34939 Normal**

**9 43.13147 Normal**

**10 45.54338 Normal**

**11 62.24082 Normal**

**12 53.59814 Normal**

**13 54.00771 Normal**

**14 51.10683 Normal**

**15 44.44159 Normal**

**16 67.86913 Normal**

**17 54.97850 Normal**

**18 30.33383 Normal**

**19 57.01356 Normal**

**20 45.27209 Normal**

**21 39.32176 Normal**

**22 47.82025 Normal**

**23 39.73996 Normal**

**24 42.71109 Normal**

**25 43.74961 Normal**

**26 33.13307 Normal**

**27 58.37787 Normal**

**28 51.53373 Normal**

**29 38.61863 Normal**

**30 62.53815 Normal**

**31 54.26464 Normal**

**32 47.04929 Normal**

**33 58.95126 Normal**

**34 58.78133 Normal**

**35 58.21581 Normal**

**36 56.88640 Normal**

**37 55.53918 Normal**

**38 49.38088 Normal**

**39 46.94037 Normal**

**40 46.19529 Normal**

**41 43.05293 Normal**

**42 47.92083 Normal**

**43 37.34604 Normal**

**44 71.68956 Normal**

**45 62.07962 Normal**

**46 38.76891 Normal**

**47 45.97115 Normal**

**48 45.33345 Normal**

**49 57.79965 Normal**

**50 49.16631 Normal**

**51 52.53319 Normal**

**52 49.71453 Normal**

**53 49.57130 Normal**

**54 63.68602 Normal**

**55 47.74229 Normal**

**56 65.16471 Normal**

**57 34.51247 Normal**

**58 55.84614 Normal**

**59 51.23854 Normal**

**60 52.15942 Normal**

**61 53.79639 Normal**

**62 44.97677 Normal**

**63 46.66793 Normal**

**64 39.81425 Normal**

**65 39.28209 Normal**

**66 53.03529 Normal**

**67 54.48210 Normal**

**68 50.53004 Normal**

**69 59.22267 Normal**

**70 70.50085 Normal**

**71 45.08969 Normal**

**72 26.90831 Normal**

**73 60.05739 Normal**

**74 42.90799 Normal**

**75 43.11991 Normal**

**76 60.25571 Normal**

**77 47.15227 Normal**

**78 37.79282 Normal**

**79 51.81303 Normal**

**80 48.61109 Normal**

**81 50.05764 Normal**

**82 53.85280 Normal**

**83 46.29340 Normal**

**84 56.44377 Normal**

**85 47.79513 Normal**

**86 53.31782 Normal**

**87 60.96839 Normal**

**88 54.35181 Normal**

**89 46.74068 Normal**

**90 61.48808 Normal**

**91 59.93504 Normal**

**92 55.48397 Normal**

**93 52.38732 Normal**

**94 43.72094 Normal**

**95 63.60652 Normal**

**96 43.99740 Normal**

**97 71.87333 Normal**

**98 65.32611 Normal**

**99 150.00 Anomaly**

**100 160.00 Anomaly**

**# Visualization with anomalies highlighted**

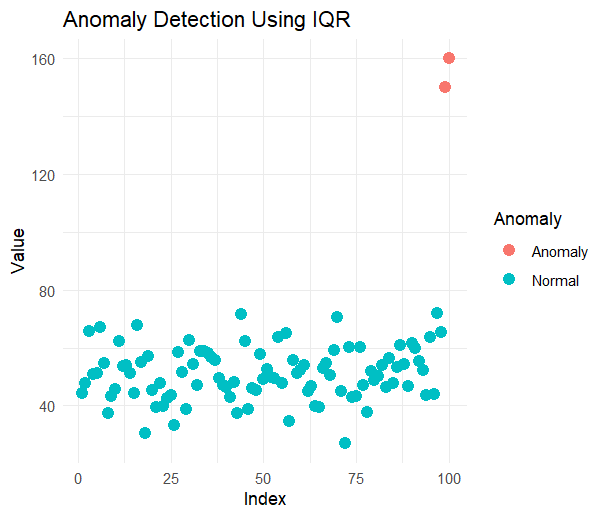
**> ggplot(data, aes(x = 1:nrow(data), y = Value, color = Anomaly)) +**

**+ geom\_point(size = 3) +**

**+ ggtitle("Anomaly Detection Using IQR") +**

**+ xlab("Index") + ylab("Value") +**

**+ theme\_minimal()**

****

**Practical 11 Output**

|  |
| --- |
| **# Install required packages (if not installed)**  **> install.packages("tm")**  **> install.packages("tm")**  **package ‘tm’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpOQJOuj\downloaded\_packages**  **> install.packages("caret")**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpOQJOuj\downloaded\_packages**  **> install.packages("e1071")**  **package ‘e1071’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpOQJOuj\downloaded\_packages**  **# Load necessary libraries**  **> library(tm) # Text mining**  **> library(caret) # Machine learning**  **> library(e1071) # Naive Bayes**  **# Load dataset (SMS Spam Collection)**  **> data <- read.csv("sms\_spam.csv", stringsAsFactors = FALSE)**  **> colnames(data) <- c("label", "message") # Rename columns**  **> data$label <- factor(data$label) # Convert label to factor**  **# Text Preprocessing**  **> corpus <- VCorpus(VectorSource(data$message))**  **# Text Preprocessing**  **> corpus <- VCorpus(VectorSource(data$message))**  **> corpus <- tm\_map(corpus, content\_transformer(tolower)) # Convert to lowercase**  **> corpus <- tm\_map(corpus, removePunctuation) # Remove punctuation**  **> corpus <- tm\_map(corpus, removeNumbers) # Remove numbers**  **> corpus <- tm\_map(corpus, removeWords, stopwords("en")) # Remove stopwords**  **> corpus <- tm\_map(corpus, stripWhitespace) # Remove extra spaces**  **# Create Document-Term Matrix (DTM)**  **> dtm <- DocumentTermMatrix(corpus)**  **> dtm <- removeSparseTerms(dtm, 0.99) # Remove sparse terms**  **> dataset <- as.data.frame(as.matrix(dtm))**  **> dataset$label <- data$label # Add label to dataset**  **# Split data into training and testing sets**  **> set.seed(123)**  **> trainIndex <- createDataPartition(dataset$label, p = 0.8, list = FALSE)**  **> trainData <- dataset[trainIndex, ]**  **> testData <- dataset[-trainIndex, ]**  **# Train Naïve Bayes Model**  **> model <- naiveBayes(label ~ ., data = trainData)**  **# Predict on test data**  **> predictions <- predict(model, testData)**  **# Evaluate model performance**  **> conf\_matrix <- confusionMatrix(predictions, testData$label)**  **# Evaluate model performance**  **> conf\_matrix <- confusionMatrix(predictions, testData$label)**  **> print(conf\_matrix)**  **Confusion Matrix and Statistics**  **Reference**  **Prediction ham spam**  **ham 265 2**  **spam 700 147**    **Accuracy : 0.3698**  **95% CI : (0.3414, 0.399)**  **No Information Rate : 0.8662**  **P-Value [Acc > NIR] : 1**  **Kappa : 0.0876**  **Mcnemar's Test P-Value : <2e-16**  **Sensitivity : 0.2746**  **Specificity : 0.9866**  **Pos Pred Value : 0.9925**  **Neg Pred Value : 0.1736**  **Prevalence : 0.8662**  **Detection Rate : 0.2379**  **Detection Prevalence : 0.2397**  **Balanced Accuracy : 0.6306**  **'Positive' Class : ham**    **# Display sample output**  **> print("Sample Predictions:")**  **[1] "Sample Predictions:"**  **> print(head(data.frame(Actual = testData$label, Predicted = predictions), 10))**  **Actual Predicted**  **1 ham spam**  **2 spam spam**  **3 spam spam**  **4 ham ham**  **5 ham spam**  **6 ham spam**  **7 ham ham**  **8 ham spam**  **9 spam spam**  **10 spam spam** |
|  |
| **Practical 12 Output**   |  | | --- | | **#Install required packages (if not installed)**  **> install.packages("tidyverse")**  **> install.packages("tidyverse")**  **package ‘tidyverse’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpSuoYf5\downloaded\_packages**  **> install.packages("tidytext")**  **package ‘tidytext’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpSuoYf5\downloaded\_packages**  **> install.packages("tm")**  **package ‘tm’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpSuoYf5\downloaded\_packages**  **> install.packages("wordcloud")**  **package ‘wordcloud’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpSuoYf5\downloaded\_packages**  **> install.packages("syuzhet")**  **package ‘syuzhet’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpSuoYf5\downloaded\_packages**  **> install.packages("ggplot2")**  **package ‘ggplot2’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpSuoYf5\downloaded\_packages**  **# Load necessary libraries**  **> library(tidyverse)**  **── 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 3.5.1 ✔ tibble 3.2.1**  **✔ lubridate 1.9.4 ✔ tidyr 1.3.1**  **✔ purrr 1.0.2**  **> library(tidytext)**  **> library(tm)**  **> library(wordcloud)**  **> library(syuzhet)**  **> library(ggplot2)**  **#Load dataset (Amazon Reviews dataset example)**  **> data <- read.csv("amazon\_reviews.csv", stringsAsFactors = FALSE)**  **> colnames(data) <- c("id", "rating", "text") # Rename columns**  **# Convert text to lowercase**  **> data$text <- tolower(data$text)**  **# Remove punctuation, numbers, and stopwords**  **> corpus <- VCorpus(VectorSource(data$text))**  **> corpus <- tm\_map(corpus, removePunctuation)**  **> corpus <- tm\_map(corpus, removeNumbers)**  **> corpus <- tm\_map(corpus, removeWords, stopwords("en"))**  **> corpus <- tm\_map(corpus, stripWhitespace)**  **# Convert to tidy format**  **> text\_df <- data.frame(text = sapply(corpus, as.character), stringsAsFactors = FALSE)**  **# Sentiment Analysis using Syuzhet**  **> sentiments <- get\_nrc\_sentiment(text\_df$text)**  **# Summarize sentiment scores**  **> sentiment\_scores <- colSums(sentiments)**  **> print(sentiment\_scores)**    **anger anticipation disgust fear joy sadness surprise trust negative positive**    **0 0 0 0 0 0 0 0 0 0**  **# Visualize sentiment scores**  **> sentiment\_df <- data.frame(sentiment = names(sentiment\_scores), count = sentiment\_scores)**  **> ggplot(sentiment\_df, aes(x = sentiment, y = count, fill = sentiment)) +**  **+ geom\_bar(stat = "identity") +**  **+ theme\_minimal() +**  **+ labs(title = "Sentiment Analysis using EDA in R", x = "Sentiment", y = "Count")** | |  | | |  | | --- | |  | |   **Practical 13 Output**  **#Install required packages (if not installed)**  **> install.packages("ggplot2")**  **> install.packages("ggplot2")**  **package ‘ggplot2’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpqorKhy\downloaded\_packages**  **> install.packages("caret")**  **package ‘caret’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpqorKhy\downloaded\_packages**  **> install.packages("class")**  **package ‘class’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpqorKhy\downloaded\_packages**  **> install.packages("gridExtra")**  **package ‘gridExtra’ successfully unpacked and MD5 sums checked**  **The downloaded binary packages are in**  **C:\Users\ADMIN\AppData\Local\Temp\RtmpqorKhy\downloaded\_packages**  **# Load libraries**  **> library(ggplot2)**  **> library(caret)**  **> library(class)**  **> library(gridExtra)**  **# Load IRIS dataset**  **> data(iris)**  **# Data Visualization - Scatter Plot**  **> plot1 <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +**  **+ geom\_point(size = 3) + ggtitle("Sepal Length vs Width")**  **> plot2 <- ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) +**  **+ geom\_point(size = 3) + ggtitle("Petal Length vs Width")**  **> grid.arrange(plot1, plot2, nrow = 1) # Display plots side by side**    **# Splitting dataset into training (80%) and testing (20%)**  **> set.seed(123)**  **> trainIndex <- createDataPartition(iris$Species, p = 0.8, list = FALSE)**  **> trainData <- iris[trainIndex, ]**  **> testData <- iris[-trainIndex, ]**  **# Applying K-Nearest Neighbors (KNN) Algorithm**  **> trainLabels <- trainData$Species**  **> testLabels <- testData$Species**  **> knn\_model <- knn(train = trainData[, 1:4], test = testData[, 1:4], cl = trainLabels, k = 5)**  **# Confusion Matrix - Model Evaluation**  **> conf\_matrix <- confusionMatrix(knn\_model, testLabels)**  **> print(conf\_matrix)**  **Confusion Matrix and Statistics**  **Reference**  **Prediction setosa versicolor virginica**  **setosa 10 0 0**  **versicolor 0 10 0**  **virginica 0 0 10**  **Overall Statistics**    **Accuracy : 1**  **95% CI : (0.8843, 1)**  **No Information Rate : 0.3333**  **P-Value [Acc > NIR] : 4.857e-15**  **Kappa : 1**  **Mcnemar's Test P-Value : NA**  **Statistics by Class:**  **Class: setosa Class: versicolor Class: virginica**  **Sensitivity 1.0000 1.0000 1.0000**  **Specificity 1.0000 1.0000 1.0000**  **Pos Pred Value 1.0000 1.0000 1.0000**  **Neg Pred Value 1.0000 1.0000 1.0000**  **Prevalence 0.3333 0.3333 0.3333**  **Detection Rate 0.3333 0.3333 0.3333**  **Detection Prevalence 0.3333 0.3333 0.3333**  **Balanced Accuracy 1.0000 1.0000 1.0000**  **# Accuracy Calculation**  **> accuracy <- sum(knn\_model == testLabels) / length(testLabels)**  **> print(paste("Accuracy:", round(accuracy \* 100, 2), "%"))**  **[1] "Accuracy: 100 %"**   |  | | --- | |  | |

**Practical 14 Output**

**#Install required packages (if not installed)**

**> install.packages("xgboost")**

**> install.packages("xgboost")**

**package ‘xgboost’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpmOK0Vu\downloaded\_packages**

**> install.packages("caret")**

**package ‘caret’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpmOK0Vu\downloaded\_packages**

**> install.packages("ggplot2")**

**package ‘ggplot2’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\RtmpmOK0Vu\downloaded\_packages**

**> install.packages("Matrix")**

**package ‘Matrix’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\Rtmp8wT9Zy\downloaded\_packages**

**# Load libraries**

**> library(xgboost)**

**> library(caret)**

**Loading required package: ggplot2**

**Loading required package: lattice**

**> library(ggplot2)**

**> library(Matrix)**

**# Load the dataset (Using Iris dataset as an example)**

**> data(iris)**

**# Encode species as numeric for XGBoost**

**> iris$Species <- as.numeric(factor(iris$Species)) - 1**

**# Split data into training (80%) and testing (20%)**

**> set.seed(123)**

**> trainIndex <- createDataPartition(iris$Species, p = 0.8, list = FALSE)**

**> trainData <- iris[trainIndex, ]**

**> testData <- iris[-trainIndex, ]**

**# Convert data to matrix format required by XGBoost**

**> trainMatrix <- xgb.DMatrix(data = as.matrix(trainData[, -5]), label = trainData$Species)**

**> testMatrix <- xgb.DMatrix(data = as.matrix(testData[, -5]), label = testData$Species)**

**# Set XGBoost parameters**

**> params <- list(**

**+ objective = "multi:softmax", # Multi-class classification**

**+ num\_class = 3, # Number of classes (Setosa, Versicolor, Virginica)**

**+ eval\_metric = "mlogloss",**

**+ max\_depth = 3,**

**+ eta = 0.3,**

**+ nthread = 2**

**+ )**

**# Train the XGBoost model**

**> xgb\_model <- xgboost(params = params, data = trainMatrix, nrounds = 100, verbose = 0)**

**# Make predictions on test data**

**> predictions <- predict(xgb\_model, testMatrix)**

**# Ensure predictions match actual species levels**

**> predicted\_labels <- factor(predictions, levels = levels(testData$Species))**

**# Convert testData$Species to factor explicitly**

**> testData$Species <- factor(testData$Species, levels = unique(testData$Species))**

**# Convert predictions to factor with matching levels**

**> predicted\_labels <- factor(predictions, levels = levels(testData$Species))**

**# Create Confusion Matrix**

**> conf\_matrix <- confusionMatrix(predicted\_labels, testData$Species)**

**# Print Confusion Matrix**

**> print(conf\_matrix)**

**Confusion Matrix and Statistics**

**Reference**

**Prediction 0 1 2**

**0 11 0 0**

**1 0 9 1**

**2 0 0 9**

**Overall Statistics**

**Accuracy : 0.9667**

**95% CI : (0.8278, 0.9992)**

**No Information Rate : 0.3667**

**P-Value [Acc > NIR] : 4.476e-12**

**Kappa : 0.9499**

**Mcnemar's Test P-Value : NA**

**Statistics by Class:**

**Class: 0 Class: 1 Class: 2**

**Sensitivity 1.000 1.0000 0.9000**

**Specificity 1.0000 0.9524 1.0000**

**Pos Pred Value 1.0000 0.9000 1.0000**

**Neg Pred Value 1.0000 1.0000 0.9524**

**Prevalence 0.3667 0.3000 0.3333**

**Detection Rate 0.3667 0.3000 0.3000**

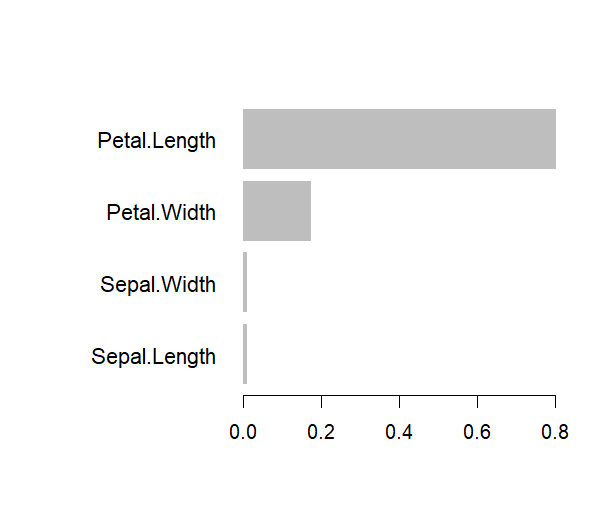
**Detection Pre. 0.3667 0.3333 0.3000**

**Balanced Acc. 1.0000 0.9762 0.9500**

**# Feature Importance Plot**

**> importance\_matrix <- xgb.importance(model = xgb\_model)**

**> xgb.plot.importance(importance\_matrix)**

****

**Practical 15 Output**

**#Install required packages (if not installed)**

**> install.packages("arules")**

**package ‘arules’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\Rtmp8wT9Zy\downloaded\_packages**

**> install.packages("arulesViz")**

**package ‘arulesViz’ successfully unpacked and MD5 sums checked**

**The downloaded binary packages are in**

**C:\Users\ADMIN\AppData\Local\Temp\Rtmp824iuL\downloaded\_packages**

**# Load libraries**

**> library(arules)**

**> library(arulesViz)**

**# Load a sample transactional dataset (Groceries dataset)**

**> data("Groceries")**

**# View summary of dataset**

**> summary(Groceries)**

**transactions as itemMatrix in sparse format with**

**9835 rows (elements/itemsets/transactions) and**

**169 columns (items) and a density of 0.02609146**

**most frequent items:**

**whole milk other vegetables rolls/buns soda yogurt (Other)**

**2513 1903 1809 1715 1372 34055**

**element (itemset/transaction) length distribution:**

**sizes**

**1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16**

**2159 1643 1299 1005 855 645 545 438 350 246 182 117 78 77 55 46**

**17 18 19 20 21 22 23 24 26 27 28 29 32**

**29 14 14 9 11 4 6 1 1 1 1 3 1**

**Min. 1st Qu. Median Mean 3rd Qu. Max.**

**1.000 2.000 3.000 4.409 6.000 32.000**

**includes extended item information - examples:**

**labels level2 level1**

**1 frankfurter sausage meat and sausage**

**2 sausage sausage meat and sausage**

**3 liver loaf sausage meat and sausage**

**# Apply Apriori algorithm to find association rules**

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

**Apriori**

**Parameter specification:**

**Confidence minval smax arem aval originalSupport maxtime support**

**0.5 0.1 1 none FALSE TRUE 5 0.01**

**Minlen maxlen target ext**

**2 10 rules TRUE**

**Algorithmic control:**

**filter tree heap memopt load sort verbose**

**0.1 TRUE TRUE FALSE TRUE 2 TRUE**

**Absolute minimum support count: 98**

**set item appearances ...[0 item(s)] done [0.00s].**

**set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].**

**sorting and recoding items ... [88 item(s)] done [0.00s].**

**creating transaction tree ... done [0.00s].**

**checking subsets of size 1 2 3 4 done [0.00s].**

**writing ... [15 rule(s)] done [0.00s].**

**creating S4 object ... done [0.00s].**

**# Inspect top 10 rules**

**> inspect(head(rules, 10))**

**lhs rhs support confidence coverage lift count**

**[1]{curd, yogurt} => {whole milk} 0.01 0.582 0.017 2.279 99**

**[2]{other vegetables, butter} => {whole milk} 0.01 0.57 0.02 2.244 113**

**[3]{other vegs, domestic egg} => {whole milk} 0.0123 0.552 0.022 2.162 121**

**[4]{yogurt, whipped/sour cream} => {whole milk} 0.01 0.524 0.02 2.052 107**

**[5]{other vegetables, whipped/sour cream} => {whole milk} 0.01 0.5 0.028 1.98 144**

**[6]{pip fruit, other vegetables} => {whole milk} 0.01 0.517 0.026 2.02 133**

**[7]{citrus fruit, root vegetables}=> {other vegetables} 0.01 0.586 0.017 3.02 102**

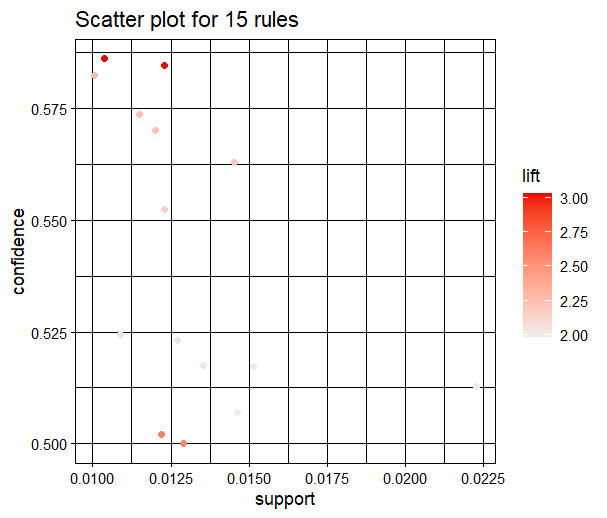
**[8]{tropical fruit, root vegetables}=> {other vegetables} 0.012 0.584 0.021 3.02 121**

**[9]{tropical fruit, root vegetables}=> {whole milk} 0.011 0.57 0.021 2.230 118**

**[10{tropical fruit, yogurt} => {whole milk} 0.015 0.51 0.029 2.024 149**

**# Visualize the rules using scatter plot**

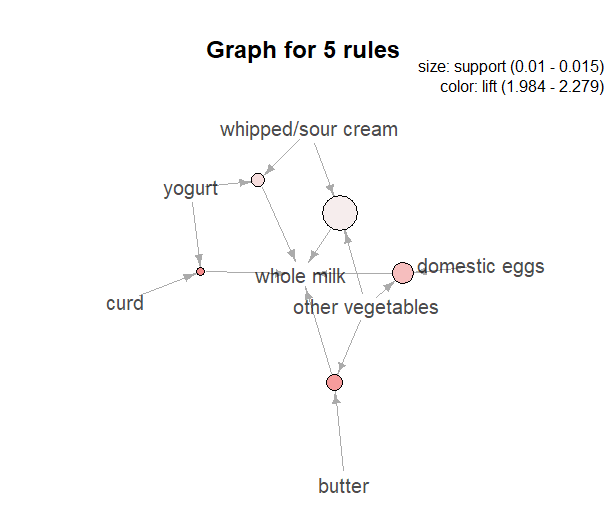
**> plot(rules, method = "scatterplot", measure = c("support", "confidence"), shading = "lift")**

****

**# Graph-based visualization of top 5 rules**

**> subrules <- head(rules, 5)**

**> plot(subrules, method = "graph", engine = "igraph")**

****