```r
Data <- read.csv("C:\\Users\\bensh\\Downloads\\data (1).csv")
# Rename the data frame to "Breast Cancer Data Set"
colnames(Data) <- "Breast Cancer Data Set"                 # renaming data se
t

# Read the data and rename the data frame
`Breastcancerdata` <- read.csv("C:\\Users\\bensh\\Downloads\\data (1).csv")
#reading dataset

library(janitor)
```

```
## Warning: package 'janitor' was built under R version 4.3.3
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```r
Breastcancer_data2 <- clean_names(Breastcancerdata)

Breastcancer_data2 <- remove_empty(Breastcancer_data2, which = c("rows", "col
s"), quiet = FALSE) # removing duplicate columns
```

```
## No empty rows to remove.
```

```
## Removing 1 empty columns of 33 columns total (Removed: x).
```

```r
# Load the dplyr package
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## 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
```

```r
# Assuming your dataset is named "data3", remove duplicate rows
Breastcancer_data2<- distinct(Breastcancer_data2)

# Get the dimensions (shape) of the dataset
dim(Breastcancer_data2)
```

```
## [1] 569  32
```

```r
# Get a summary description of the dataset
summary(Breastcancer_data2)
```

```
##        id             diagnosis          radius_mean      texture_mean
##  Min.   :     8670   Length:569         Min.   : 6.981   Min.   : 9.71
##  1st Qu.:   869218   Class :character   1st Qu.:11.700   1st Qu.:16.17
##  Median :   906024   Mode  :character   Median :13.370   Median :18.84
##  Mean   : 30371831                      Mean   :14.127   Mean   :19.29
##  3rd Qu.:  8813129                      3rd Qu.:15.780   3rd Qu.:21.80
##  Max.   :911320502                      Max.   :28.110   Max.   :39.28
##  perimeter_mean     area_mean      smoothness_mean   compactness_mean
##  Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.01938
##  1st Qu.: 75.17   1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492
##  Median : 86.24   Median : 551.1   Median :0.09587   Median :0.09263
##  Mean   : 91.97   Mean   : 654.9   Mean   :0.09636   Mean   :0.10434
##  3rd Qu.:104.10   3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040
##  Max.   :188.50   Max.   :2501.0   Max.   :0.16340   Max.   :0.34540
##  concavity_mean    concave_points_mean symmetry_mean    fractal_dimension_
mean
##  Min.   :0.00000   Min.   :0.00000     Min.   :0.1060   Min.   :0.04996
##  1st Qu.:0.02956   1st Qu.:0.02031     1st Qu.:0.1619   1st Qu.:0.05770
##  Median :0.06154   Median :0.03350     Median :0.1792   Median :0.06154
##  Mean   :0.08880   Mean   :0.04892     Mean   :0.1812   Mean   :0.06280
##  3rd Qu.:0.13070   3rd Qu.:0.07400     3rd Qu.:0.1957   3rd Qu.:0.06612
##  Max.   :0.42680   Max.   :0.20120     Max.   :0.3040   Max.   :0.09744
##    radius_se        texture_se      perimeter_se       area_se
##  Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  6.802
##  1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850
##  Median :0.3242   Median :1.1080   Median : 2.287   Median : 24.530
##  Mean   :0.4052   Mean   :1.2169   Mean   : 2.866   Mean   : 40.337
##  3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190
##  Max.   :2.8730   Max.   :4.8850   Max.   :21.980   Max.   :542.200
##  smoothness_se      compactness_se      concavity_se     concave_points_se
##  Min.   :0.001713   Min.   :0.002252   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638
##  Median :0.006380   Median :0.020450   Median :0.02589   Median :0.010930
##  Mean   :0.007041   Mean   :0.025478   Mean   :0.03189   Mean   :0.011796
##  3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710
##  Max.   :0.031130   Max.   :0.135400   Max.   :0.39600   Max.   :0.052790
##   symmetry_se       fractal_dimension_se  radius_worst   texture_worst
##  Min.   :0.007882   Min.   :0.0008948    Min.   : 7.93   Min.   :12.02
##  1st Qu.:0.015160   1st Qu.:0.0022480    1st Qu.:13.01   1st Qu.:21.08
##  Median :0.018730   Median :0.0031870    Median :14.97   Median :25.41
##  Mean   :0.020542   Mean   :0.0037949    Mean   :16.27   Mean   :25.68
##  3rd Qu.:0.023480   3rd Qu.:0.0045580    3rd Qu.:18.79   3rd Qu.:29.72
##  Max.   :0.078950   Max.   :0.0298400    Max.   :36.04   Max.   :49.54
##  perimeter_worst    area_worst     smoothness_worst  compactness_worst
##  Min.   : 50.41   Min.   : 185.2   Min.   :0.07117   Min.   :0.02729
##  1st Qu.: 84.11   1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720
##  Median : 97.66   Median : 686.5   Median :0.13130   Median :0.21190
```

```
##  Mean   :107.26   Mean   : 880.6   Mean   :0.13237   Mean   :0.25427
##  3rd Qu.:125.40   3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910
##  Max.   :251.20   Max.   :4254.0   Max.   :0.22260   Max.   :1.05800
##  concavity_worst  concave_points_worst symmetry_worst   fractal_dimension_
worst
##  Min.   :0.0000   Min.   :0.00000      Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.1145   1st Qu.:0.06493      1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2267   Median :0.09993      Median :0.2822   Median :0.08004
##  Mean   :0.2722   Mean   :0.11461      Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3829   3rd Qu.:0.16140      3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :1.2520   Max.   :0.29100      Max.   :0.6638   Max.   :0.20750
```

```
# Columns to be dropped
columns_to_drop <- c("concavity_mean", "concave_points_mean", "radius_se", "t
exture_se", "perimeter_se", "area_se",
                     "smoothness_se", "compactness_se", "concavity_se",
                     "concave_points_se", "symmetry_se", "fractal_dimension_s
e",
                     "radius_worst", "texture_worst", "perimeter_worst",
                     "area_worst", "smoothness_worst", "compactness_worst",
                     "concavity_worst", "concave_points_worst", "symmetry_wor
st",
                     "fractal_dimension_worst")

# Drop the specified columns from the Breast_cancer dataset
Breastcancer_data2 <- Breastcancer_data2[, !names(Breastcancer_data2) %in% co
lumns_to_drop]

# Normality test for radius_mean variable using Shapiro-Wilk test
print("Normality test for radius_mean:")
```

```
## [1] "Normality test for radius_mean:"
```

```
shapiro_test_radius <- shapiro.test(Breastcancer_data2$radius_mean)
print(paste("Shapiro-Wilk test p-value:", shapiro_test_radius$p.value))
```

```
## [1] "Shapiro-Wilk test p-value: 3.10564350835627e-14"
```

```
# Check if data is normal based on Shapiro-Wilk test
if (shapiro_test_radius$p.value > 0.05) {
  print("Data is normally distributed for radius_mean")
} else {
  print("Data is not normally distributed for radius_mean")
}
```

```
## [1] "Data is not normally distributed for radius_mean"
```

```
# Normality test for texture_mean variable using Shapiro-Wilk test
print("Normality test for texture_mean:")
```

```
## [1] "Normality test for texture_mean:"
```

```r
shapiro_test_texture <- shapiro.test(Breastcancer_data2$texture_mean)
print(paste("Shapiro-Wilk test p-value:", shapiro_test_texture$p.value))
```

```
## [1] "Shapiro-Wilk test p-value: 7.2835810327422e-08"
```

```r
# Check if data is normal based on Shapiro-Wilk test
if (shapiro_test_texture$p.value > 0.05) {
  print("Data is normally distributed for texture_mean")
} else {
  print("Data is not normally distributed for texture_mean")
}
```

```
## [1] "Data is not normally distributed for texture_mean"
```

```r
# Normality test for perimeter_mean variable using Shapiro-Wilk test
print("Normality test for perimeter_mean:")
```

```
## [1] "Normality test for perimeter_mean:"
```

```r
shapiro_test_perimeter <- shapiro.test(Breastcancer_data2$perimeter_mean)
print(paste("Shapiro-Wilk test p-value:", shapiro_test_perimeter$p.value))
```

```
## [1] "Shapiro-Wilk test p-value: 7.01140152099188e-15"
```

```r
# Check if data is normal based on Shapiro-Wilk test
if (shapiro_test_perimeter$p.value > 0.05) {
  print("Data is normally distributed for perimeter_mean")
} else {
  print("Data is not normally distributed for perimeter_mean")
}
```

```
## [1] "Data is not normally distributed for perimeter_mean"
```

```r
# Normality test for area_mean variable using Shapiro-Wilk test
print("Normality test for area_mean:")
```

```
## [1] "Normality test for area_mean:"
```

```r
shapiro_test_area <- shapiro.test(Breastcancer_data2$area_mean)
print(paste("Shapiro-Wilk test p-value:", shapiro_test_area$p.value))
```

```
## [1] "Shapiro-Wilk test p-value: 3.19626432303972e-22"
```

```r
# Check if data is normal based on Shapiro-Wilk test
if (shapiro_test_area$p.value > 0.05) {
  print("Data is normally distributed for area_mean")
} else {
  print("Data is not normally distributed for area_mean")
}
```

```
## [1] "Data is not normally distributed for area_mean"
```

```r
# Normality test for smoothness_mean variable using Shapiro-Wilk test
print("Normality test for smoothness_mean:")
```

```
## [1] "Normality test for smoothness_mean:"
```

```r
shapiro_test_smoothness <- shapiro.test(Breastcancer_data2$smoothness_mean)
print(paste("Shapiro-Wilk test p-value:", shapiro_test_smoothness$p.value))
```

```
## [1] "Shapiro-Wilk test p-value: 8.60083255698697e-05"
```

```r
# Check if data is normal based on Shapiro-Wilk test
if (shapiro_test_smoothness$p.value > 0.05) {
  print("Data is normally distributed for smoothness_mean")
} else {
  print("Data is not normally distributed for smoothness_mean")
}
```

```
## [1] "Data is not normally distributed for smoothness_mean"
```

```r
# Normality test for compactness_mean variable using Shapiro-Wilk test
print("Normality test for compactness_mean:")
```

```
## [1] "Normality test for compactness_mean:"
```

```r
shapiro_test_compactness <- shapiro.test(Breastcancer_data2$compactness_mean)
print(paste("Shapiro-Wilk test p-value:", shapiro_test_compactness$p.value))
```

```
## [1] "Shapiro-Wilk test p-value: 3.96720386388631e-17"
```

```r
# Check if data is normal based on Shapiro-Wilk test
if (shapiro_test_compactness$p.value > 0.05) {
  print("Data is normally distributed for compactness_mean")
} else {
  print("Data is not normally distributed for compactness_mean")
}
```

```
## [1] "Data is not normally distributed for compactness_mean"
```

```r
# Normality test for symmetry_mean variable using Shapiro-Wilk test
print("Normality test for symmetry_mean:")
```

```
## [1] "Normality test for symmetry_mean:"
```

```r
shapiro_test_symmetry <- shapiro.test(Breastcancer_data2$symmetry_mean)
print(paste("Shapiro-Wilk test p-value:", shapiro_test_symmetry$p.value))
```

```
## [1] "Shapiro-Wilk test p-value: 7.8847728112006e-09"
```

```r
# Check if data is normal based on Shapiro-Wilk test
if (shapiro_test_symmetry$p.value > 0.05) {
  print("Data is normally distributed for symmetry_mean")
} else {
```

```r
  print("Data is not normally distributed for symmetry_mean")
}
```

```
## [1] "Data is not normally distributed for symmetry_mean"
```

```r
# Normality test for fractal_dimension_mean variable using Shapiro-Wilk test
print("Normality test for fractal_dimension_mean:")
```

```
## [1] "Normality test for fractal_dimension_mean:"
```

```r
shapiro_test_fractal_dimension <- shapiro.test(Breastcancer_data2$fractal_dim
ension_mean)
print(paste("Shapiro-Wilk test p-value:", shapiro_test_fractal_dimension$p.va
lue))
```

```
## [1] "Shapiro-Wilk test p-value: 1.95657476081236e-16"
```

```r
# Check if data is normal based on Shapiro-Wilk test
if (shapiro_test_fractal_dimension$p.value > 0.05) {
  print("Data is normally distributed for fractal_dimension_mean")
} else {
  print("Data is not normally distributed for fractal_dimension_mean")
}
```

```
## [1] "Data is not normally distributed for fractal_dimension_mean"
```

```r
# Ensure the ggplot2 package is installed and loaded
if (!require(ggplot2)) install.packages('ggplot2')
```

```
## Loading required package: ggplot2
```

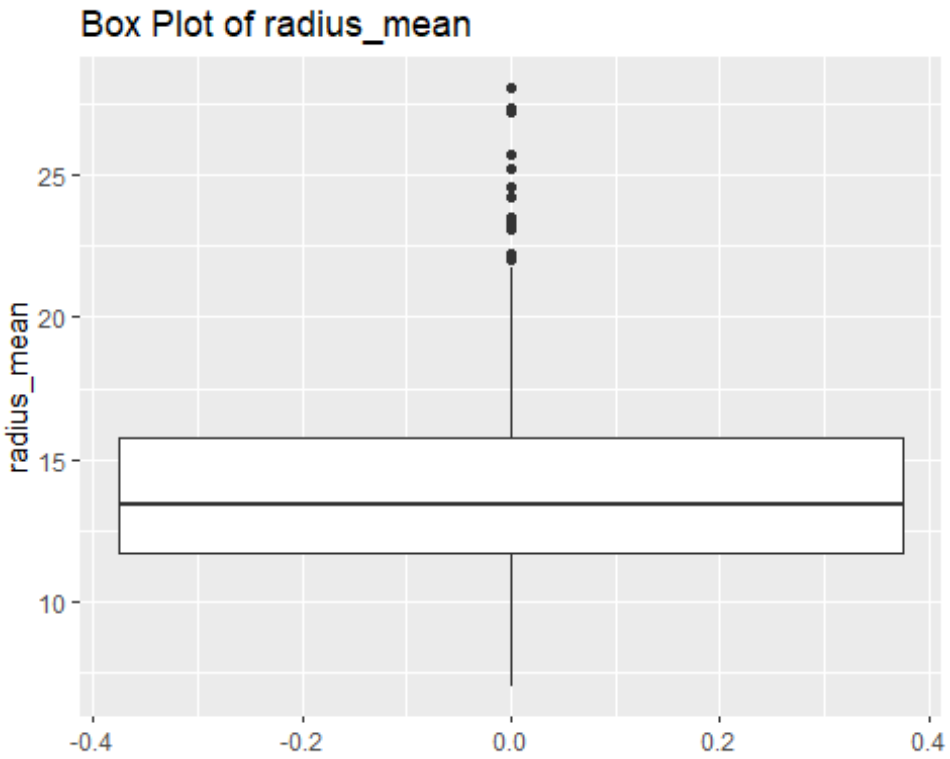
```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```r
# List of variables
variables <- c("radius_mean", "texture_mean", "perimeter_mean", "area_mean",
               "smoothness_mean", "compactness_mean", "symmetry_mean", "fract
al_dimension_mean")

# Create box plots for each variable
plots <- lapply(variables, function(var) {
  ggplot(Breastcancer_data2, aes_string(y = var)) +
    geom_boxplot() +
    labs(title = paste("Box Plot of", var), y = var)
})
```

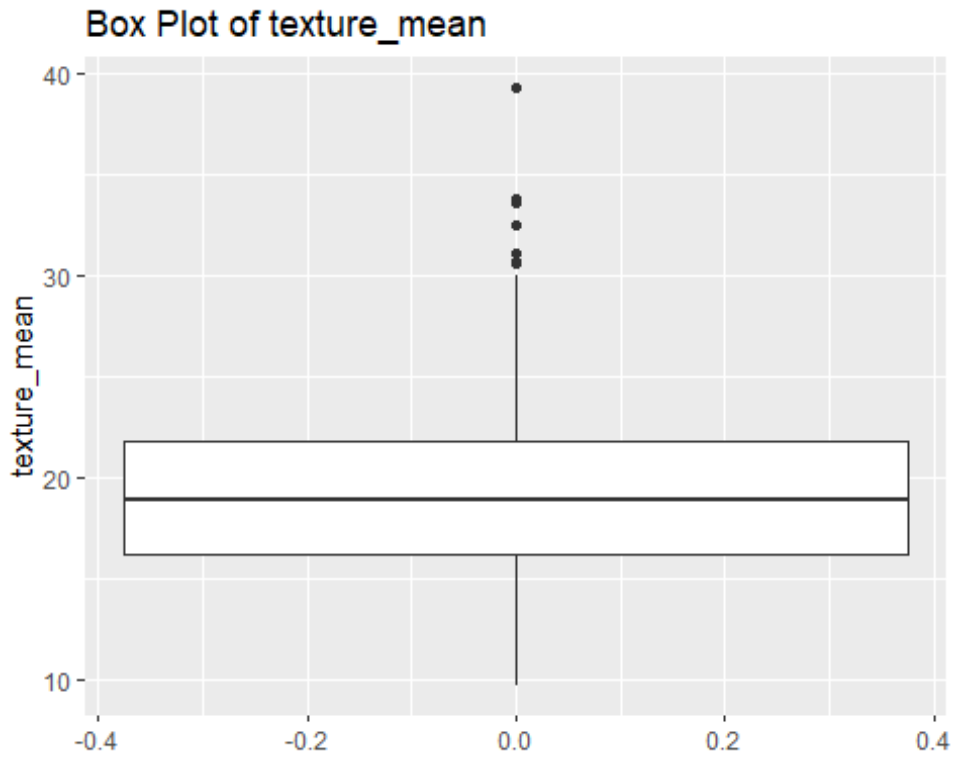
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# Display the plots
plots
```

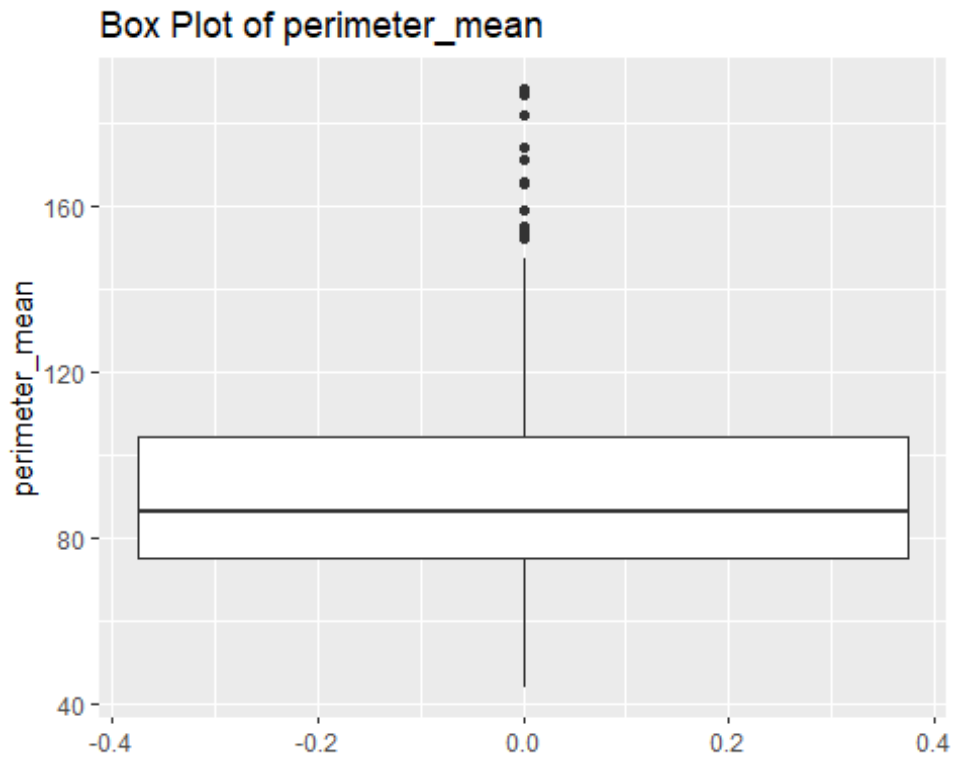
```
## [[1]]
```



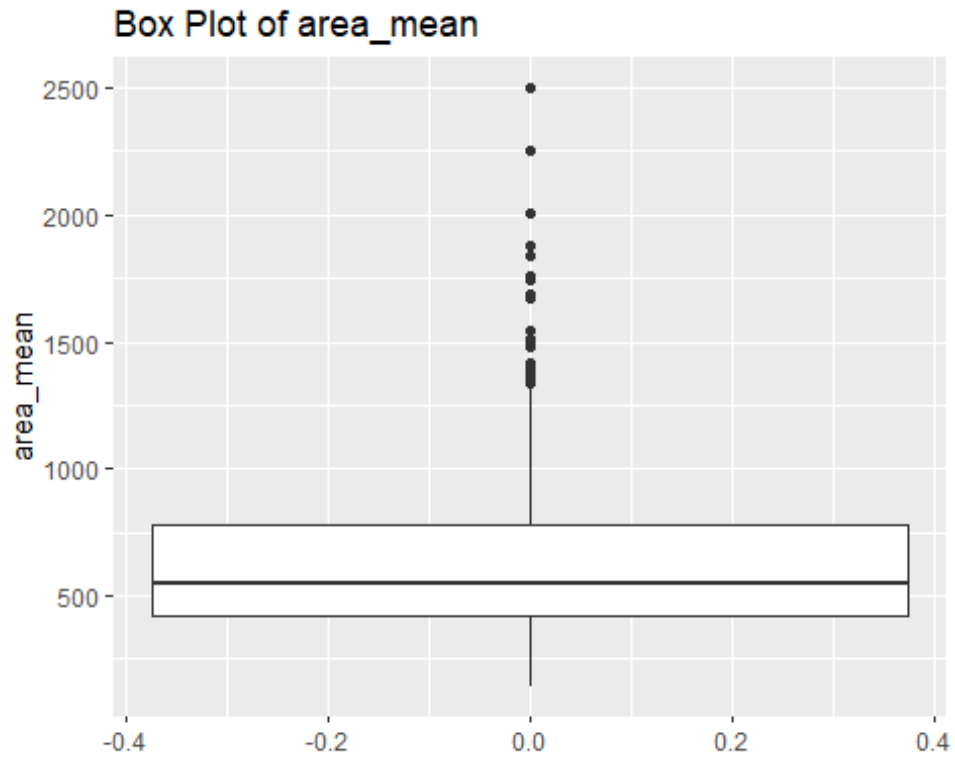

```
##
## [[2]]
```


Box Plot of texture_mean
texture_mean
40
30
20
10
-0.4
-0.2
0.0
0.2
0.4

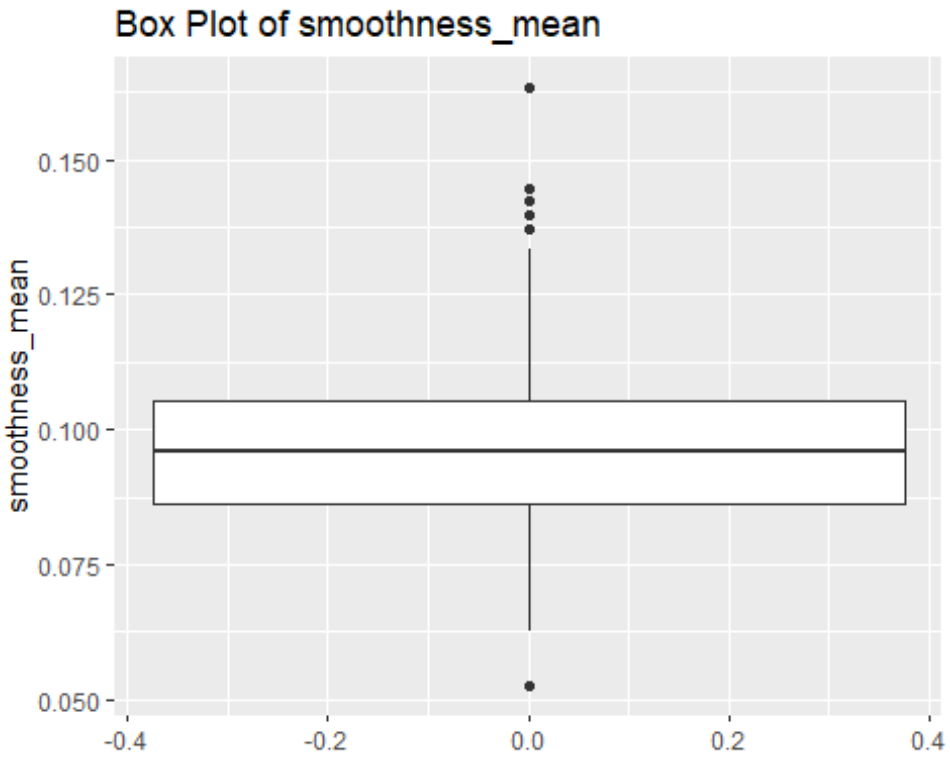
```
##
## [[3]]
```


Box Plot of perimeter_mean
perimeter_mean
160
120
80
40
-0.4
-0.2
0.0
0.2
0.4

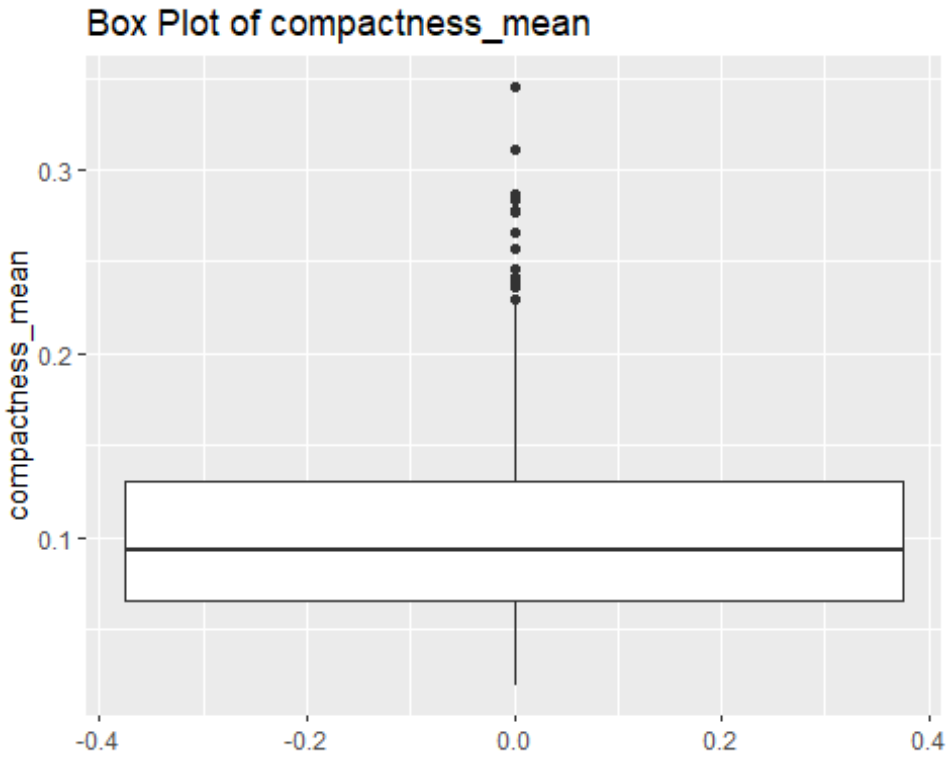
```
##
## [[4]]
```


Box Plot of area_mean
area_mean
2500
2000
1500
1000
500
-0.4
-0.2
0.0
0.2
0.4

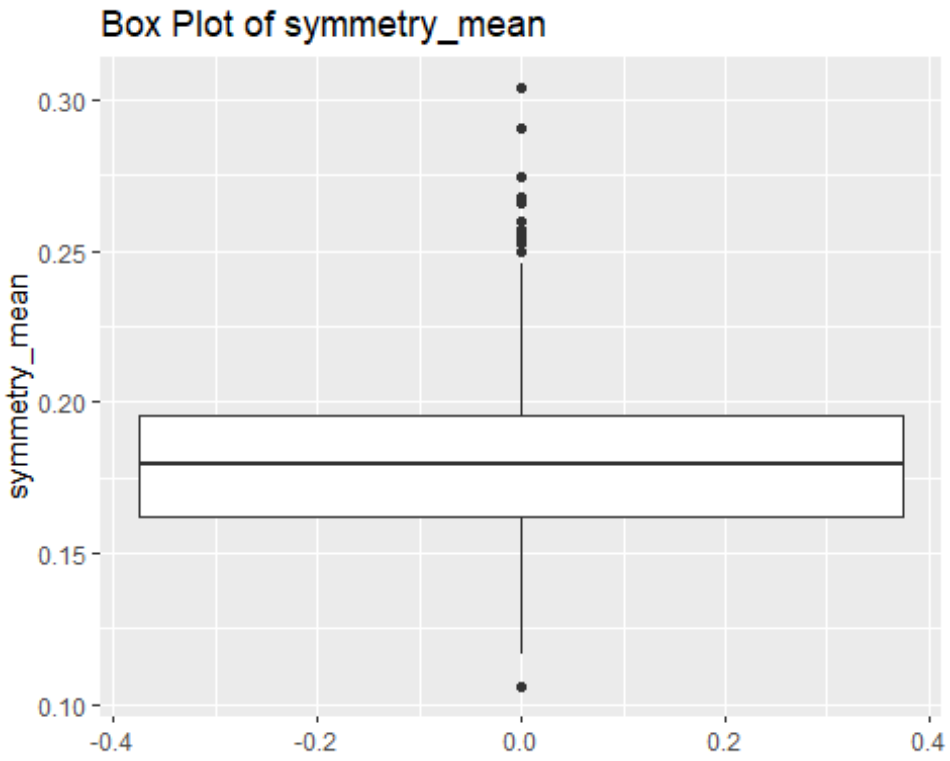
```
##
## [[5]]
```


Box Plot of smoothness_mean
smoothness_mean
0.150
0.125
0.100
0.075
0.050
-0.4
-0.2
0.0
0.2
0.4

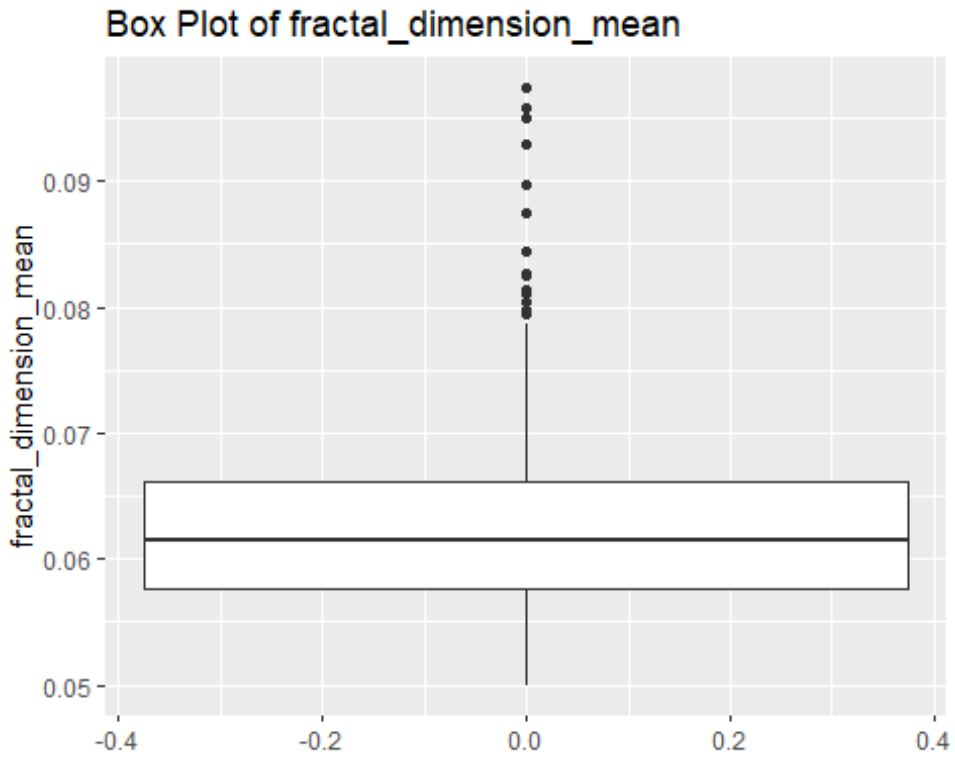
```
##
## [[6]]
```


Box Plot of compactness_mean
compactness_mean
0.3
0.2
0.1
-0.4
-0.2
0.0
0.2
0.4

```
##
## [[7]]
```

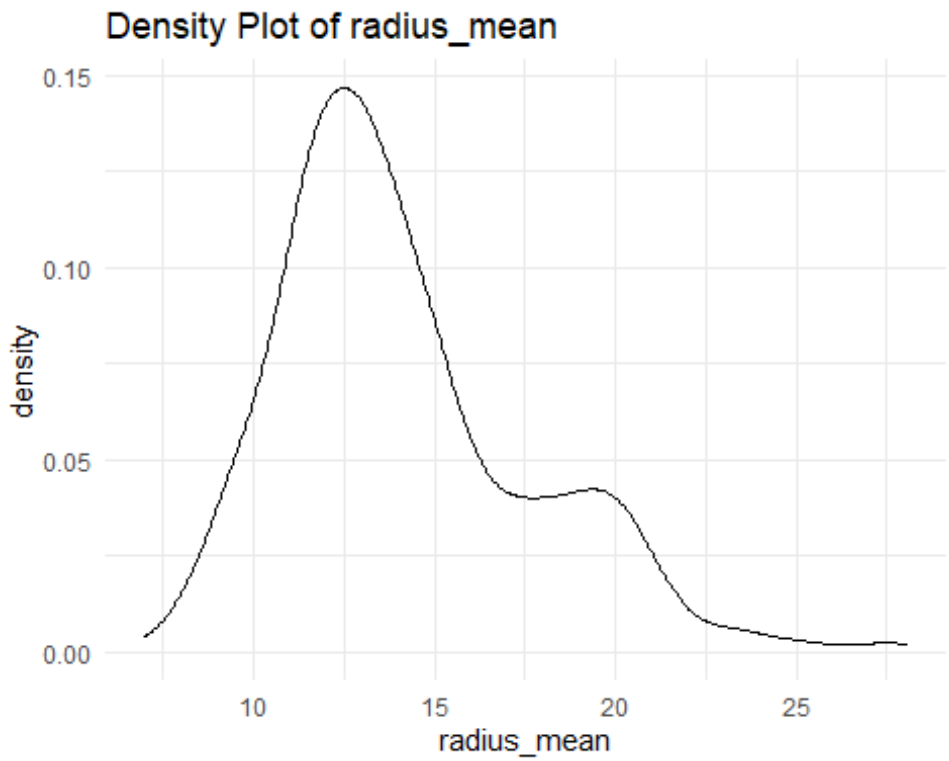

Box Plot of symmetry_mean
symmetry_mean
0.30
0.25
0.20
0.15
0.10
-0.4
-0.2
0.0
0.2
0.4

```
##
## [[8]]
```

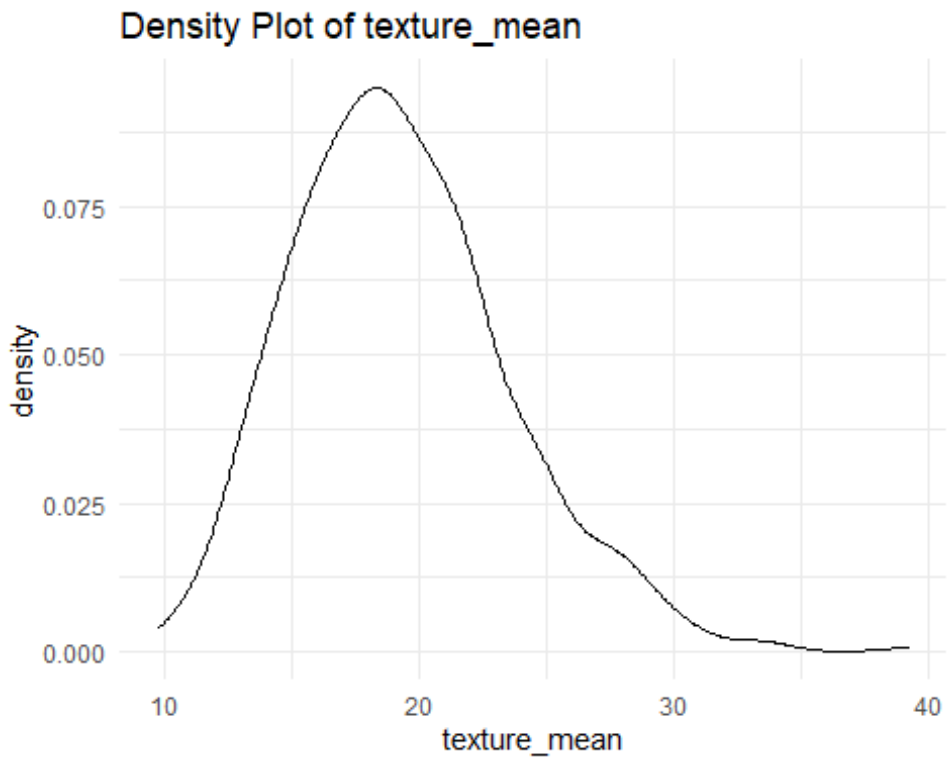



```r
# Ensure the ggplot2 package is installed and loaded
if (!require(ggplot2)) install.packages('ggplot2')

# Function to calculate skewness
skewness <- function(x) {
  mean((x - mean(x)) ^ 3) / (sd(x) ^ 3)
}

# List of variables
variables <- c("radius_mean", "texture_mean", "perimeter_mean", "area_mean",
               "smoothness_mean", "compactness_mean", "symmetry_mean", "fract
al_dimension_mean")

# Calculate skewness for each variable
skewness_values <- sapply(Breastcancer_data2[variables], skewness)

# Create density plots for each variable
plots <- lapply(variables, function(var) {
  ggplot(Breastcancer_data2, aes_string(x = var)) +
    geom_density() +
    labs(title = paste("Density Plot of", var), x = var) +
    theme_minimal()
})

# Display the plots
plots
```

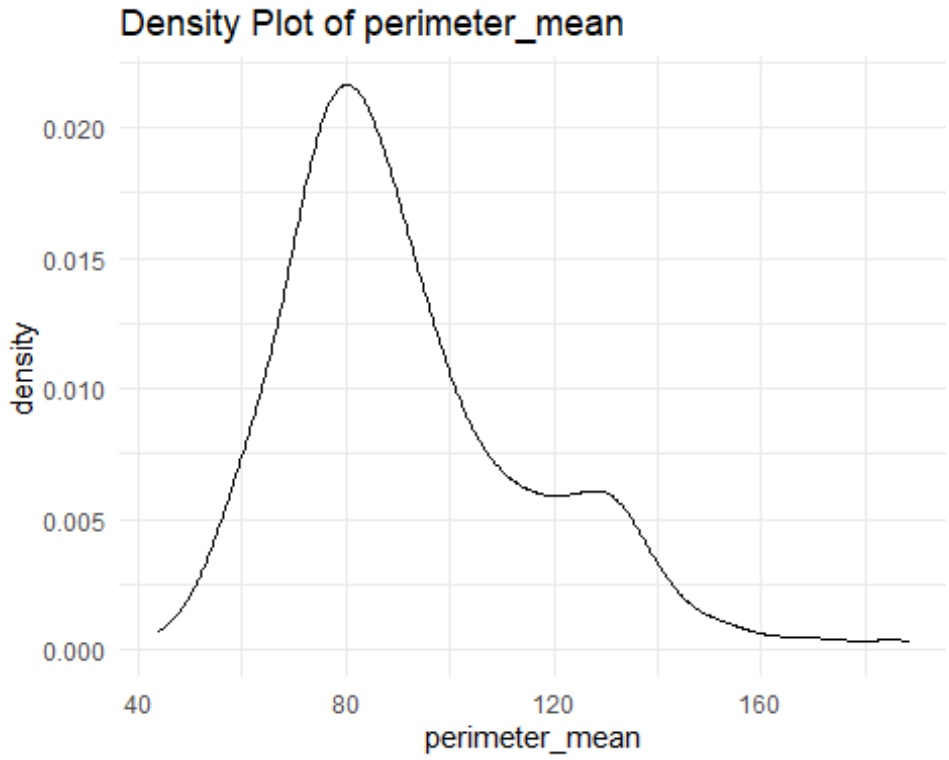
```
## [[1]]
```


Density Plot of radius_mean
0.15
0.10
0.05
0.00
density
10
15
20
25
radius_mean

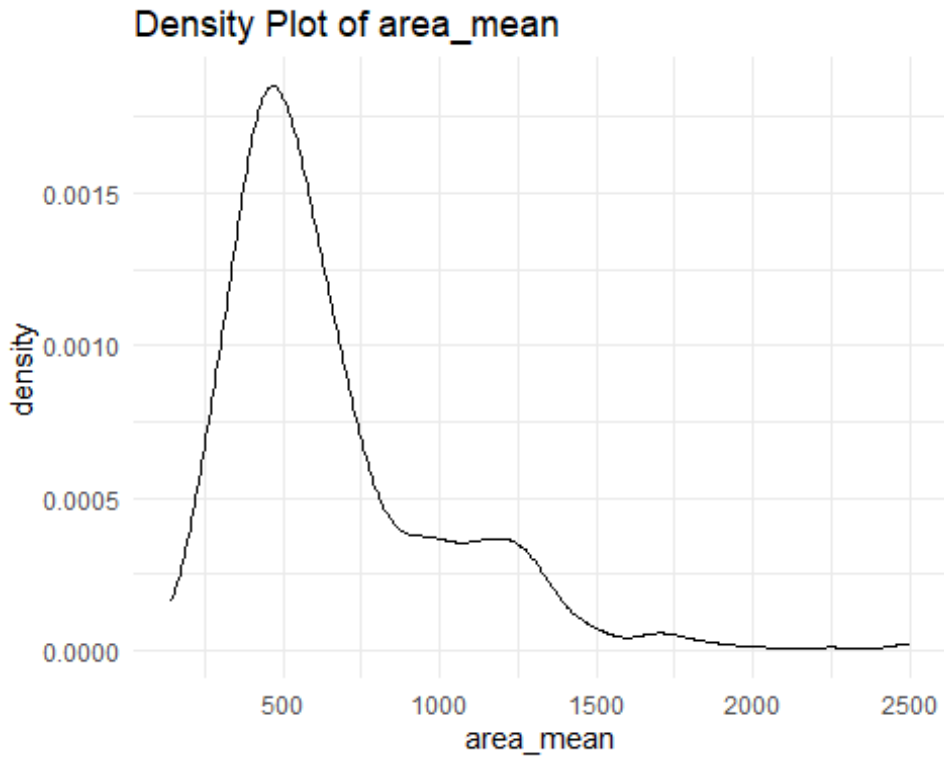
```
##
## [[2]]
```


Density Plot of texture_mean
density
0.075
0.050
0.025
0.000
10
20
30
40
texture_mean

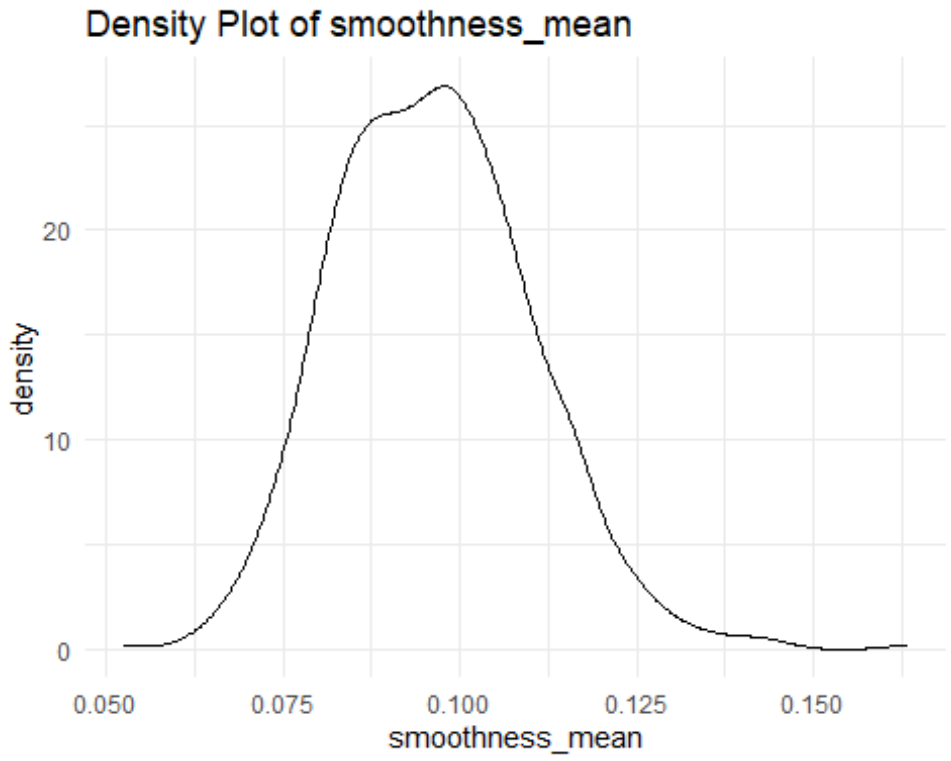
```
##
## [[3]]
```


Density Plot of perimeter_mean
density
0.020
0.015
0.010
0.005
0.000
40
80
120
160
perimeter_mean

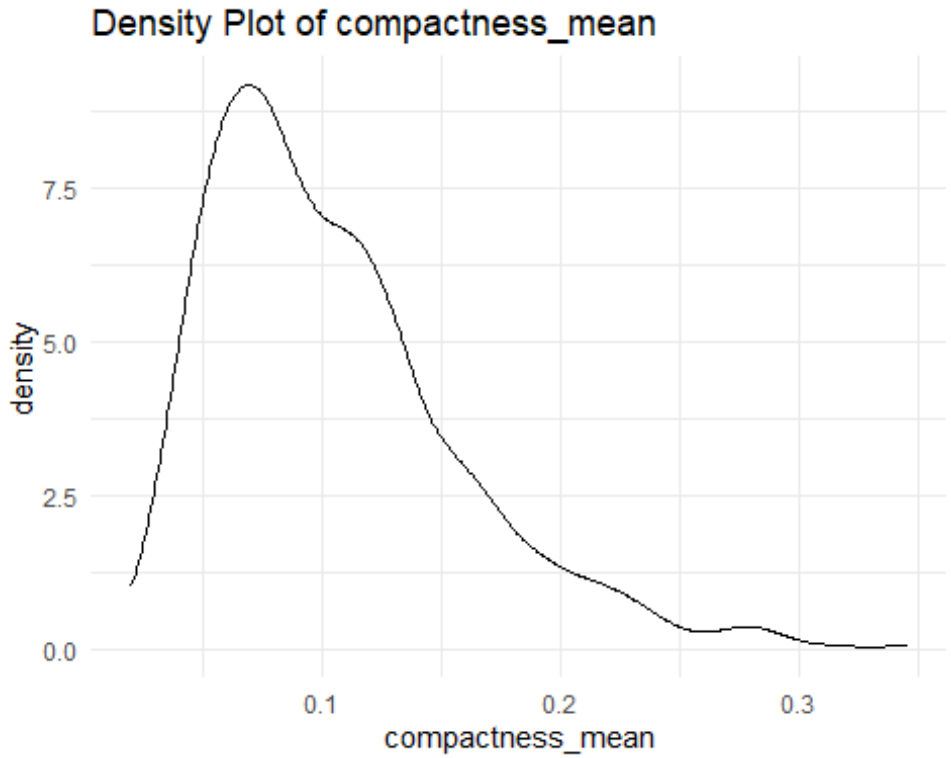
```
##
## [[4]]
```


Density Plot of area_mean
density
0.0015
0.0010
0.0005
0.0000
500
1000
1500
2000
2500
area_mean

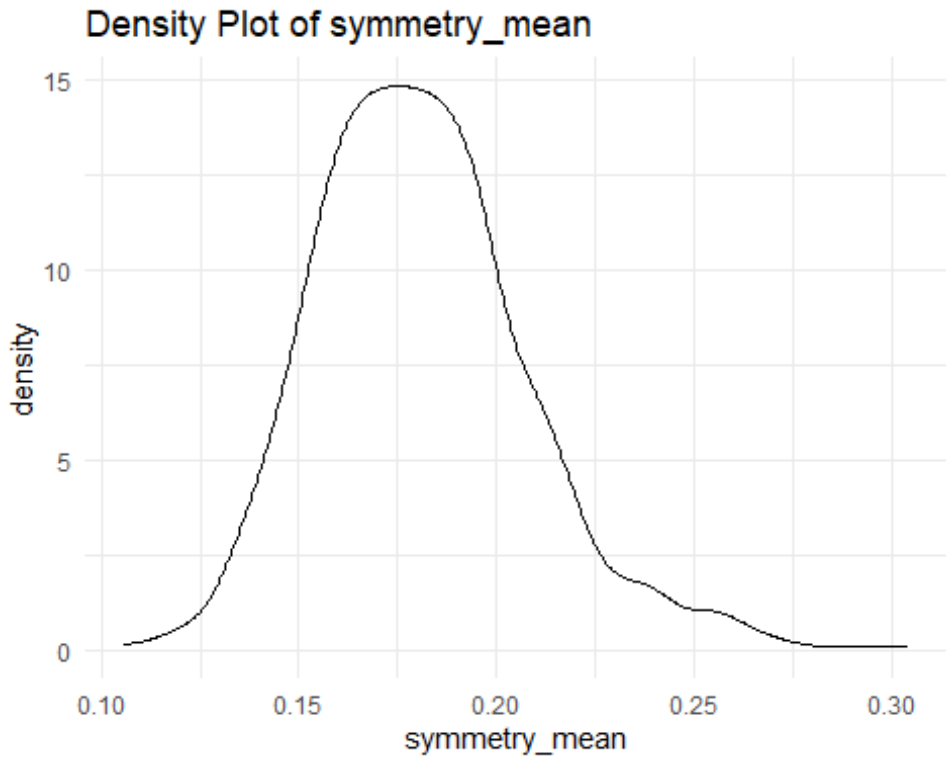
```
##
## [[5]]
```


Density Plot of smoothness_mean
density
20
10
0
0.050
0.075
0.100
0.125
0.150
smoothness_mean

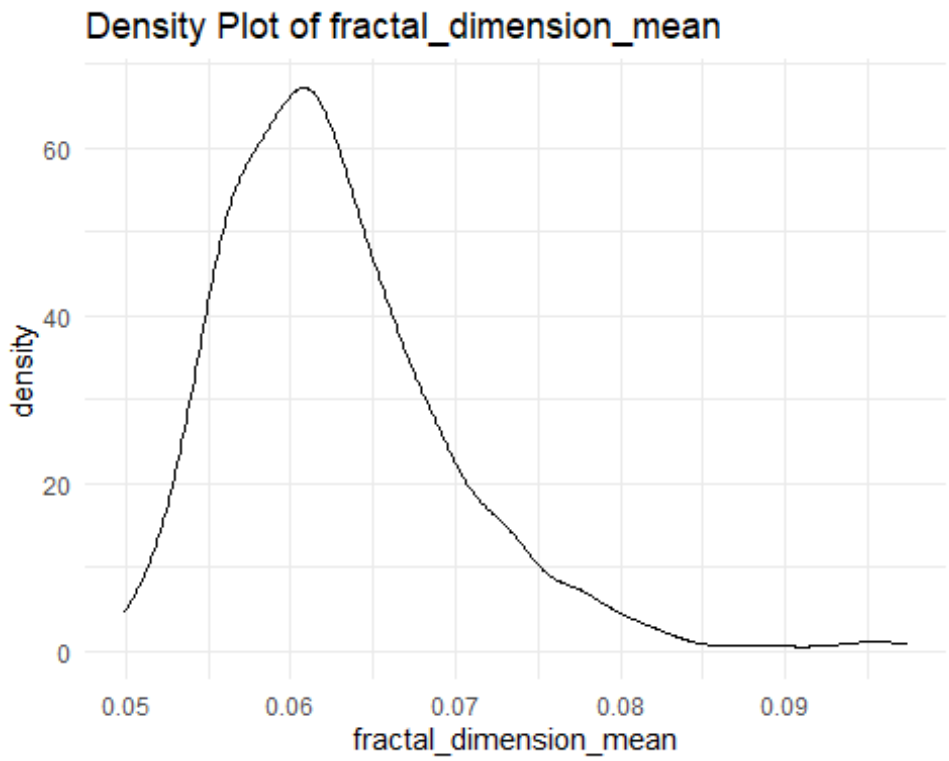
```
##
## [[6]]
```


Density Plot of compactness_mean
density
7.5
5.0
2.5
0.0
0.1
0.2
0.3
compactness_mean

```
##
## [[7]]
```


Density Plot of symmetry_mean
15
10
5
0
density
0.10
0.15
0.20
0.25
0.30
symmetry_mean

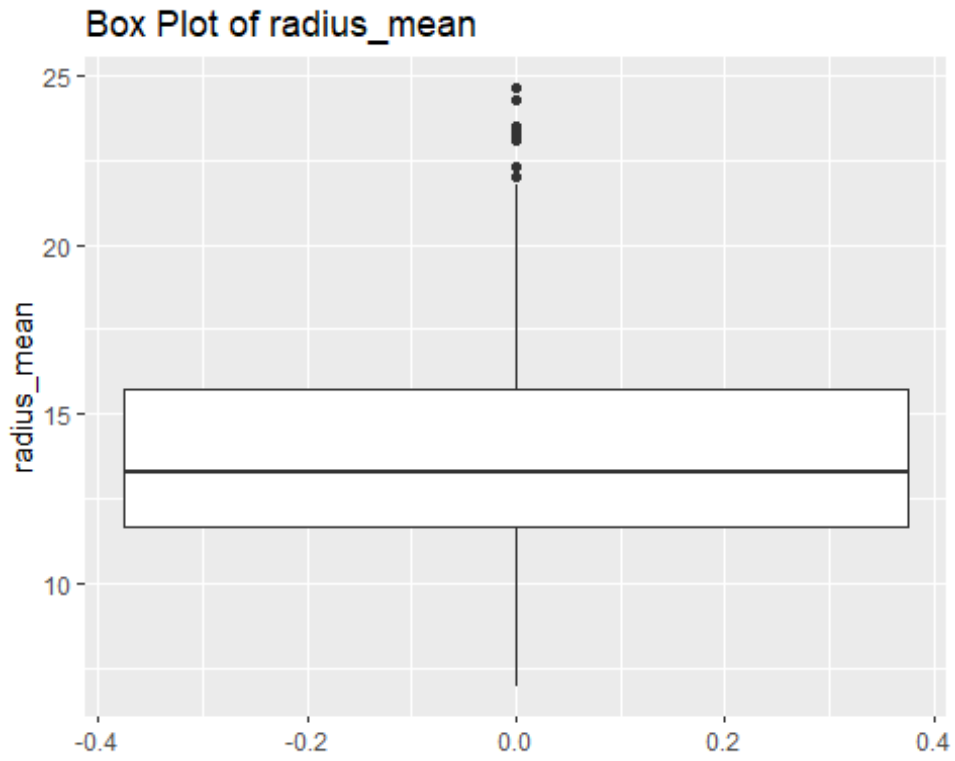
```
##
## [[8]]
```



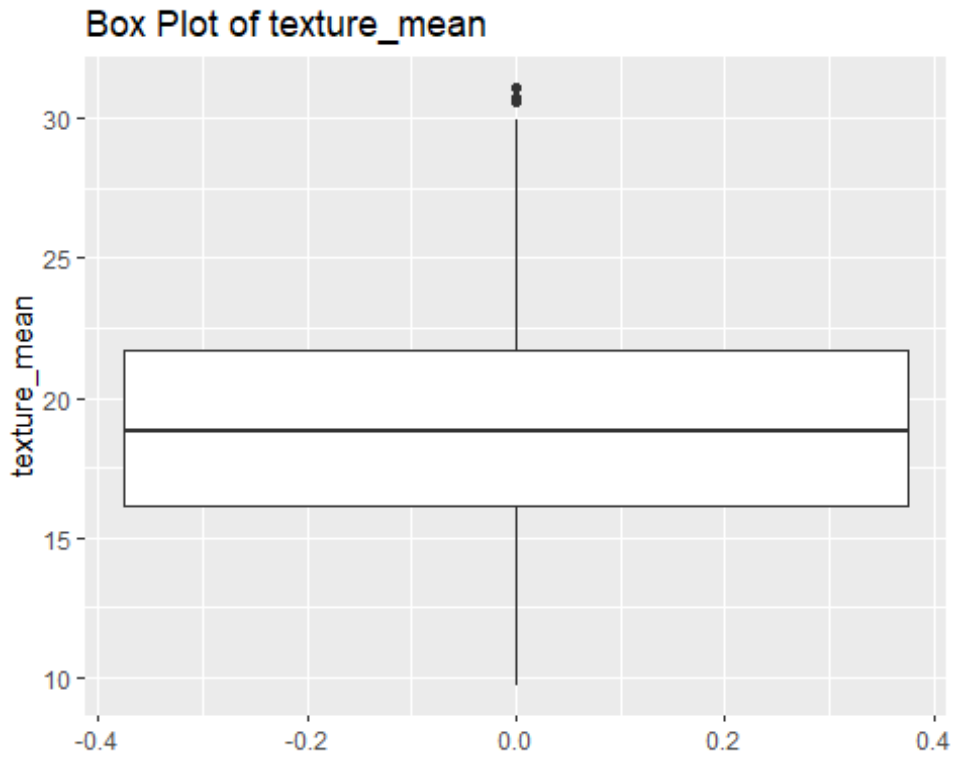

```
# Print skewness values
print(skewness_values)

##            radius_mean           texture_mean         perimeter_mean
##              0.9374168              0.6470241              0.9854334
##              area_mean        smoothness_mean       compactness_mean
##              1.6370654              0.4539207              1.1838556
##          symmetry_mean fractal_dimension_mean
##              0.7217877              1.2976191

# Ensure the ggplot2 package is installed and loaded
if (!require(ggplot2)) install.packages('ggplot2')

# List of variables
variables <- c("radius_mean", "texture_mean", "perimeter_mean", "area_mean",
               "smoothness_mean", "compactness_mean", "symmetry_mean", "fract
al_dimension_mean")

# Outlier treatment function
treat_outliers <- function(x) {
  mean_x <- mean(x, na.rm = TRUE)
  sd_x <- sd(x, na.rm = TRUE)
  threshold <- mean_x + 3 * sd_x
  x[x > threshold] <- NA
  return(x)
}
```

```r
# Apply outlier treatment and create box plots for each variable
plots <- lapply(variables, function(var) {
  Breastcancer_data2[[var]] <- treat_outliers(Breastcancer_data2[[var]]) # Ap
ply outlier treatment
  ggplot(Breastcancer_data2, aes_string(y = var)) +
    geom_boxplot() +
    labs(title = paste("Box Plot of", var), y = var)
})

# Display the plots
plots
```

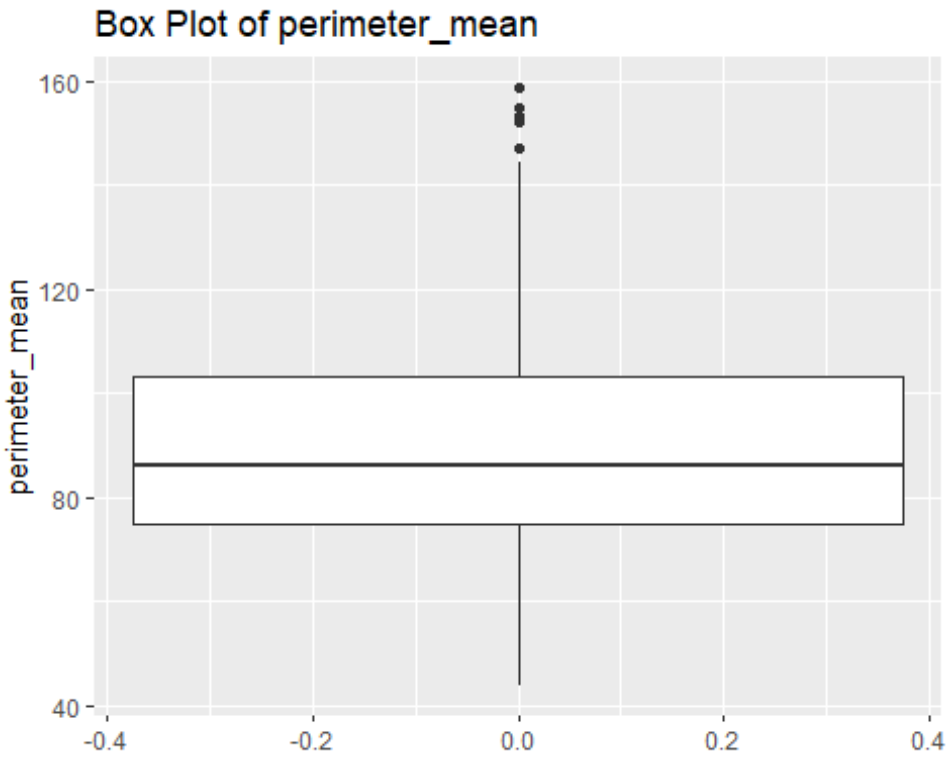
```
## [[1]]

## Warning: Removed 5 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



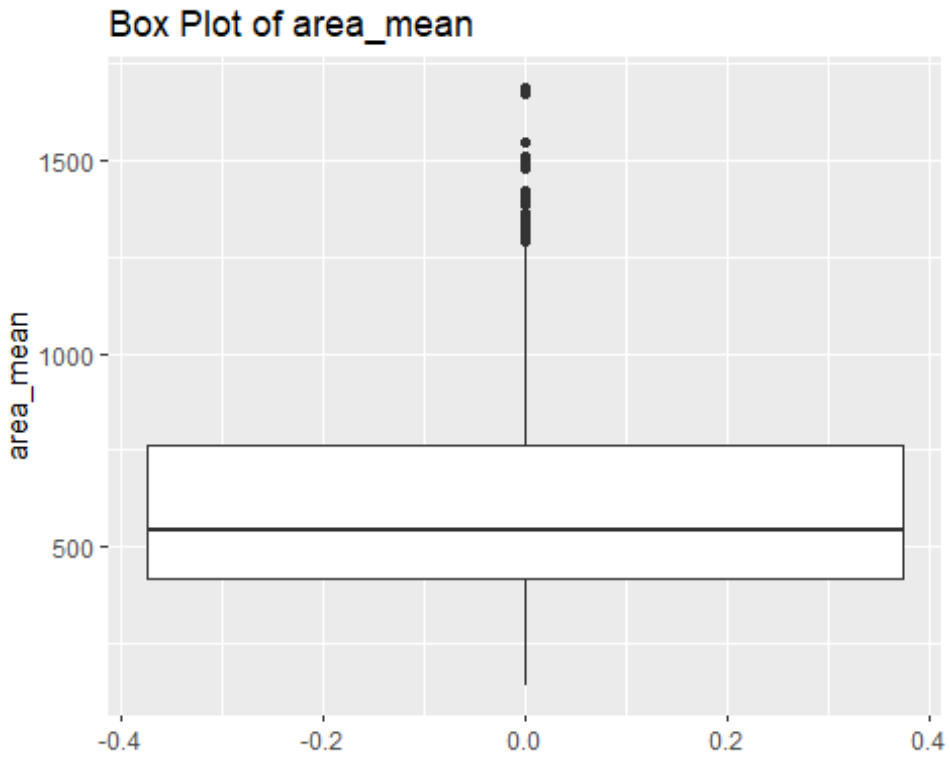

```
##
## [[2]]

## Warning: Removed 4 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



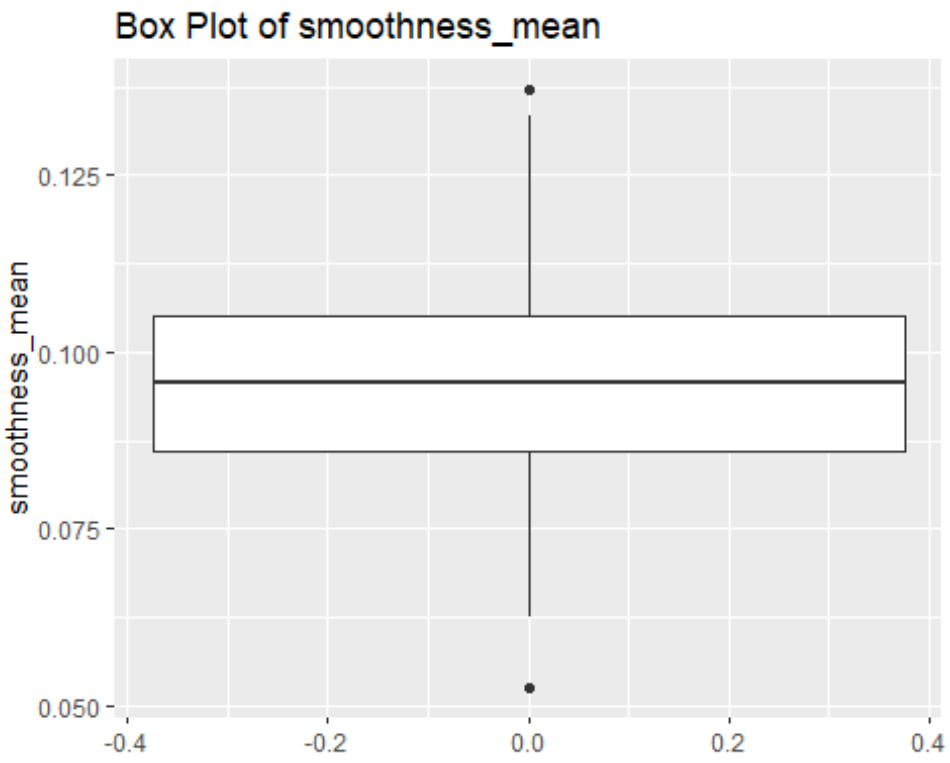

```
##
## [[3]]

## Warning: Removed 7 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



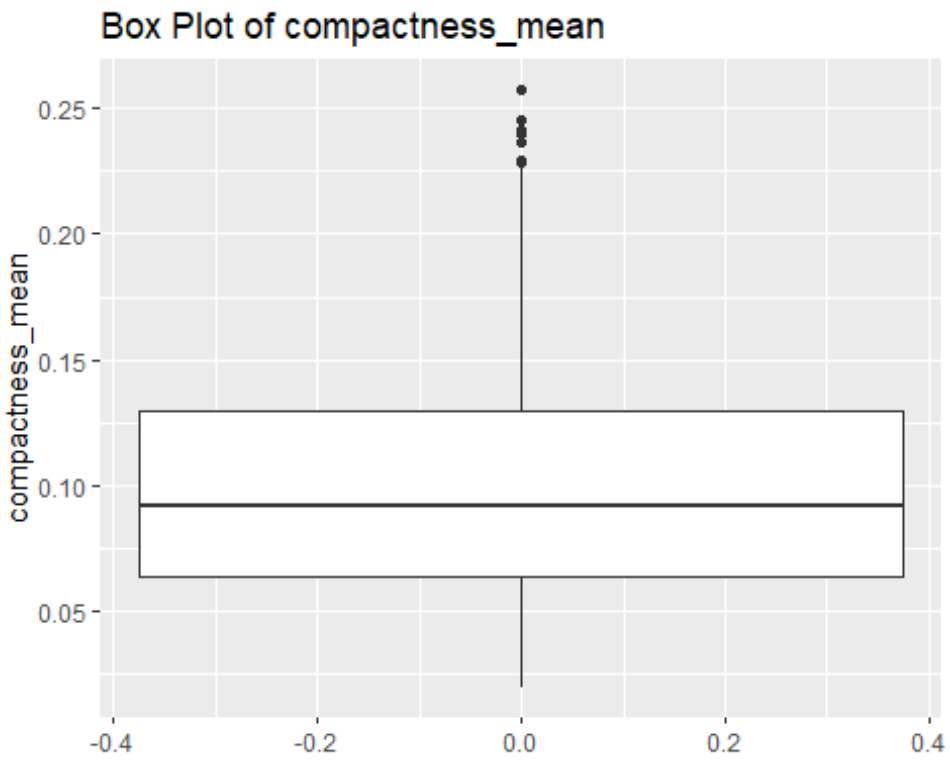

```
##
## [[4]]

## Warning: Removed 8 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



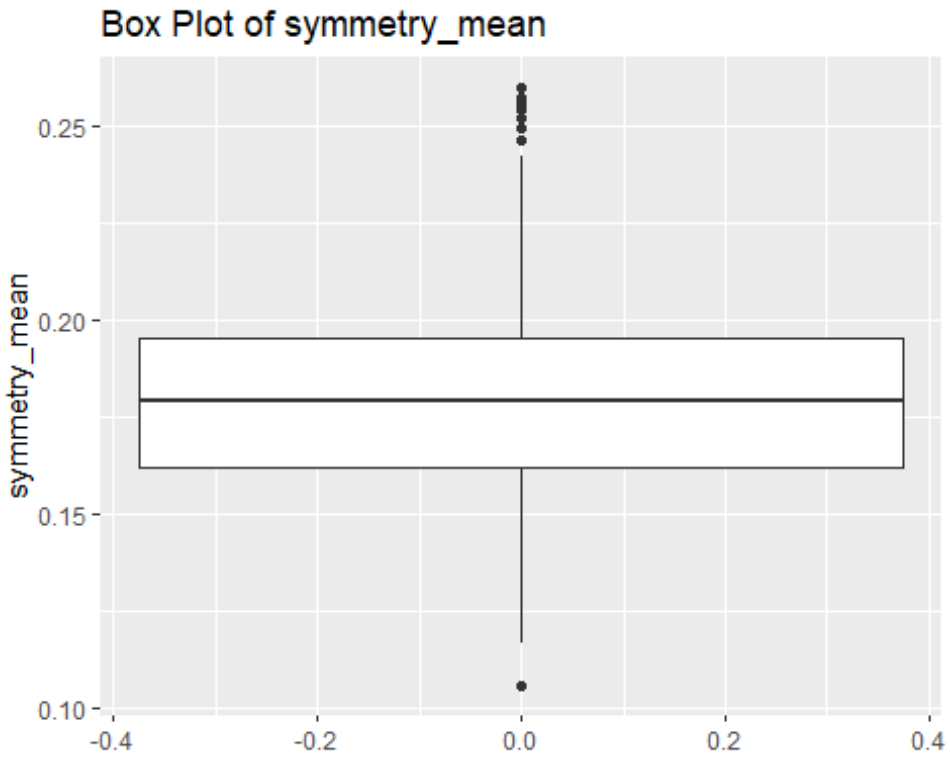

```
##
## [[5]]

## Warning: Removed 4 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



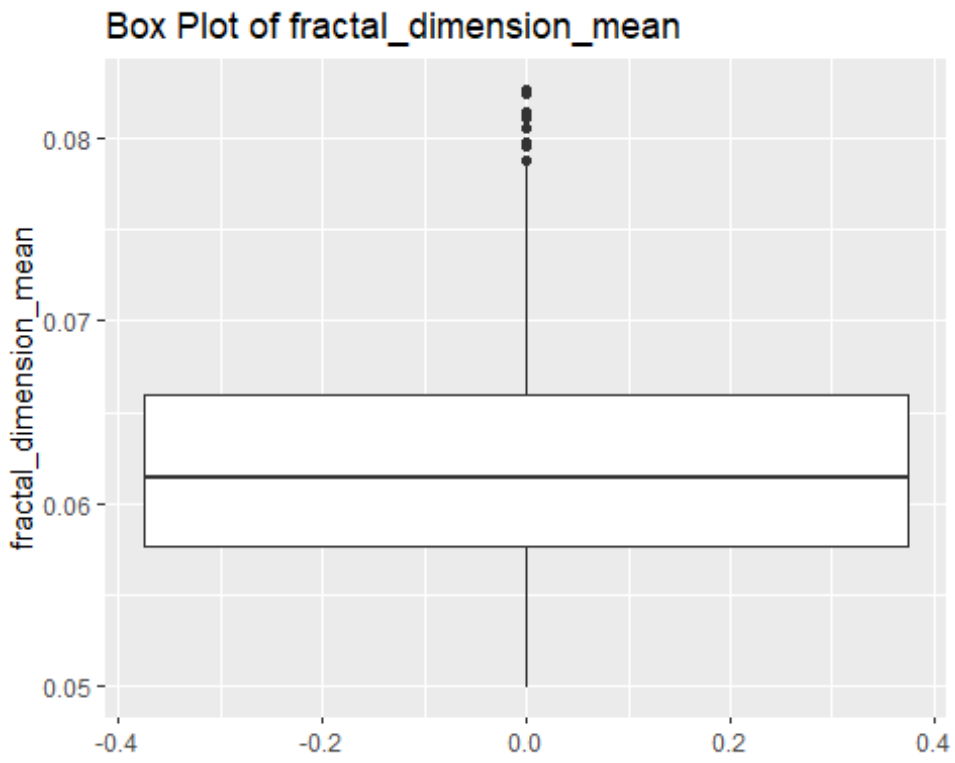

```
##
## [[6]]

## Warning: Removed 9 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```




```
##
## [[7]]

## Warning: Removed 5 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```




```
##
## [[8]]

## Warning: Removed 7 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```




```
## Calculate the quartiles of the radius_mean column
quartiles <- quantile(Breastcancer_data2$radius_mean, probs = c(0, 0.25, 0.5,
0.75, 1))

# Create a new column with the tumor size categories based on the quartile ra
nges
Breastcancer_data2$tumor_size <- cut(Breastcancer_data2$radius_mean,
                                     breaks = quartiles,
                                     labels = c("Very Small Tumors", "Small Tumors
", "Medium Tumors", "Large Tumors"),
                                     include.lowest = TRUE)

# Print the updated data frame
head(Breastcancer_data2)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean symmetry_mean fractal_dimension_mean
## 1         0.11840          0.27760        0.2419                0.07871
## 2         0.08474          0.07864        0.1812                0.05667
## 3         0.10960          0.15990        0.2069                0.05999
## 4         0.14250          0.28390        0.2597                0.09744
```

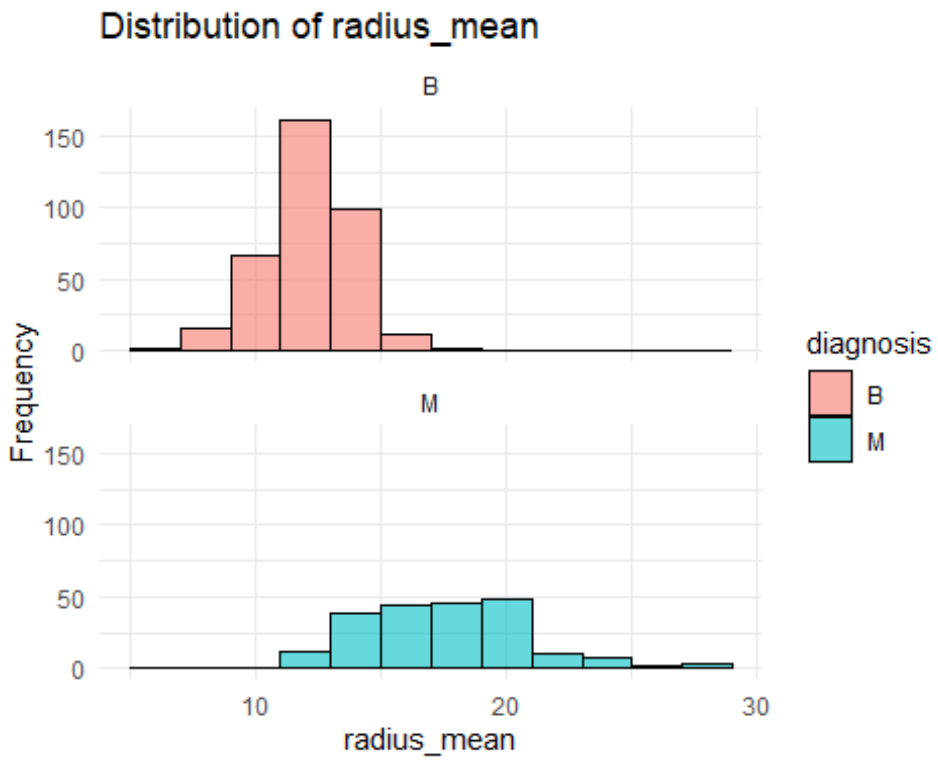
```
## 5         0.10030          0.13280        0.1809                0.05883
## 6         0.12780          0.17000        0.2087                0.07613
##          tumor_size
## 1      Large Tumors
## 2      Large Tumors
## 3      Large Tumors
## 4 Very Small Tumors
## 5      Large Tumors
## 6      Small Tumors
```

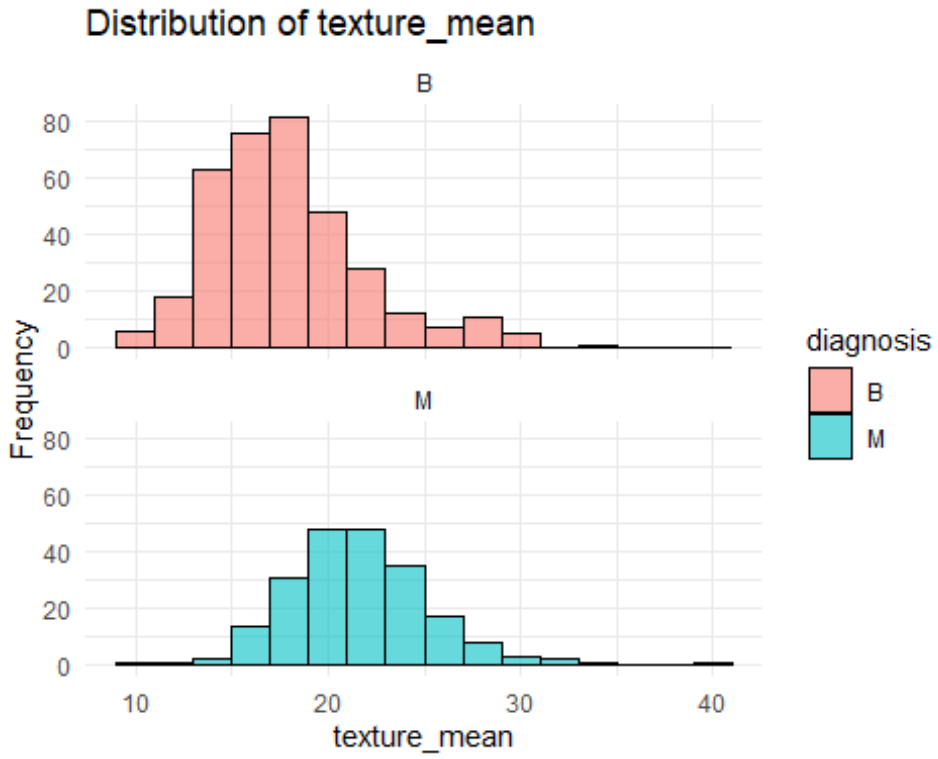
```
# Calculate the quartiles of the radius_mean column
quartiles <- quantile(Breastcancer_data2$radius_mean, probs = c(0, 0.25, 0.5,
0.75, 1))

# Create a new column with the tumor size categories based on the quartile ra
nges
Breastcancer_data2$tumor_size <- cut(Breastcancer_data2$radius_mean,
                                     breaks = quartiles,
                                     labels = c("Very Small Tumors", "Small Tu
mors", "Medium Tumors", "Large Tumors"),
                                     include.lowest = TRUE)

# Create a new column with numerical values corresponding to the categories i
n the existing "tumor_size" column
Breastcancer_data2$tumor_size_numerical <- as.integer(as.factor(Breastcancer_
data2$tumor_size))

# Print the updated data frame
head(Breastcancer_data2)
```

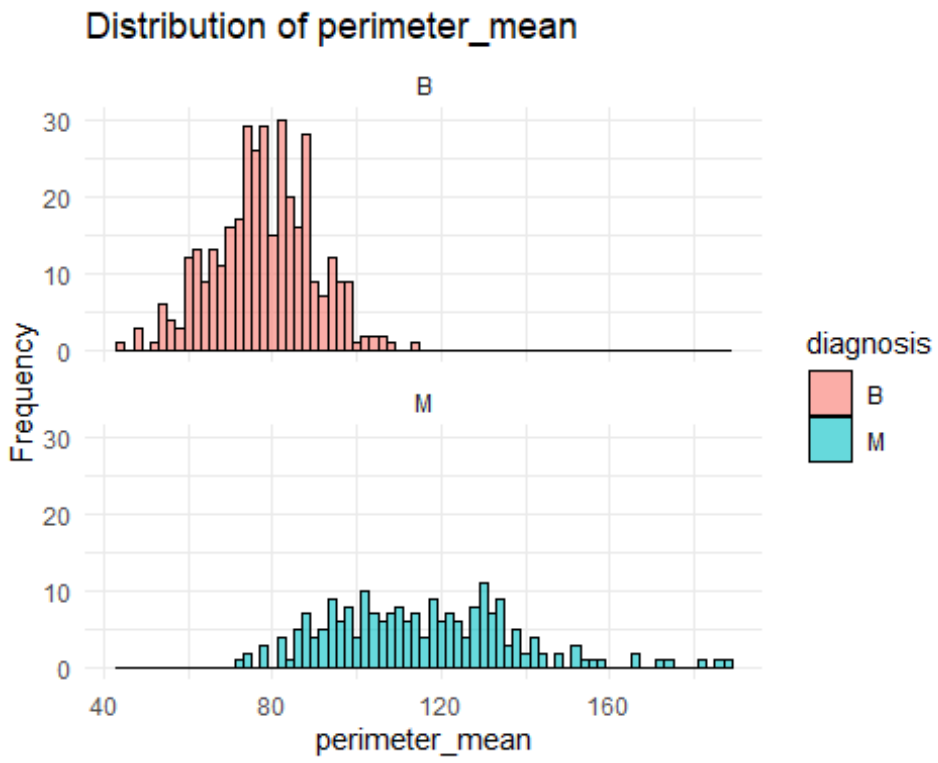
```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean symmetry_mean fractal_dimension_mean
## 1         0.11840          0.27760        0.2419                0.07871
## 2         0.08474          0.07864        0.1812                0.05667
## 3         0.10960          0.15990        0.2069                0.05999
## 4         0.14250          0.28390        0.2597                0.09744
## 5         0.10030          0.13280        0.1809                0.05883
## 6         0.12780          0.17000        0.2087                0.07613
##          tumor_size tumor_size_numerical
## 1      Large Tumors                    4
## 2      Large Tumors                    4
## 3      Large Tumors                    4
## 4 Very Small Tumors                    1
```

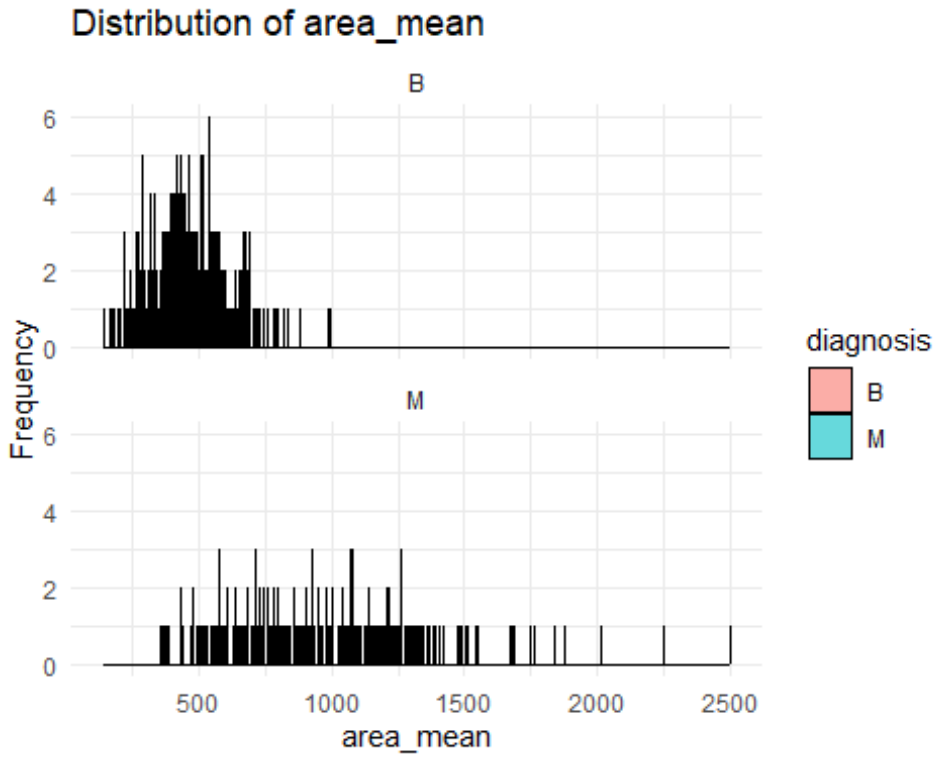
```
## 5      Large Tumors                    4
## 6      Small Tumors                    2
```

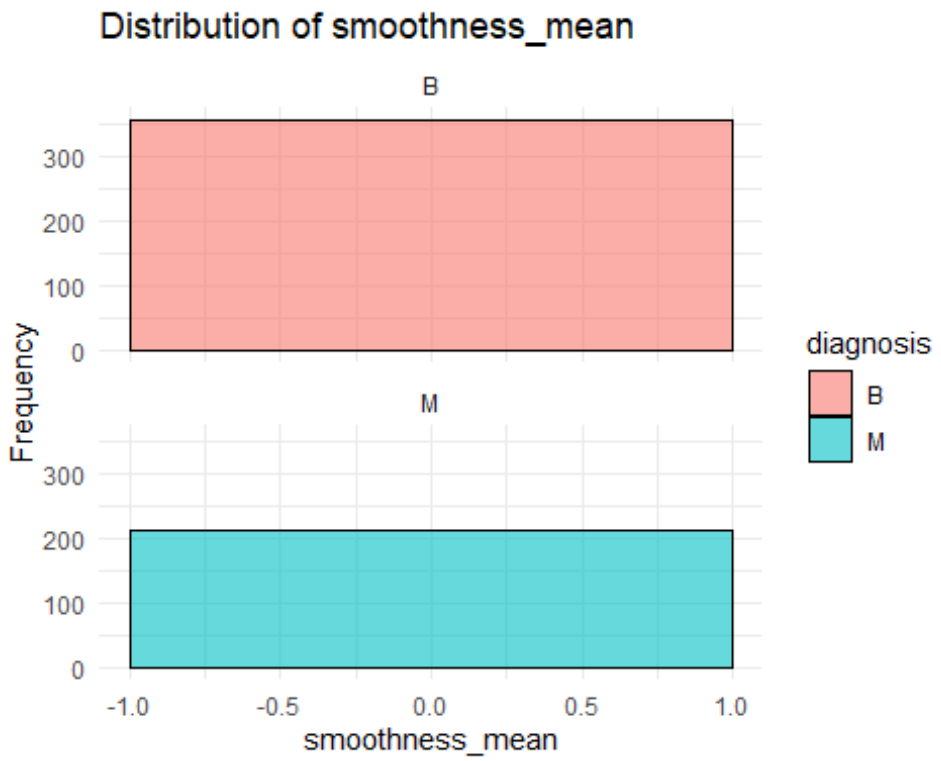
```r
# Load necessary libraries
library(ggplot2)

# Assuming the diagnosis variable is named "diagnosis" in your dataset

# Create histograms for all variables grouped by diagnosis
for (var in c("radius_mean", "texture_mean", "perimeter_mean", "area_mean",
              "smoothness_mean", "compactness_mean", "symmetry_mean")) {

  # Create histogram
  p <- ggplot(Breastcancer_data2, aes_string(x = var, fill = "diagnosis")) +
    geom_histogram(binwidth = 2, color = "black", alpha = 0.6) +
    labs(x = var, y = "Frequency", title = paste("Distribution of", var)) +
    theme_minimal() +
    facet_wrap(~ diagnosis, ncol = 1)

  # Print histogram
  print(p)
}
```

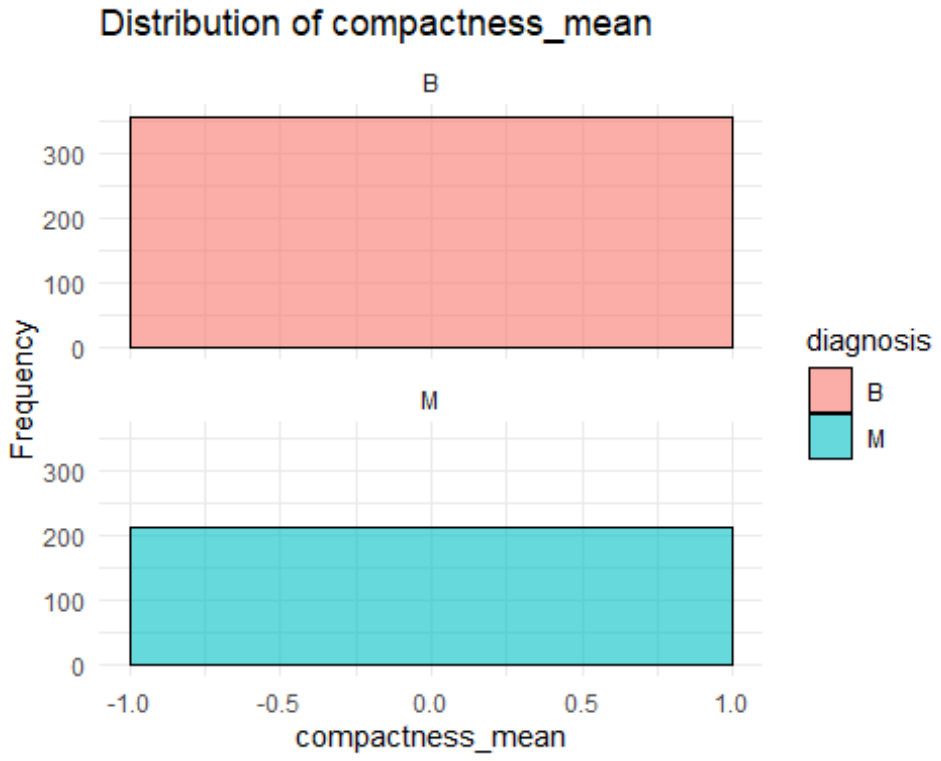

Distribution of radius_mean
B
M
Frequency
150
100
50
0
10
20
30
radius_mean
diagnosis
B
M

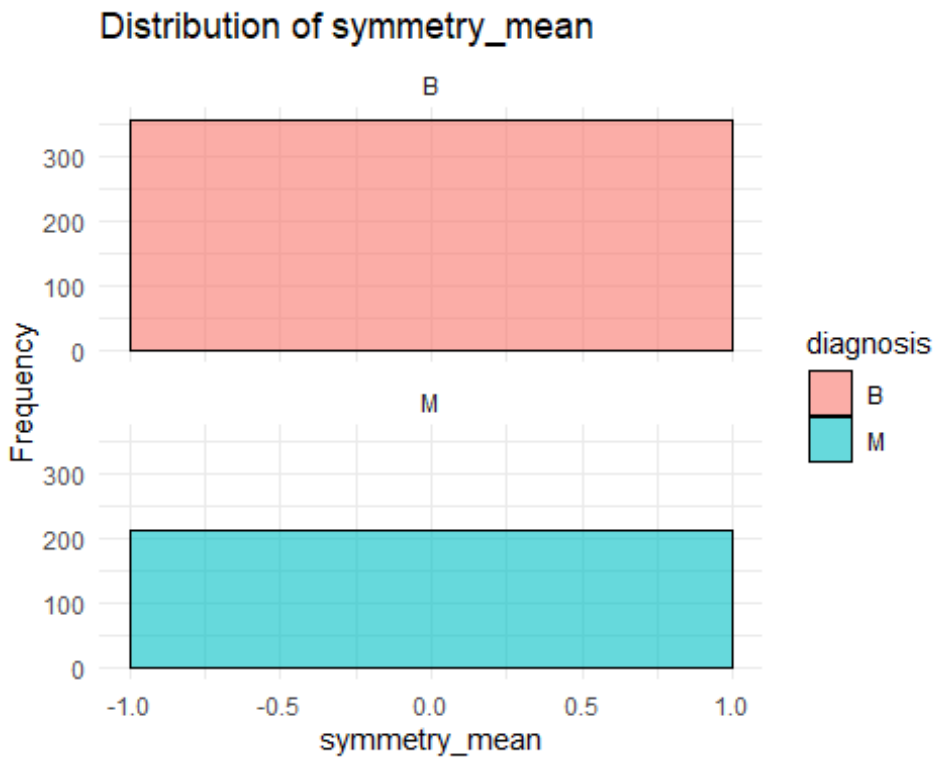

Distribution of texture_mean
B
M
Frequency
80
60
40
20
0
10
20
30
40
texture_mean
diagnosis
B
M


Distribution of perimeter_mean
B
M
Frequency
30
20
10
0
40
80
120
160
perimeter_mean
diagnosis
B
M


Distribution of area_mean
B
M
Frequency
6
4
2
0
500
1000
1500
2000
2500
area_mean
diagnosis
B
M


Distribution of smoothness_mean
B
M
300
200
100
0
Frequency
-1.0
-0.5
0.0
0.5
1.0
smoothness_mean
diagnosis
B
M


Distribution of compactness_mean
B
M
300
200
100
0
Frequency
-1.0
-0.5
0.0
0.5
1.0
compactness_mean
diagnosis
B
M

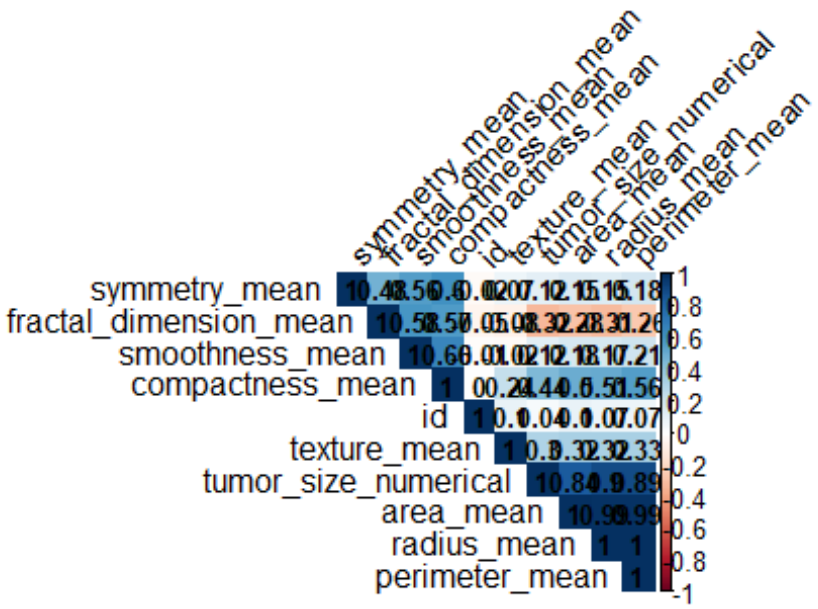



```r
# Select only numeric columns for correlation analysis
numeric_data <- Breastcancer_data2[sapply(Breastcancer_data2, is.numeric)]


# Compute the correlation matrix
cor_matrix <- cor(numeric_data)


# Print the correlation matrix
print(cor_matrix)
```

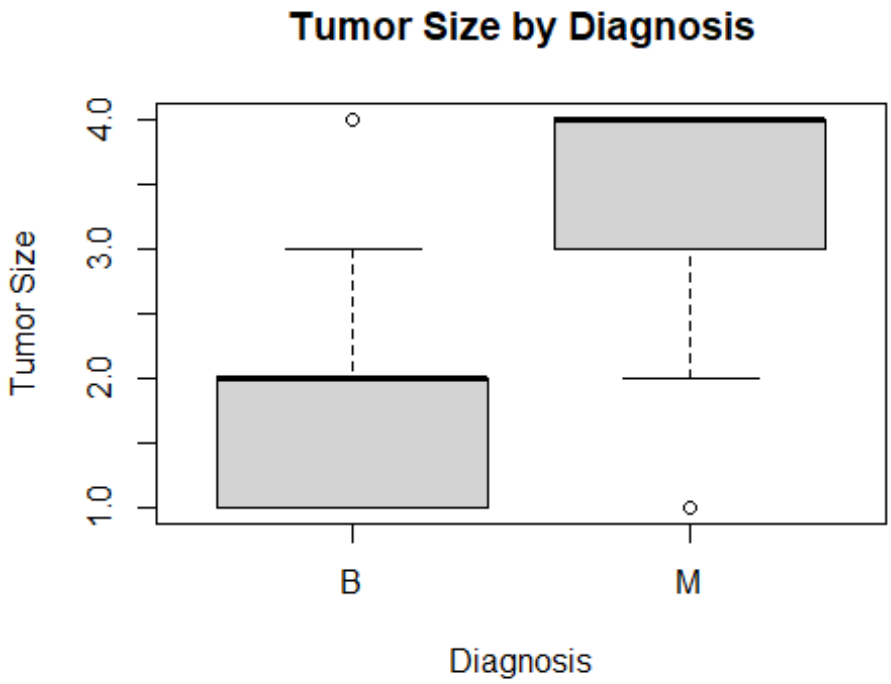
```
##                                   id radius_mean texture_mean perimeter_me
an
## id                     1.0000000000  0.07462647   0.09976989     0.073159
41
## radius_mean            0.0746264697  1.00000000   0.32378189     0.997855
28
## texture_mean           0.0997698912  0.32378189   1.00000000     0.329533
06
## perimeter_mean         0.0731594119  0.99785528   0.32953306     1.000000
00
## area_mean              0.0968928233  0.98735717   0.32108570     0.986506
80
## smoothness_mean       -0.0129681975  0.17058119  -0.02338852     0.207278
16
## compactness_mean       0.0000957011  0.50612358   0.23670222     0.556936
```

```
21
## symmetry_mean          -0.0221140609  0.14774124   0.07140098     0.183027
21
## fractal_dimension_mean -0.0525114476 -0.31163083  -0.07643718    -0.261476
91
## tumor_size_numerical    0.0360597917  0.89547746   0.30348407     0.890401
32
##                          area_mean smoothness_mean compactness_mean
## id                      0.09689282     -0.01296820     0.0000957011
## radius_mean             0.98735717      0.17058119     0.5061235775
## texture_mean            0.32108570     -0.02338852     0.2367022221
## perimeter_mean          0.98650680      0.20727816     0.5569362109
## area_mean               1.00000000      0.17702838     0.4985016822
## smoothness_mean         0.17702838      1.00000000     0.6591232152
## compactness_mean        0.49850168      0.65912322     1.0000000000
## symmetry_mean           0.15129308      0.55777479     0.6026410484
## fractal_dimension_mean -0.28310981      0.58479200     0.5653686634
## tumor_size_numerical    0.83918195      0.12175585     0.4444671476
##                         symmetry_mean fractal_dimension_mean
## id                        -0.02211406            -0.05251145
## radius_mean                0.14774124            -0.31163083
## texture_mean               0.07140098            -0.07643718
## perimeter_mean             0.18302721            -0.26147691
## area_mean                  0.15129308            -0.28310981
## smoothness_mean            0.55777479             0.58479200
## compactness_mean           0.60264105             0.56536866
## symmetry_mean              1.00000000             0.47992133
## fractal_dimension_mean     0.47992133             1.00000000
## tumor_size_numerical       0.11946552            -0.32403775
##                         tumor_size_numerical
## id                                0.03605979
## radius_mean                       0.89547746
## texture_mean                      0.30348407
## perimeter_mean                    0.89040132
## area_mean                         0.83918195
## smoothness_mean                   0.12175585
## compactness_mean                  0.44446715
## symmetry_mean                     0.11946552
## fractal_dimension_mean           -0.32403775
## tumor_size_numerical              1.00000000

# Install and load corrplot package if not already installed
if (!requireNamespace("corrplot", quietly = TRUE)) {
    install.packages("corrplot")
}
library(corrplot)

## Warning: package 'corrplot' was built under R version 4.3.3

## corrplot 0.92 loaded
```

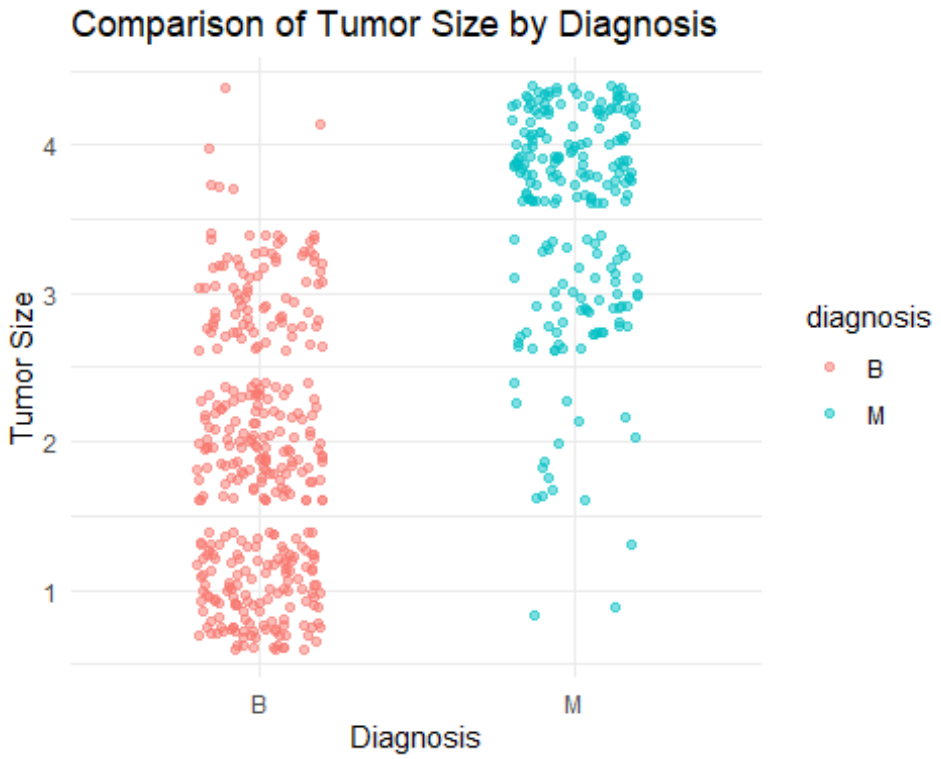
```r
# Visualize the correlation matrix with adjusted plot size and label size
corrplot(cor_matrix, method = "color", type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45, addCoef.col = "black",
         number.cex = 0.8, tl.cex = 1.0)
```



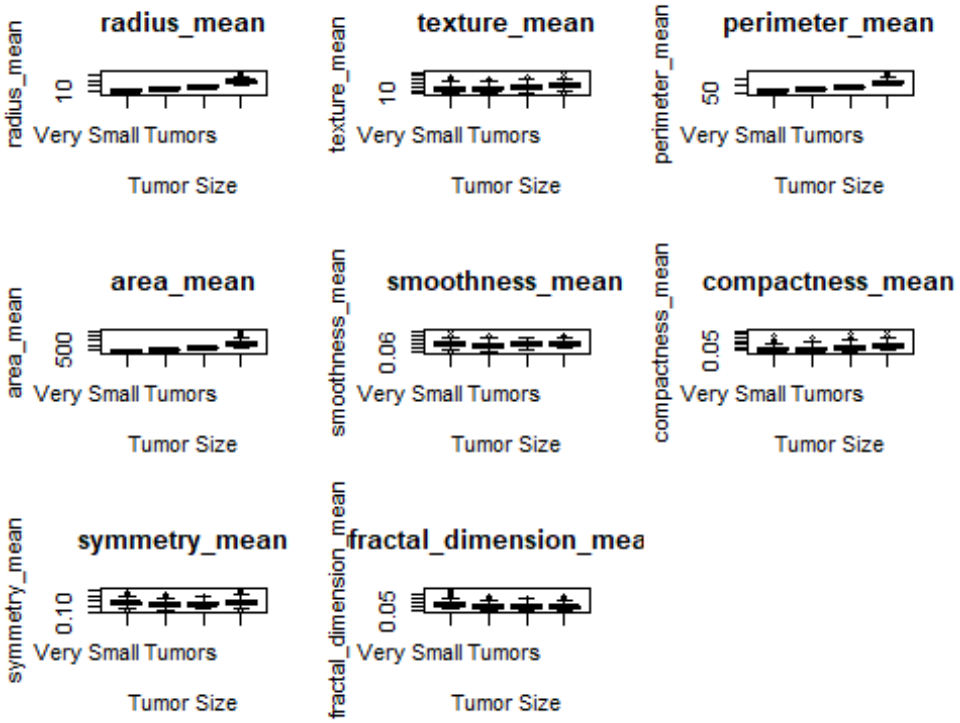

```r
# The correlation coefficients provided in the matrix indicate the strength a
nd direction of the linear relationship between different features and tumor
diagnosis. For instance, the radius_mean, perimeter_mean, and area_mean show
strong positive correlations with tumor diagnosis, with coefficients of appro
ximately 0.74, 0.73, and 0.74, respectively, indicating that larger values of
these features tend to be associated with malignant tumors. Similarly, compac
tness_mean and smoothness_mean also show moderate positive correlations with
tumor diagnosis, suggesting that higher compactness and smoothness values are
associated with malignant tumors. Conversely, fractal_dimension_mean shows a
moderate negative correlation with tumor diagnosis, implying that lower value
s of fractal dimension tend to be associated with malignant tumors.

# Comparison of tumor size by diagnosis using box plot
boxplot(tumor_size_numerical ~ diagnosis, data = Breastcancer_data2,
        main = "Tumor Size by Diagnosis", xlab = "Diagnosis", ylab = "Tumor S
ize")
```



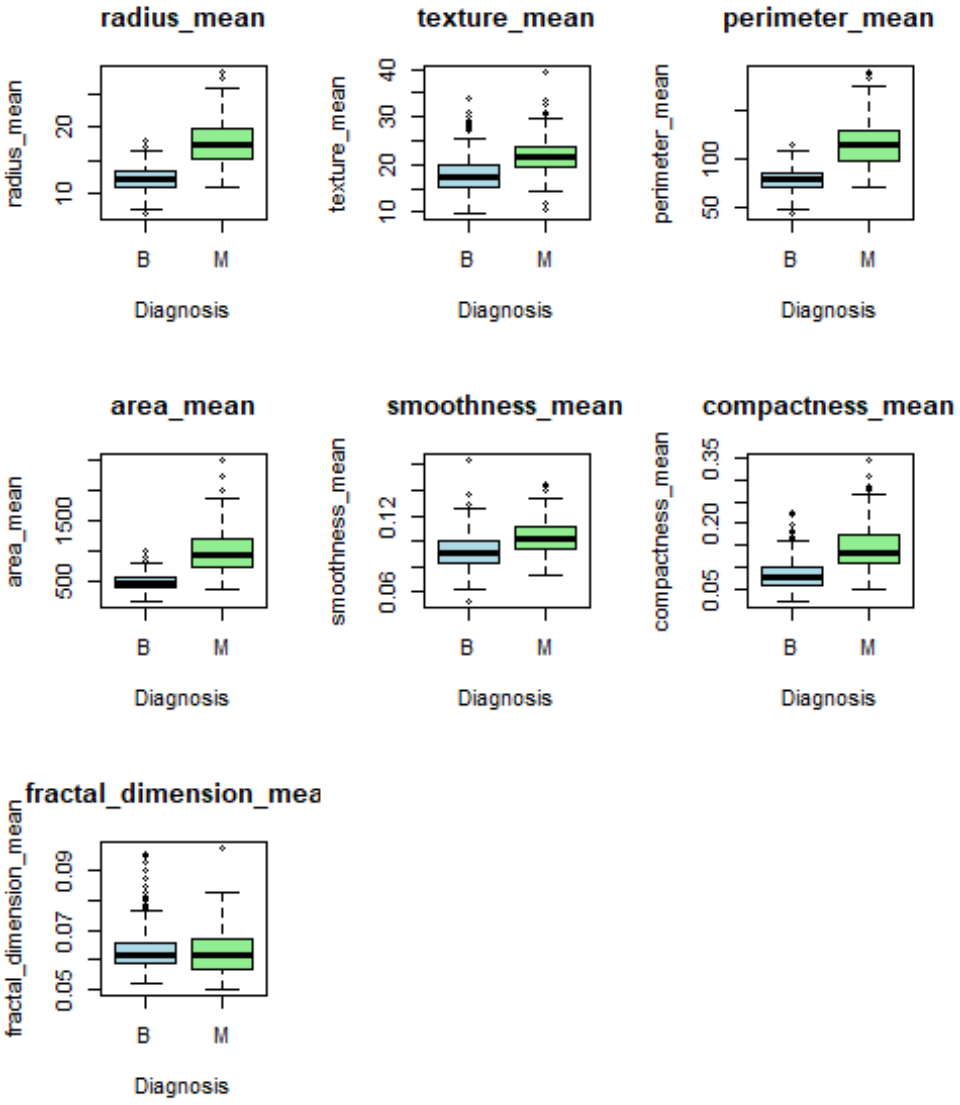

```r
# Create a scatter plot
ggplot(Breastcancer_data2, aes(x = diagnosis, y = tumor_size_numerical, color
= diagnosis)) +
  geom_point(position = position_jitter(width = 0.2), alpha = 0.5) +
  labs(x = "Diagnosis", y = "Tumor Size", title = "Comparison of Tumor Size b
y Diagnosis") +
  theme_minimal()
```




```r
# Comparison of morphological characteristics by tumor size using grouped box plots
par(mfrow=c(3, 3))  # Setting up the layout for multiple plots
for (var in c("radius_mean", "texture_mean", "perimeter_mean", "area_mean",
              "smoothness_mean", "compactness_mean", "symmetry_mean",
              "fractal_dimension_mean")) {
  boxplot(get(var) ~ tumor_size, data = Breastcancer_data2, main = var,
          xlab = "Tumor Size", ylab = var, col = c("lightblue", "lightgreen",
"lightyellow", "lightcoral"))
}
```




```r
# Comparison of multiple variables by diagnosis using grouped box plots
par(mfrow=c(2, 3))
for (var in c("radius_mean", "texture_mean", "perimeter_mean", "area_mean",
              "smoothness_mean", "compactness_mean","fractal_dimension_mean")
) {
  boxplot(get(var) ~ diagnosis, data = Breastcancer_data2, main = var,
          xlab = "Diagnosis", ylab = var, col = c("lightblue", "lightgreen"))
}
```




```
# Replace 'M' with 1 and 'D' with 0
Breastcancer_data2$diagnosis <- ifelse(Breastcancer_data2$diagnosis == "M", 1
,
                                       ifelse(Breastcancer_data2$diagnosis ==
```

```
"B", 0, Breastcancer_data2$diagnosis))

# If you want to convert the column to numeric type
Breastcancer_data2$diagnosis <- as.numeric(Breastcancer_data2$diagnosis)

# Mann-Whitney U test for radius_mean
mann_whitney_radius <- wilcox.test(radius_mean ~ diagnosis, data = Breastcanc
er_data2)
print("Mann-Whitney U test for radius_mean:")
```

```
## [1] "Mann-Whitney U test for radius_mean:"
```

```
print(mann_whitney_radius)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  radius_mean by diagnosis
## W = 4729, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
# Association result for radius_mean
association_radius <- ifelse(mann_whitney_radius$p.value < 0.05, "Significant
association", "No significant association")
print(paste("Association result for radius_mean:", association_radius))
```

```
## [1] "Association result for radius_mean: Significant association"
```

```
# Mann-Whitney U test for texture_mean
mann_whitney_texture <- wilcox.test(texture_mean ~ diagnosis, data = Breastca
ncer_data2)
print("Mann-Whitney U test for texture_mean:")
```

```
## [1] "Mann-Whitney U test for texture_mean:"
```

```
print(mann_whitney_texture)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  texture_mean by diagnosis
## W = 16967, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
# Association result for texture_mean
association_texture <- ifelse(mann_whitney_texture$p.value < 0.05, "Significa
nt association", "No significant association")
print(paste("Association result for texture_mean:", association_texture))
```

```
## [1] "Association result for texture_mean: Significant association"
```

```
# Mann-Whitney U test for perimeter_mean
mann_whitney_perimeter <- wilcox.test(perimeter_mean ~ diagnosis, data = Brea
```

```
stcancer_data2)
print("Mann-Whitney U test for perimeter_mean:")
```

```
## [1] "Mann-Whitney U test for perimeter_mean:"
```

```
print(mann_whitney_perimeter)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  perimeter_mean by diagnosis
## W = 4019, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
# Association result for perimeter_mean
association_perimeter <- ifelse(mann_whitney_perimeter$p.value < 0.05, "Signi
ficant association", "No significant association")
print(paste("Association result for perimeter_mean:", association_perimeter))
```

```
## [1] "Association result for perimeter_mean: Significant association"
```

```
# Mann-Whitney U test for area_mean
mann_whitney_area <- wilcox.test(area_mean ~ diagnosis, data = Breastcancer_d
ata2)
print("Mann-Whitney U test for area_mean:")
```

```
## [1] "Mann-Whitney U test for area_mean:"
```

```
print(mann_whitney_area)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  area_mean by diagnosis
## W = 4668.5, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
# Association result for area_mean
association_area <- ifelse(mann_whitney_area$p.value < 0.05, "Significant ass
ociation", "No significant association")
print(paste("Association result for area_mean:", association_area))
```

```
## [1] "Association result for area_mean: Significant association"
```

```
# Mann-Whitney U test for smoothness_mean
mann_whitney_smoothness <- wilcox.test(smoothness_mean ~ diagnosis, data = Br
eastcancer_data2)
print("Mann-Whitney U test for smoothness_mean:")
```

```
## [1] "Mann-Whitney U test for smoothness_mean:"
```

```
print(mann_whitney_smoothness)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  smoothness_mean by diagnosis
## W = 21037, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
# Association result for smoothness_mean
association_smoothness <- ifelse(mann_whitney_smoothness$p.value < 0.05, "Sig
nificant association", "No significant association")
print(paste("Association result for smoothness_mean:", association_smoothness
))
```

```
## [1] "Association result for smoothness_mean: Significant association"
```

```
# Mann-Whitney U test for compactness_mean
mann_whitney_compactness <- wilcox.test(compactness_mean ~ diagnosis, data =
Breastcancer_data2)
print("Mann-Whitney U test for compactness_mean:")
```

```
## [1] "Mann-Whitney U test for compactness_mean:"
```

```
print(mann_whitney_compactness)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  compactness_mean by diagnosis
## W = 10310, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
# Association result for compactness_mean
association_compactness <- ifelse(mann_whitney_compactness$p.value < 0.05, "S
ignificant association", "No significant association")
print(paste("Association result for compactness_mean:", association_compactne
ss))
```

```
## [1] "Association result for compactness_mean: Significant association"
```

```
# Mann-Whitney U test for symmetry_mean
mann_whitney_symmetry <- wilcox.test(symmetry_mean ~ diagnosis, data = Breast
cancer_data2)
print("Mann-Whitney U test for symmetry_mean:")
```

```
## [1] "Mann-Whitney U test for symmetry_mean:"
```

```
print(mann_whitney_symmetry)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  symmetry_mean by diagnosis
```

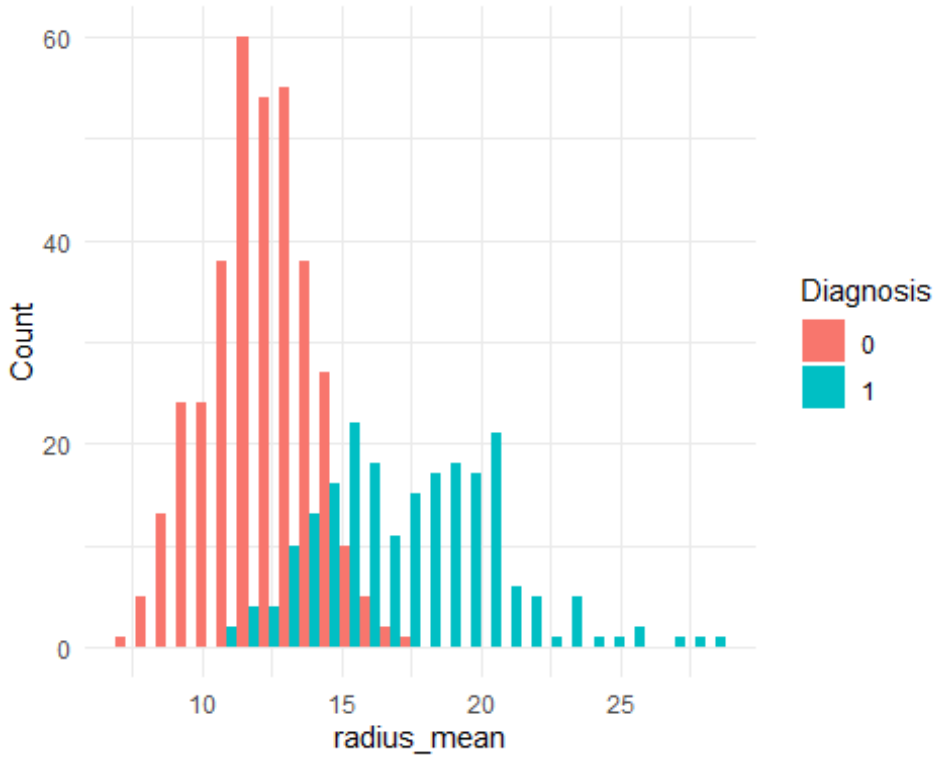
```
## W = 22814, p-value = 2.268e-15
## alternative hypothesis: true location shift is not equal to 0

# Association result for symmetry_mean
association_symmetry <- ifelse(mann_whitney_symmetry$p.value < 0.05, "Signifi
cant association", "No significant association")
print(paste("Association result for symmetry_mean:", association_symmetry))

## [1] "Association result for symmetry_mean: Significant association"

# Mann-Whitney U test for fractal_dimension_mean
mann_whitney_fractal_dimension <- wilcox.test(fractal_dimension_mean ~ diagno
sis, data = Breastcancer_data2)
print("Mann-Whitney U test for fractal_dimension_mean:")

## [1] "Mann-Whitney U test for fractal_dimension_mean:"

print(mann_whitney_fractal_dimension)

##
##  Wilcoxon rank sum test with continuity correction
##
## data:  fractal_dimension_mean by diagnosis
## W = 39013, p-value = 0.5372
## alternative hypothesis: true location shift is not equal to 0

# Association result for fractal_dimension_mean
association_fractal_dimension <- ifelse(mann_whitney_fractal_dimension$p.valu
e < 0.05, "Significant association", "No significant association")
print(paste("Association result for fractal_dimension_mean:", association_fra
ctal_dimension))

## [1] "Association result for fractal_dimension_mean: No significant associa
tion"

# Get unique values in the diagnosis column
diagnosis_levels <- unique(Breastcancer_data2$diagnosis)

# Perform chi-square test for each diagnosis variable
for (level in diagnosis_levels) {
  # Subset the data for the current diagnosis level
  subset_data <- Breastcancer_data2[Breastcancer_data2$diagnosis == level, ]

  # Create a contingency table for tumor_size and the current diagnosis level
  cont_table <- table(subset_data$tumor_size)

  # Perform chi-square test
  chi_sq_test <- chisq.test(cont_table)

  # Print the results
  print(paste("Chi-square test for", level, ":", sep = " "))
```

```r
  print(chi_sq_test)
}
```

```
## [1] "Chi-square test for 1 :"
##
##  Chi-squared test for given probabilities
##
## data:  cont_table
## X-squared = 206.53, df = 3, p-value < 2.2e-16
##
## [1] "Chi-square test for 0 :"
##
##  Chi-squared test for given probabilities
##
## data:  cont_table
## X-squared = 123.77, df = 3, p-value < 2.2e-16
```

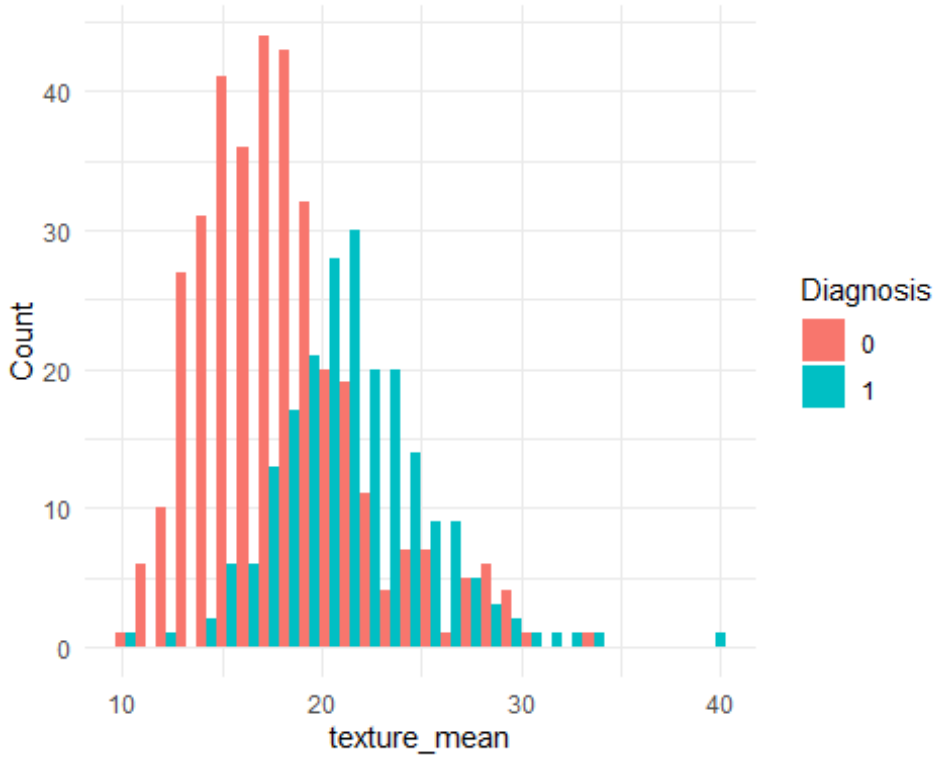
```r
# Load the necessary libraries
library(ggplot2)
library(dplyr)

# Prepare the data
Breastcancer_data2$diagnosis <- as.factor(Breastcancer_data2$diagnosis)

# Create a function to plot the association between diagnosis and a given fea
ture
plot_feature_association <- function(feature_name) {
  ggplot(Breastcancer_data2, aes(x = !!sym(feature_name), fill = diagnosis))
+
    geom_histogram(position = "dodge") +
    labs(x = feature_name, y = "Count", fill = "Diagnosis") +
    theme_minimal()
}

# Plot the association for each feature
plot_feature_association("radius_mean")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



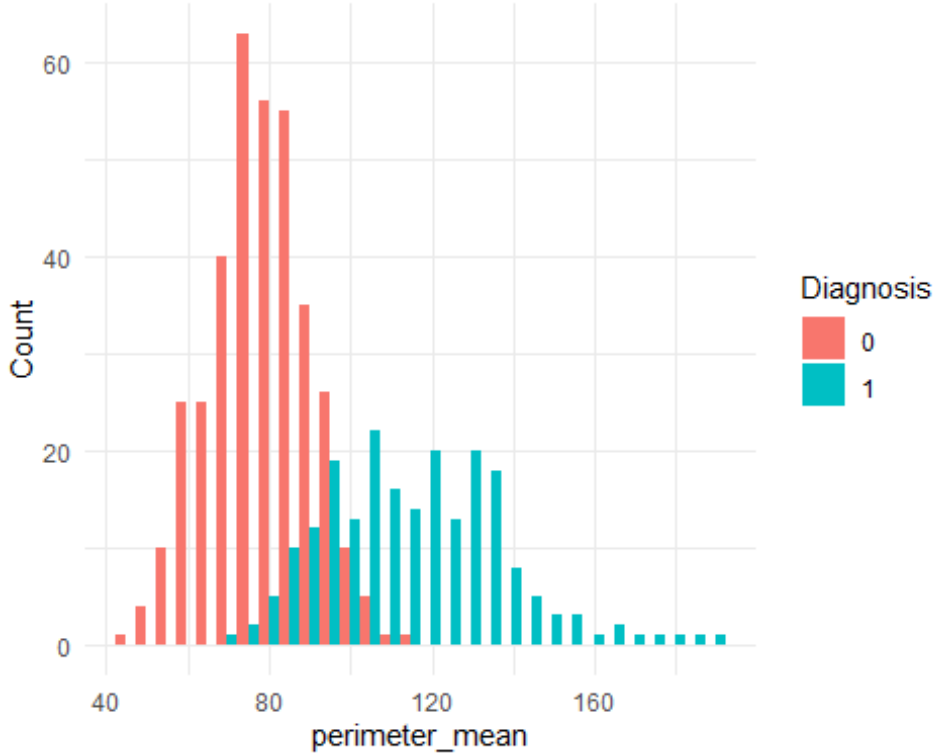

```
plot_feature_association("texture_mean")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



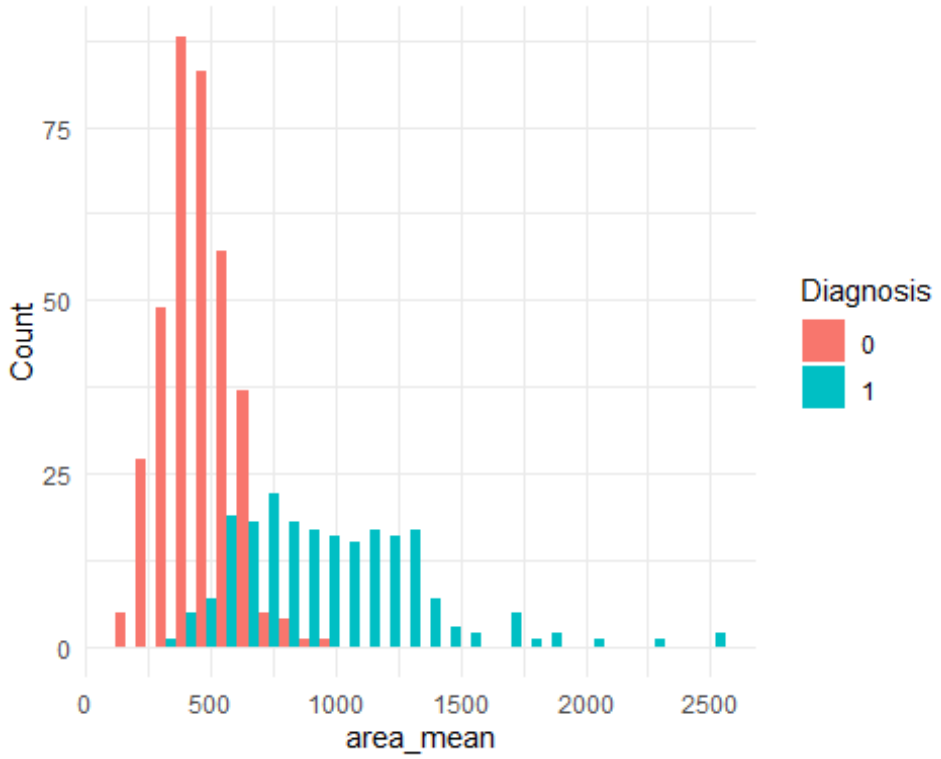
```
plot_feature_association("perimeter_mean")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



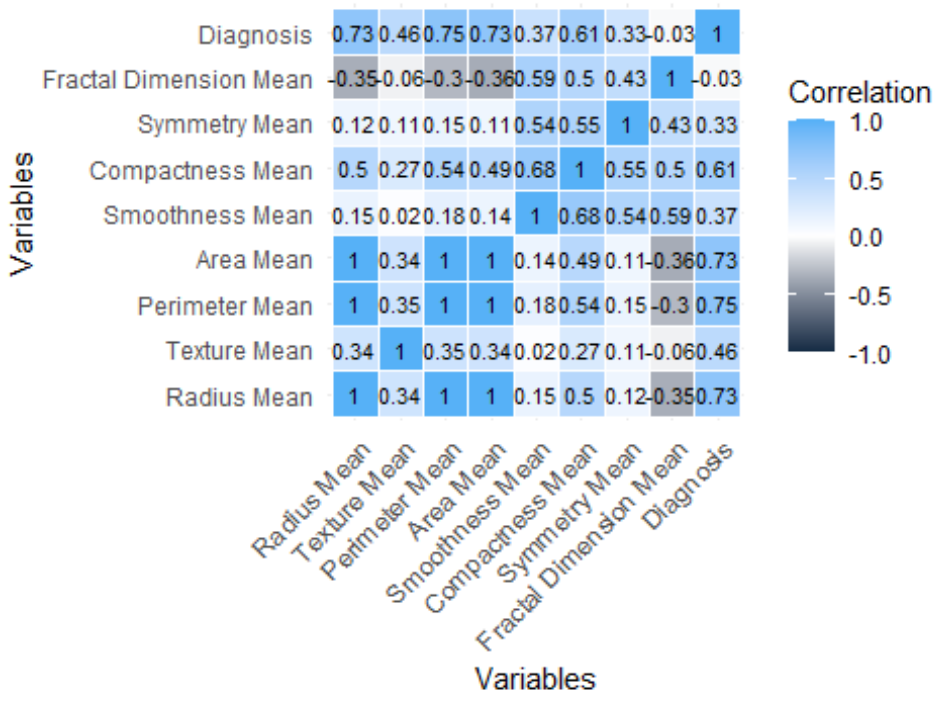

```
plot_feature_association("area_mean")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



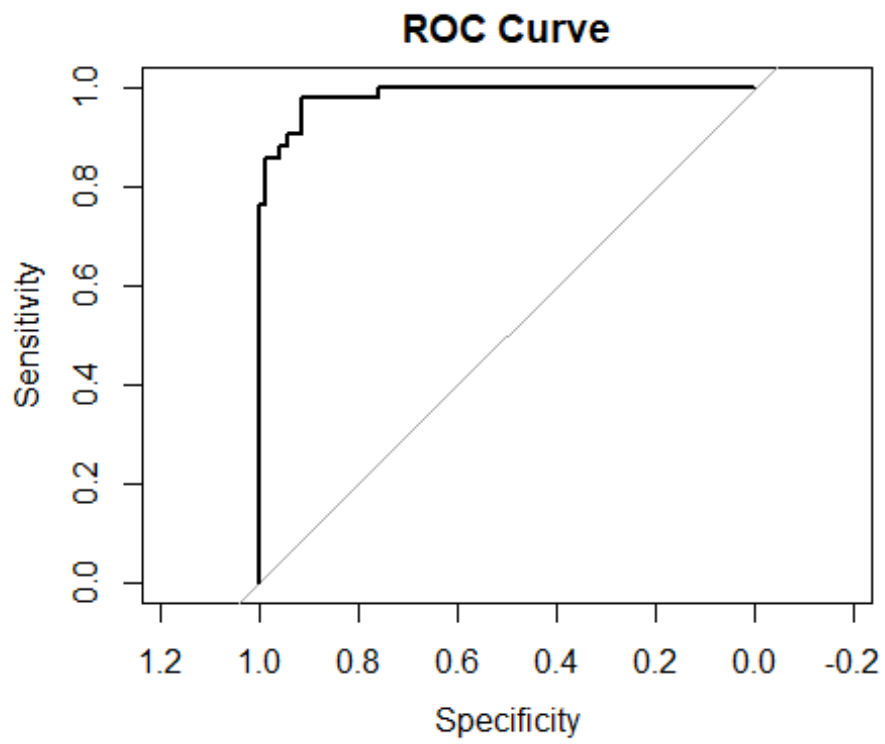

```r
# Convert "diagnosis" variable to numeric
Breastcancer_data2$diagnosis <- as.numeric(as.factor(Breastcancer_data2$diagn
osis))

# Specify variables for correlation analysis (including "diagnosis")
correlation_variables <- c("radius_mean", "texture_mean", "perimeter_mean",
                           "area_mean", "smoothness_mean", "compactness_mean
",
                           "symmetry_mean", "fractal_dimension_mean", "diagn
osis")

# Calculate Spearman correlation matrix
correlation_matrix_spearman <- cor(Breastcancer_data2[, correlation_variables
], method = "spearman")

# Convert correlation matrix to data frame
correlation_df_spearman <- as.data.frame(as.table(correlation_matrix_spearman
))

# Add variable names
correlation_df_spearman$Var1 <- factor(correlation_df_spearman$Var1, levels =
correlation_variables)
correlation_df_spearman$Var2 <- factor(correlation_df_spearman$Var2, levels =
correlation_variables)

# Plot heatmap with Spearman correlation values displayed
```

```r
library(ggplot2)

ggplot(correlation_df_spearman, aes(Var1, Var2, fill = Freq, label = round(Freq, 2))) +
  geom_tile(color = "white") +
  geom_text(color = "black", size = 3) +
  scale_fill_gradient2(low = "#132B43", high = "#56B1F7", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 10, hjust = 1)) +
  coord_fixed() +
  labs(x = "Variables", y = "Variables") +  # Adjust axis labels
  scale_x_discrete(labels = c("Radius Mean", "Texture Mean", "Perimeter Mean",
                              "Area Mean", "Smoothness Mean", "Compactness Mean",
                              "Symmetry Mean", "Fractal Dimension Mean", "Diagnosis")) +  # Include variable names
 scale_y_discrete(labels = c("Radius Mean", "Texture Mean", "Perimeter Mean",
                              "Area Mean", "Smoothness Mean", "Compactness Mean",
                              "Symmetry Mean", "Fractal Dimension Mean", "Diagnosis"))  # Include variable names
```



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

## Warning: package 'caret' was built under R version 4.3.3

## Loading required package: lattice

library(dplyr)
library(e1071)

## Warning: package 'e1071' was built under R version 4.3.3

##
## Attaching package: 'e1071'

## The following object is masked _by_ '.GlobalEnv':
##
##     skewness

# Prepare the data
Breastcancer_data2$diagnosis <- as.factor(Breastcancer_data2$diagnosis)

# Split the data into training and test sets
set.seed(123)
train_index <- createDataPartition(Breastcancer_data2$diagnosis, p = 0.8, lis
t = FALSE)
train_data <- Breastcancer_data2[train_index, ]
test_data <- Breastcancer_data2[-train_index, ]

# Train the SVM model for classification
svm_model <- svm(diagnosis ~ ., data = train_data, type = "C-classification",
kernel = "radial")

# Evaluate the model on the test set
predictions <- predict(svm_model, newdata = test_data)
accuracy <- sum(predictions == test_data$diagnosis) / nrow(test_data)
print(paste("Accuracy:", accuracy))

## [1] "Accuracy: 0.920353982300885"

# Plot the association between diagnosis and each feature
for (feature in names(Breastcancer_data2)[2:ncol(Breastcancer_data2)]) {
  plot_feature_association(feature)
}

# Function to plot the association between diagnosis and a feature
plot_feature_association <- function(feature_name) {
  ggplot(Breastcancer_data2, aes_string(x = feature_name, fill = "diagnosis")
) +
    geom_histogram(position = "dodge", binwidth = 1) +
```

```
    labs(x = feature_name, y = "Count", fill = "Diagnosis") +
    theme_minimal()
}

library(pROC)

## Warning: package 'pROC' was built under R version 4.3.3

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(caret)
library(ggplot2)
library(e1071)

# Train the SVM model for classification with probability estimates
svm_model <- svm(diagnosis ~ ., data = train_data, type = "C-classification",
kernel = "radial", probability = TRUE)

# Predict probabilities
probabilities <- predict(svm_model, newdata = test_data, probability = TRUE)

# Extract probabilities for the positive class
# Assuming the positive class is the second factor level, adjust if necessary
probs <- attr(probabilities, "probabilities")[,2]

# ROC curve using the pROC package
roc_obj <- roc(test_data$diagnosis, probs)

## Setting levels: control = 1, case = 2

## Setting direction: controls > cases

plot(roc_obj, main = "ROC Curve")
```




```
print(paste("AUC:", auc(roc_obj)))
```

```
## [1] "AUC: 0.984574111334675"
```

```
predictions <- predict(svm_model, newdata = test_data)

# Generate the confusion matrix
conf_matrix <- confusionMatrix(as.factor(predictions), as.factor(test_data$di
agnosis))

# Print the confusion matrix
print(conf_matrix$table)
```

```
##           Reference
## Prediction  1  2
##          1 66  4
##          2  5 38
```

```
# print the summary of the confusion matrix
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 66  4
##          2  5 38
```

```
##
##                Accuracy : 0.9204
##                  95% CI : (0.8542, 0.9629)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 9.656e-13
##
##                   Kappa : 0.8303
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9296
##             Specificity : 0.9048
##          Pos Pred Value : 0.9429
##          Neg Pred Value : 0.8837
##              Prevalence : 0.6283
##          Detection Rate : 0.5841
##    Detection Prevalence : 0.6195
##       Balanced Accuracy : 0.9172
##
##        'Positive' Class : 1
##
```

```
# Convert diagnosis to a factor if it's not already
Breastcancer_data2$diagnosis <- as.factor(Breastcancer_data2$diagnosis)

# Split the data into training and test sets
set.seed(123)  # for reproducibility
train_index <- createDataPartition(Breastcancer_data2$diagnosis, p = 0.8, lis
t = FALSE)
train_data <- Breastcancer_data2[train_index, ]
test_data <- Breastcancer_data2[-train_index, ]


logistic_model <- glm(diagnosis ~ ., data = train_data, family = binomial())

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# Predicting probabilities
probabilities <- predict(logistic_model, newdata = test_data, type = "respons
e")

# Predicting class labels
predicted_classes <- ifelse(probabilities > 0.5, levels(train_data$diagnosis)
[2], levels(train_data$diagnosis)[1])

conf_matrix <- confusionMatrix(as.factor(predicted_classes), test_data$diagno
sis)
print(conf_matrix$table)

##           Reference
## Prediction  1  2
```

```
##          1 69  2
##          2  2 40
```

```
print(conf_matrix$overall['Accuracy'])
```

```
##  Accuracy
## 0.9646018
```

```
# print the summary of the confusion matrix
print(conf_matrix)
```

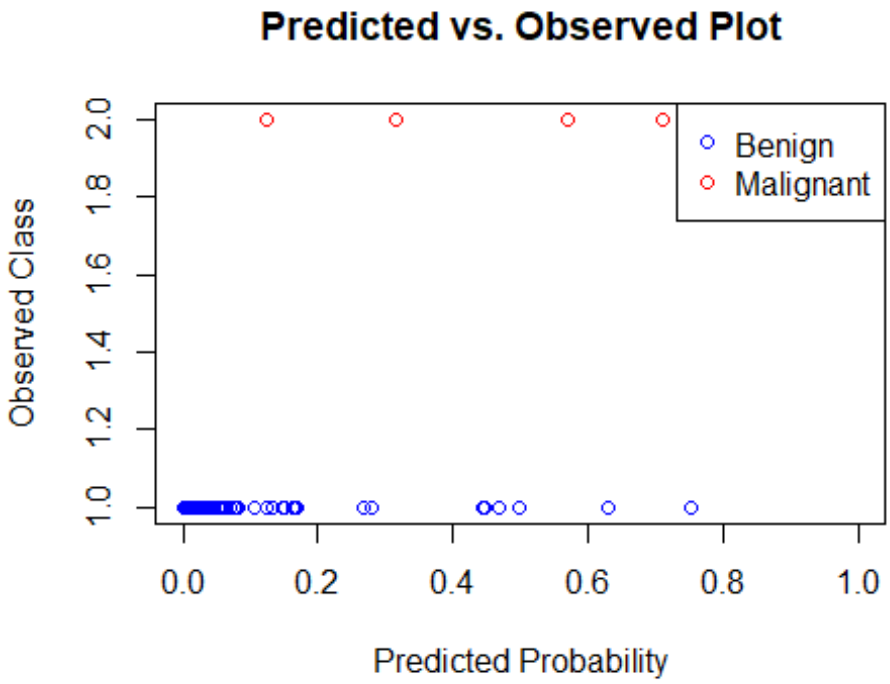
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 69  2
##          2  2 40
##
##                Accuracy : 0.9646
##                  95% CI : (0.9118, 0.9903)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9242
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9718
##             Specificity : 0.9524
##          Pos Pred Value : 0.9718
##          Neg Pred Value : 0.9524
##              Prevalence : 0.6283
##          Detection Rate : 0.6106
##    Detection Prevalence : 0.6283
##       Balanced Accuracy : 0.9621
##
##        'Positive' Class : 1
##
```

```
# Get summary of the logistic regression model
summary_glm <- summary(logistic_model)
print(summary_glm)
```

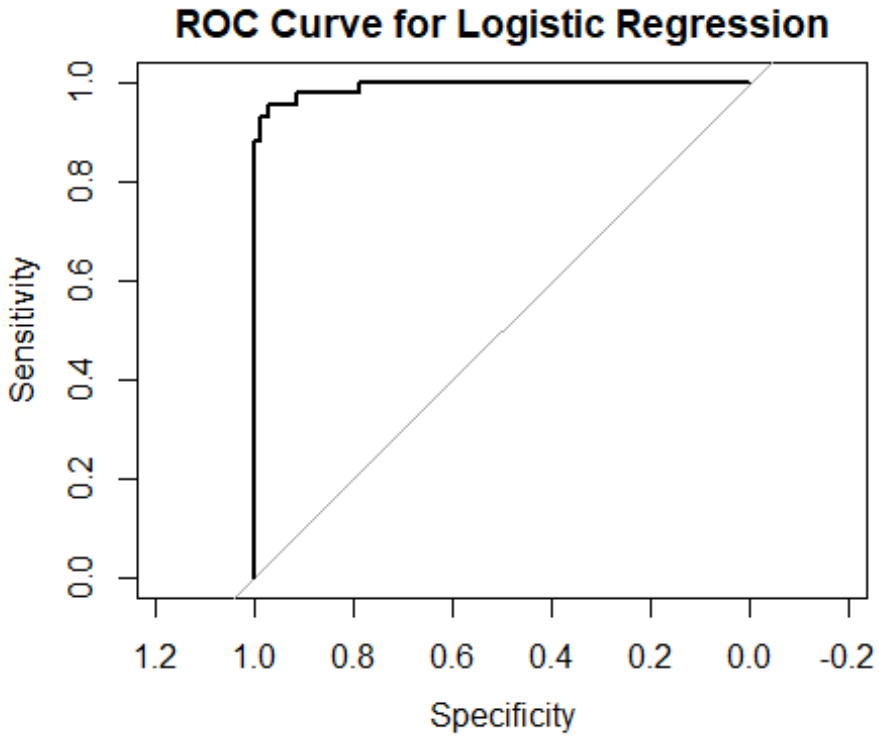
```
##
## Call:
## glm(formula = diagnosis ~ ., family = binomial(), data = train_data)
##
## Coefficients: (1 not defined because of singularities)
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              1.100e+01  1.383e+01   0.795 0.426644
## id                      -4.446e-09  4.914e-09  -0.905 0.365551
```

```
## radius_mean              -6.822e+00  3.781e+00  -1.804 0.071166 .
## texture_mean              3.694e-01  6.820e-02   5.416 6.09e-08 ***
## perimeter_mean            1.221e-01  5.360e-01   0.228 0.819790
## area_mean                 8.439e-02  2.153e-02   3.919 8.88e-05 ***
## smoothness_mean           9.816e+01  2.632e+01   3.729 0.000192 ***
## compactness_mean          2.320e+01  2.367e+01   0.980 0.326943
## symmetry_mean             3.040e+01  1.341e+01   2.267 0.023377 *
## fractal_dimension_mean   -9.267e+01  9.473e+01  -0.978 0.327981
## tumor_sizeSmall Tumors    2.383e+00  1.421e+00   1.677 0.093623 .
## tumor_sizeMedium Tumors   2.596e+00  1.784e+00   1.455 0.145631
## tumor_sizeLarge Tumors    6.610e-01  2.512e+00   0.263 0.792411
## tumor_size_numerical             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 602.31  on 455  degrees of freedom
## Residual deviance: 127.51  on 443  degrees of freedom
## AIC: 153.51
##
## Number of Fisher Scoring iterations: 9

#Interpretation of the coefficients suggests that texture_mean, area_mean, an
d smoothness_mean are statistically significant predictors of tumor diagnosis
, as their p-values are less than the conventional threshold of 0.05. This im
plies that changes in these features are associated with changes in the likel
ihood of tumor diagnosis. On the other hand, radius_mean and symmetry_mean al
so show some significance with p-values close to the threshold, indicating th
ey might have some predictive value but require further investigation. Howeve
r, other variables such as compactness_mean, perimeter_mean, and fractal_dime
nsion_mean do not appear to be statistically significant predictors in this m
odel.

# View summary of the logistic regression model
summary(logistic_model)

##
## Call:
## glm(formula = diagnosis ~ ., family = binomial(), data = train_data)
##
## Coefficients: (1 not defined because of singularities)
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               1.100e+01  1.383e+01   0.795 0.426644
## id                       -4.446e-09  4.914e-09  -0.905 0.365551
## radius_mean              -6.822e+00  3.781e+00  -1.804 0.071166 .
## texture_mean              3.694e-01  6.820e-02   5.416 6.09e-08 ***
## perimeter_mean            1.221e-01  5.360e-01   0.228 0.819790
## area_mean                 8.439e-02  2.153e-02   3.919 8.88e-05 ***
## smoothness_mean           9.816e+01  2.632e+01   3.729 0.000192 ***
```

```
## compactness_mean          2.320e+01  2.367e+01   0.980 0.326943
## symmetry_mean             3.040e+01  1.341e+01   2.267 0.023377 *
## fractal_dimension_mean   -9.267e+01  9.473e+01  -0.978 0.327981
## tumor_sizeSmall Tumors    2.383e+00  1.421e+00   1.677 0.093623 .
## tumor_sizeMedium Tumors   2.596e+00  1.784e+00   1.455 0.145631
## tumor_sizeLarge Tumors    6.610e-01  2.512e+00   0.263 0.792411
## tumor_size_numerical             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 602.31  on 455  degrees of freedom
## Residual deviance: 127.51  on 443  degrees of freedom
## AIC: 153.51
##
## Number of Fisher Scoring iterations: 9

# Predicting probabilities
probabilities <- predict(logistic_model, newdata = test_data, type = "respons
e")

# Creating a data frame with predicted probabilities and observed classes
predicted_observed <- data.frame(Probability = probabilities, Observed = test
_data$diagnosis)

# Plot predicted vs. observed
plot(predicted_observed$Probability, predicted_observed$Observed,
     main = "Predicted vs. Observed Plot", xlab = "Predicted Probability", yl
ab = "Observed Class",
     col = ifelse(predicted_observed$Observed == "2", "red", "blue"))
legend("topright", legend = c("Benign", "Malignant"), col = c("blue", "red"),
pch = 1)
```




```r
# Load necessary library
library(pROC)

# Create ROC curve object
roc_obj <- roc(test_data$diagnosis, probabilities)

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

# Plot ROC curve
plot(roc_obj, main = "ROC Curve for Logistic Regression")
```




```r
# Print AUC
print(paste("Area Under Curve (AUC):", auc(roc_obj)))
```

```
## [1] "Area Under Curve (AUC): 0.991616364855802"
```

```r
conf_matrix <- confusionMatrix(as.factor(predicted_classes), test_data$diagno
sis)
print(conf_matrix$table)
```

```
##           Reference
## Prediction  1  2
##          1 69  2
##          2  2 40
```

```r
print(conf_matrix$overall['Accuracy'])
```

```
##  Accuracy
## 0.9646018
```

```r
# print the summary of the confusion matrix
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 69  2
##          2  2 40
```

```
##
##                Accuracy : 0.9646
##                  95% CI : (0.9118, 0.9903)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9242
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9718
##             Specificity : 0.9524
##          Pos Pred Value : 0.9718
##          Neg Pred Value : 0.9524
##              Prevalence : 0.6283
##          Detection Rate : 0.6106
##    Detection Prevalence : 0.6283
##       Balanced Accuracy : 0.9621
##
##        'Positive' Class : 1
##
```

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

# Sample 10 random observations from the test set
sample_test_data <- test_data[sample(nrow(test_data), 10), ]

# Predict with Logistic Regression
logistic_predictions <- predict(logistic_model, newdata = sample_test_data, t
ype = "response")
logistic_predicted_classes <- ifelse(logistic_predictions > 0.5, levels(train
_data$diagnosis)[2], levels(train_data$diagnosis)[1])

# Combine actual and predicted into a data frame for easy comparison
comparison_data <- data.frame(
  Actual = sample_test_data$diagnosis,
  Predicted_Logistic = logistic_predicted_classes
)

# Print results
print("Model Predictions Comparison:")
```

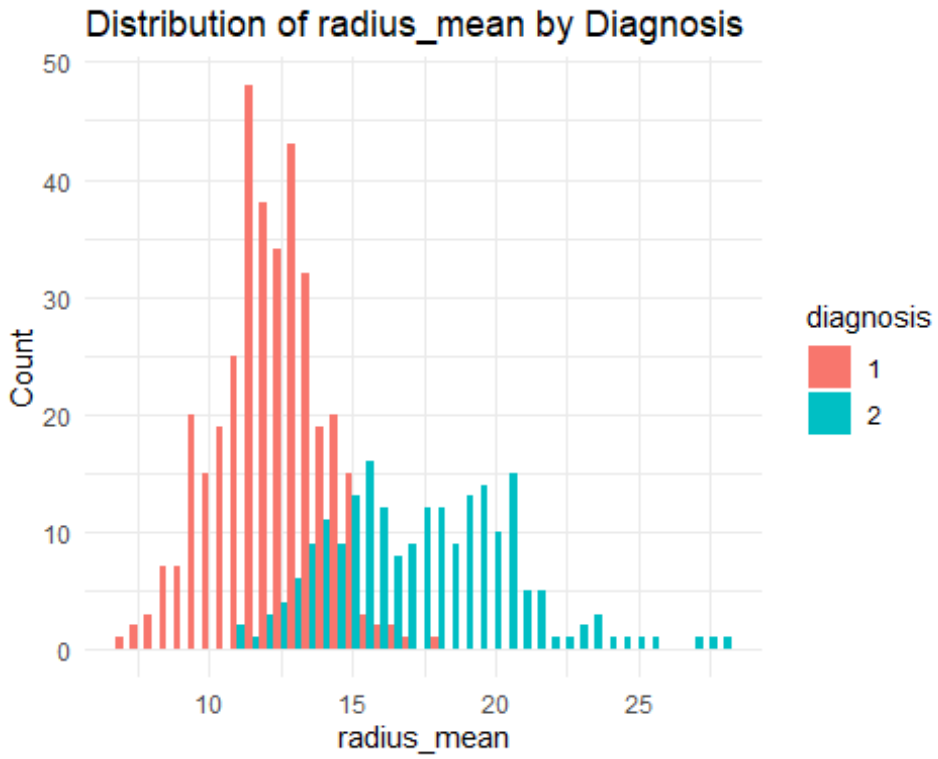
```
## [1] "Model Predictions Comparison:"
```

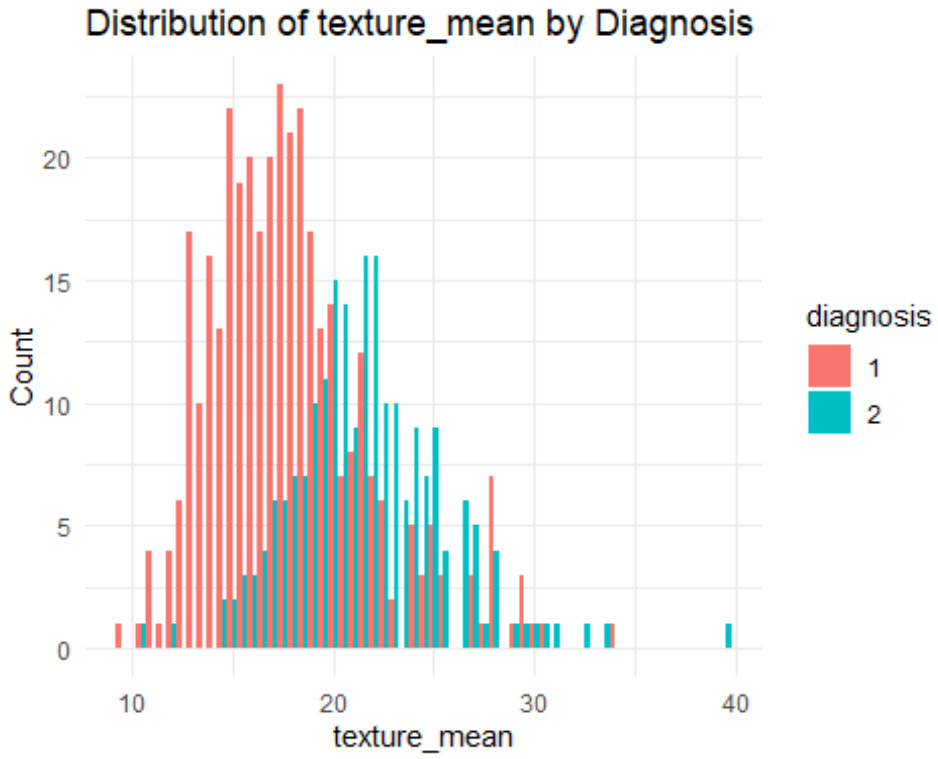
```r
print(comparison_data)
```

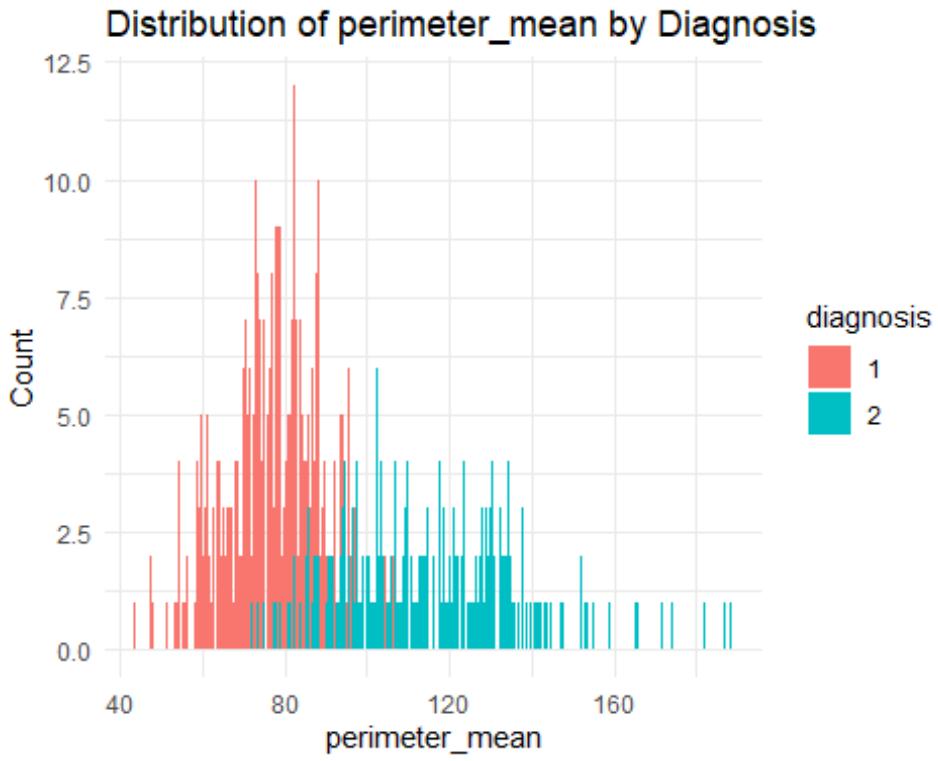
```
##     Actual Predicted_Logistic
## 151      1                  1
## 395      1                  1
```

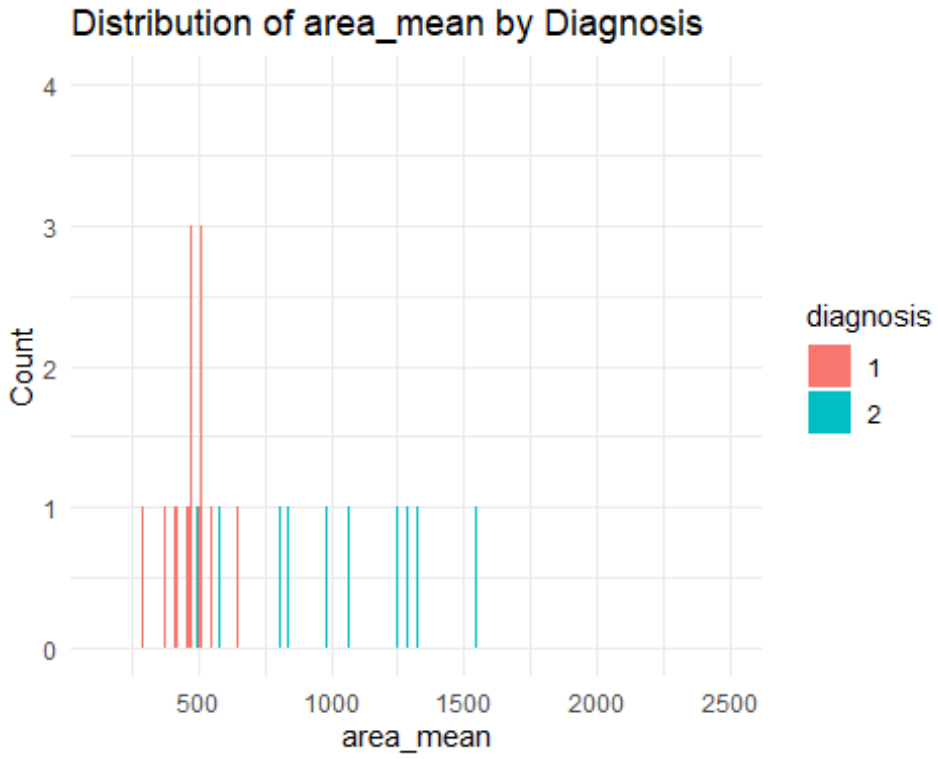
```
## 269      1                   1
## 69       1                   1
## 336      2                   2
## 233      1                   1
## 267      1                   1
## 240      2                   2
## 510      2                   2
## 551      1                   1
```

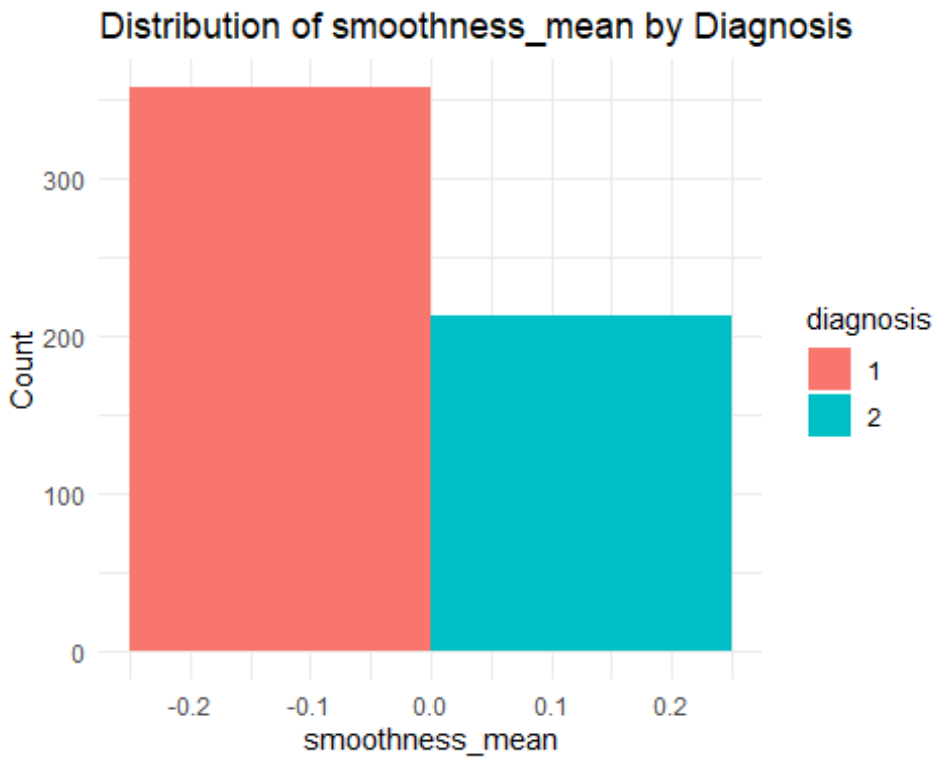
```r
library(ggplot2)

# Assuming 'Breastcancer_data2' is your dataset
features <- c("radius_mean", "texture_mean", "perimeter_mean", "area_mean", "
smoothness_mean")

# Loop through features and create histograms
for (feature in features) {
  print(ggplot(Breastcancer_data2, aes_string(x = feature, fill = "diagnosis"
)) +
    geom_histogram(position = "dodge", binwidth = 0.5) +
    labs(title = paste("Distribution of", feature, "by Diagnosis"),
         x = feature,
         y = "Count") +
    theme_minimal())
}
```


Distribution of radius_mean by Diagnosis
Count
radius_mean
diagnosis
1
2

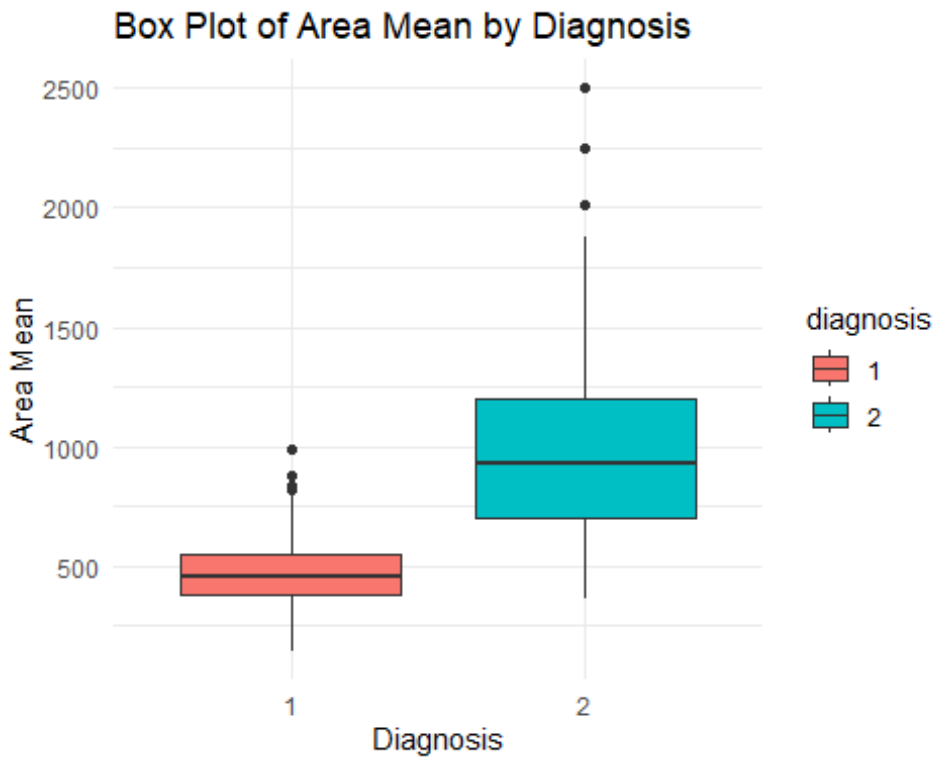

Distribution of texture_mean by Diagnosis
Count
texture_mean
diagnosis
1
2


Distribution of perimeter_mean by Diagnosis
12.5
10.0
7.5
5.0
2.5
0.0
Count
40
80
120
160
perimeter_mean
diagnosis
1
2

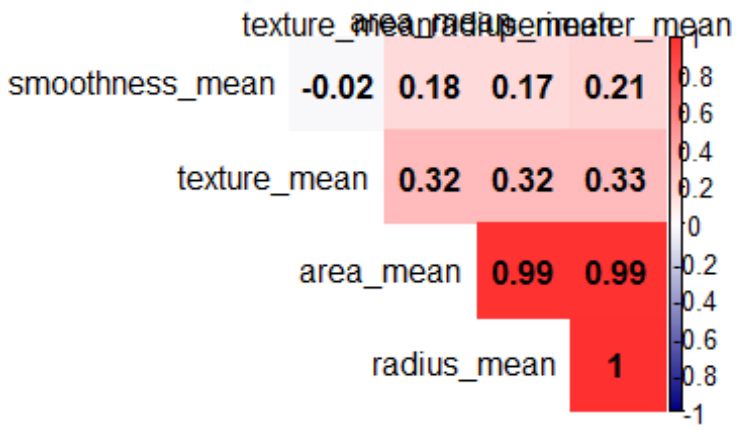

Distribution of area_mean by Diagnosis
4
3
2
1
0
Count
500
1000
1500
2000
2500
area_mean
diagnosis
1
2



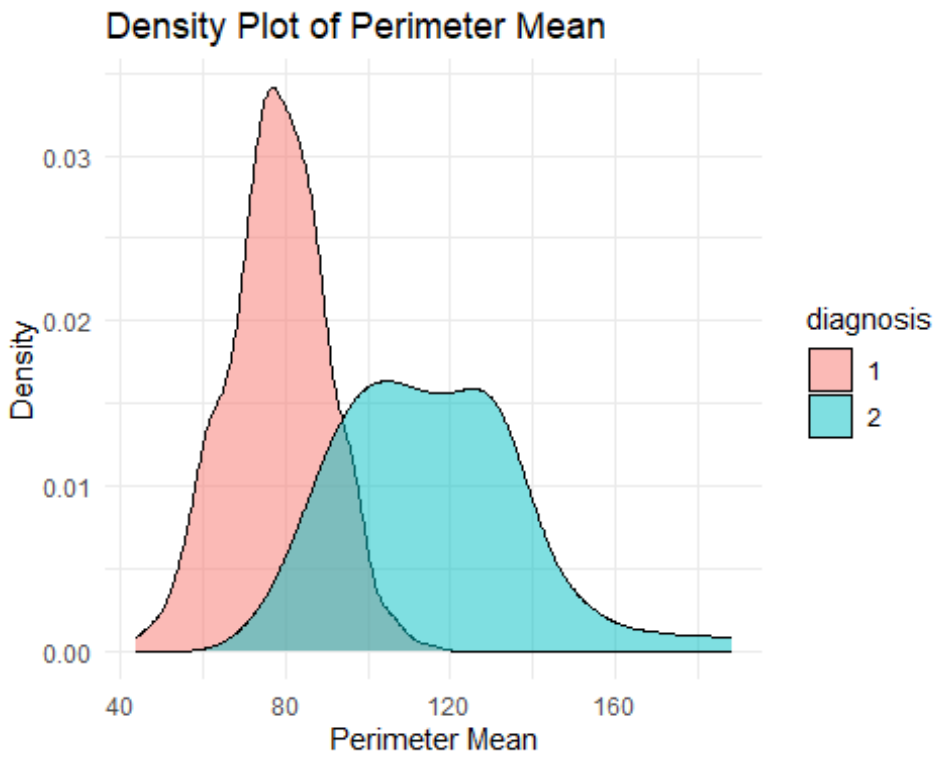

```r
ggplot(Breastcancer_data2, aes(x = diagnosis, y = area_mean, fill = diagnosis
)) +
  geom_boxplot() +
  labs(title = "Box Plot of Area Mean by Diagnosis",
       x = "Diagnosis",
       y = "Area Mean") +
  theme_minimal()
```




```r
library(corrplot)

# Compute the correlation matrix
cor_matrix <- cor(Breastcancer_data2[, c("radius_mean", "texture_mean", "perimeter_mean", "area_mean", "smoothness_mean")])

# Plot the correlation matrix with a straightforward and safe color scheme
corrplot(cor_matrix, method = "color",
         col = colorRampPalette(c("navy", "white", "firebrick1"))(200),  # Simple and effective gradient: navy to white to firebrick
         type = "upper",  # Display only the upper part of the matrix
         order = "hclust",  # Order variables by hierarchical clustering
         tl.col = "black",  # Text label color
         tl.srt = 0,  # Text label rotation set to 0 to avoid problems
         addCoef.col = "black",  # Color for the correlation coefficients
         number.cex = 1.0,  # Adjust size of the coefficient labels for clarity
         diag = FALSE)  # Avoid showing the diagonal (self-correlation is always 1)
```




```r
library(caret)
# Split data into training and test sets
set.seed(123)
index <- createDataPartition(Breastcancer_data2$diagnosis, p = 0.75, list = F
ALSE)
train_set <- Breastcancer_data2[index,]
test_set <- Breastcancer_data2[-index,]

# Fit the logistic regression model
model <- glm(diagnosis ~ ., data = train_set, family = "binomial")

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# Predict probabilities
predictions <- predict(model, newdata = test_set, type = "response")

ggplot(Breastcancer_data2, aes(x = perimeter_mean, fill = diagnosis)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of Perimeter Mean",
       x = "Perimeter Mean",
       y = "Density") +
  theme_minimal()
```

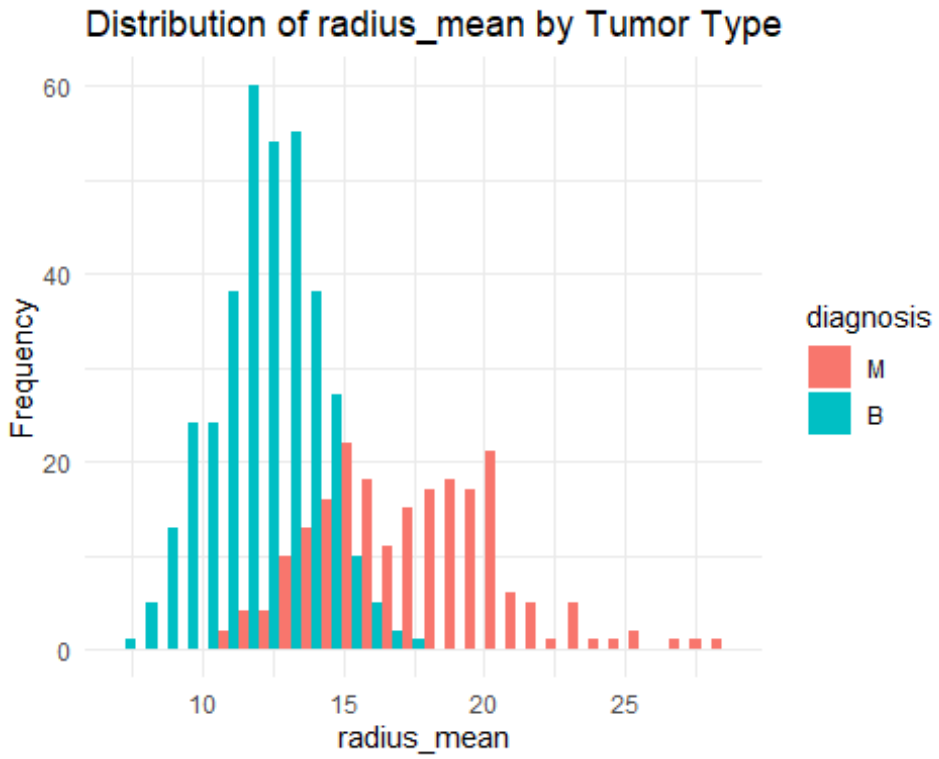



```r
# Read and clean data
BreastCancerData <- read.csv("C:\\Users\\bensh\\Downloads\\data (1).csv")
BreastCancerData <- BreastCancerData %>%
  clean_names() %>%
  distinct() %>%
  select(-starts_with("x")) %>%
  mutate(diagnosis = factor(diagnosis, levels = c("M", "B")))

# Perform logistic regression for each feature and summarize results
logistic_results <- lapply(features, function(feature) {
  model <- glm(as.formula(paste("diagnosis ~", feature)), family = binomial,
data = BreastCancerData)
  summary(model)$coefficients
})

# Print results
names(logistic_results) <- features
print(logistic_results)

## $radius_mean
##               Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) 15.245871 1.32462600  11.50957 1.180708e-30
## radius_mean -1.033589 0.09310645 -11.10115 1.238377e-28
##
## $texture_mean
##                Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)   5.1257724 0.52638000  9.737780 2.080506e-22
```

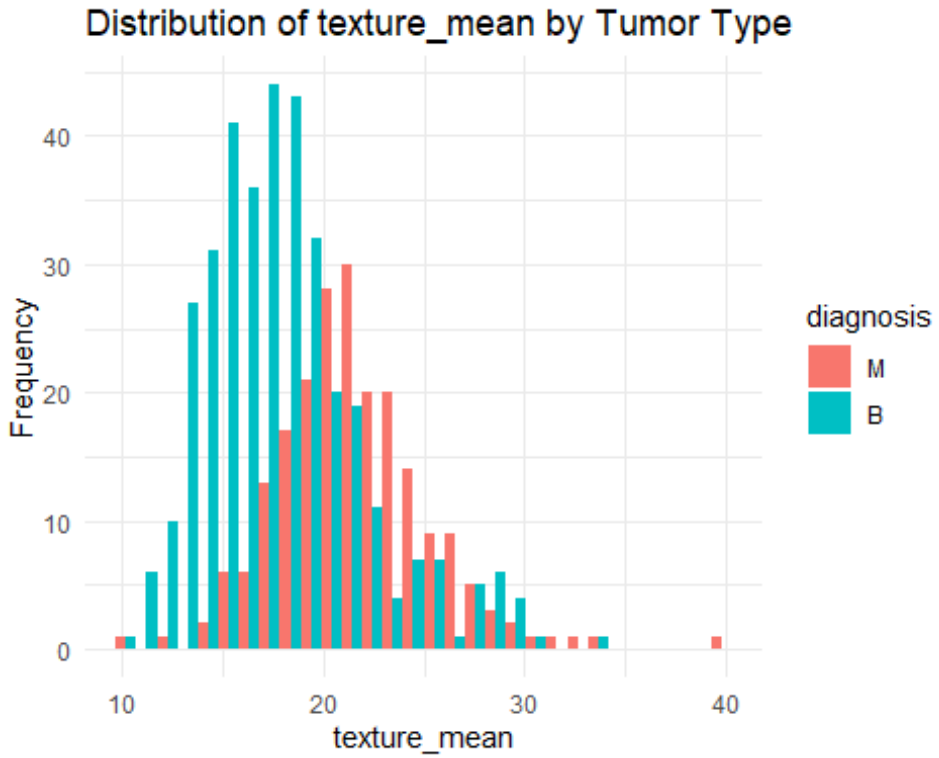
```
## texture_mean -0.2346406 0.02614308 -8.975245 2.827221e-19
##
## $perimeter_mean
##                  Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)    15.7133279 1.37475538  11.42991 2.964217e-30
## perimeter_mean -0.1639859 0.01485368 -11.04009 2.447935e-28
##
## $area_mean
##                Estimate  Std. Error   z value     Pr(>|z|)
## (Intercept)  7.97409315 0.682863094  11.67744 1.662256e-31
## area_mean   -0.01176793 0.001089694 -10.79929 3.468757e-27
##
## $smoothness_mean
##                   Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)       6.377306   0.747421  8.532415 1.433238e-17
## smoothness_mean -60.085710   7.549692 -7.958697 1.738602e-15
```

```
# Visualizations for each feature comparing malignant and benign
plots <- lapply(features, function(feature) {
  ggplot(BreastCancerData, aes_string(x = feature, fill = "diagnosis")) +
    geom_histogram(position = "dodge", bins = 30) +
    labs(title = paste("Distribution of", feature, "by Tumor Type"), x = feat
ure, y = "Frequency") +
    theme_minimal()
})
print(plots)
```

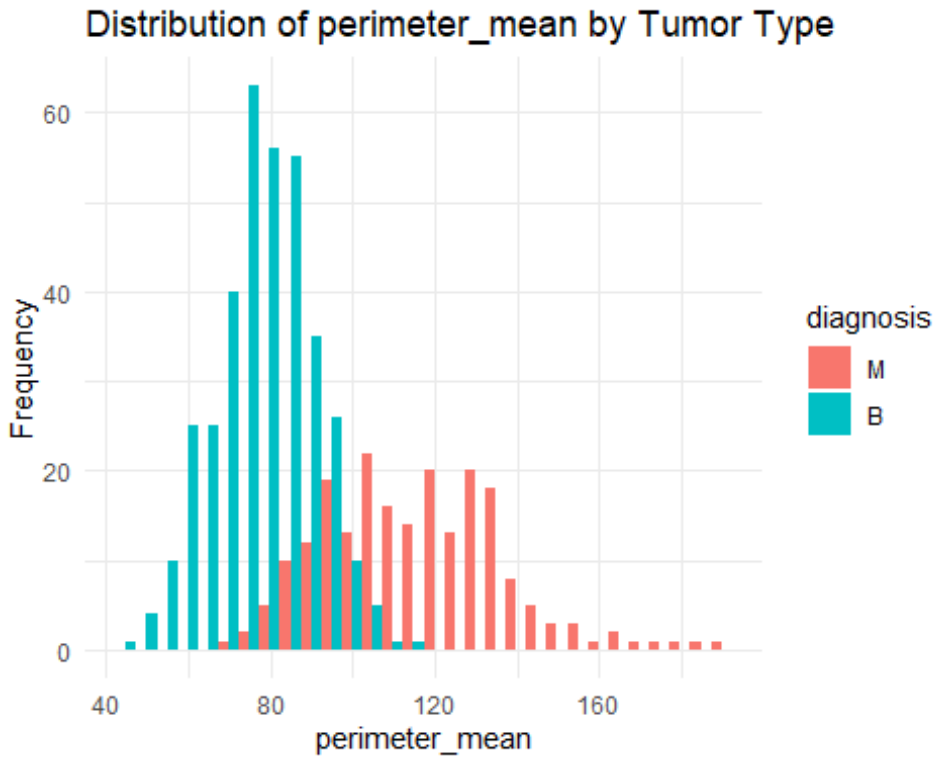
```
## [[1]]
```


Distribution of radius_mean by Tumor Type
Frequency
radius_mean
diagnosis
M
B
0
20
40
60
10
15
20
25

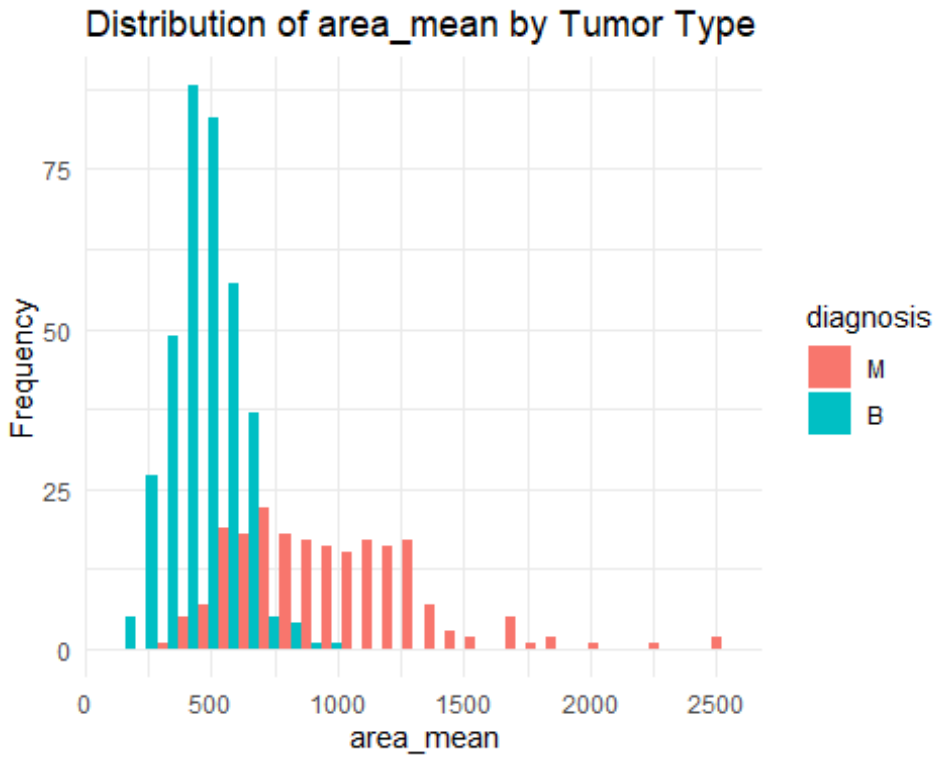
```
##
## [[2]]
```


Distribution of texture_mean by Tumor Type
Frequency
texture_mean
diagnosis
M
B
0
10
20
30
40
10
20
30
40

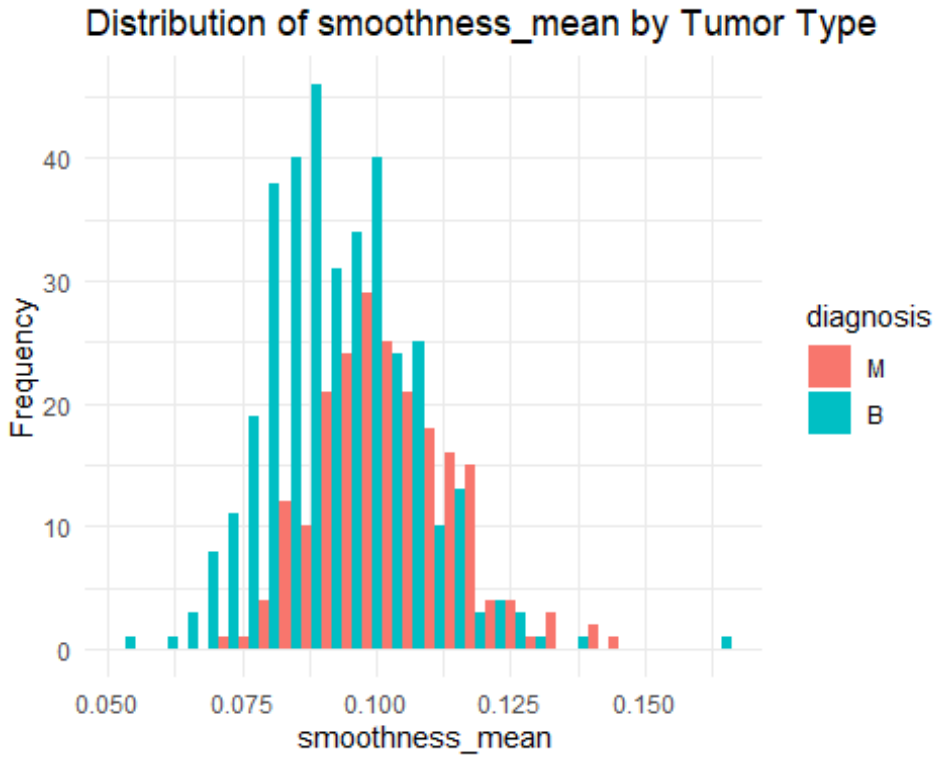
```
##
## [[3]]
```


Distribution of perimeter_mean by Tumor Type
Frequency
0
20
40
60
40
80
120
160
perimeter_mean
diagnosis
M
B

```
##
## [[4]]
```


Distribution of area_mean by Tumor Type
Frequency
75
50
25
0
0
500
1000
1500
2000
2500
area_mean
diagnosis
M
B

```
##
## [[5]]
```


Distribution of smoothness_mean by Tumor Type
Frequency
40
30
20
10
0
0.050
0.075
0.100
0.125
0.150
smoothness_mean
diagnosis
M
B

```r
# Check unique values of the diagnosis column
unique_values <- unique(BreastCancerData$diagnosis)
print(unique_values)
```

```
## [1] M B
## Levels: M B
```

```r
# Check for missing values in the diagnosis column
missing_values <- sum(is.na(BreastCancerData$diagnosis))
print(missing_values)
```

```
## [1] 0
```