1.INSTALLING PACKAGES

This code installs packages/modules or dependencies required here

```
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
```

```
    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)

    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)

    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)
```

2.LOADING LIBRARIES

This code imports libraries/modules into the current program for use.

```
library(ggplot2)
library(dplyr)
library(tidyr)
```

```
    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
```

```
# Using read_csv() from readr package
install.packages("readr")

library(readr)
```

```
    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)
```

3.READING DATA

This code reads data from a specified source, such as a file or a database, into the program for further processing or analysis.

```
data <- read.csv("/content/wisconsin.csv")
```

```
data
```

A data.frame: 699 × 10

Cl.thickness	Cell.size	Cell.shape	Marg.adhesion	Epith.c.size	Bare.nucle
<int>	<int>	<int>	<int>	<int>	<int
5	1	1	1	2	
5	4	4	5	7	
3	1	1	1	2	
6	8	8	1	3	
4	1	1	3	2	
8	10	10	8	7	
1	1	1	1	2	
2	1	2	1	2	
2	1	1	1	2	
4	2	1	1	2	
1	1	1	1	1	
2	1	1	1	2	

5	3	3	3	2
1	1	1	1	2
8	7	5	10	7
7	4	6	4	6
4	1	1	1	2
4	1	1	1	2
10	7	7	6	4
6	1	1	1	2
7	3	2	10	5
10	5	5	3	6
3	1	1	1	2
8	4	5	1	2
1	1	1	1	2
5	2	3	4	2
3	2	1	1	1
5	1	1	1	2
2	1	1	1	2
1	1	3	1	2
⋮	⋮	⋮	⋮	⋮
5	10	10	8	5
3	10	7	8	5
3	2	1	2	2
2	1	1	1	2
5	3	2	1	3
1	1	1	1	2
4	1	4	1	2
1	1	2	1	2
5	1	1	1	2
1	1	1	1	2
2	1	1	1	2

10	10	10	10	5
5	10	10	10	4
5	1	1	1	2
1	1	1	1	2
1	1	1	1	2
1	1	1	1	2
1	1	1	1	2
3	1	1	1	2
4	1	1	1	2
1	1	1	1	2
1	1	1	3	2
5	10	10	5	4
3	1	1	1	2
3	1	1	1	2
3	1	1	1	3
2	1	1	1	2
5	10	10	3	7
4	8	6	4	3
4	8	8	5	4

```
head(data, 5)
```

A data.frame: 5 × 10

	Cl.thickness	Cell.size	Cell.shape	Marg.adhesion	Epith.c.size	Bare.nu
	<int>	<int>	<int>	<int>	<int>	<
1	5	1	1	1	2	
2	5	4	4	5	7	
3	3	1	1	1	2	
4	6	8	8	1	3	
5	4	1	1	3	2	

```
tail(data, 5)
```

A data.frame: 5 × 10

	Cl.thickness	Cell.size	Cell.shape	Marg.adhesion	Epith.c.size	Bare.
	<int>	<int>	<int>	<int>	<int>	
695	3	1	1	1	3	
696	2	1	1	1	2	
697	5	10	10	3	7	
698	4	8	6	4	3	
699	4	8	8	5	4	

```
str(data)
```

```
'data.frame':   699 obs. of  10 variables:
 $ Cl.thickness   : int  5 5 3 6 4 8 1 2 2 4 ...
 $ Cell.size      : int  1 4 1 8 1 10 1 1 1 2 ...
 $ Cell.shape     : int  1 4 1 8 1 10 1 2 1 1 ...
 $ Marg.adhesion  : int  1 5 1 1 3 8 1 1 1 1 ...
 $ Epith.c.size   : int  2 7 2 3 2 7 2 2 2 2 ...
 $ Bare.nuclei    : int  1 10 2 4 1 10 10 1 1 1 ...
 $ Bl.cromatin    : int  3 3 3 3 3 9 3 3 1 2 ...
 $ Normal.nucleoli: int  1 2 1 7 1 7 1 1 1 1 ...
 $ Mitoses        : int  1 1 1 1 1 1 1 1 5 1 ...
 $ Class          : chr  "benign" "benign" "benign" "benign" ...
```

```
dim(data)
```

699 · 10

```
sapply(data, class)
```

Cl.thickness: 'integer' Cell.size: 'integer' Cell.shape: 'integer' Marg.adhesion: 'integer' Epith.c.size: 'integer' Bare.nuclei: 'integer' Bl.cromatin: 'integer' Normal.nucleoli: 'integer' Mitoses: 'integer' Class: 'character'

```
# Summary statistics for the data frame
summary(data)
```

```
 Cl.thickness      Cell.size        Cell.shape     Marg.adhesion
Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 1.000
Median : 4.000   Median : 1.000   Median : 1.000   Median : 1.000
Mean   : 4.418   Mean   : 3.134   Mean   : 3.207   Mean   : 2.807
3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.000
Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000

 Epith.c.size     Bare.nuclei      Bl.cromatin     Normal.nucleoli
Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000
Median : 2.000   Median : 1.000   Median : 3.000   Median : 1.000
Mean   : 3.216   Mean   : 3.545   Mean   : 3.438   Mean   : 2.867
3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 4.000
Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
                 NA's   :16
   Mitoses          Class
Min.   : 1.000   Length:699
1st Qu.: 1.000   Class :character
Median : 1.000   Mode  :character
Mean   : 1.589
3rd Qu.: 1.000
Max.   :10.000
```

4.DATA CLEANING

This code processes and manipulates data to remove errors, inconsistencies, or irrelevant information, ensuring the data is suitable for analysis or use in a program.

CHANGE VARIABLE

```
data$Class <- as.factor(data$Class)
```

CHECKING MISSING VALUES HANDLING

```
# Count missing values in each column
missing_counts <- colSums(is.na(data))

# Sort in descending order
missing_counts <- sort(missing_counts, decreasing = TRUE)

# Print the result
print(missing_counts)
```

```
        Bare.nuclei    Cl.thickness       Cell.size      Cell.shape   Marg.adhe
                 16               0               0               0
       Epith.c.size     Bl.cromatin Normal.nucleoli         Mitoses           (
                  0               0               0               0
```

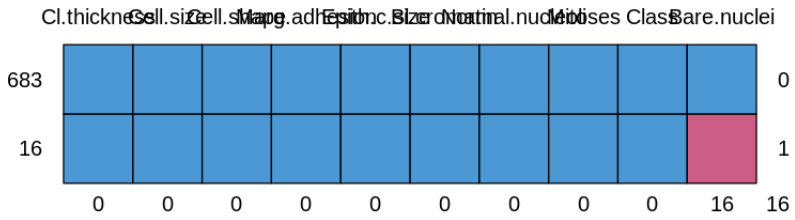
```
install.packages("mice")
library(mice)
```

```
    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)

    also installing the dependencies ‘minqa’, ‘nloptr’, ‘ucminf’, ‘numDeriv’,


    Attaching package: ‘mice’


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

        filter


    The following objects are masked from ‘package:base’:

        cbind, rbind
```

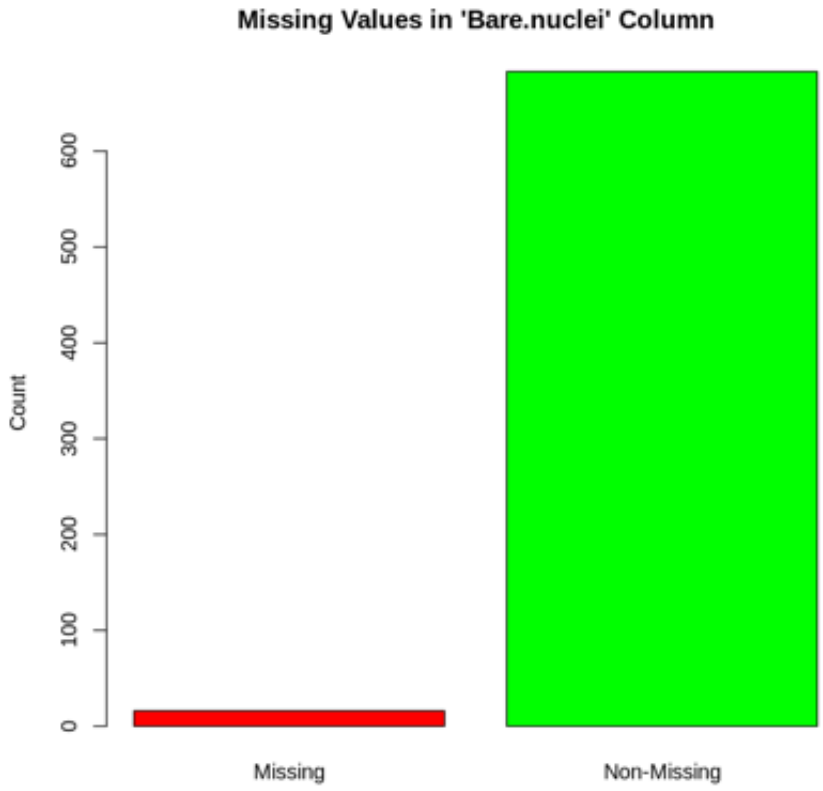
```
# Generate missing data pattern plot
md.pattern(data)
```

A matrix: 3 × 11 of type dbl

	Cl.thickness	Cell.size	Cell.shape	Marg.adhesion	Epith.c.size	Bl.cr
683	1	1	1	1	1	
16	1	1	1	1	1	
	0	0	0	0	0	



```
# Count missing values in 'Bare.nuclei' column
missing_values <- sum(is.na(data$Bare.nuclei))

# Count non-missing values in 'Bare.nuclei' column
non_missing_values <- sum(!is.na(data$Bare.nuclei))

# Create a bar plot
barplot(c(missing_values, non_missing_values),
        names.arg = c("Missing", "Non-Missing"),
        col = c("red", "green"),
        main = "Missing Values in 'Bare.nuclei' Column",
        ylab = "Count")
```



HANDLING MISSING VALUES

```
install.packages("tidyverse")
library(tidyverse)

# Assuming 'data' is your dataframe with missing values
# Replace 'Bare.nuclei' with the actual name of the column containing missing va

df <- data %>%
  mutate(Bare.nuclei = ifelse(is.na(Bare.nuclei), mean(Bare.nuclei, na.rm = TRUE
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

── Attaching core tidyverse packages ──────────────────────── tidyverse
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ lubridate 1.9.3     ✔ tibble    3.2.1
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_con
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
```

```
df
```

A data.frame: 699 × 10

Cl.thickness	Cell.size	Cell.shape	Marg.adhesion	Epith.c.size	Bare.nucle
<int>	<int>	<int>	<int>	<int>	<db
5	1	1	1	2	1.00000
5	4	4	5	7	10.0000
3	1	1	1	2	2.00000
6	8	8	1	3	4.00000
4	1	1	3	2	1.00000
8	10	10	8	7	10.0000
1	1	1	1	2	10.0000
2	1	2	1	2	1.00000
2	1	1	1	2	1.00000
4	2	1	1	2	1.00000
1	1	1	1	1	1.00000
2	1	1	1	2	1.00000

5	3	3	3	2	3.00000
1	1	1	1	2	3.00000
8	7	5	10	7	9.00000
7	4	6	4	6	1.00000
4	1	1	1	2	1.00000
4	1	1	1	2	1.00000
10	7	7	6	4	10.00000
6	1	1	1	2	1.00000
7	3	2	10	5	10.00000
10	5	5	3	6	7.00000
3	1	1	1	2	1.00000
8	4	5	1	2	3.54465
1	1	1	1	2	1.00000
5	2	3	4	2	7.00000
3	2	1	1	1	1.00000
5	1	1	1	2	1.00000
2	1	1	1	2	1.00000
1	1	3	1	2	1.00000
⋮	⋮	⋮	⋮	⋮	
5	10	10	8	5	
3	10	7	8	5	
3	2	1	2	2	
2	1	1	1	2	
5	3	2	1	3	
1	1	1	1	2	
4	1	4	1	2	
1	1	2	1	2	
5	1	1	1	2	
1	1	1	1	2	
2	1	1	1	2	

10	10	10	10	5
5	10	10	10	4
5	1	1	1	2
1	1	1	1	2
1	1	1	1	2
1	1	1	1	2
1	1	1	1	2
3	1	1	1	2
4	1	1	1	2
1	1	1	1	2
1	1	1	3	2
5	10	10	5	4
3	1	1	1	2
3	1	1	1	2
3	1	1	1	3
2	1	1	1	2
5	10	10	3	7
4	8	6	4	3
4	8	8	5	4

```
# Check for missing values in each column
missing_values <- colSums(is.na(df))

# Print the result
print(missing_values)
```

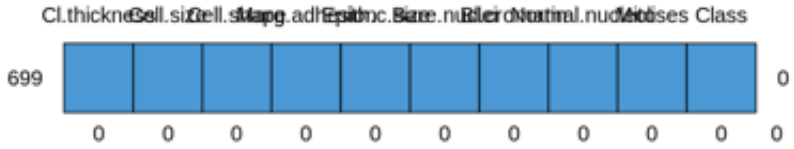
```
        Cl.thickness       Cell.size      Cell.shape   Marg.adhesion    Epith.c
                   0               0               0               0
         Bare.nuclei     Bl.cromatin Normal.nucleoli         Mitoses           (
                   0               0               0               0
```

```
# Generate missing data pattern plot
md.pattern(df)
```

```
 /\     /\
{  `---'  }
{  O   O  }
==>  V <==  No need for mice. This data set is completely observed.
 \  \|/  /
  `-----'
```

A matrix: 2 × 11 of type dbl

	Cl.thickness	Cell.size	Cell.shape	Marg.adhesion	Epith.c.size	Bare.
699	1	1	1	1	1	
	0	0	0	0	0	



```
round(prop.table(table(df$Class)), 2)
```

```
   benign malignant 
     0.66      0.34 
```

5.EXPLORATORY DATA ANALYSIS

This code performs initial investigation and analysis of data sets to summarize their main characteristics, often using statistical graphics and other data visualization techniques.

STATISTICS SUMMARY

```
# Summary statistics for the data frame
summary(df)
```

```
 Cl.thickness      Cell.size        Cell.shape     Marg.adhesion
Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 1.000
Median : 4.000   Median : 1.000   Median : 1.000   Median : 1.000
Mean   : 4.418   Mean   : 3.134   Mean   : 3.207   Mean   : 2.807
3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.000
Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
 Epith.c.size     Bare.nuclei      Bl.cromatin     Normal.nucleoli
Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000
Median : 2.000   Median : 1.000   Median : 3.000   Median : 1.000
Mean   : 3.216   Mean   : 3.545   Mean   : 3.438   Mean   : 2.867
3rd Qu.: 4.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.000
Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
   Mitoses            Class
Min.   : 1.000   benign   :458
1st Qu.: 1.000   malignant:241
Median : 1.000
Mean   : 1.589
3rd Qu.: 1.000
Max.   :10.000
```

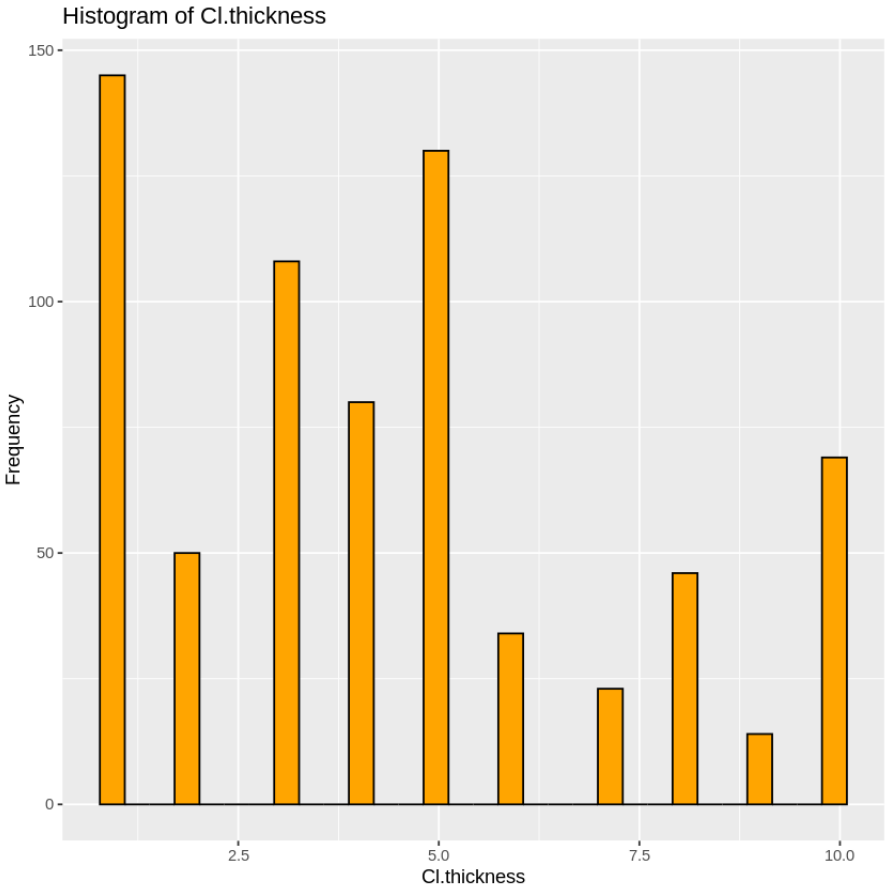
UNIVARIATE ANALYSIS

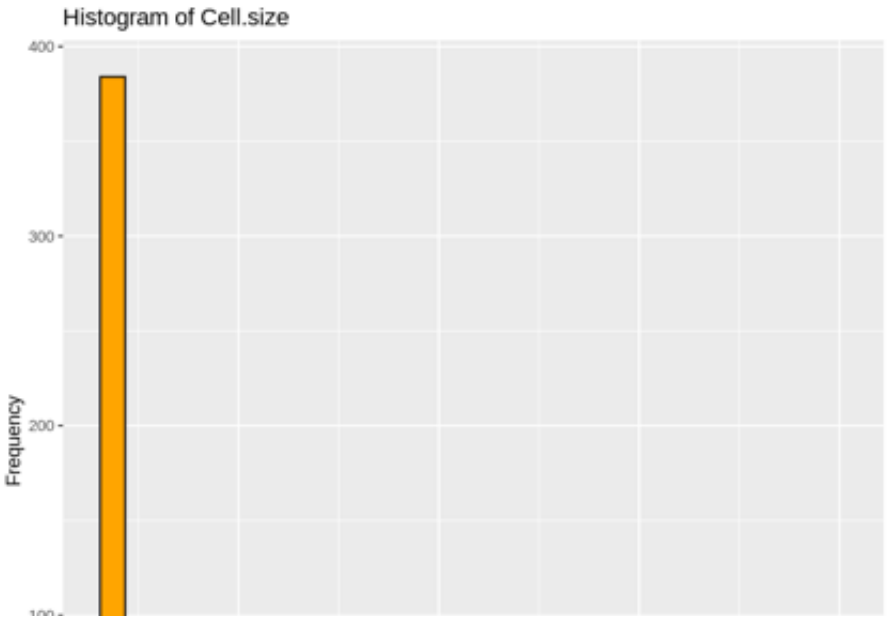
NUMERICAL VARIABLES

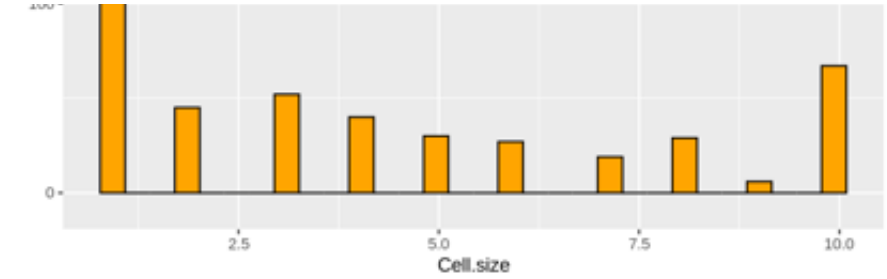
to create histograms for each numerical feature

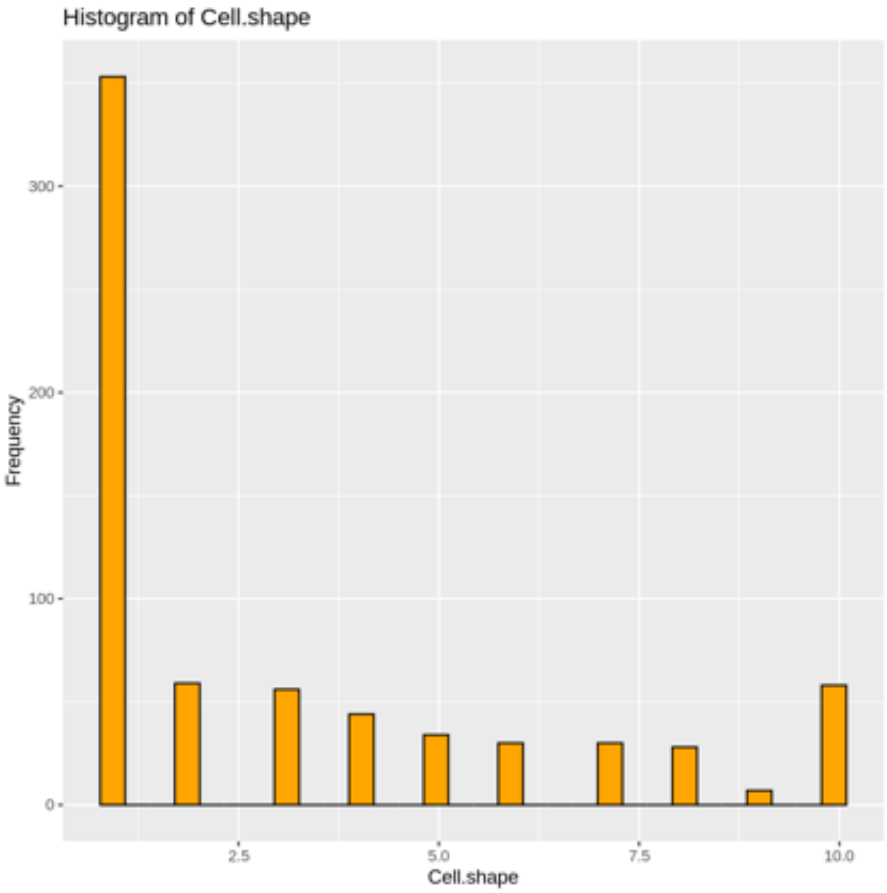
```
library(ggplot2)

# Get the numerical columns in the dataframe
numeric_cols <- sapply(df, is.numeric)

# Create histograms for each numerical feature
for (col in names(df)[numeric_cols]) {
```

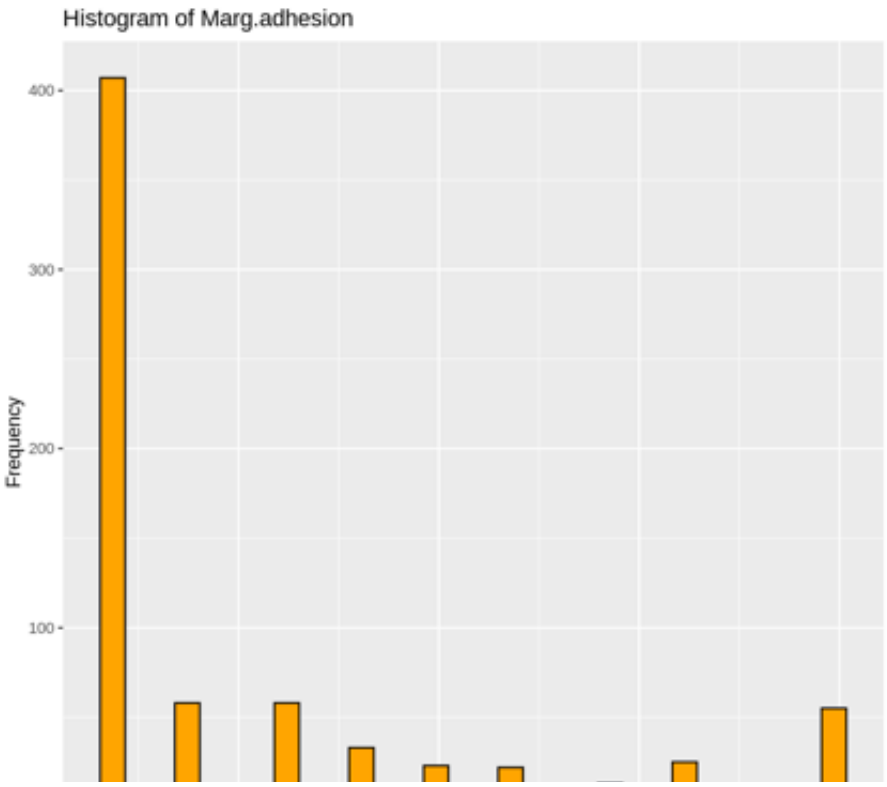
```
  plot_data <- data.frame(x = df[[col]])

  plot <- ggplot(plot_data, aes(x = x)) +
    geom_histogram(fill = "orange", color = "black", bins = 30) +
    ggtitle(paste("Histogram of", col)) +
    labs(x = col, y = "Frequency")

  print(plot)
}
```

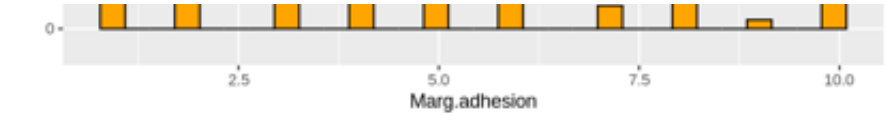


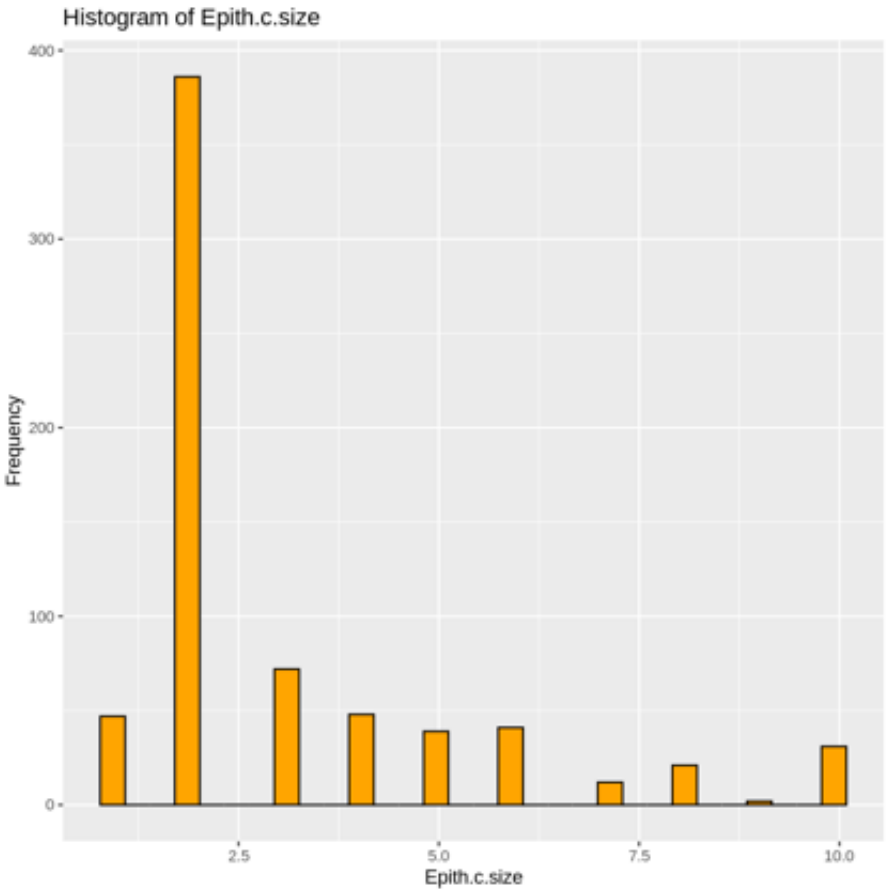


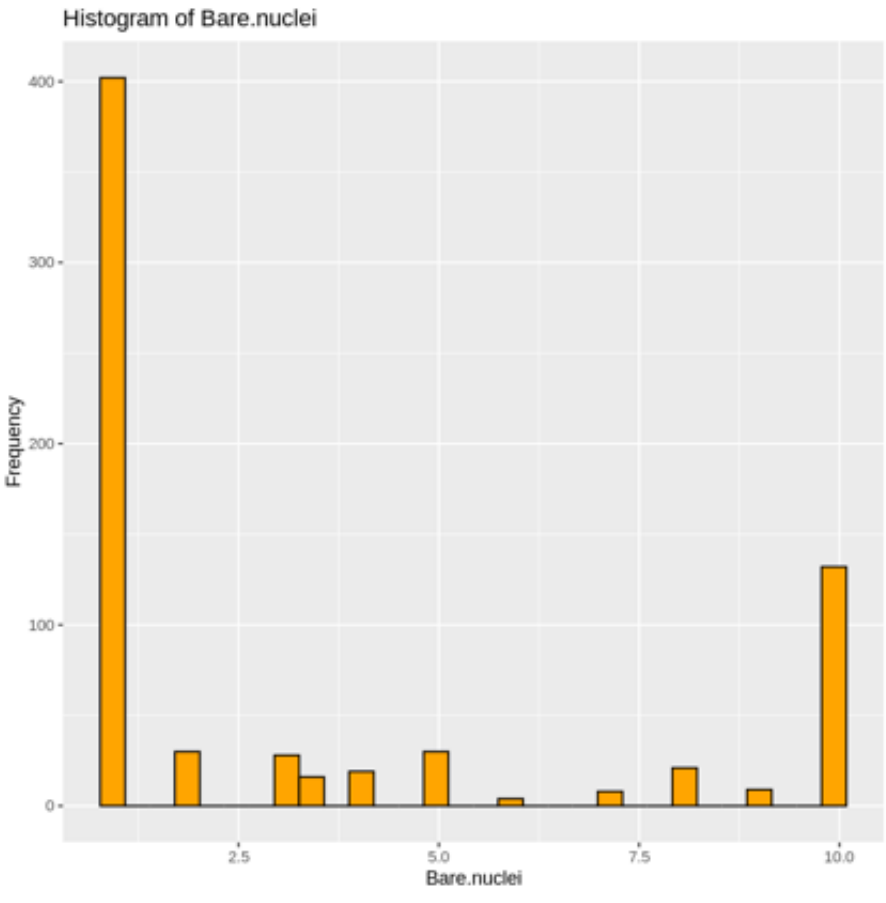

0
2.5
5.0
7.5
10.0
Cell.size


Histogram of Cell.shape
Frequency
300
200
100
0
2.5
5.0
7.5
10.0
Cell.shape

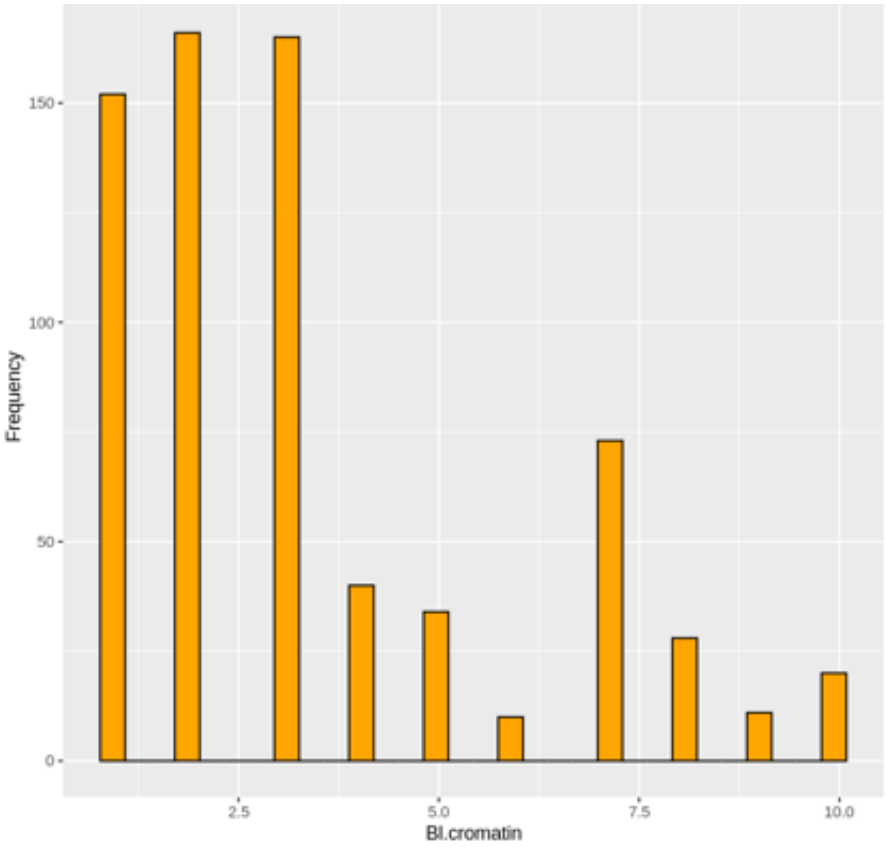

Histogram of Marg.adhesion
Frequency
400
300
200
100


0
2.5
5.0
7.5
10.0
Marg.adhesion

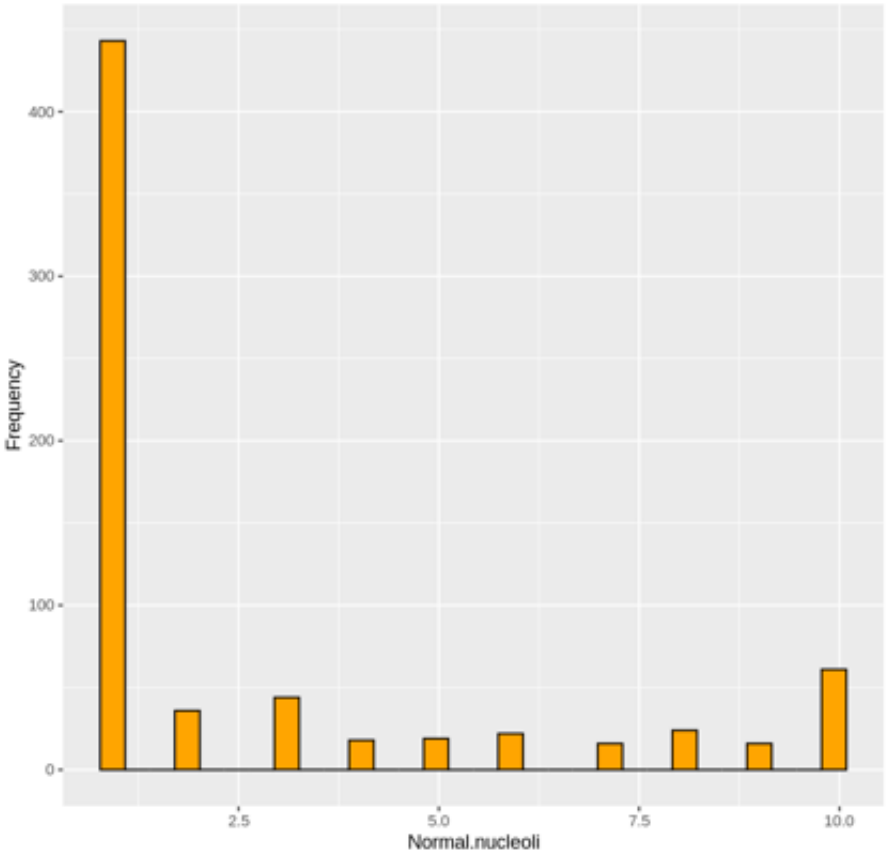

Histogram of Epith.c.size
400
300
200
100
0
Frequency
2.5
5.0
7.5
10.0
Epith.c.size


Histogram of Bare.nuclei
400
300
200
100
0
Frequency
2.5
5.0
7.5
10.0
Bare.nuclei

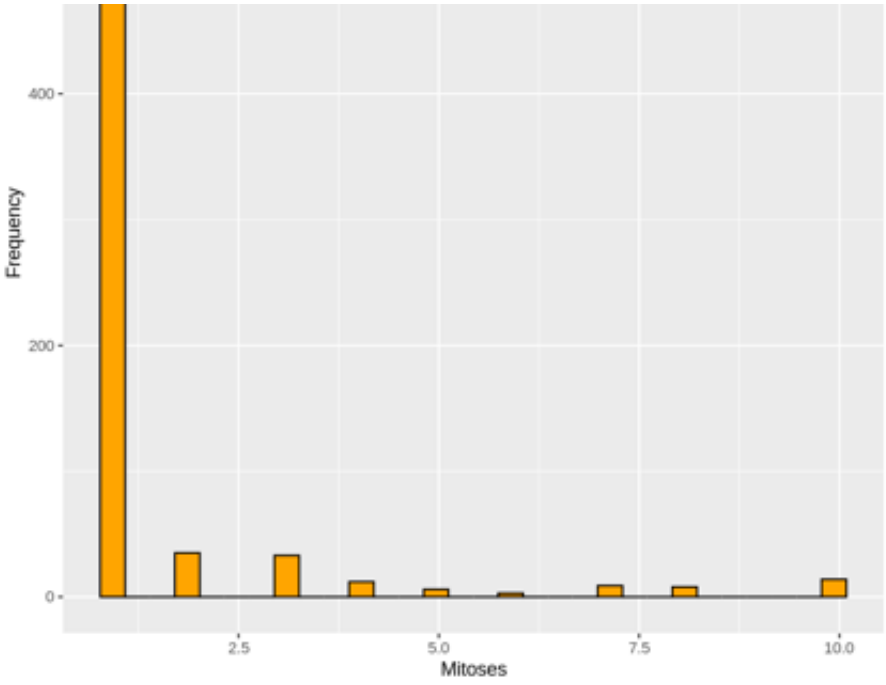

Histogram of Bl.cromatin



Histogram of Normal.nucleoli



Histogram of Mitoses

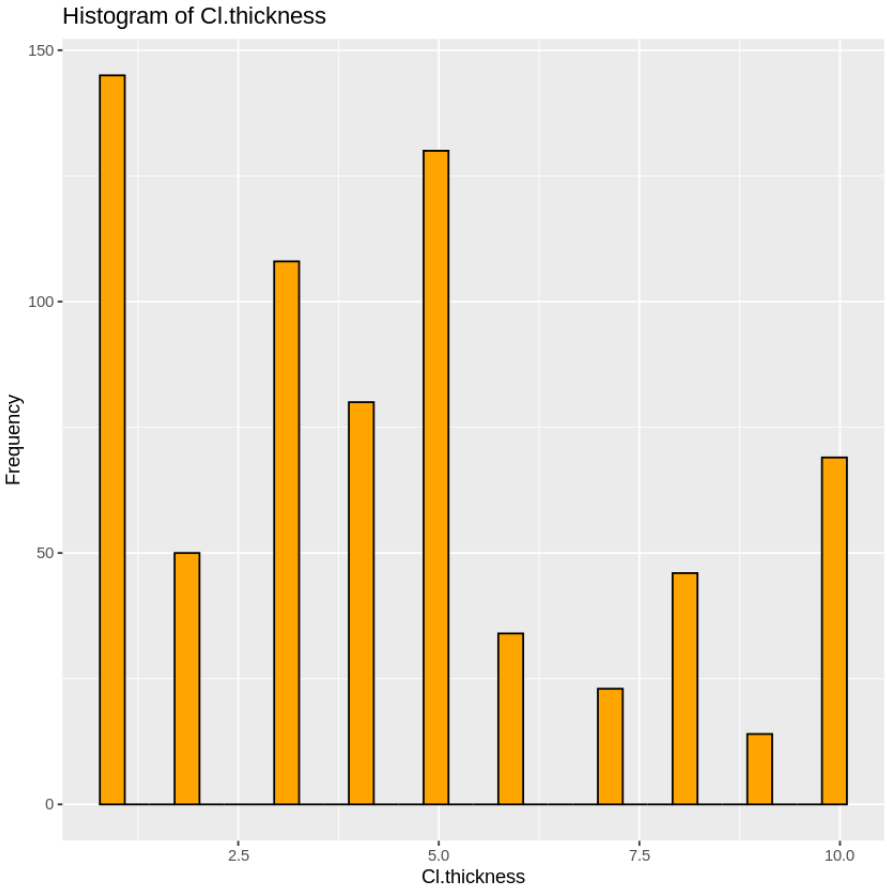


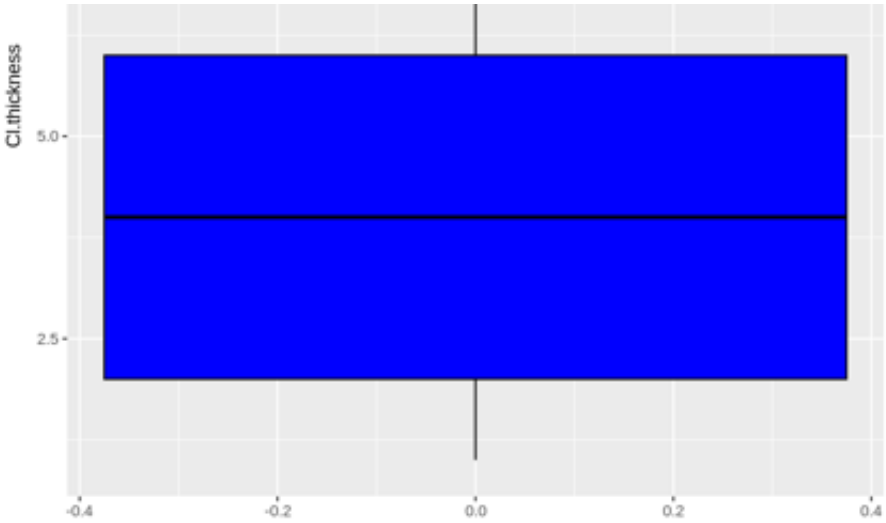
to find the skewness of numerical columns

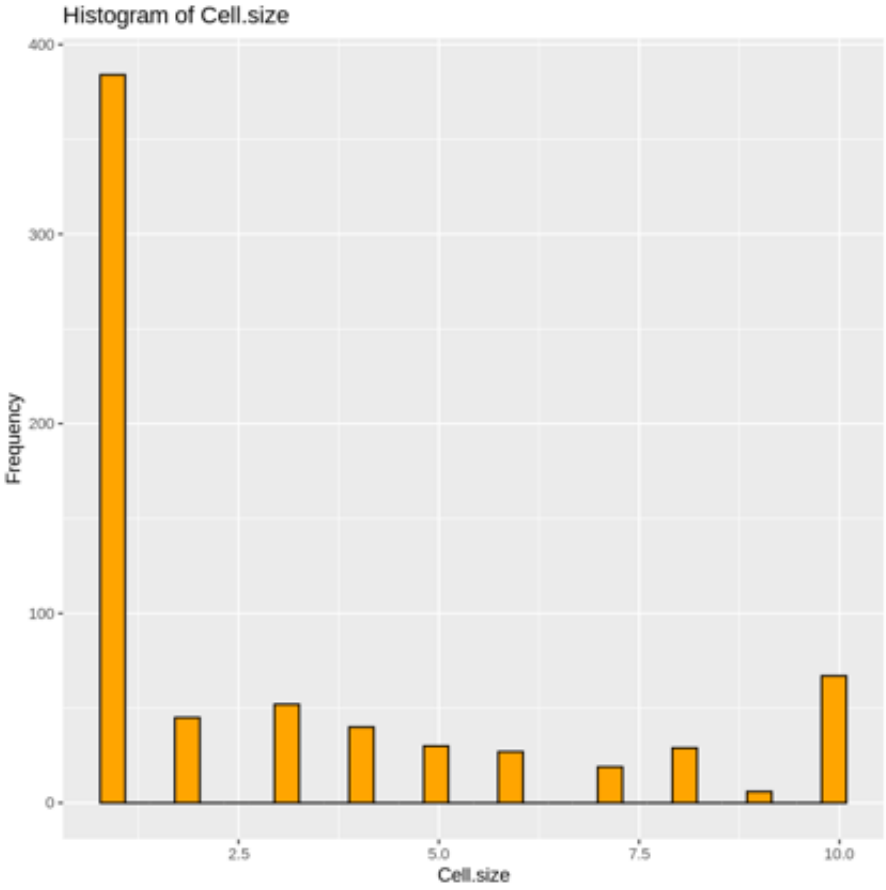
```
install.packages("e1071")

library(ggplot2)
library(e1071)

# Get the numerical columns in the dataframe
numeric_cols <- sapply(df, is.numeric)

# Function to calculate skewness
calculate_skewness <- function(x) {
  skewness <- skewness(x)
  return(skewness)
}

# Calculate skewness for each numerical column
skewness_values <- sapply(df[, numeric_cols], calculate_skewness)

# Create histograms with box plots for each numerical feature
for (col in names(df)[numeric_cols]) {
  plot_data <- data.frame(x = df[[col]])

  hist_plot <- ggplot(plot_data, aes(x = x)) +
    geom_histogram(fill = "orange", color = "black", bins = 30) +
    ggtitle(paste("Histogram of", col)) +
    labs(x = col, y = "Frequency")

  box_plot <- ggplot(plot_data, aes(y = x)) +
```

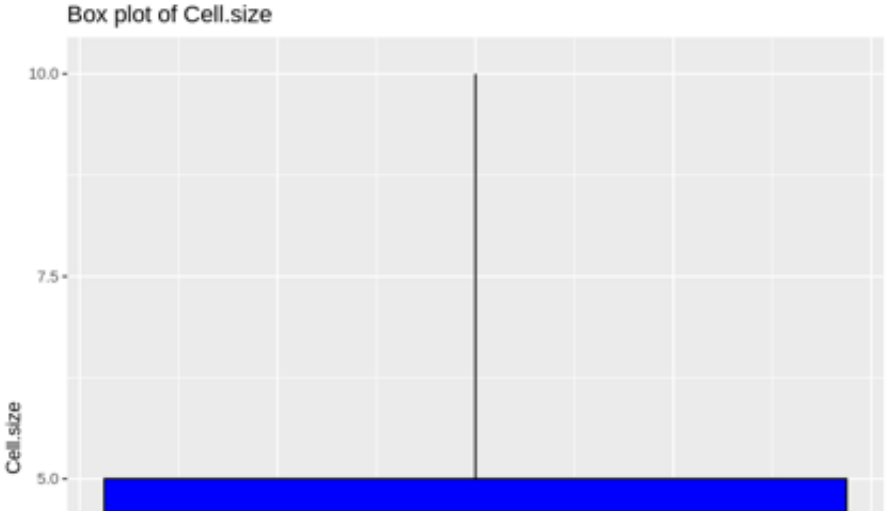
```
    geom_boxplot(fill = "blue", color = "black") +
    ggtitle(paste("Box plot of", col)) +
    labs(x = "", y = col)

  print(hist_plot)
  print(box_plot)
}

# Print skewness values
print(skewness_values)
```

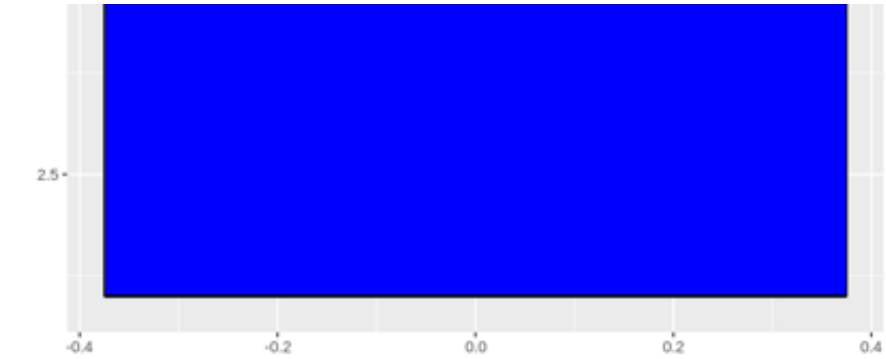
```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependency ‘proxy’
```

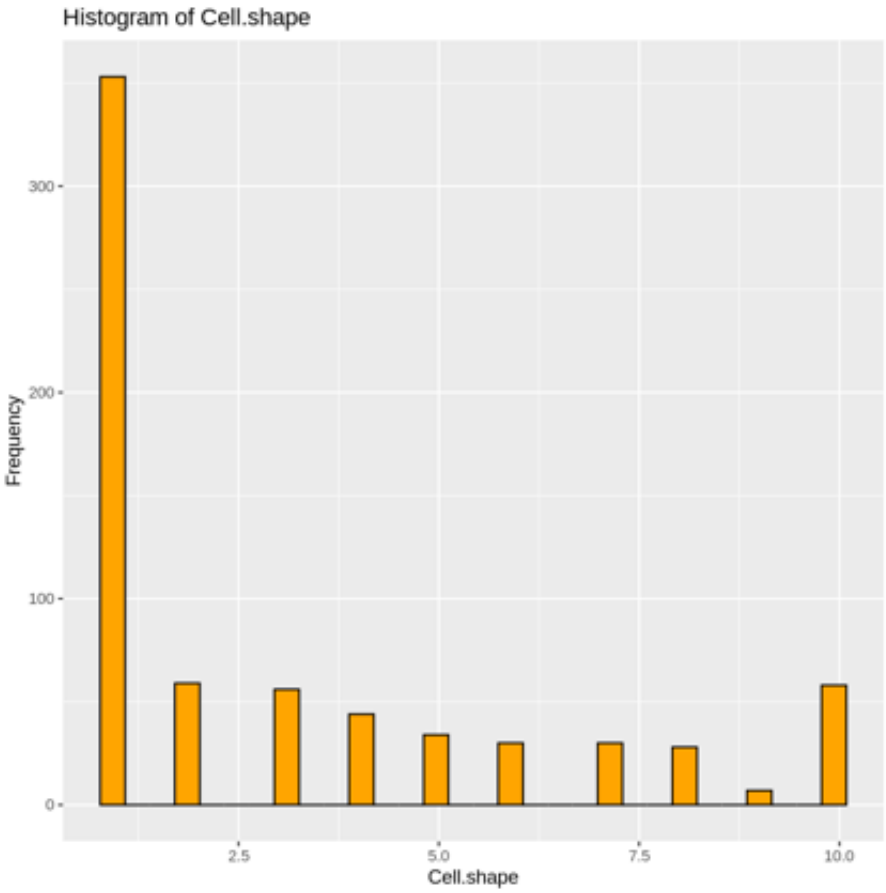


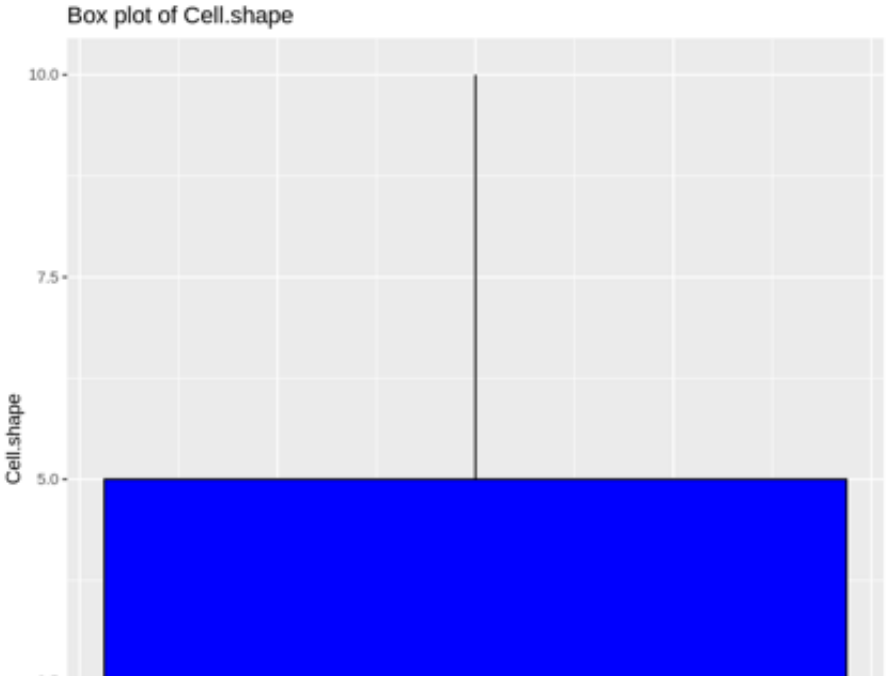

Cl.thickness
5.0
2.5
-0.4
-0.2
0.0
0.2
0.4

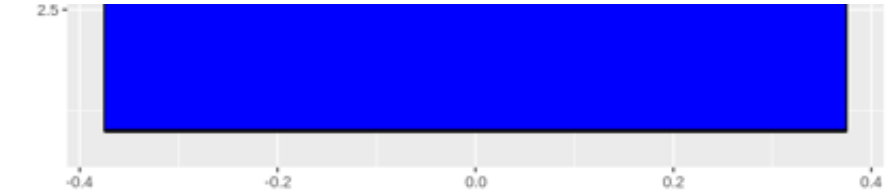

Histogram of Cell.size
Frequency
400
300
200
100
0
2.5
5.0
7.5
10.0
Cell.size

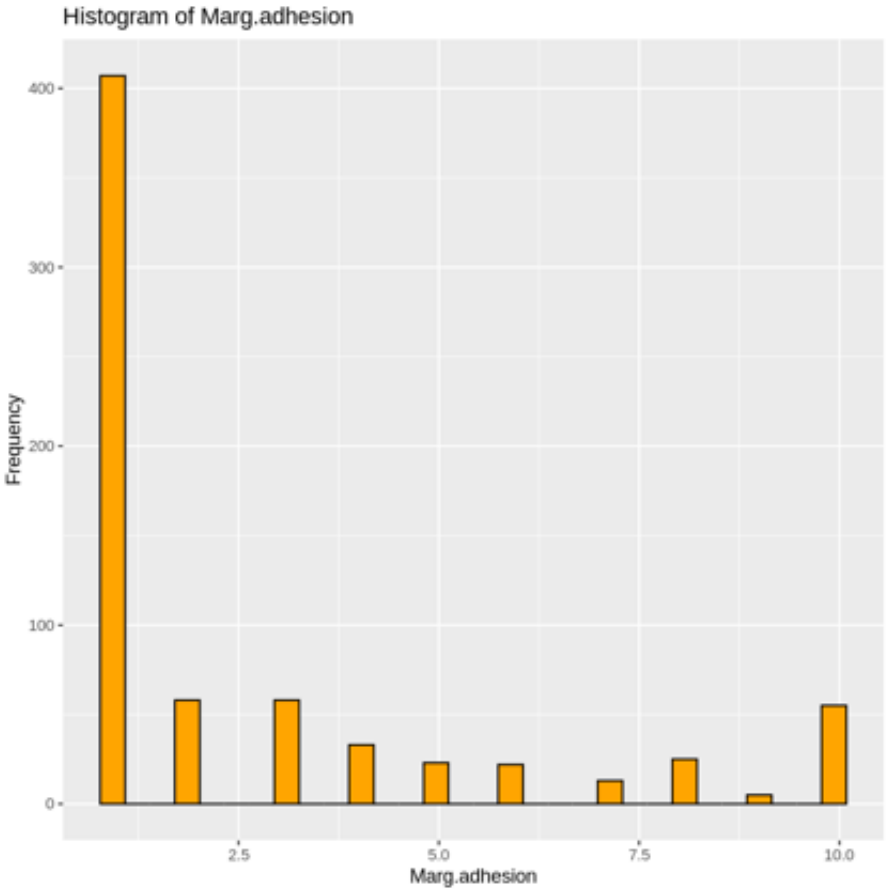

Box plot of Cell.size
Cell.size
10.0
7.5
5.0

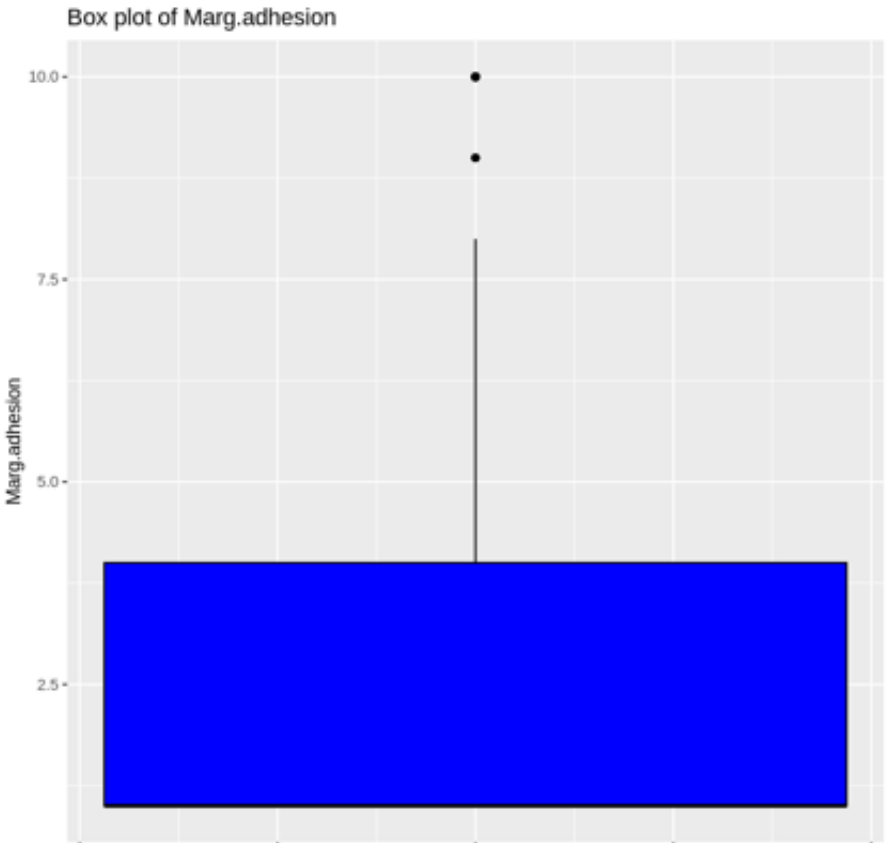

2.5
-0.4
-0.2
0.0
0.2
0.4

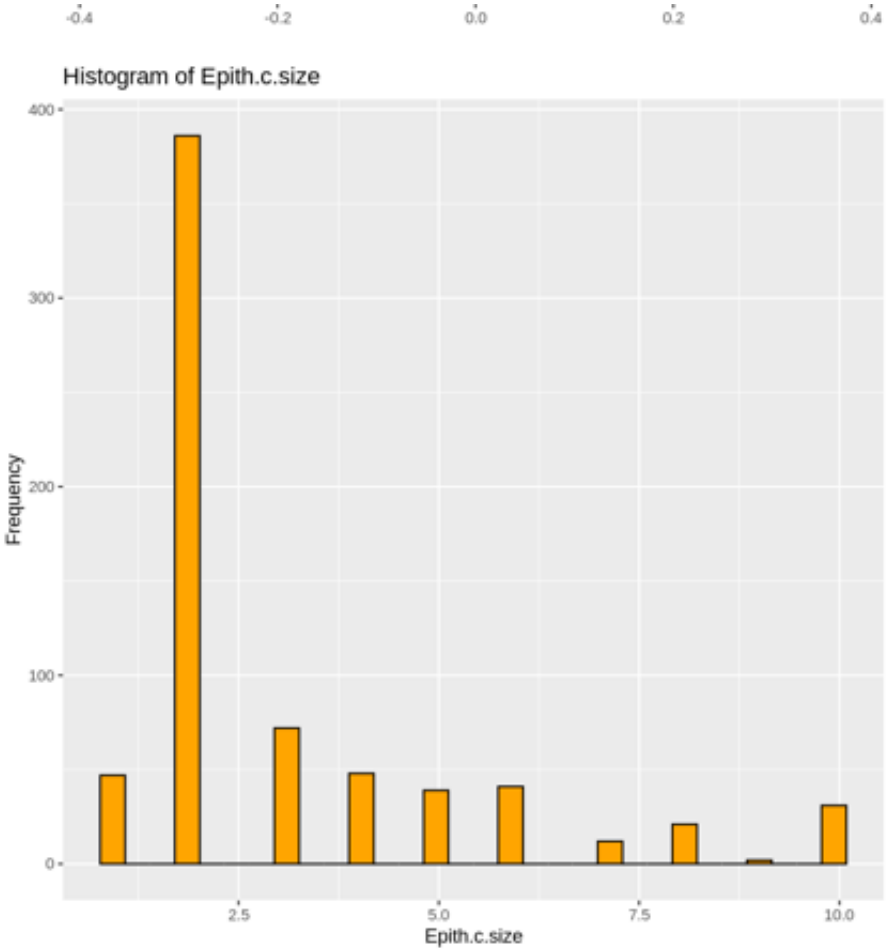

Histogram of Cell.shape
Frequency
300
200
100
0
2.5
5.0
7.5
10.0
Cell.shape

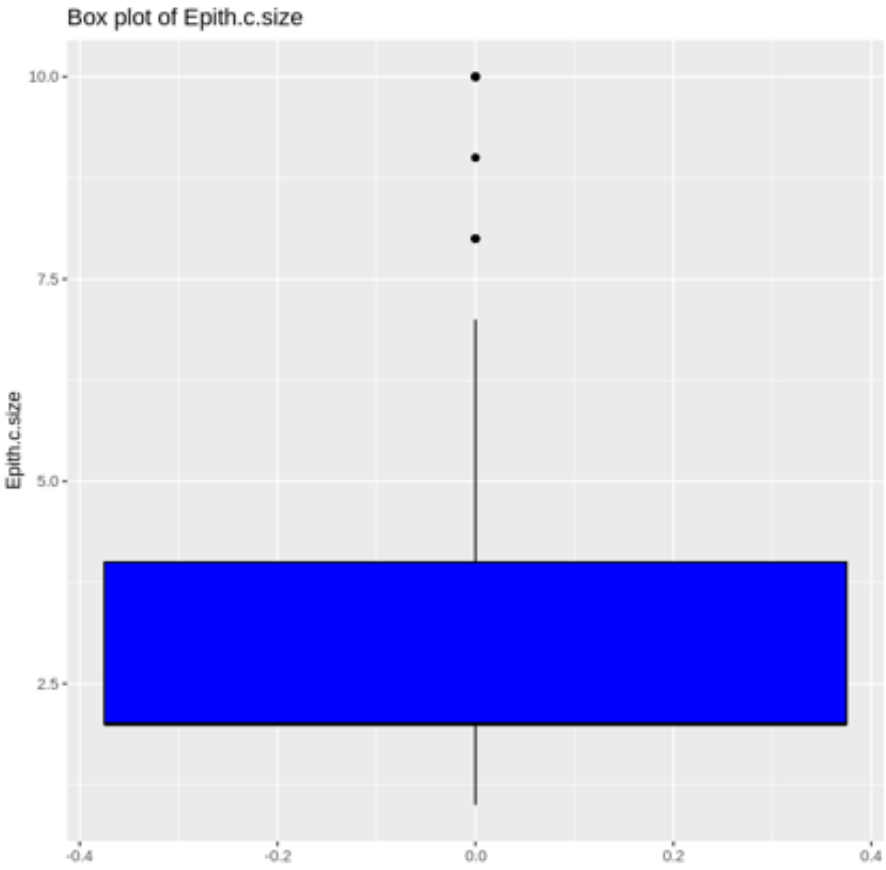

Box plot of Cell.shape
Cell.shape
10.0
7.5
5.0

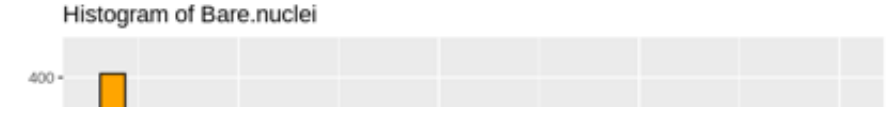

2.5
-0.4
-0.2
0.0
0.2
0.4

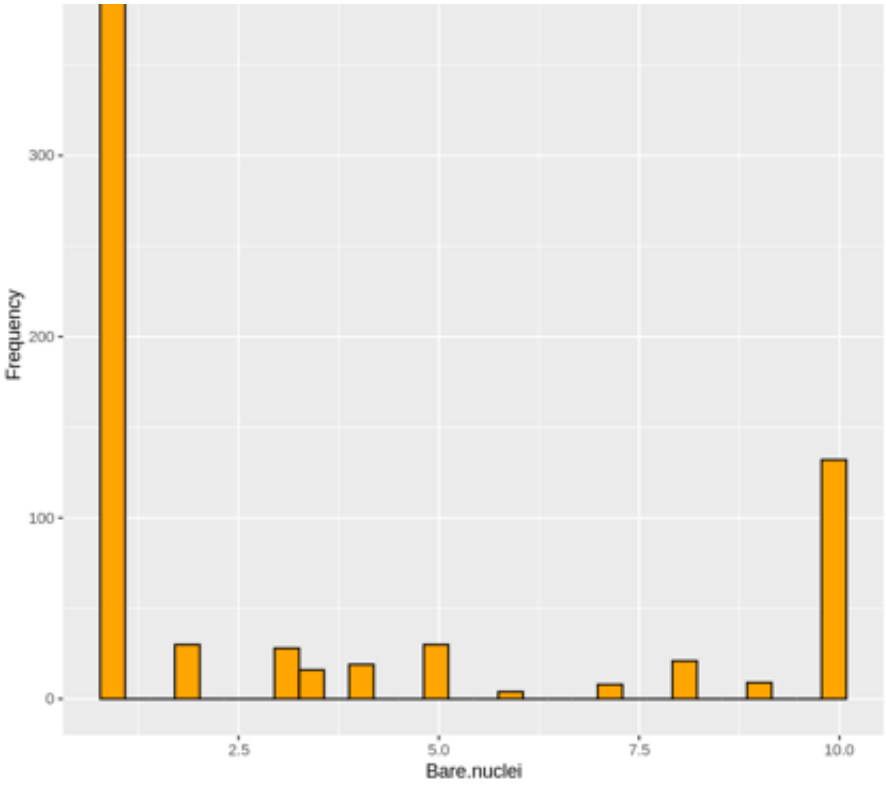

Histogram of Marg.adhesion
Frequency
400
300
200
100
0
2.5
5.0
7.5
10.0
Marg.adhesion


Box plot of Marg.adhesion
Marg.adhesion
10.0
7.5
5.0
2.5

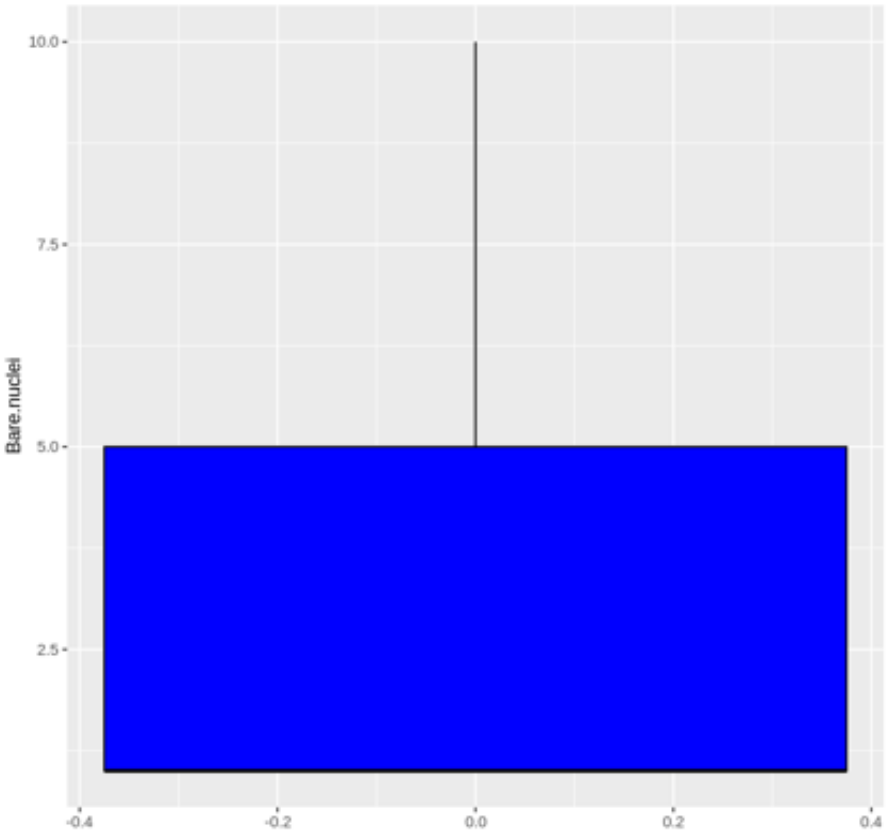

-0.4
-0.2
0.0
0.2
0.4
Histogram of Epith.c.size
400
300
200
100
0
Frequency
2.5
5.0
7.5
10.0
Epith.c.size


Box plot of Epith.c.size
10.0
7.5
5.0
2.5
Epith.c.size
-0.4
-0.2
0.0
0.2
0.4

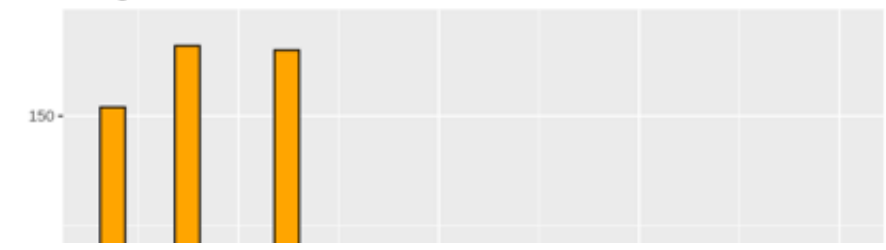

Histogram of Bare.nuclei
400

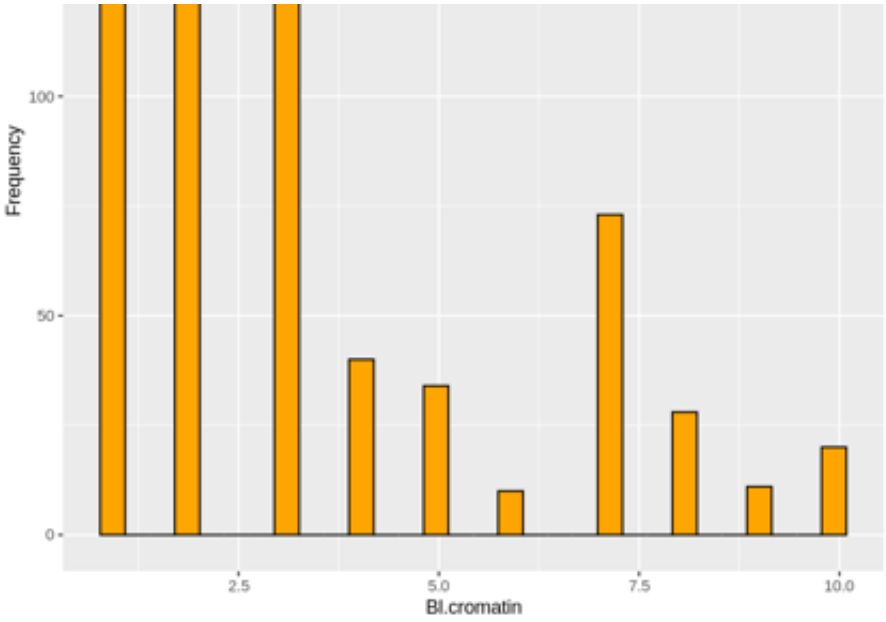


Box plot of Bare.nuclei

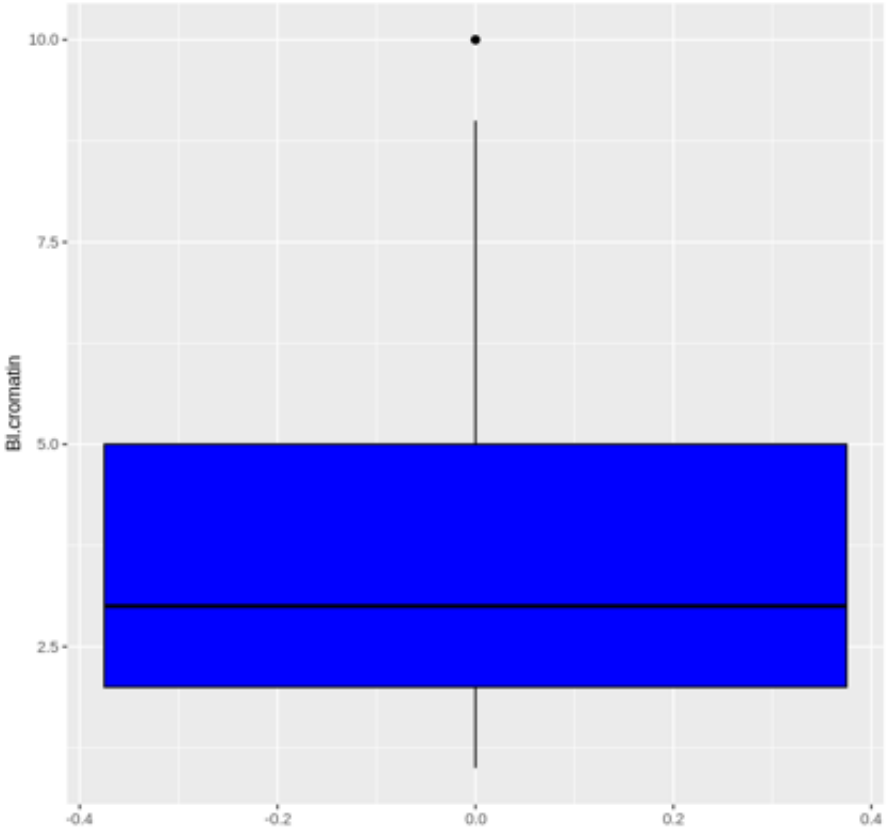


Histogram of Bl.cromatin

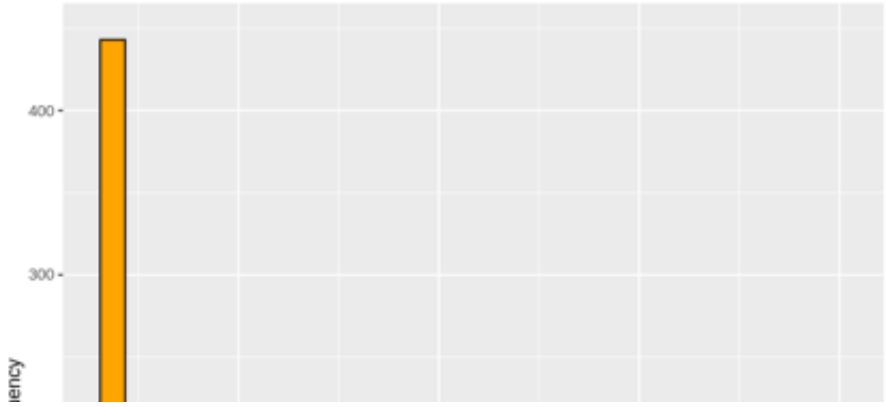


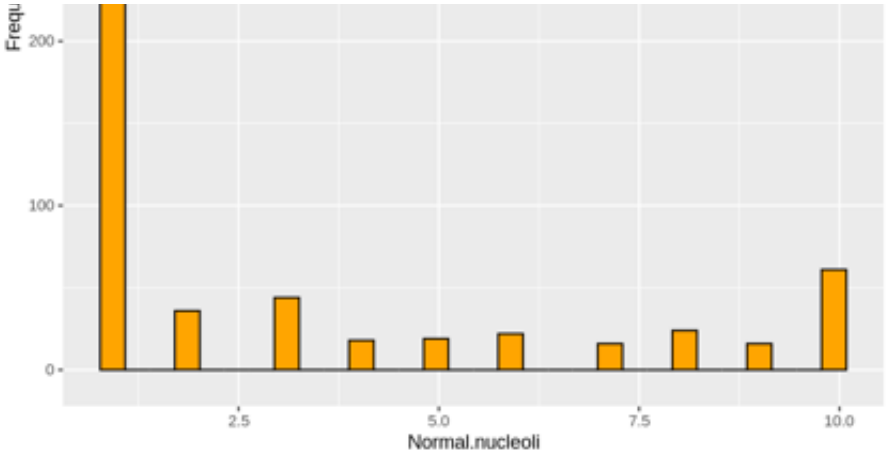


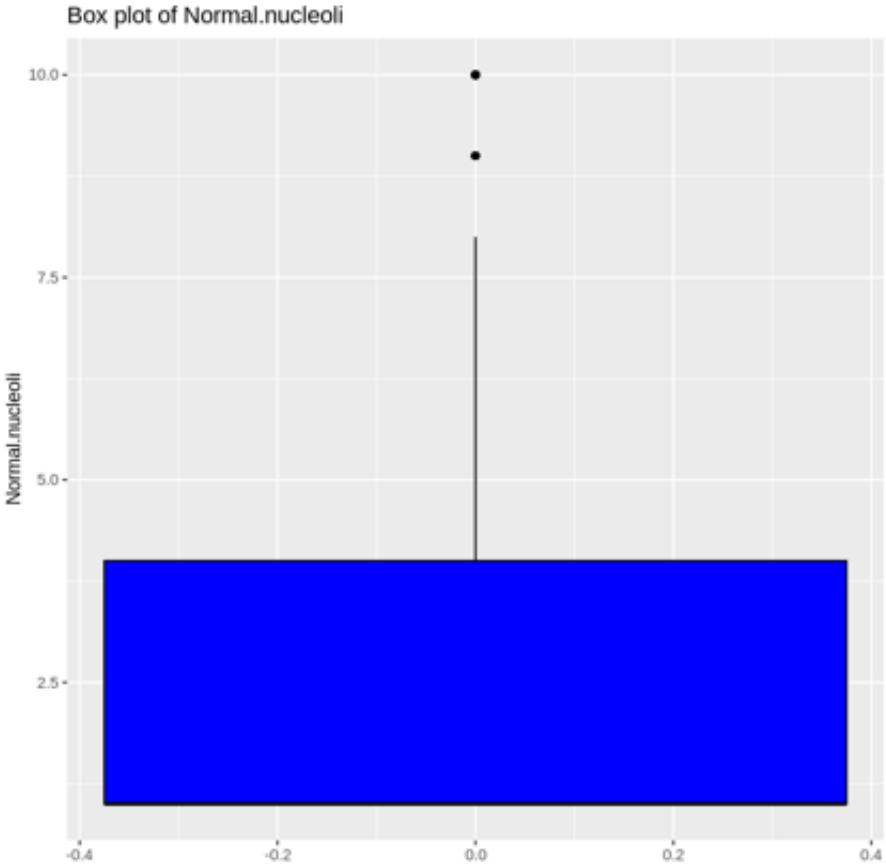
Box plot of Bl.cromatin

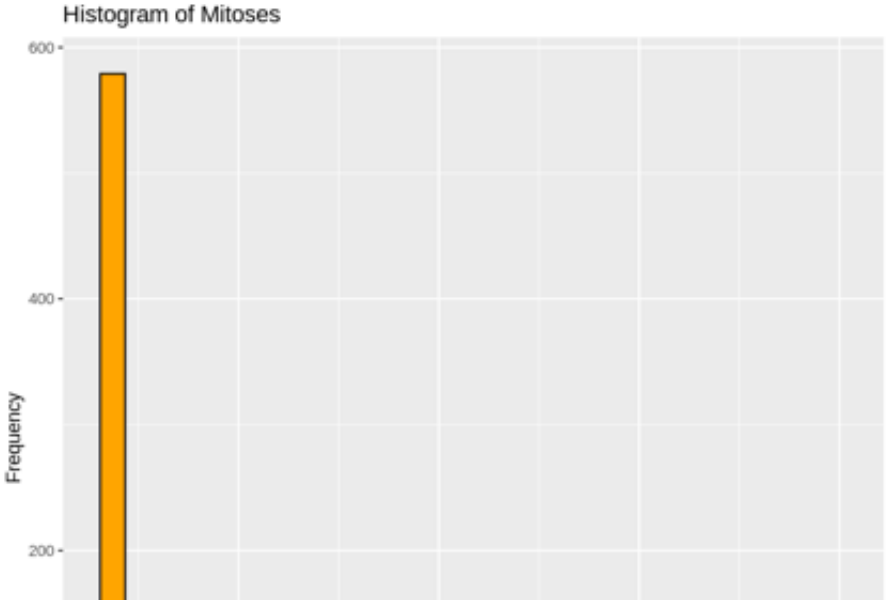


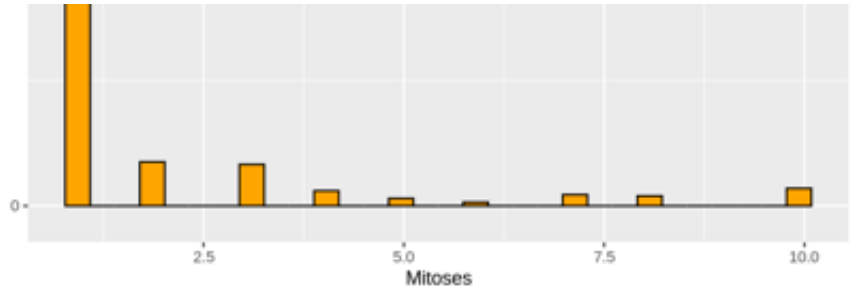
Histogram of Normal.nucleoli



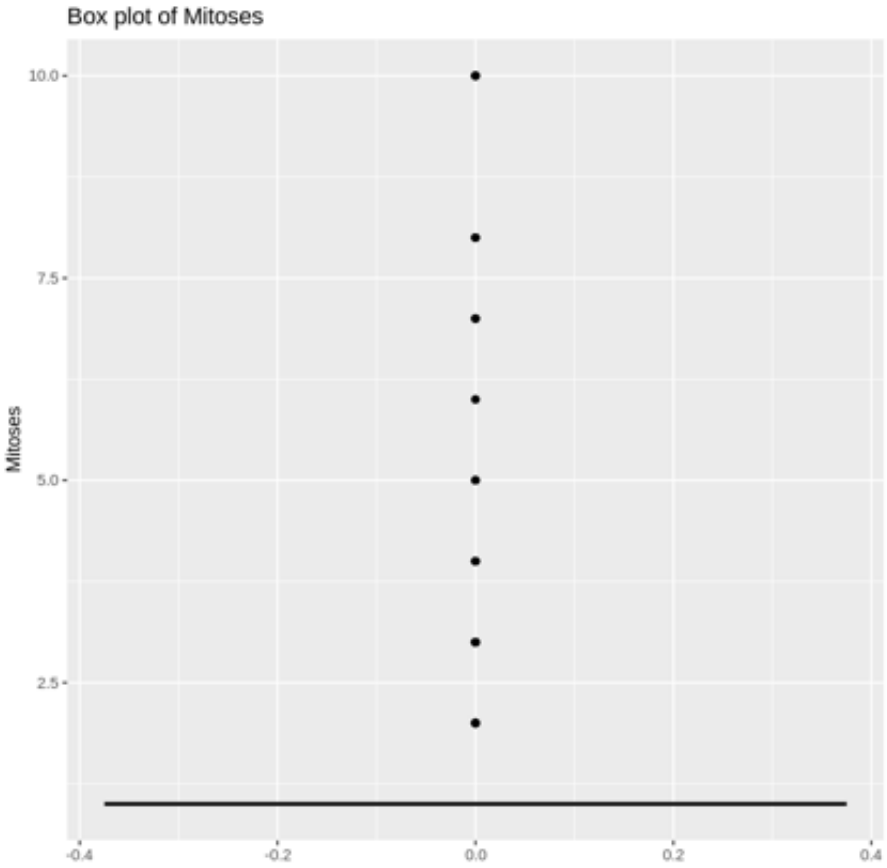

Frequ
200
100
0
2.5
5.0
7.5
10.0
Normal.nucleoli


Box plot of Normal.nucleoli
10.0
7.5
5.0
2.5
Normal.nucleoli
-0.4
-0.2
0.0
0.2
0.4


Histogram of Mitoses
600
400
200
Frequency

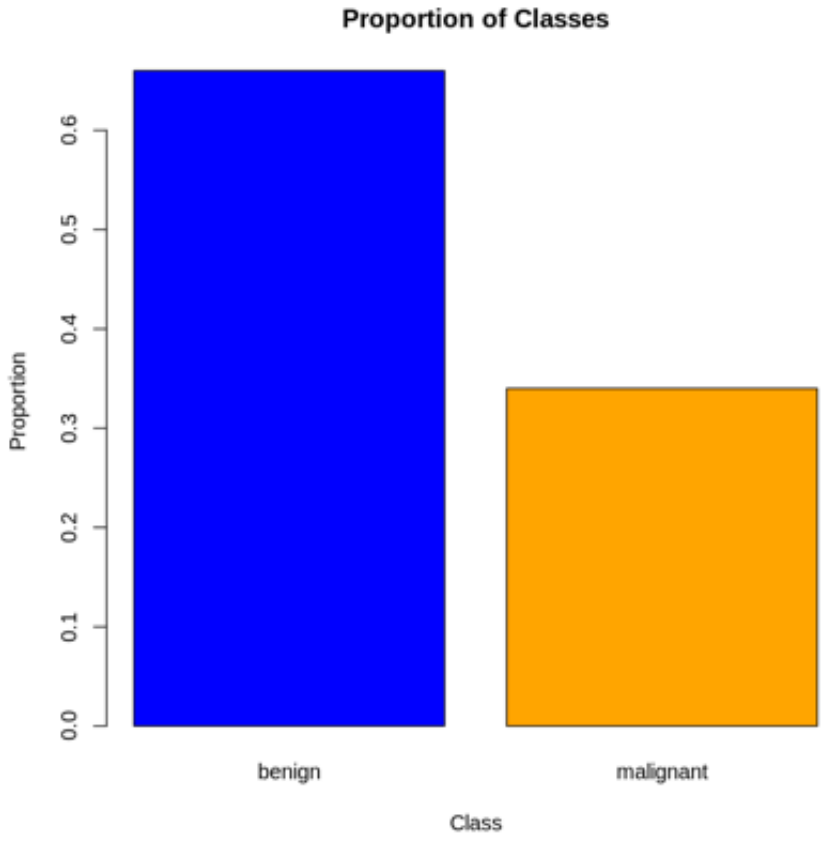



```
Cl.thickness       Cell.size      Cell.shape    Marg.adhesion     Epith.c.
   0.5903165       1.2278492       1.1568774       1.5179315       1.704
 Bare.nuclei     Bl.cromatin Normal.nucleoli         Mitoses
   0.9971999       1.0952527       1.4161630       3.5453906
```



CATEGORICAL VARIABLES

```
# Calculate proportions and round them to 2 decimal places
prop_class <- round(prop.table(table(df$Class)), 2)

# Define colors for each bar
bar_colors <- c("blue", "orange")

# Create a bar plot
barplot(prop_class, main="Proportion of Classes", xlab="Class", ylab="Proporti
```



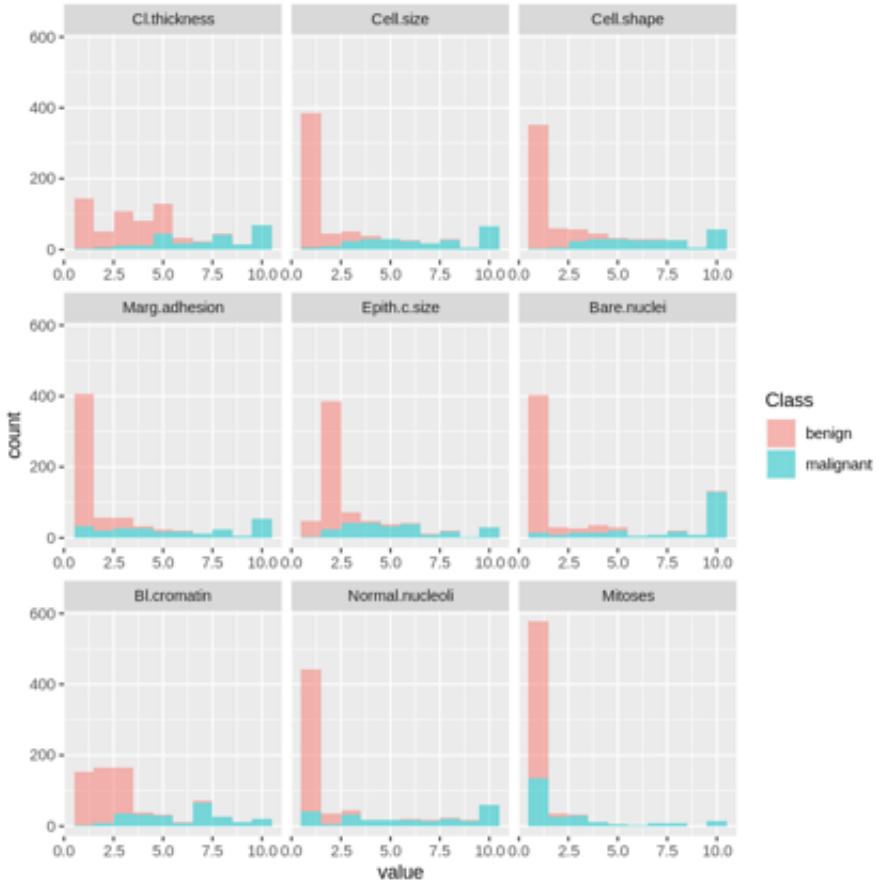
DISTRIBUTION OF HISTOGRAMS

Plot histograms of variables group by CLASS

```
install.packages("reshape2")
library(ggplot2)
library(reshape2)  # For the melt function

# Assuming your data frame is named "df"

# Melt the data frame
melted_df <- melt(df, id.var = "Class")
```

```
# Plot histograms
ggplot(data = melted_df, aes(x = value)) +
  geom_histogram(bins = 10, aes(fill = Class), alpha = 0.5) +
  facet_wrap(~variable, scales = 'free_x')
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘plyr’, ‘Rcpp’


Attaching package: ‘reshape2’


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

    smiths
```



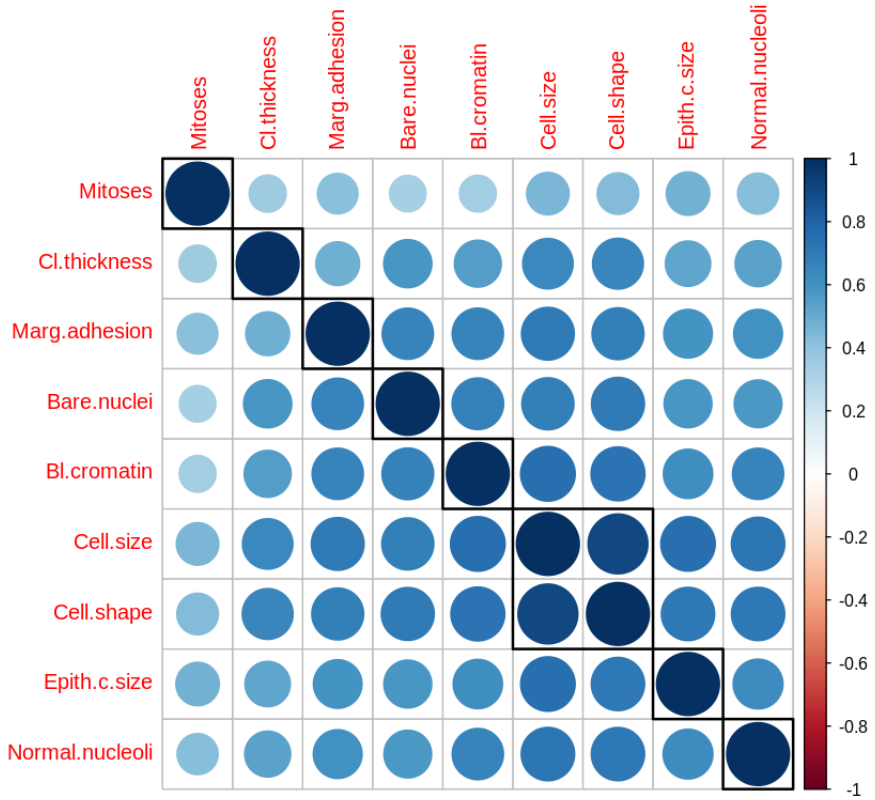
BIVARIATE/MULTIVARIATE ANALYSIS

Here, Correlation checks which variables are more correlated

```
# Install corrplot package
install.packages("corrplot")
```

```
    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)
```

```
df_corr <- cor(df %>% select(-Class))
corrplot::corrplot(df_corr, order = "hclust", tl.cex = 1, addrect = 8)
```



6.MULTICOLLINEARITY HANDLING

This code addresses multicollinearity, a condition where independent variables in a regression model are highly correlated, by employing techniques such as feature selection, dimensionality reduction, or regularization to mitigate its effects and improve model performance.

```
install.packages("caret")
library(caret)
```

```
    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)

    also installing the dependencies ‘listenv’, ‘parallelly’, ‘future’, ‘globa

    Loading required package: lattice

    Attaching package: ‘caret’

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

        lift
```

```
# The findcorrelation() function from caret package remove highly correlated p
# based on whose correlation is above 0.9. This function uses a heuristic algo
# to determine which variable should be removed instead of selecting blindly
df2 <- df %>% select(-findCorrelation(df_corr, cutoff = 0.9))

#Number of columns for our new data frame
ncol(df2)
```

```
    9
```

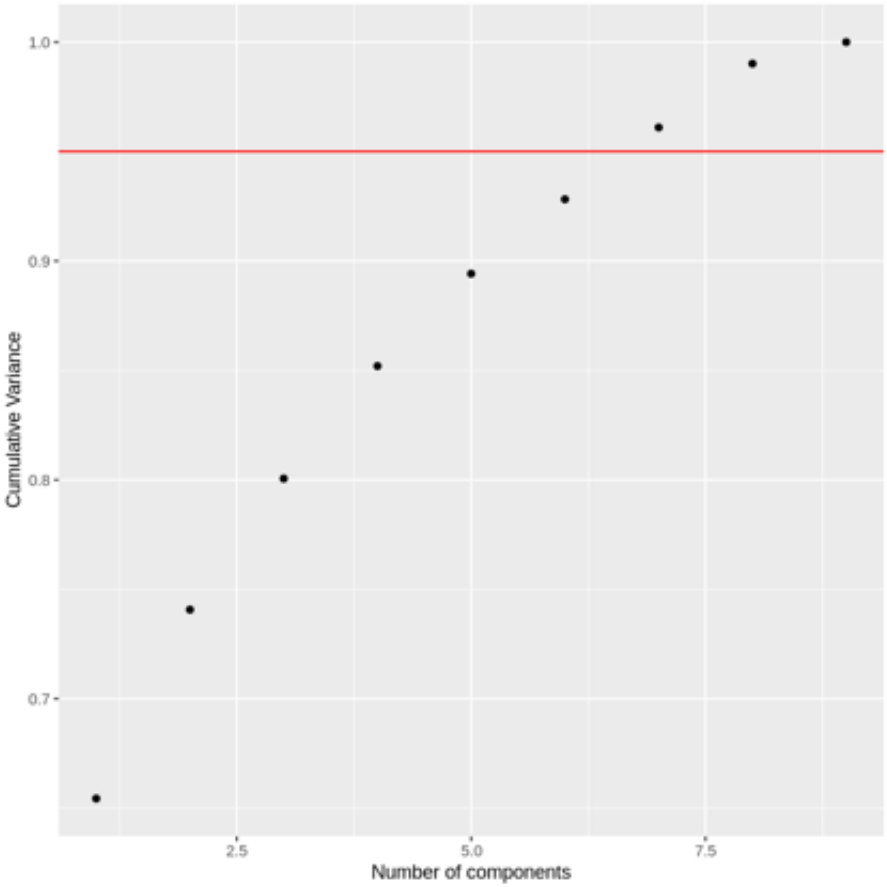
This code preprocesses data using Principal Component Analysis (PCA), a technique used for dimensionality reduction, feature extraction, and data visualization, to prepare it for further analysis or modeling.

USING PCA

```
# Select only numeric columns excluding the 'Class' column
numeric_df <- df %>% select(-Class) %>% select_if(is.numeric)

# Perform PCA
preproc_pca_df <- prcomp(numeric_df, scale = TRUE, center = TRUE)

# Summary of PCA
summary(preproc_pca_df)
```

```
Importance of components:
                          PC1    PC2     PC3     PC4    PC5     PC6
PC7
Standard deviation     2.4268 0.8813 0.73406 0.68034 0.6163 0.55311
0.54347
Proportion of Variance 0.6544 0.0863 0.05987 0.05143 0.0422 0.03399
0.03282
Cumulative Proportion  0.6544 0.7407 0.80056 0.85199 0.8942 0.92818
0.96100
                           PC8     PC9
Standard deviation     0.51219 0.29775
Proportion of Variance 0.02915 0.00985
```

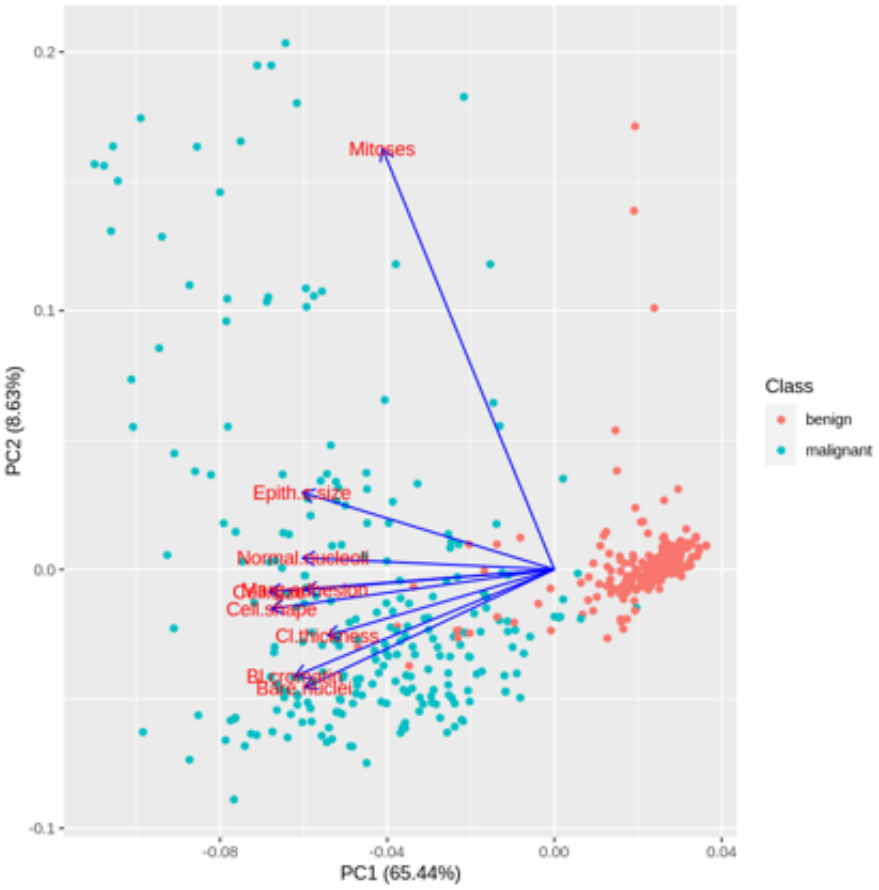
```
# Calculate the proportion of variance explained
pca_df_var <- preproc_pca_df$sdev^2
pve_df <- pca_df_var / sum(pca_df_var)
cum_pve <- cumsum(pve_df)
pve_table <- tibble(comp = seq(1:ncol(df %>% select(-Class))), pve_df, cum_pve

ggplot(pve_table, aes(x = comp, y = cum_pve)) +
  geom_point() +
  geom_abline(intercept = 0.95, color = "red", slope = 0) +
  labs(x = "Number of components", y = "Cumulative Variance")
```




```
pca_df <- as_tibble(preproc_pca_df$x)

ggplot(pca_df, aes(x = PC1, y = PC2, col = df$Class)) + geom_point()
```

```
install.packages("ggfortify")
library(ggfortify)
autoplot(preproc_pca_df, data = df,  colour = 'Class',
                    loadings = FALSE, loadings.label = TRUE, loadings.colour =
```

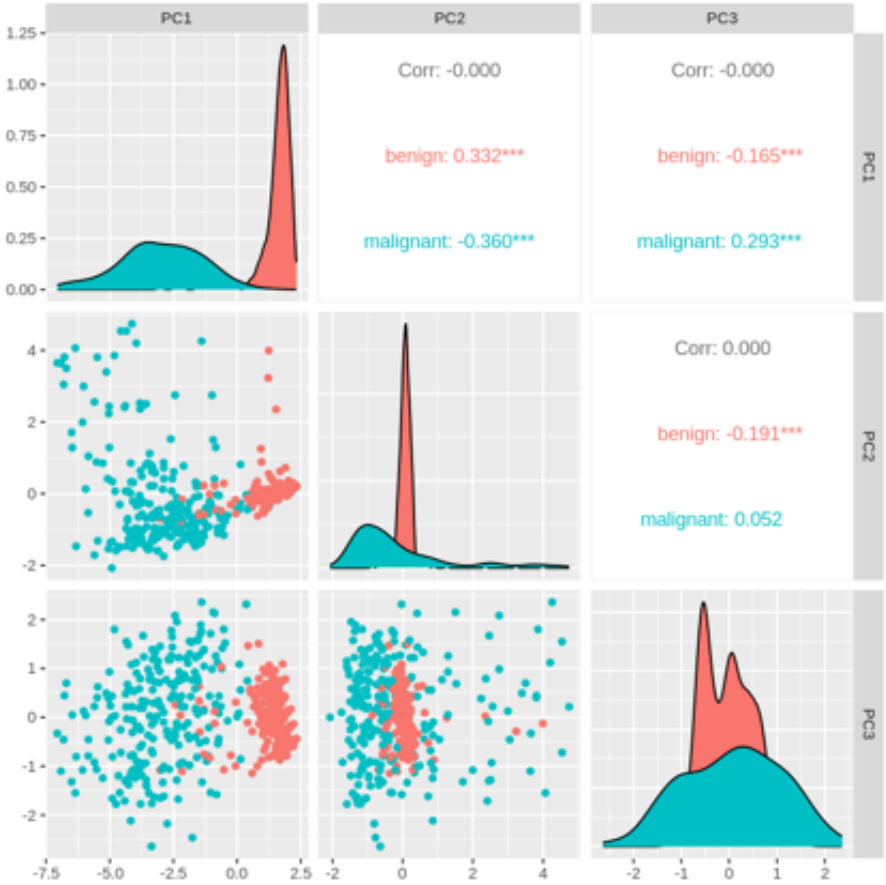
```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependency ‘gridExtra’
```



```
install.packages("GGally")
library(GGally)
```

```
    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)

    also installing the dependencies ‘labelled’, ‘broom.helpers’, ‘patchwork’,


    Registered S3 method overwritten by 'GGally':
      method from
      +.gg   ggplot2
```

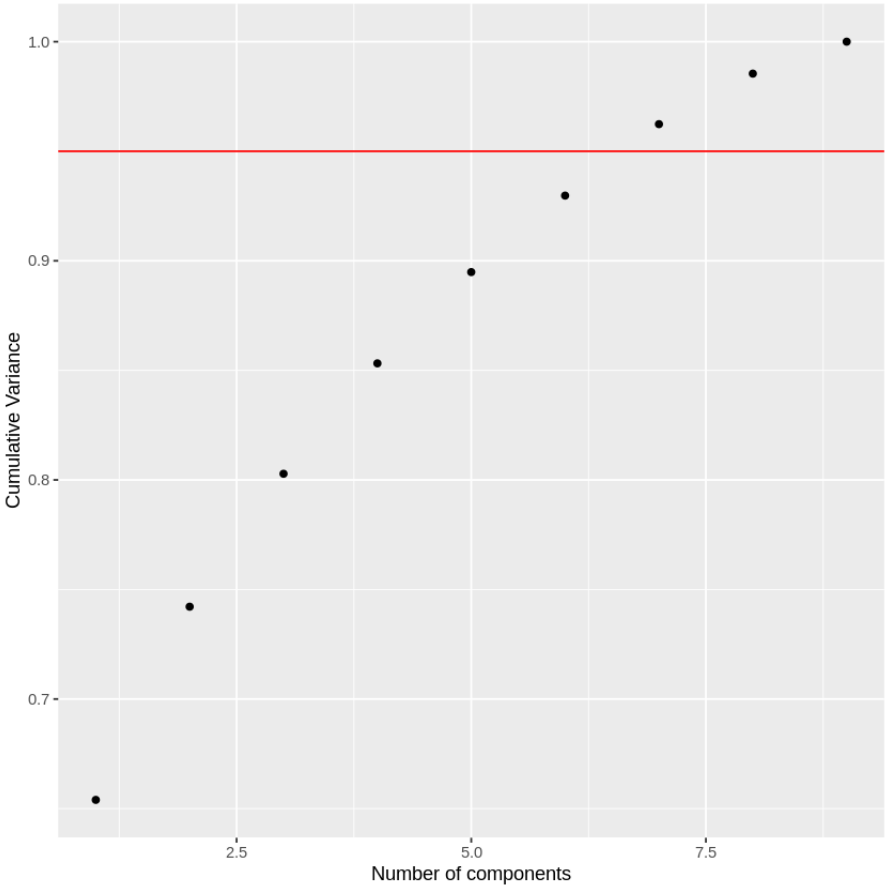
```
df_pcs <- cbind(as_tibble(df$Class), as_tibble(preproc_pca_df$x))
GGally::ggpairs(df_pcs, columns = 2:4, ggplot2::aes(color = value))
```



```
# Check for missing values
if (anyNA(df2)) {
  # Impute missing values with column means
  df2[is.na(df2)] <- colMeans(df2, na.rm = TRUE)
}

# Check data type of each column
if (!all(sapply(df2, is.numeric))) {
  # Convert non-numeric columns to numeric if possible
  df2 <- sapply(df2, as.numeric)
}

# Perform PCA
preproc_pca_df2 <- prcomp(df2, scale = TRUE, center = TRUE)
summary(preproc_pca_df2)
```

```
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6
PC7
Standard deviation     2.426 0.89074 0.73893 0.67341 0.61227 0.5605
0.54179
Proportion of Variance 0.654 0.08816 0.06067 0.05039 0.04165 0.0349
0.03261
Cumulative Proportion  0.654 0.74214 0.80281 0.85319 0.89485 0.9297
0.96236
                           PC8     PC9
Standard deviation     0.45554 0.36222
Proportion of Variance 0.02306 0.01458
```

```
pca_df2_var <- preproc_pca_df2$sdev^2

# proportion of variance explained
pve_df2 <- pca_df2_var / sum(pca_df2_var)
cum_pve_df2 <- cumsum(pve_df2)
pve_table_df2 <- tibble(comp = seq(1:ncol(df2)), pve_df2, cum_pve_df2)

ggplot(pve_table_df2, aes(x = comp, y = cum_pve_df2)) +
  geom_point() +
  geom_abline(intercept = 0.95, color = "red", slope = 0) +
  labs(x = "Number of components", y = "Cumulative Variance")
```



7.MODEL THE DATA

This code trains a predictive model using the preprocessed data to learn patterns and relationships within the dataset, enabling it to make predictions or classifications on new, unseen data.

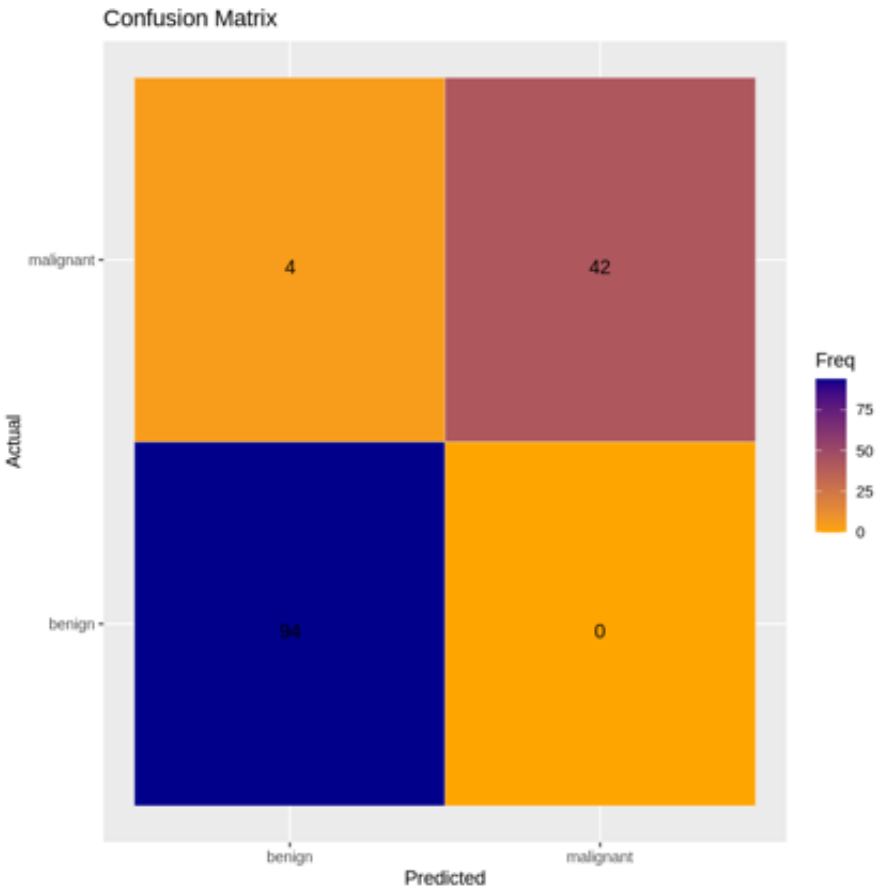
```
library(caret)

# Assuming 'df' contains your data

# Identify categorical variables
categorical_cols <- sapply(df, is.factor)

# Convert categorical variables to factors if they are not already
df[categorical_cols] <- lapply(df[categorical_cols], as.factor)

# Perform one-hot encoding
dummy_df <- dummyVars(~., data = df[categorical_cols])
encoded_df <- predict(dummy_df, newdata = df)

# Combine encoded dataframe with non-categorical variables
final_df <- cbind(df[!categorical_cols], encoded_df)

# Now 'final_df' contains your encoded dataframe ready for modeling



# Split Data
X <- df[, -which(names(df) == "Class")] # Features
y <- df$Class  # Target variable

# Split Data into Training and Testing Sets
set.seed(123) # For reproducibility
train_indices <- sample(1:nrow(df), 0.8 * nrow(df)) # 80% for training
X_train <- X[train_indices, ]
X_test <- X[-train_indices, ]
y_train <- y[train_indices]
y_test <- y[-train_indices]

# Now you can proceed with modeling using X_train, X_test, y_train, and y_test
```

KNN

```
# Load the caret library
library(caret)

# Train the KNN Model
k <- 5  # Choose the number of neighbors (you can tune this parameter)
model <- train(x = X_train, y = y_train, method = "knn", trControl = trainCont

# Predict using the trained model
predictions <- predict(model, newdata = X_test)

# Evaluate the Model
confusion_matrix <- confusionMatrix(data = predictions, reference = y_test)
print(confusion_matrix)
```

```
    Confusion Matrix and Statistics

              Reference
    Prediction  benign malignant
      benign        94         0
      malignant      4        42

                  Accuracy : 0.9714
                    95% CI : (0.9285, 0.9922)
       No Information Rate : 0.7
       P-Value [Acc > NIR] : <2e-16

                     Kappa : 0.9338

     Mcnemar's Test P-Value : 0.1336

                Sensitivity : 0.9592
                Specificity : 1.0000
             Pos Pred Value : 1.0000
             Neg Pred Value : 0.9130
                 Prevalence : 0.7000
             Detection Rate : 0.6714
       Detection Prevalence : 0.6714
          Balanced Accuracy : 0.9796

           'Positive' Class : benign
```

```
library(ggplot2)

# Create a data frame for the confusion matrix
conf_matrix <- matrix(c(94, 4, 0, 42), nrow = 2, byrow = TRUE)
colnames(conf_matrix) <- c("benign", "malignant")
rownames(conf_matrix) <- c("benign", "malignant")
conf_matrix_df <- as.data.frame(as.table(conf_matrix))

# Plot the confusion matrix
ggplot(data = conf_matrix_df, aes(x = Var1, y = Var2, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), vjust = 1) +
  scale_fill_gradient(low = "orange", high = "darkblue") +
  labs(title = "Confusion Matrix", x = "Predicted", y = "Actual")
```



SVM

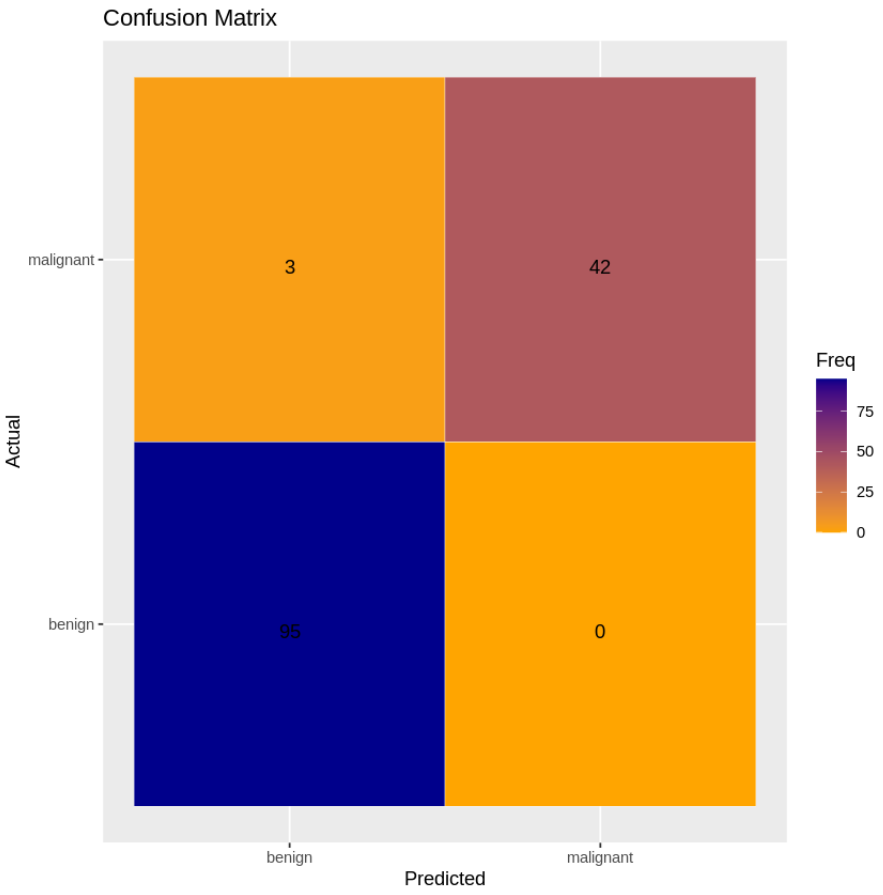
```
# Load the necessary library for SVM
install.packages("e1071")
library(e1071)
```

```
    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)
```

```
# Train the SVM Model
svm_model <- svm(Class ~ ., data = data.frame(X_train, Class = y_train), kerne

# Make predictions
svm_pred <- predict(svm_model, X_test)

# Evaluate the Model
confusion_matrix <- confusionMatrix(data = svm_pred, reference = y_test)
print(confusion_matrix)
```

```
    Confusion Matrix and Statistics

              Reference
    Prediction  benign malignant
      benign        95         0
      malignant      3        42

                   Accuracy : 0.9786
                     95% CI : (0.9387, 0.9956)
       No Information Rate : 0.7
       P-Value [Acc > NIR] : <2e-16

                      Kappa : 0.95

     Mcnemar's Test P-Value : 0.2482

                Sensitivity : 0.9694
                Specificity : 1.0000
             Pos Pred Value : 1.0000
             Neg Pred Value : 0.9333
                 Prevalence : 0.7000
             Detection Rate : 0.6786
       Detection Prevalence : 0.6786
          Balanced Accuracy : 0.9847

           'Positive' Class : benign
```

```
library(ggplot2)

# Confusion matrix data
conf_matrix <- matrix(c(95, 3, 0, 42), nrow = 2, byrow = TRUE)
colnames(conf_matrix) <- c("benign", "malignant")
rownames(conf_matrix) <- c("benign", "malignant")
conf_matrix_df <- as.data.frame(as.table(conf_matrix))

# Plot the confusion matrix
ggplot(data = conf_matrix_df, aes(x = Var1, y = Var2, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), vjust = 1) +
  scale_fill_gradient(low = "orange", high = "darkblue") +
  labs(title = "Confusion Matrix", x = "Predicted", y = "Actual")
```



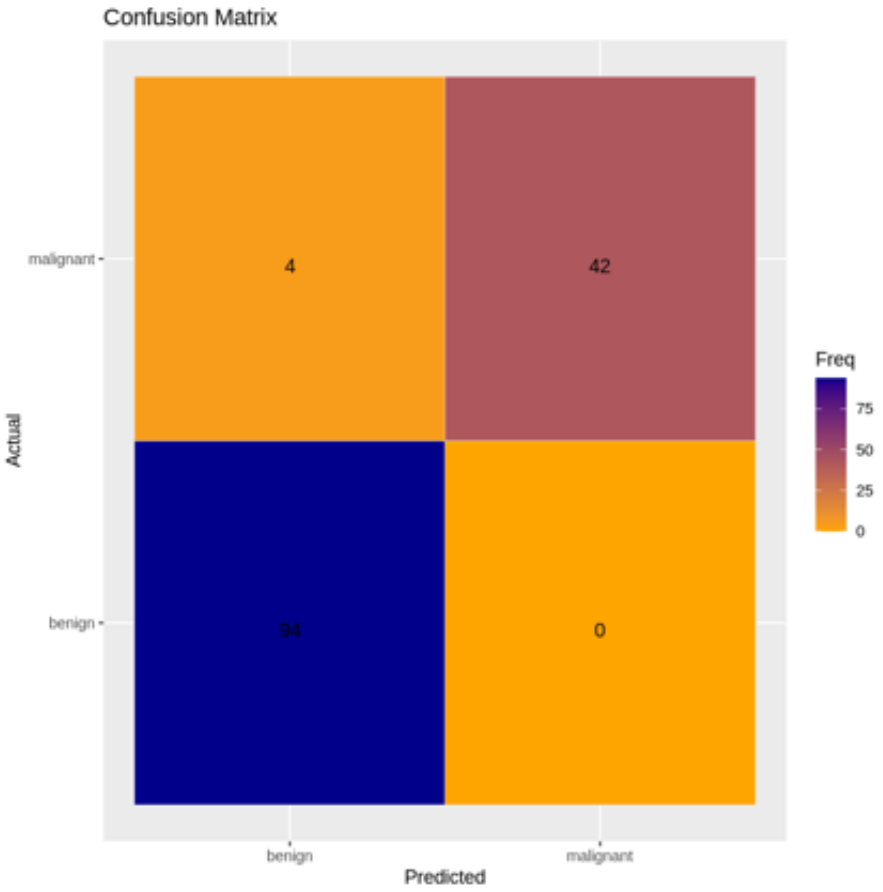
SVM with PCA

```
# Load necessary libraries
library(caret)
library(e1071)
library(MASS)
```

```
# Perform PCA
pca <- prcomp(X_train, scale. = TRUE)
X_train_pca <- predict(pca, X_train)
X_test_pca <- predict(pca, X_test)

# Train and Evaluate SVM Model with PCA
svm_model_pca <- svm(Class ~ ., data = data.frame(X_train_pca, Class = y_train
svm_pred_pca <- predict(svm_model_pca, X_test_pca)

# Evaluate the Model
confusion_matrix <- confusionMatrix(data = svm_pred_pca, reference = y_test)
print(confusion_matrix)
```

```
Attaching package: ‘MASS’


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

    select


Confusion Matrix and Statistics

          Reference
Prediction  benign malignant
  benign        94         0
  malignant      4        42

               Accuracy : 0.9714
                 95% CI : (0.9285, 0.9922)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9338

 Mcnemar's Test P-Value : 0.1336

            Sensitivity : 0.9592
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9130
             Prevalence : 0.7000
         Detection Rate : 0.6714
   Detection Prevalence : 0.6714
      Balanced Accuracy : 0.9796

       'Positive' Class : benign
```

```
library(ggplot2)

# Confusion matrix data
conf_matrix <- matrix(c(94, 4, 0, 42), nrow = 2, byrow = TRUE)
colnames(conf_matrix) <- c("benign", "malignant")
rownames(conf_matrix) <- c("benign", "malignant")
conf_matrix_df <- as.data.frame(as.table(conf_matrix))

# Plot the confusion matrix
ggplot(data = conf_matrix_df, aes(x = Var1, y = Var2, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), vjust = 1) +
  scale_fill_gradient(low = "orange", high = "darkblue") +
  labs(title = "Confusion Matrix", x = "Predicted", y = "Actual")
```




```
## Naive Bayes
```

Naive Bayes

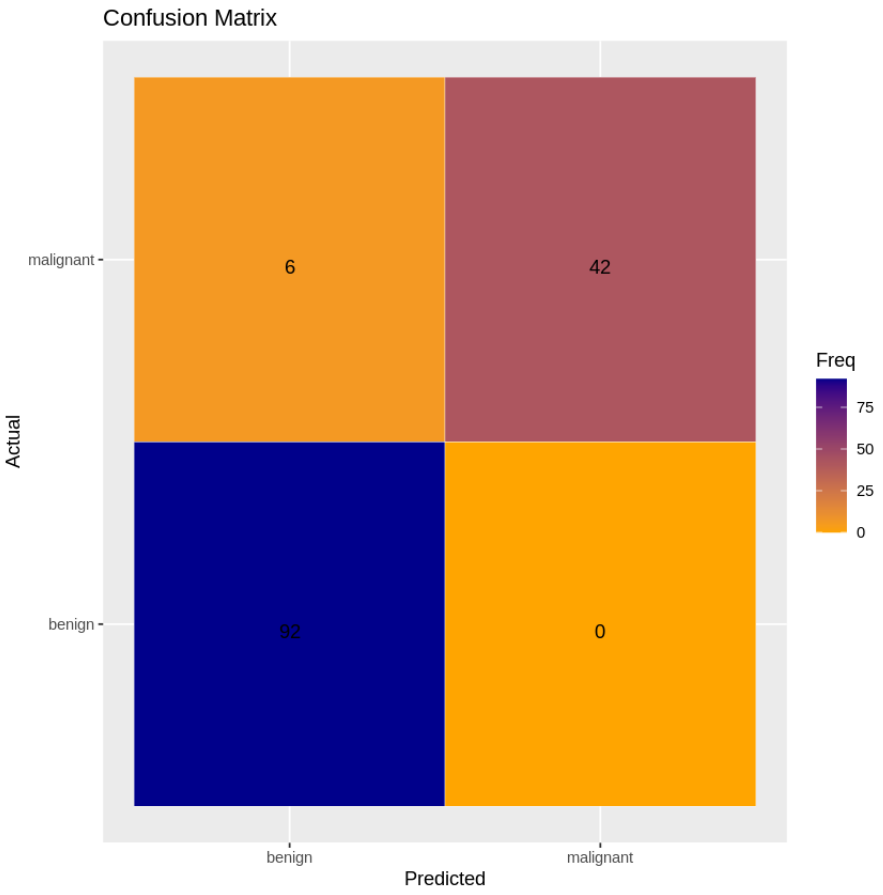
```
install.packages("naivebayes")
```

```
    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)
```

```
# Load the required library for Naive Bayes
library(e1071)

# Train the Naive Bayes Model
nb_model <- naiveBayes(Class ~ ., data = data.frame(X_train, Class = y_train))

# Make predictions
nb_pred <- predict(nb_model, X_test)

# Evaluate the Model
confusion_matrix <- confusionMatrix(data = nb_pred, reference = y_test)
print(confusion_matrix)
```

```
    Confusion Matrix and Statistics

              Reference
    Prediction  benign malignant
      benign        92         0
      malignant      6        42

                   Accuracy : 0.9571
                     95% CI : (0.9091, 0.9841)
        No Information Rate : 0.7
        P-Value [Acc > NIR] : 1.333e-14

                      Kappa : 0.902

     Mcnemar's Test P-Value : 0.04123

                Sensitivity : 0.9388
                Specificity : 1.0000
             Pos Pred Value : 1.0000
             Neg Pred Value : 0.8750
                 Prevalence : 0.7000
             Detection Rate : 0.6571
       Detection Prevalence : 0.6571
          Balanced Accuracy : 0.9694

           'Positive' Class : benign
```

```
library(ggplot2)

# Confusion matrix data
conf_matrix <- matrix(c(92, 6, 0, 42), nrow = 2, byrow = TRUE)
colnames(conf_matrix) <- c("benign", "malignant")
rownames(conf_matrix) <- c("benign", "malignant")
conf_matrix_df <- as.data.frame(as.table(conf_matrix))

# Plot the confusion matrix
ggplot(data = conf_matrix_df, aes(x = Var1, y = Var2, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), vjust = 1) +
  scale_fill_gradient(low = "orange", high = "darkblue") +
  labs(title = "Confusion Matrix", x = "Predicted", y = "Actual")
```



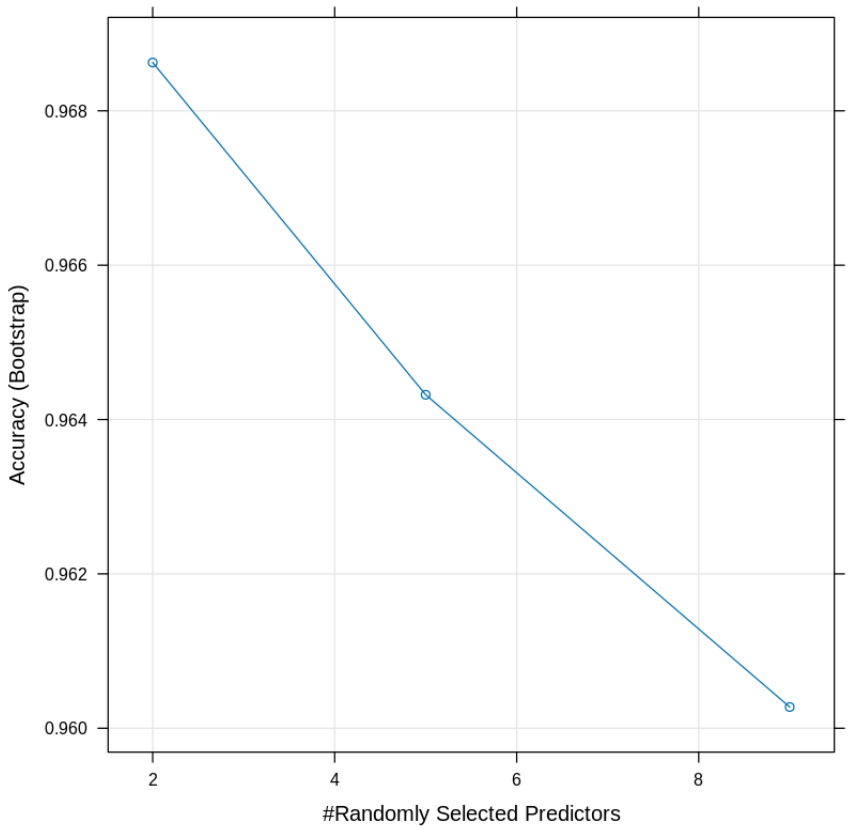
Random Forest

```
install.packages("randomForest")
# Load the required library for Random Forest
library(randomForest)

# Train the Random Forest Model
rf_model <- randomForest(Class ~ ., data = data.frame(X_train, Class = y_train

# Make predictions
rf_pred <- predict(rf_model, X_test)

# Evaluate the Model
confusion_matrix <- confusionMatrix(data = rf_pred, reference = y_test)
print(confusion_matrix)
```

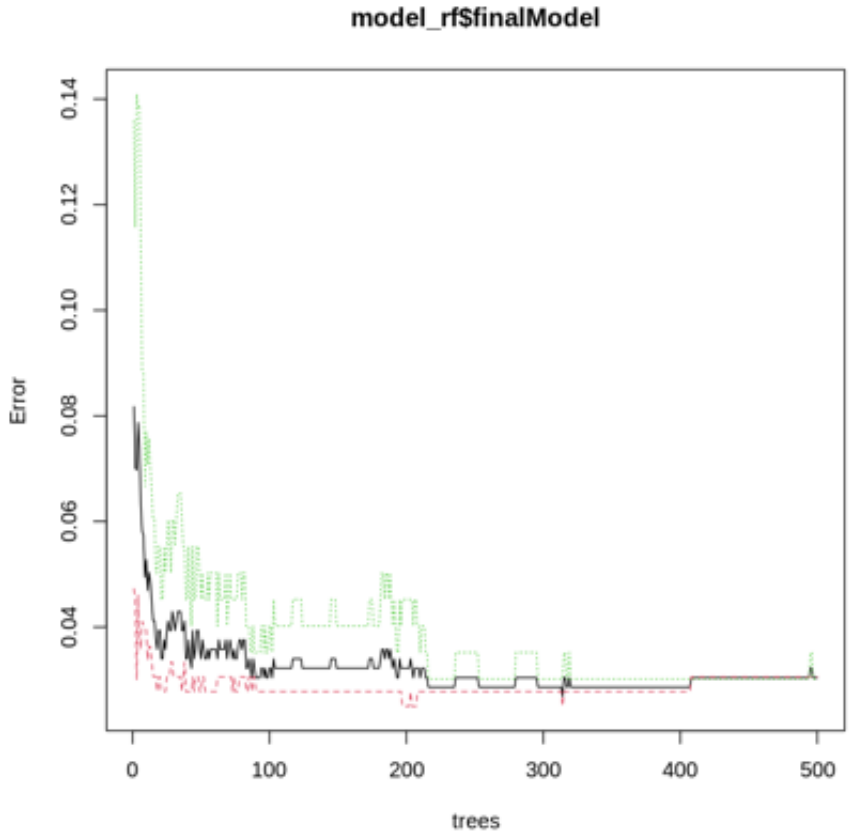
```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Confusion Matrix and Statistics

          Reference
Prediction  benign malignant
  benign        95         0
  malignant      3        42

               Accuracy : 0.9786
                 95% CI : (0.9387, 0.9956)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.95

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9694
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9333
             Prevalence : 0.7000
         Detection Rate : 0.6786
   Detection Prevalence : 0.6786
      Balanced Accuracy : 0.9847

       'Positive' Class : benign
```

```
# Load the necessary libraries
library(caret)
library(randomForest)
```

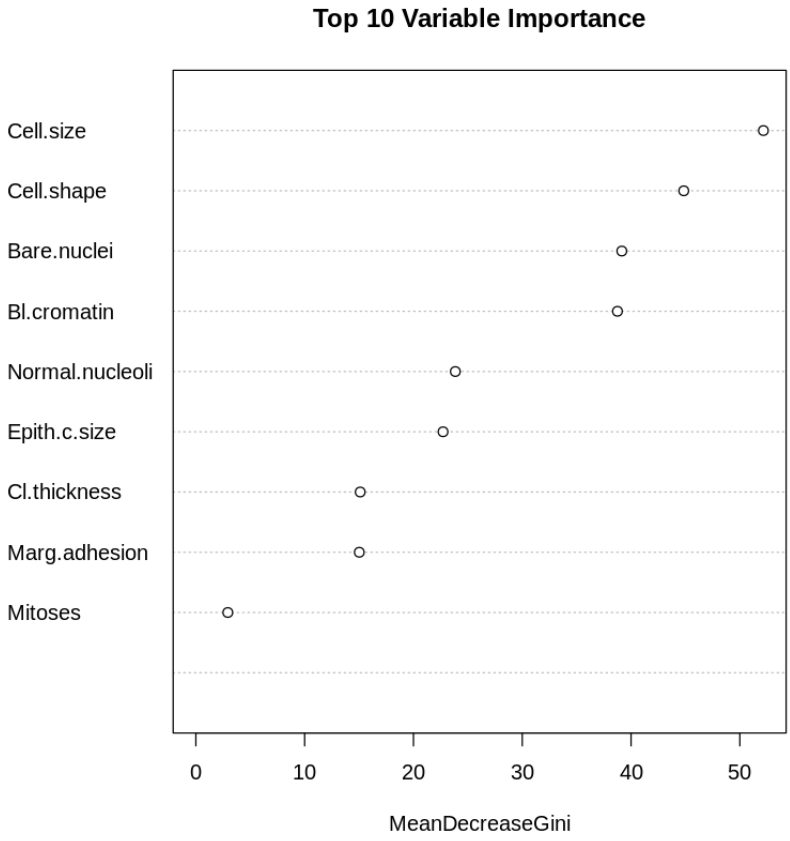
```
plot(model_rf)
```



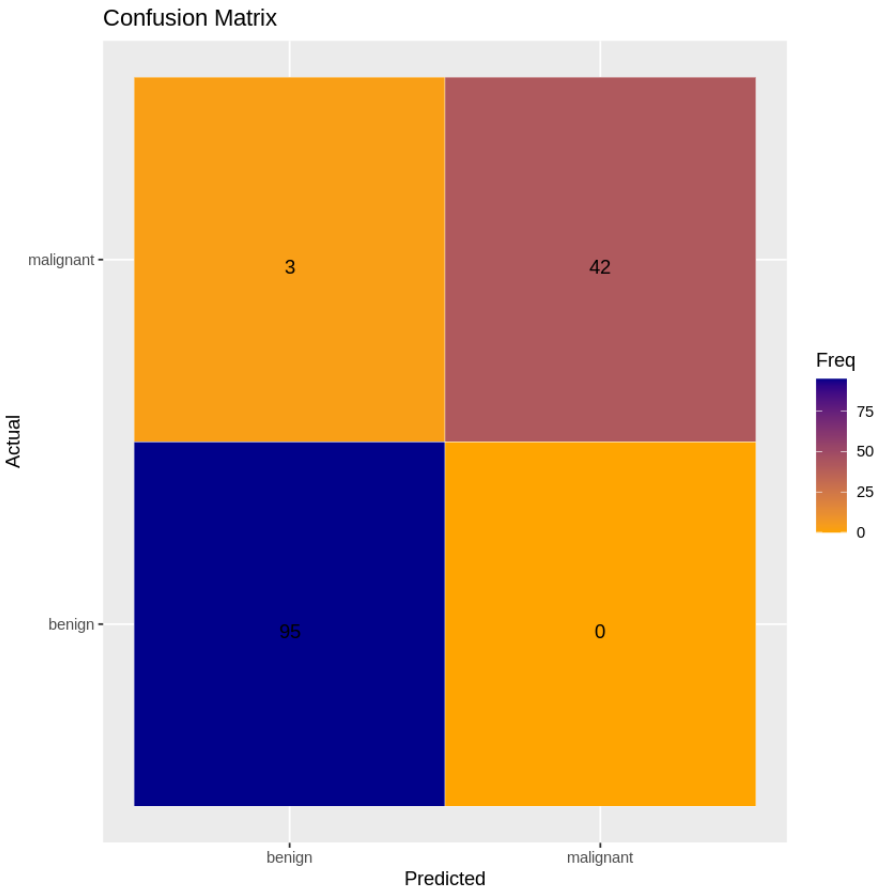
```
plot(model_rf$finalModel)
```



```
# Train the Random Forest model
model_rf <- train(x = X_train, y = y_train, method = "rf")

# Get variable importance scores
var_importance <- varImp(model_rf$finalModel)

# Plot the variable importance
varImpPlot(model_rf$finalModel, sort = TRUE, n.var = 10, main = "Top 10 Variab
```



```
library(ggplot2)

# Confusion matrix data
conf_matrix <- matrix(c(95, 3, 0, 42), nrow = 2, byrow = TRUE)
colnames(conf_matrix) <- c("benign", "malignant")
rownames(conf_matrix) <- c("benign", "malignant")
conf_matrix_df <- as.data.frame(as.table(conf_matrix))

# Plot the confusion matrix
ggplot(data = conf_matrix_df, aes(x = Var1, y = Var2, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), vjust = 1) +
  scale_fill_gradient(low = "orange", high = "darkblue") +
  labs(title = "Confusion Matrix", x = "Predicted", y = "Actual")
```



8.MODEL EVALUATION

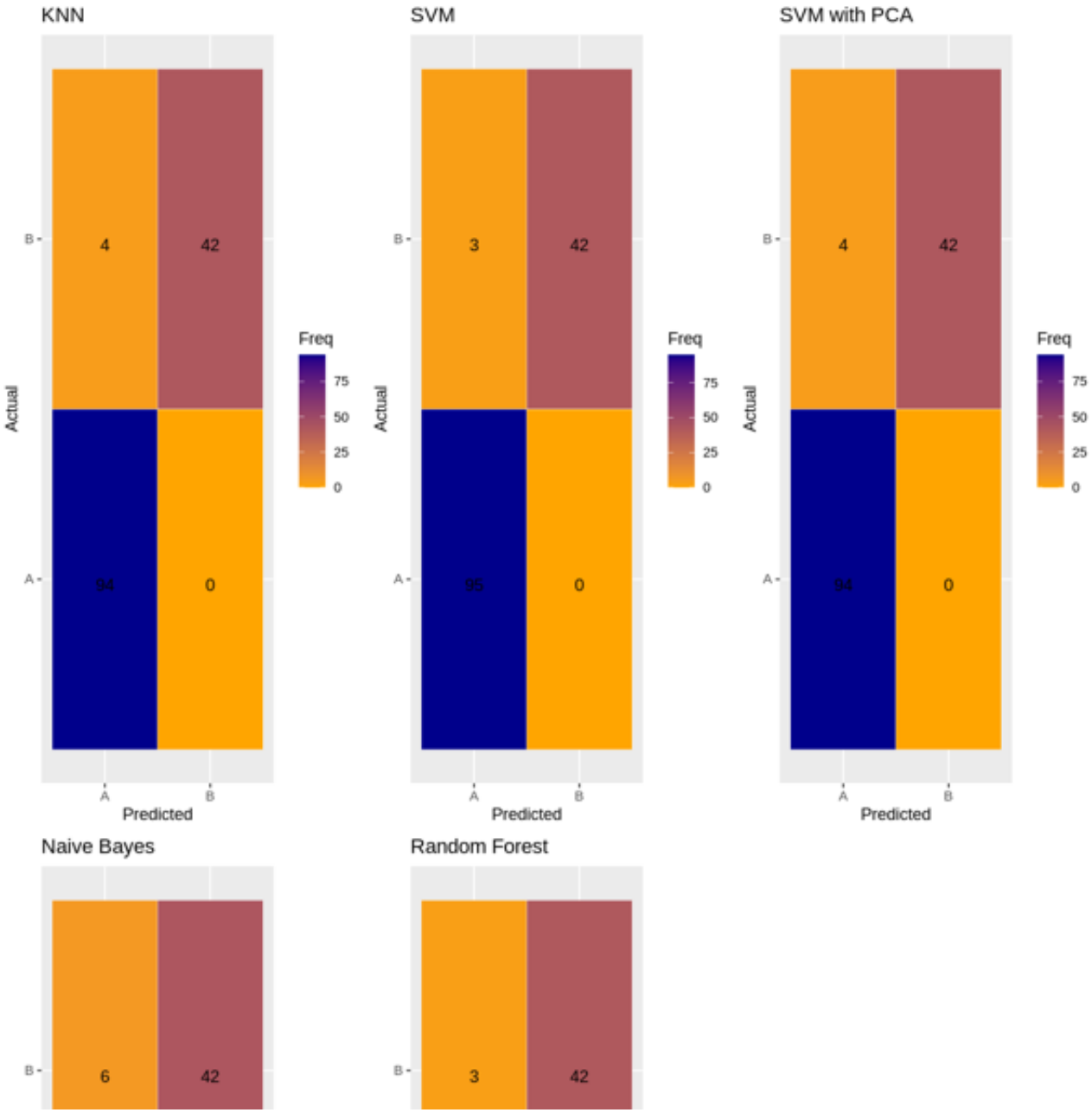
This code assesses the performance and effectiveness of a trained model by evaluating its predictions against known outcomes or ground truth data using various metrics such as accuracy, precision, recall, or Test score.

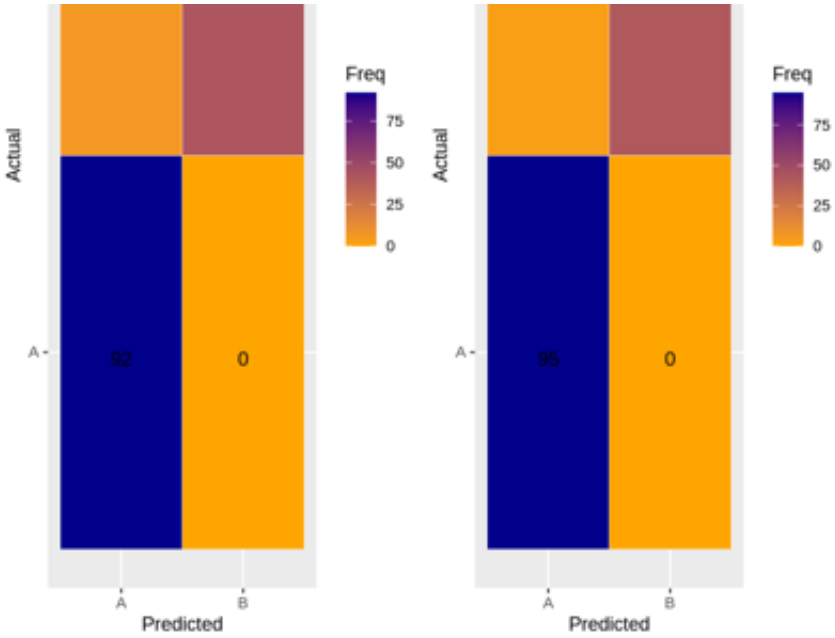
CONFUSION MATRIX

```
install.packages("gridExtra")
library(gridExtra)
library(ggplot2)

# Function to plot confusion matrix
plot_confusion_matrix <- function(conf_matrix, title) {
  conf_matrix_df <- as.data.frame(as.table(conf_matrix))
  ggplot(data = conf_matrix_df, aes(x = Var1, y = Var2, fill = Freq)) +
    geom_tile(color = "white") +
    geom_text(aes(label = Freq), vjust = 1) +
    scale_fill_gradient(low = "orange", high = "darkblue") +
    labs(title = title, x = "Predicted", y = "Actual")
}

# Confusion matrix data
conf_matrix_knn <- matrix(c(94, 4, 0, 42), nrow = 2, byrow = TRUE)
conf_matrix_svm <- matrix(c(95, 3, 0, 42), nrow = 2, byrow = TRUE)
conf_matrix_svm_pca <- matrix(c(94, 4, 0, 42), nrow = 2, byrow = TRUE)
conf_matrix_nb <- matrix(c(92, 6, 0, 42), nrow = 2, byrow = TRUE)
conf_matrix_rf <- matrix(c(95, 3, 0, 42), nrow = 2, byrow = TRUE)

# Create plots for each confusion matrix
plot_knn <- plot_confusion_matrix(conf_matrix_knn, "KNN")
plot_svm <- plot_confusion_matrix(conf_matrix_svm, "SVM")
plot_svm_pca <- plot_confusion_matrix(conf_matrix_svm_pca, "SVM with PCA")
plot_nb <- plot_confusion_matrix(conf_matrix_nb, "Naive Bayes")
plot_rf <- plot_confusion_matrix(conf_matrix_rf, "Random Forest")

# Combine plots
library(gridExtra)
combined_plot <- grid.arrange(plot_knn, plot_svm, plot_svm_pca, plot_nb, plot_rf

# Set the size of the plot
ggsave("confusion_matrix_combined.png", combined_plot, width = 80, height = 30)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
Error in `ggsave()`:
! Dimensions exceed 50 inches (`height` and `width` are specified in
  inches not pixels).
i If you're sure you want a plot that big, use `limitsize = FALSE`.
Traceback:

1. ggsave("confusion_matrix_combined.png", combined_plot, width = 80,
 .     height = 30)
2. plot_dim(c(width, height), scale = scale, units = units, limitsize =
limitsize,
 .     dpi = dpi)
3. cli::cli_abort(c(msg, i = "If you're sure you want a plot that big, use
{.code limitsize = FALSE}.\n    "),
 .     call = call)
4. rlang::abort(message, ..., call = call, use_cli_format = TRUE,
 .     .frame = .frame)
5. signal_abort(cnd, .file)
```





TEST SCORE

```
install.packages("e1071")
install.packages("randomForest")
install.packages("MASS")
install.packages("class")
```

```
    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)

    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)

    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)

    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)
```

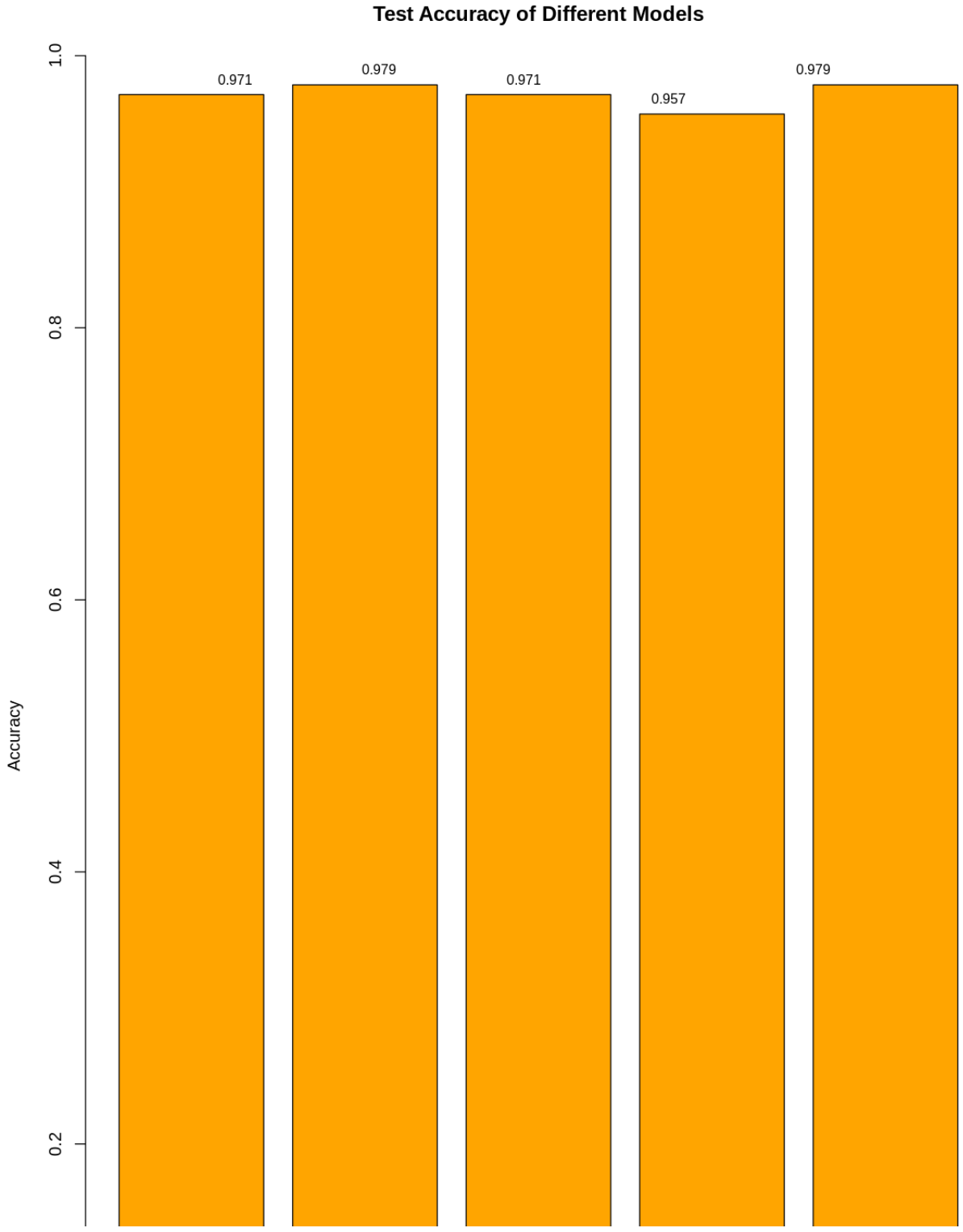
```
# Load necessary libraries
library(e1071)
library(randomForest)
library(MASS)
library(class)

# Perform PCA
```

```
# Perform PCA
pca <- prcomp(X_train, scale. = TRUE)
X_train_pca <- predict(pca, X_train)
X_test_pca <- predict(pca, X_test)

# Train and Evaluate KNN Model
k <- 5  # Choose the number of neighbors
knn_model <- knn(train = X_train, test = X_test, cl = y_train, k = k)
confusion_matrix_knn <- confusionMatrix(data = knn_model, reference = y_test)

# Train and Evaluate SVM Model
svm_model <- svm(Class ~ ., data = data.frame(X_train, Class = y_train))
svm_pred <- predict(svm_model, X_test)
confusion_matrix_svm <- confusionMatrix(data = svm_pred, reference = y_test)

# Train and Evaluate SVM Model with PCA
svm_model_pca <- svm(Class ~ ., data = data.frame(X_train_pca, Class = y_train))
svm_pred_pca <- predict(svm_model_pca, X_test_pca)
confusion_matrix_svm_pca <- confusionMatrix(data = svm_pred_pca, reference = y_t

# Train and Evaluate Naive Bayes Model
nb_model <- naiveBayes(Class ~ ., data = data.frame(X_train, Class = y_train))
nb_pred <- predict(nb_model, X_test)
confusion_matrix_nb <- confusionMatrix(data = nb_pred, reference = y_test)

# Train and Evaluate Random Forest Model
rf_model <- randomForest(x = X_train, y = y_train)
rf_pred <- predict(rf_model, X_test)
confusion_matrix_rf <- confusionMatrix(data = rf_pred, reference = y_test)

# Combine test scores and sort in descending order
test_scores <- c(
  KNN = confusion_matrix_knn$overall["Accuracy"],
  SVM = confusion_matrix_svm$overall["Accuracy"],
  `SVM with PCA` = confusion_matrix_svm_pca$overall["Accuracy"],
  `Naive Bayes` = confusion_matrix_nb$overall["Accuracy"],
  `Random Forest` = confusion_matrix_rf$overall["Accuracy"]
)

# Print test scores in descending order
sorted_test_scores <- sort(test_scores, decreasing = TRUE)
print(sorted_test_scores)
```

```
             SVM.Accuracy Random Forest.Accuracy           KNN.Accuracy
                0.9785714              0.9785714              0.9714286
SVM with PCA.Accuracy   Naive Bayes.Accuracy
                0.9714286              0.9571429
```

```
# Load necessary libraries
library(caret)
library(e1071)
library(randomForest)
library(MASS)

# Perform PCA
pca <- prcomp(X_train, scale. = TRUE)
X_train_pca <- predict(pca, X_train)
X_test_pca <- predict(pca, X_test)

# Train and Evaluate KNN Model
k <- 5  # Choose the number of neighbors
knn_model <- knn(train = X_train, test = X_test, cl = y_train, k = k)
accuracy_knn <- confusionMatrix(data = knn_model, reference = y_test)$overall[

# Train and Evaluate SVM Model
svm_model <- svm(Class ~ ., data = data.frame(X_train, Class = y_train))
svm_pred <- predict(svm_model, X_test)
accuracy_svm <- confusionMatrix(data = svm_pred, reference = y_test)$overall["

# Train and Evaluate SVM Model with PCA
svm_model_pca <- svm(Class ~ ., data = data.frame(X_train_pca, Class = y_train
svm_pred_pca <- predict(svm_model_pca, X_test_pca)
accuracy_svm_pca <- confusionMatrix(data = svm_pred_pca, reference = y_test)$o

# Train and Evaluate Naive Bayes Model
nb_model <- naiveBayes(Class ~ ., data = data.frame(X_train, Class = y_train))
nb_pred <- predict(nb_model, X_test)
accuracy_nb <- confusionMatrix(data = nb_pred, reference = y_test)$overall["Ac

# Train and Evaluate Random Forest Model
rf_model <- randomForest(x = X_train, y = y_train)
rf_pred <- predict(rf_model, X_test)
accuracy_rf <- confusionMatrix(data = rf_pred, reference = y_test)$overall["Ac

# Create a larger plot
png("accuracy_bar_chart1.png", width = 1000, height = 1000)

# Create a bar chart
models <- c("KNN", "SVM", "SVM with PCA", "Naive Bayes", "Random Forest")
accuracies <- c(accuracy_knn, accuracy_svm, accuracy_svm_pca, accuracy_nb, acc
barplot(accuracies, names.arg = models, main = "Test Accuracy of Different Mod
text(x = 1:length(models), y = accuracies, labels = round(accuracies, digits =
```

```
# Create a larger plot
options(repr.plot.width = 10, repr.plot.height = 15)

# Create a bar chart
models <- c("KNN", "SVM", "SVM with PCA", "Naive Bayes", "Random Forest")
accuracies <- c(accuracy_knn, accuracy_svm, accuracy_svm_pca, accuracy_nb, acc
barplot(accuracies, names.arg = models, main = "Test Accuracy of Different Mod
text(x = 1:length(models), y = accuracies, labels = round(accuracies, digits =
```



F1 SCORE

```
install.packages(c("caret", "data.table", "dplyr"))
library(caret)
library(data.table)
library(dplyr)
```

```
Installing packages into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)


Attaching package: ‘data.table’


The following objects are masked from ‘package:lubridate’:

    hour, isoweek, mday, minute, month, quarter, second, wday, week,
    yday, year


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

    transpose


The following objects are masked from ‘package:dplyr’:

    between, first, last


The following objects are masked from ‘package:reshape2’:

    dcast, melt
```

```
# Compute performance metrics for each model
metrics <- list(
  RF = metrics_rf,
  SVM = metrics_svm,
  `SVM with PCA` = metrics_svm_pca,
  `Naive Bayes` = metrics_nb,
  KNN = metrics_knn
)

# Create a data frame to store the metrics
metrics_df <- data.frame(
  metric = rownames(metrics[[1]]),
  stringsAsFactors = FALSE
)

# Populate the data frame with metric values for each model
for (model_name in names(metrics)) {
  metrics_df[[model_name]] <- unlist(metrics[[model_name]])
}

# Print the metrics data frame
print(metrics_df)
```

```
                  metric        RF       SVM SVM with PCA Naive Bayes       
    1           Sensitivity 0.9693878 0.9693878    0.9591837   0.9387755 0.959
    2           Specificity 1.0000000 1.0000000    1.0000000   1.0000000 1.000
    3        Pos Pred Value 1.0000000 1.0000000    1.0000000   1.0000000 1.000
    4        Neg Pred Value 0.9333333 0.9333333    0.9130435   0.8750000 0.913
    5             Precision 1.0000000 1.0000000    1.0000000   1.0000000 1.000
    6                Recall 0.9693878 0.9693878    0.9591837   0.9387755 0.959
    7                    F1 0.9844560 0.9844560    0.9791667   0.9684211 0.979
    8            Prevalence 0.7000000 0.7000000    0.7000000   0.7000000 0.700
    9        Detection Rate 0.6785714 0.6785714    0.6714286   0.6571429 0.671
    10 Detection Prevalence 0.6785714 0.6785714    0.6714286   0.6571429 0.671
    11    Balanced Accuracy 0.9846939 0.9846939    0.9795918   0.9693878 0.979
```

9.INSIGHTS

TEST SCORE

1. **Model Performance**:
    - The SVM model and the Random Forest model perform the best, both achieving an accuracy of 97.86%.
    - The KNN model follows closely with an accuracy of 97.14%.
    - The SVM with PCA and Naive Bayes models have slightly lower accuracies of 97.14% and 95.71% respectively.
2. **Insights**:
    - All models perform quite well with high accuracy scores ranging from 95.71% to 97.86%.
    - The high sensitivity and specificity values for all models indicate good performance in correctly identifying both benign and malignant cases.
    - There is a slight imbalance in the dataset (Prevalence: 70%), which could potentially affect model performance, but overall, the models seem to handle this imbalance well.
    - The Kappa statistics for all models indicate substantial to almost perfect agreement between predicted and actual classifications.
    - The balanced accuracy scores, which take into account imbalanced datasets, are also quite high for all models, indicating their robustness in handling class imbalance.
    - The models' positive predictive values (Pos Pred Value) are all 1.0000 for the benign class, indicating a high proportion of correctly predicted benign cases among the total predicted benign cases.

In summary, the SVM and Random Forest models are the top performers, followed closely by KNN. The insights suggest that all models perform well in distinguishing between benign and malignant cases, with high accuracy and robustness in handling imbalanced data.

F1 SCORE

The F1 score represents the harmonic mean of precision and recall, and the model with the highest F1 score is typically considered the best performer in terms of overall classification performance.

From the provided metrics:

- **RF (Random Forest)**: F1 Score = 0.9844560
- **SVM (Support Vector Machine)**: F1 Score = 0.9844560
- **SVM with PCA (Support Vector Machine with Principal Component Analysis)**: F1 Score = 0.9791667
- **Naive Bayes**: F1 Score = 0.9684211
- **KNN (K-Nearest Neighbors)**: F1 Score = 0.9791667

Both Random Forest (RF) and Support Vector Machine (SVM) have the highest F1 score of 0.9844560, indicating that these two models achieve the best balance between precision and recall. Therefore, based on the F1 score, either the RF model or the SVM model can be considered the top-performing model.

SUMMARY

The high performance of these machine learning models in accurately predicting breast cancer cases from diagnostic features provides several important messages to breast cancer diagnosis and treatment:

1. **Early Detection**: The models' high accuracy indicates that they can effectively identify patterns in diagnostic features that differentiate between benign and malignant tumors. Early detection of breast cancer is crucial for successful treatment and improved outcomes.
2. **Precision Medicine**: Machine learning models allow for the analysis of multiple diagnostic features simultaneously, providing personalized predictions based on individual patient data. This enables precision medicine approaches tailored to each patient's unique characteristics, leading to more effective treatment strategies.
3. **Reduced Misdiagnosis**: High sensitivity and specificity values of the models indicate a low rate of misdiagnosis, reducing the likelihood of false positives (incorrectly diagnosing a benign tumor as malignant) and false negatives (incorrectly diagnosing a malignant tumor as benign). This helps in avoiding unnecessary treatments or missed opportunities for early intervention.
4. **Enhanced Decision Support**: Clinicians can use these models as decision support tools to complement their expertise in diagnosing breast cancer. By incorporating machine learning predictions into the diagnostic process, clinicians can make more informed decisions, leading to improved patient care and outcomes.
5. **Improved Screening Programs**: Integrating machine learning models into breast cancer screening programs can enhance their effectiveness by accurately identifying individuals at higher risk of developing breast cancer. This can lead to targeted screening strategies and more efficient allocation of healthcare resources.

Overall, the high performance of machine learning models in breast cancer diagnosis underscores their potential to improve patient outcomes, enhance clinical decision-making, and contribute to the advancement of breast cancer detection and treatment strategies.

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.