Paquetes necesarios

```
if (!require('dplyr'))
  install.packages('dplyr')
```

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## 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
```

```
library(dplyr)

if (!require('scales'))
  install.packages('scales')
```

```
## Loading required package: scales
```

```
## Warning: package 'scales' was built under R version 4.2.3
```

```
library(scales)

if (!require('gridExtra'))
  install.packages('gridExtra')
```

```
## Loading required package: gridExtra
```

```
## Warning: package 'gridExtra' was built under R version 4.2.3
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(gridExtra)

if (!require('corrplot'))
  install.packages('corrplot')
```

```
## Loading required package: corrplot
```

```
## corrplot 0.92 loaded
```

```
library(corrplot)

if (!require('ggplot2'))
  install.packages('ggplot2')
```

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

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```r
library(ggplot2)

if (!require('caret'))
  install.packages('caret')
```

```
## Loading required package: caret
## Warning: package 'caret' was built under R version 4.2.3
## Loading required package: lattice
```

```r
library(caret)

if (!require('car'))
  install.packages('car')
```

```
## Loading required package: car
## Warning: package 'car' was built under R version 4.2.3
## Loading required package: carData
##
## Attaching package: 'car'
## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
library(car)

if (!require('randomForest'))
  install.packages('randomForest')
```

```
## Loading required package: randomForest
## Warning: package 'randomForest' was built under R version 4.2.3
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:gridExtra':
##
##     combine
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(randomForest)

if (!require('rpart'))
  install.packages('rpart')
```

```
## Loading required package: rpart
```

```
library(rpart)

if (!require('class'))
  install.packages('class')
```

```
## Loading required package: class
```

```
library(class)

if (!require('e1071'))
  install.packages('e1071')
```

```
## Loading required package: e1071

## Warning: package 'e1071' was built under R version 4.2.3
```

```
library(e1071)
```

Preprocesado

Carga de datos

```
data <- read.csv("C:/Users/ariad/Desktop/MASTER/6 SEM/TFM/Dataset/breast-cancer-wisconsin.csv", sep=",")
str(data)
```

```
## 'data.frame':    569 obs. of  33 variables:
##  $ id                      : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844
##  $ diagnosis               : chr  "M" "M" "M" "M" ...
##  $ radius_mean             : num  18 20.6 19.7 11.4 20.3 ...
##  $ texture_mean            : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ perimeter_mean          : num  122.8 132.9 130 77.6 135.1 ...
##  $ area_mean               : num  1001 1326 1203 386 1297 ...
##  $ smoothness_mean         : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ compactness_mean        : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ concavity_mean          : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ concave.points_mean     : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ symmetry_mean           : num  0.242 0.181 0.207 0.26 0.181 ...
##  $ fractal_dimension_mean  : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ radius_se               : num  1.095 0.543 0.746 0.496 0.757 ...
##  $ texture_se              : num  0.905 0.734 0.787 1.156 0.781 ...
##  $ perimeter_se            : num  8.59 3.4 4.58 3.44 5.44 ...
##  $ area_se                 : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se           : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ compactness_se          : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se            : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ concave.points_se       : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ symmetry_se             : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ fractal_dimension_se    : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ radius_worst            : num  25.4 25 23.6 14.9 22.5 ...
##  $ texture_worst           : num  17.3 23.4 25.5 26.5 16.7 ...
##  $ perimeter_worst         : num  184.6 158.8 152.5 98.9 152.2 ...
##  $ area_worst              : num  2019 1956 1709 568 1575 ...
##  $ smoothness_worst        : num  0.162 0.124 0.144 0.21 0.137 ...
##  $ compactness_worst       : num  0.666 0.187 0.424 0.866 0.205 ...
```

```
##  $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
##  $ concave.points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
##  $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
##  $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
##  $ X                      : logi  NA NA NA NA NA NA ...
```

Calcular estadísticas descriptivas para variables numéricas

```
numvars_stats <- sapply(data[, sapply(data, is.numeric)], function(x) c(mean = mean(x, na.rm = TRUE),
                                                                        sd = sd(x, na.rm = TRUE),
                                                                        min = min(x, na.rm = TRUE),
                                                                        max = max(x, na.rm = TRUE)))


# Mostrar las estadísticas descriptivas transpuestas
numvars_stats <- t(numvars_stats)
colnames(numvars_stats) <- c("Mean", "SD", "Min", "Max")
numvars_stats
```

```
##                                    Mean                  SD          Min
## id                       30371831.432337433 125020585.612223655 8670.0000000
## radius_mean                     14.127291740         3.524048826    6.9810000
## texture_mean                    19.289648506         4.301035768    9.7100000
## perimeter_mean                  91.969033392        24.298981039   43.7900000
## area_mean                      654.889103691       351.914129182  143.5000000
## smoothness_mean                  0.096360281         0.014064128    0.0526300
## compactness_mean                 0.104340984         0.052812758    0.0193800
## concavity_mean                   0.088799316         0.079719809    0.0000000
## concave.points_mean              0.048919146         0.038802845    0.0000000
## symmetry_mean                    0.181161863         0.027414281    0.1060000
## fractal_dimension_mean           0.062797610         0.007060363    0.0499600
## radius_se                        0.405172056         0.277312733    0.1115000
## texture_se                       1.216853427         0.551648393    0.3602000
## perimeter_se                     2.866059227         2.021854554    0.7570000
## area_se                         40.337079086        45.491005516    6.8020000
## smoothness_se                    0.007040979         0.003002518    0.0017130
## compactness_se                   0.025478139         0.017908179    0.0022520
## concavity_se                     0.031893716         0.030186060    0.0000000
## concave.points_se                0.011796137         0.006170285    0.0000000
## symmetry_se                      0.020542299         0.008266372    0.0078820
## fractal_dimension_se             0.003794904         0.002646071    0.0008948
## radius_worst                    16.269189807         4.833241580    7.9300000
## texture_worst                   25.677223199         6.146257623   12.0200000
## perimeter_worst                107.261212654        33.602542269   50.4100000
## area_worst                     880.583128295       569.356992670  185.2000000
## smoothness_worst                 0.132368594         0.022832429    0.0711700
## compactness_worst                0.254265044         0.157336489    0.0272900
## concavity_worst                  0.272188483         0.208624281    0.0000000
## concave.points_worst             0.114606223         0.065732341    0.0000000
## symmetry_worst                   0.290075571         0.061867468    0.1565000
## fractal_dimension_worst          0.083945817         0.018061267    0.0550400
##                                      Max
## id                       911320502.00000
## radius_mean                     28.11000
## texture_mean                    39.28000
```

```
## perimeter_mean               188.50000
## area_mean                   2501.00000
## smoothness_mean                0.16340
## compactness_mean               0.34540
## concavity_mean                 0.42680
## concave.points_mean            0.20120
## symmetry_mean                  0.30400
## fractal_dimension_mean         0.09744
## radius_se                      2.87300
## texture_se                     4.88500
## perimeter_se                  21.98000
## area_se                      542.20000
## smoothness_se                  0.03113
## compactness_se                 0.13540
## concavity_se                   0.39600
## concave.points_se              0.05279
## symmetry_se                    0.07895
## fractal_dimension_se           0.02984
## radius_worst                  36.04000
## texture_worst                 49.54000
## perimeter_worst              251.20000
## area_worst                  4254.00000
## smoothness_worst               0.22260
## compactness_worst              1.05800
## concavity_worst                1.25200
## concave.points_worst           0.29100
## symmetry_worst                 0.66380
## fractal_dimension_worst        0.20750
```

Valores nulos

```
colSums(is.na(data))
```

```
##                      id               diagnosis             radius_mean
##                       0                       0                       0
##            texture_mean         perimeter_mean              area_mean
##                       0                       0                       0
##         smoothness_mean        compactness_mean          concavity_mean
##                       0                       0                       0
##     concave.points_mean          symmetry_mean  fractal_dimension_mean
##                       0                       0                       0
##               radius_se             texture_se           perimeter_se
##                       0                       0                       0
##                 area_se          smoothness_se         compactness_se
##                       0                       0                       0
##           concavity_se       concave.points_se            symmetry_se
##                       0                       0                       0
##    fractal_dimension_se            radius_worst          texture_worst
##                       0                       0                       0
##         perimeter_worst              area_worst        smoothness_worst
##                       0                       0                       0
##       compactness_worst         concavity_worst    concave.points_worst
##                       0                       0                       0
##          symmetry_worst fractal_dimension_worst                       X
```

```
##                       0                       0                     569
```

Corregir inconsistencias

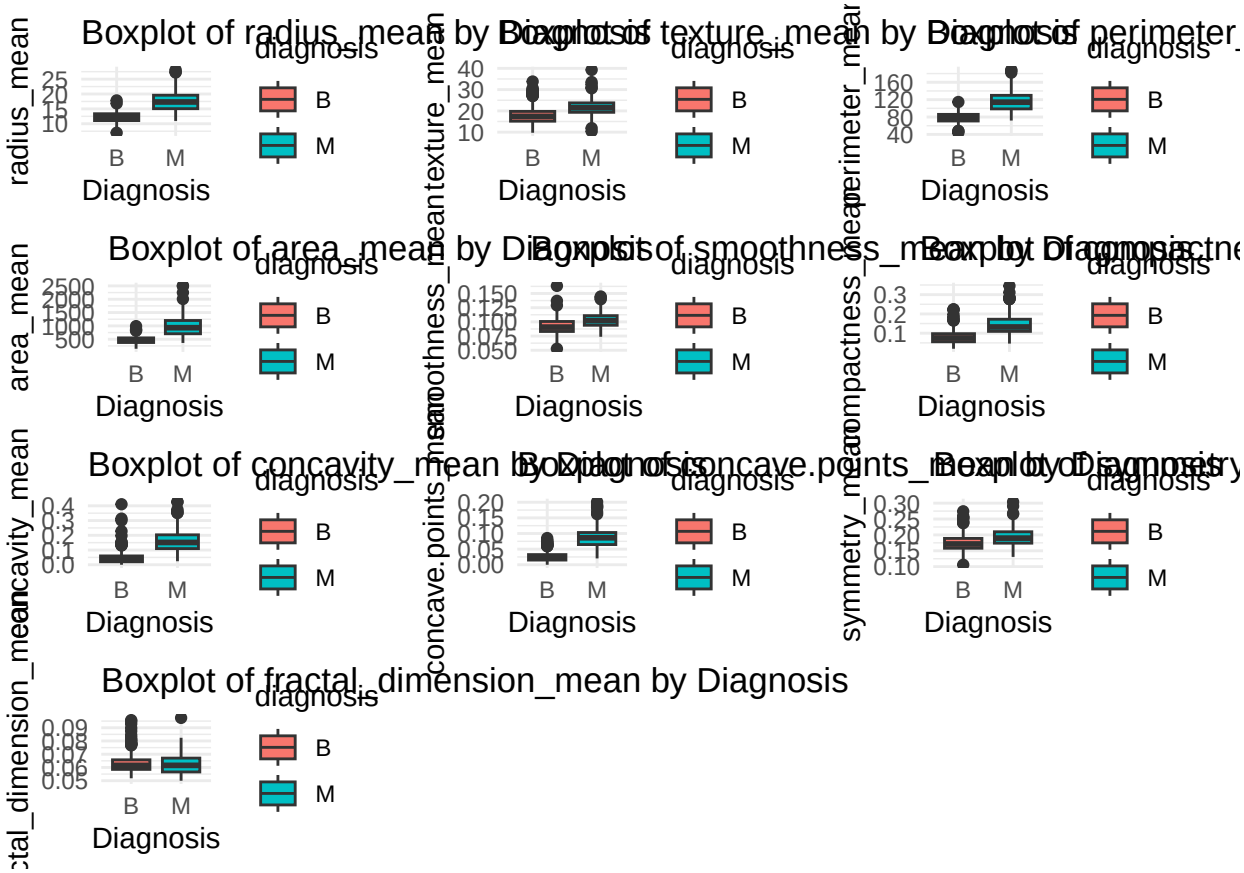
```
df <- subset(data, select = -c(X, id))
head(df)
```

```
##   diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1         M       17.99        10.38         122.80    1001.0         0.11840
## 2         M       20.57        17.77         132.90    1326.0         0.08474
## 3         M       19.69        21.25         130.00    1203.0         0.10960
## 4         M       11.42        20.38          77.58     386.1         0.14250
## 5         M       20.29        14.34         135.10    1297.0         0.10030
## 6         M       12.45        15.70          82.57     477.1         0.12780
##   compactness_mean concavity_mean concave.points_mean symmetry_mean
## 1          0.27760         0.3001              0.14710        0.2419
## 2          0.07864         0.0869              0.07017        0.1812
## 3          0.15990         0.1974              0.12790        0.2069
## 4          0.28390         0.2414              0.10520        0.2597
## 5          0.13280         0.1980              0.10430        0.1809
## 6          0.17000         0.1578              0.08089        0.2087
##   fractal_dimension_mean radius_se texture_se perimeter_se area_se
## 1                0.07871    1.0950     0.9053        8.589  153.40
## 2                0.05667    0.5435     0.7339        3.398   74.08
## 3                0.05999    0.7456     0.7869        4.585   94.03
## 4                0.09744    0.4956     1.1560        3.445   27.23
## 5                0.05883    0.7572     0.7813        5.438   94.44
## 6                0.07613    0.3345     0.8902        2.217   27.19
##   smoothness_se compactness_se concavity_se concave.points_se symmetry_se
## 1      0.006399        0.04904      0.05373           0.01587     0.03003
## 2      0.005225        0.01308      0.01860           0.01340     0.01389
## 3      0.006150        0.04006      0.03832           0.02058     0.02250
## 4      0.009110        0.07458      0.05661           0.01867     0.05963
## 5      0.011490        0.02461      0.05688           0.01885     0.01756
## 6      0.007510        0.03345      0.03672           0.01137     0.02165
##   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## 1             0.006193        25.38         17.33          184.60     2019.0
## 2             0.003532        24.99         23.41          158.80     1956.0
## 3             0.004571        23.57         25.53          152.50     1709.0
## 4             0.009208        14.91         26.50           98.87      567.7
## 5             0.005115        22.54         16.67          152.20     1575.0
## 6             0.005082        15.47         23.75          103.40      741.6
##   smoothness_worst compactness_worst concavity_worst concave.points_worst
## 1           0.1622            0.6656          0.7119               0.2654
## 2           0.1238            0.1866          0.2416               0.1860
## 3           0.1444            0.4245          0.4504               0.2430
## 4           0.2098            0.8663          0.6869               0.2575
## 5           0.1374            0.2050          0.4000               0.1625
## 6           0.1791            0.5249          0.5355               0.1741
##   symmetry_worst fractal_dimension_worst
## 1         0.4601                 0.11890
## 2         0.2750                 0.08902
## 3         0.3613                 0.08758
## 4         0.6638                 0.17300
```

```
## 5         0.2364                0.07678
## 6         0.3985                0.12440
```

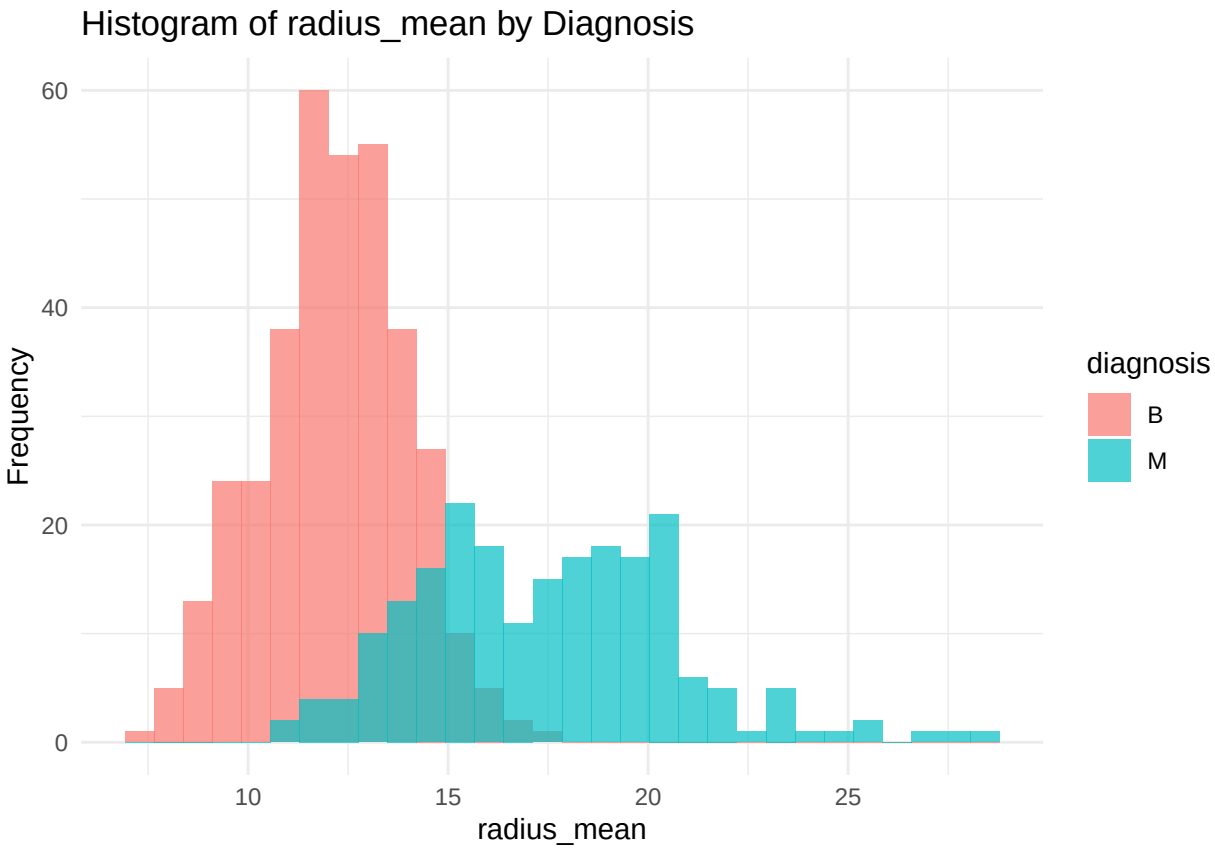
Diagramas de caja y presencia de outliers

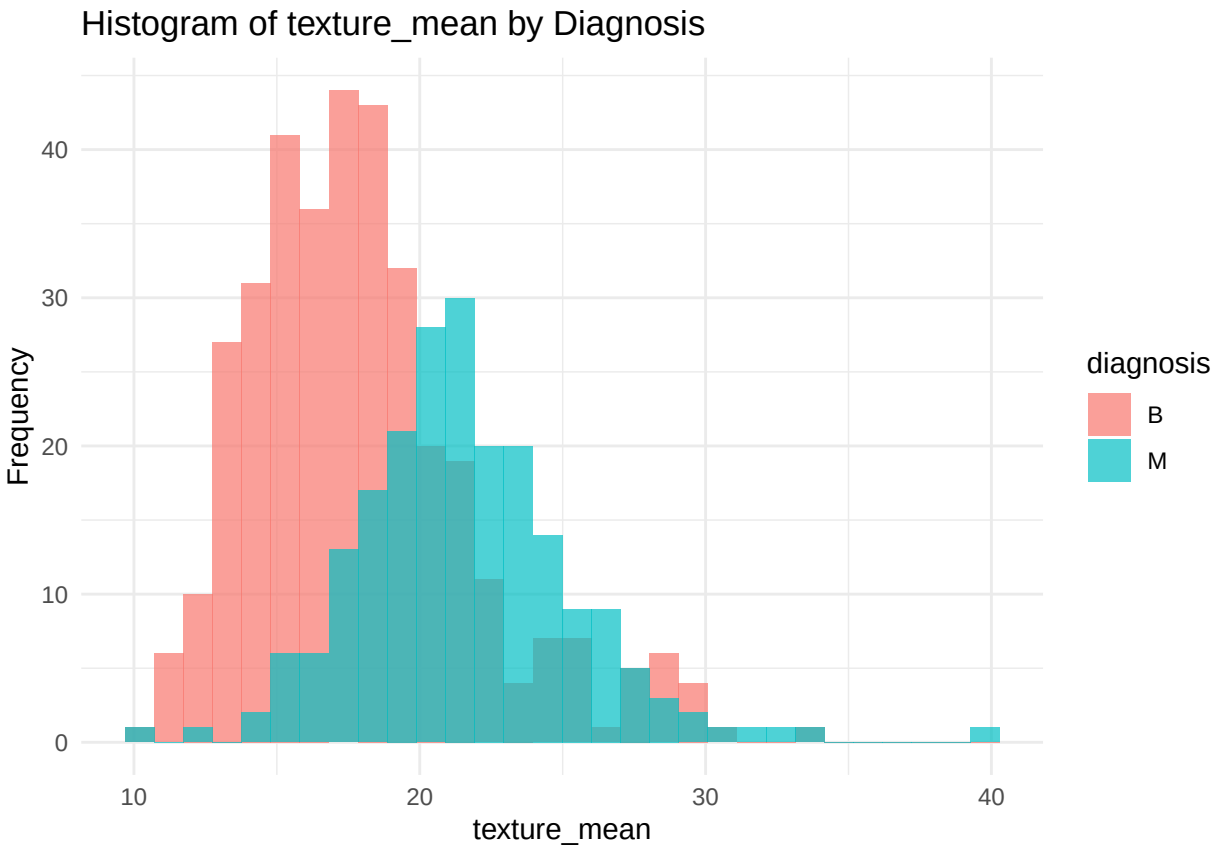
```r
# Seleccionar las 10 primeras columnas después de la variable "diagnosis"
df_subset <- df[, 2:11]


boxplot_by_diagnosis <- function(variable_name) {
  ggplot(df, aes(x = diagnosis, y = !!sym(variable_name), fill = diagnosis)) +
    geom_boxplot() +
    labs(title = paste("Boxplot of", variable_name, "by Diagnosis"),
         x = "Diagnosis", y = variable_name) +
    theme_minimal()
}

# Crear una lista para almacenar todos los gráficos
plots <- list()

# Crear diagramas de caja para cada variable numérica frente a la variable "diagnosis"
for (variable in colnames(df_subset)) {
  plots[[variable]] <- boxplot_by_diagnosis(variable)
}

# Organizar los gráficos en una cuadrícula
grid.arrange(grobs = plots, ncol = 3)
```

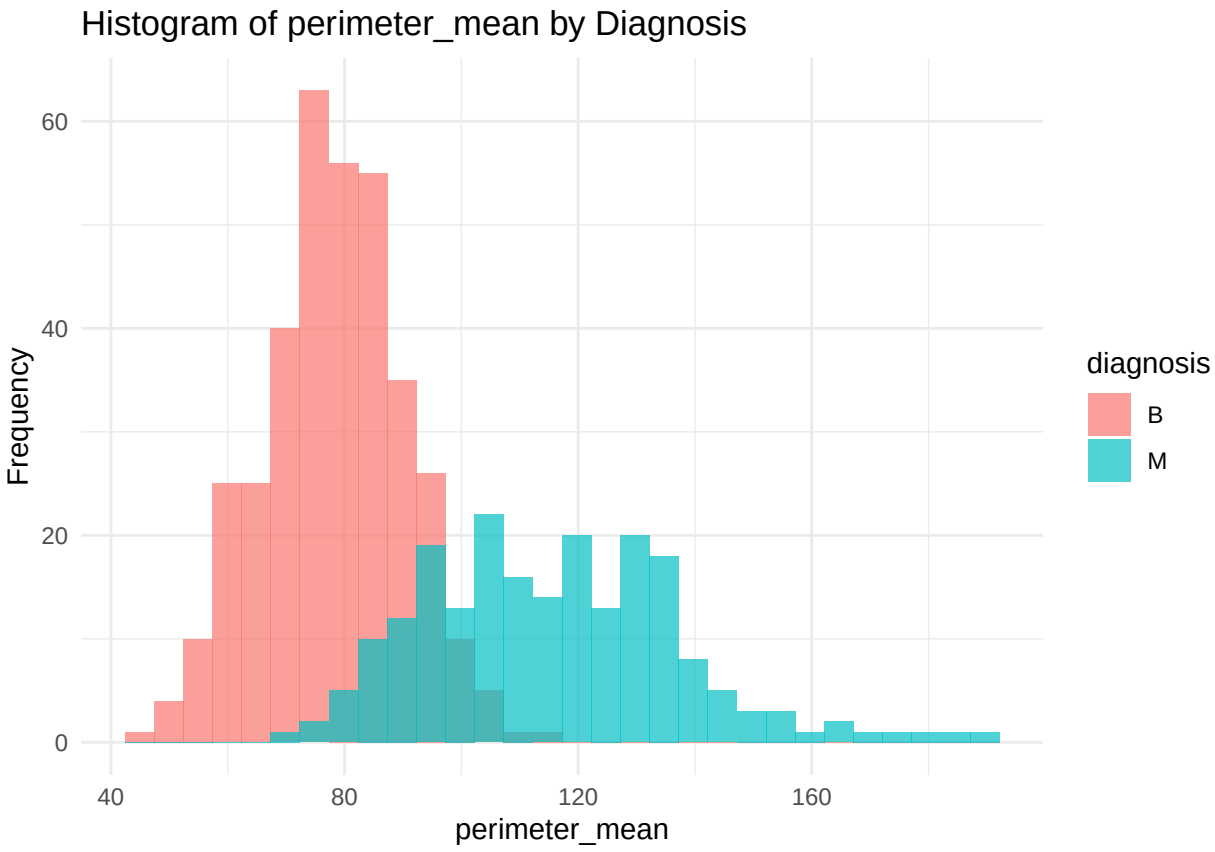


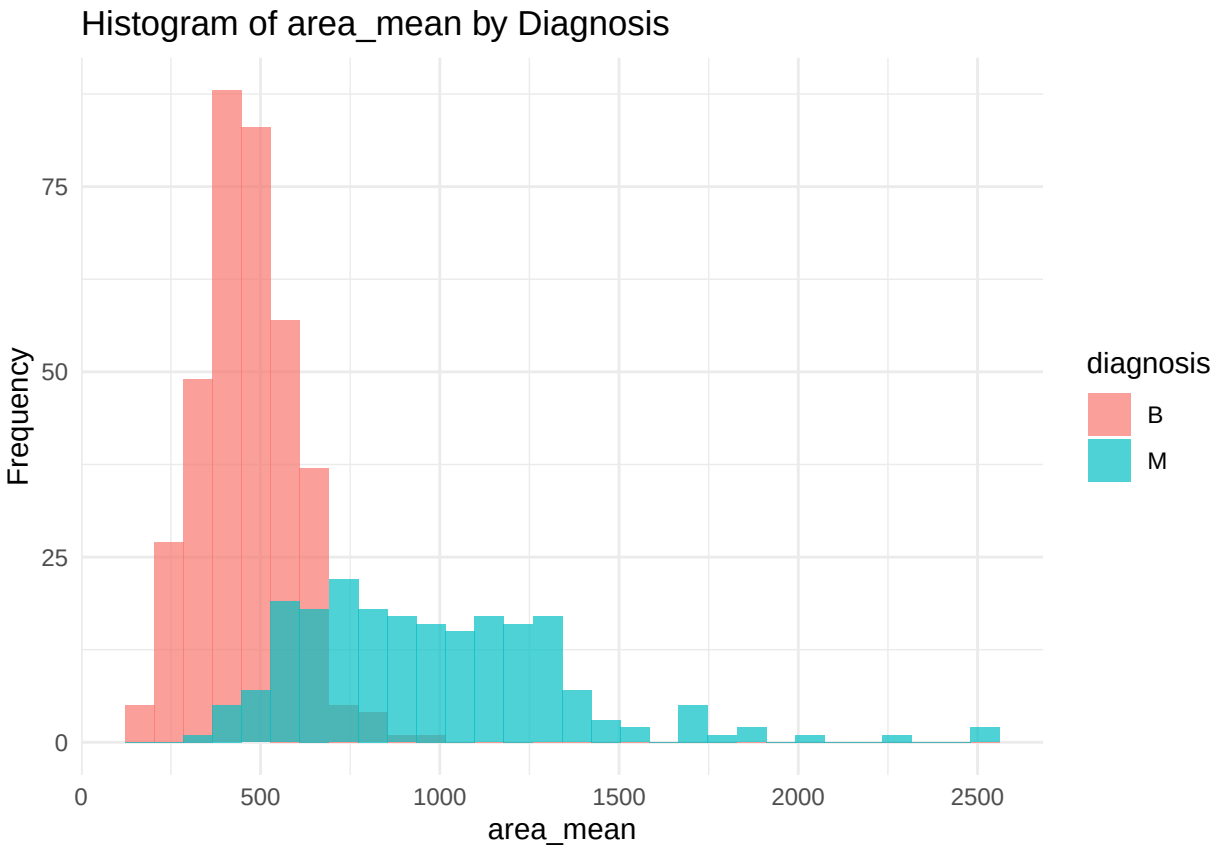
Histogramas (diagramas de barras) y distribución visual

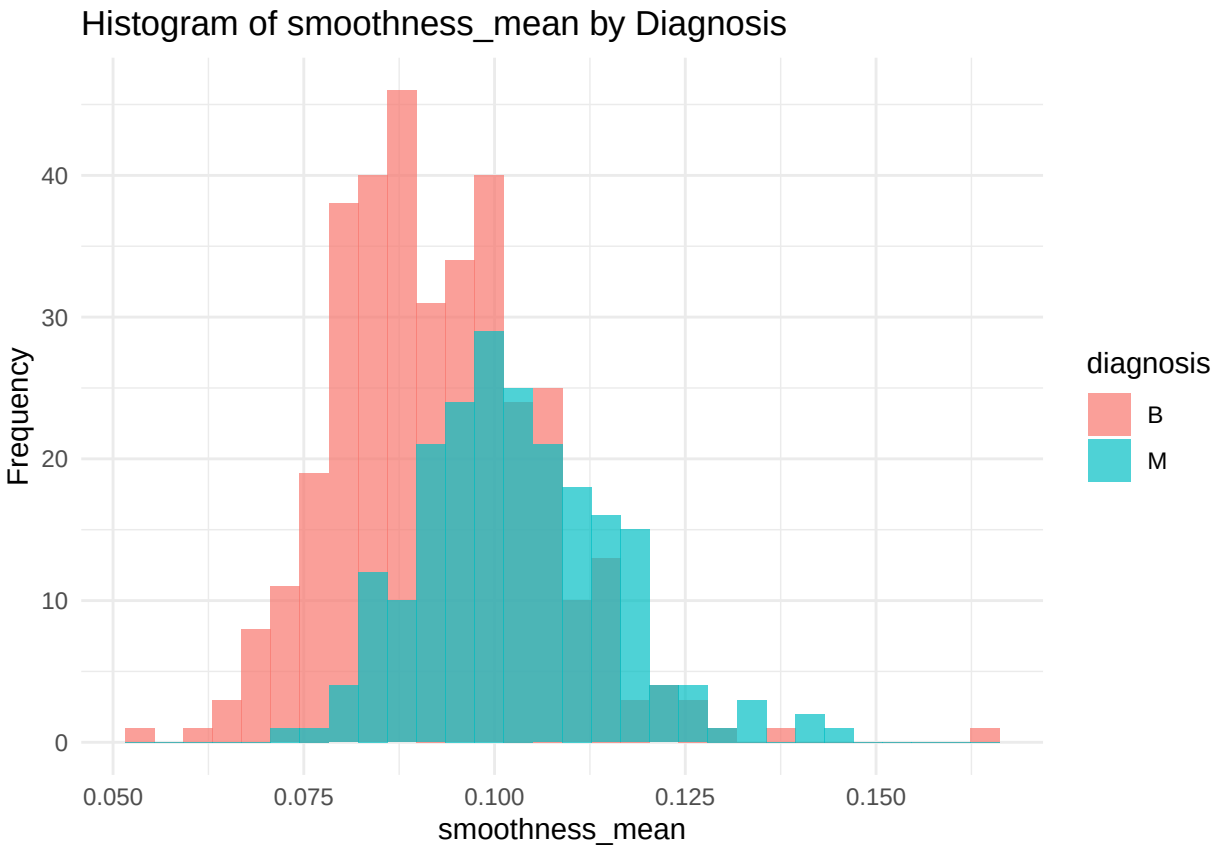
```r
# Función para crear histogramas
histogram_by_diagnosis <- function(variable_name) {
  ggplot(df, aes(x = !!sym(variable_name), fill = diagnosis)) +
    geom_histogram(bins = 30, alpha = 0.7, position = "identity") +
    labs(title = paste("Histogram of", variable_name, "by Diagnosis"),
         x = variable_name, y = "Frequency") +
    theme_minimal()
}

for (variable in colnames(df_subset)) {
  print(histogram_by_diagnosis(variable))
}
```

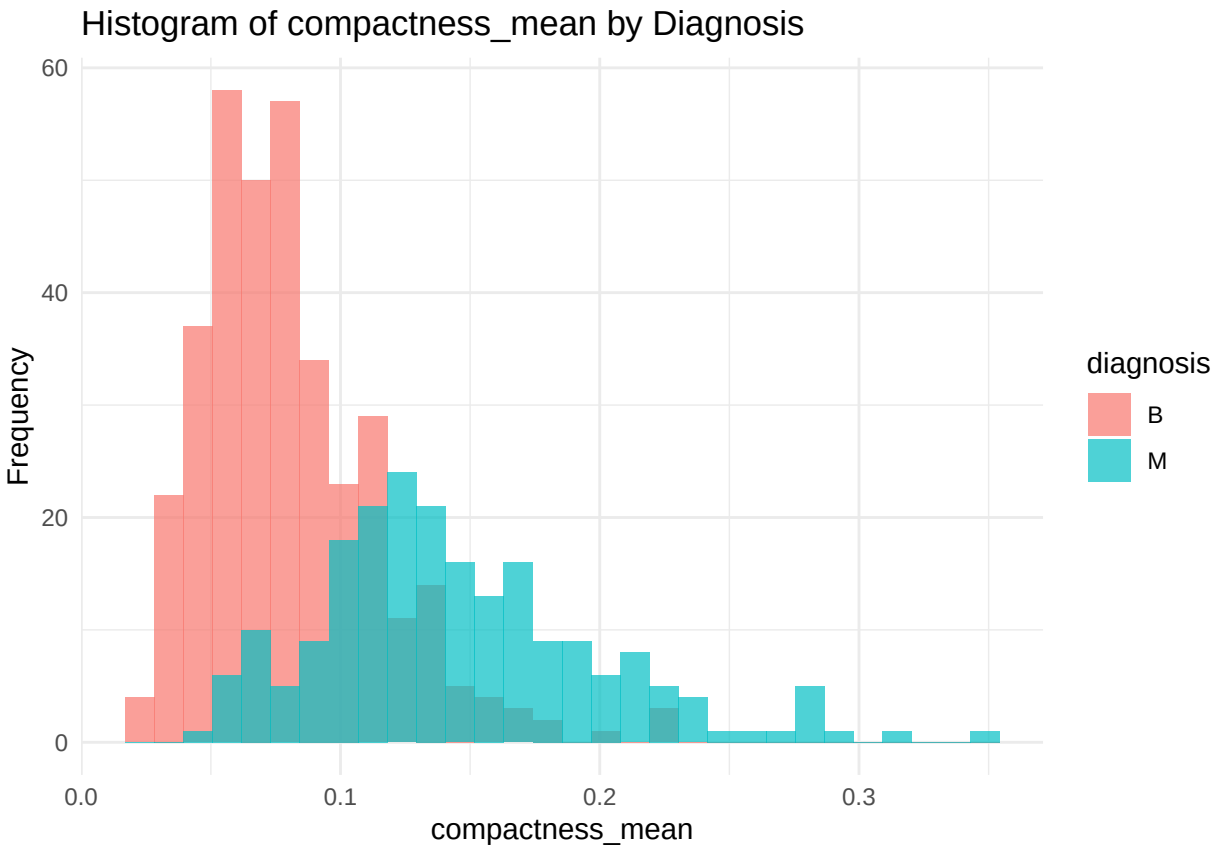

Histogram of radius_mean by Diagnosis
Frequency
60
40
20
0
10
15
20
25
radius_mean
diagnosis
B
M

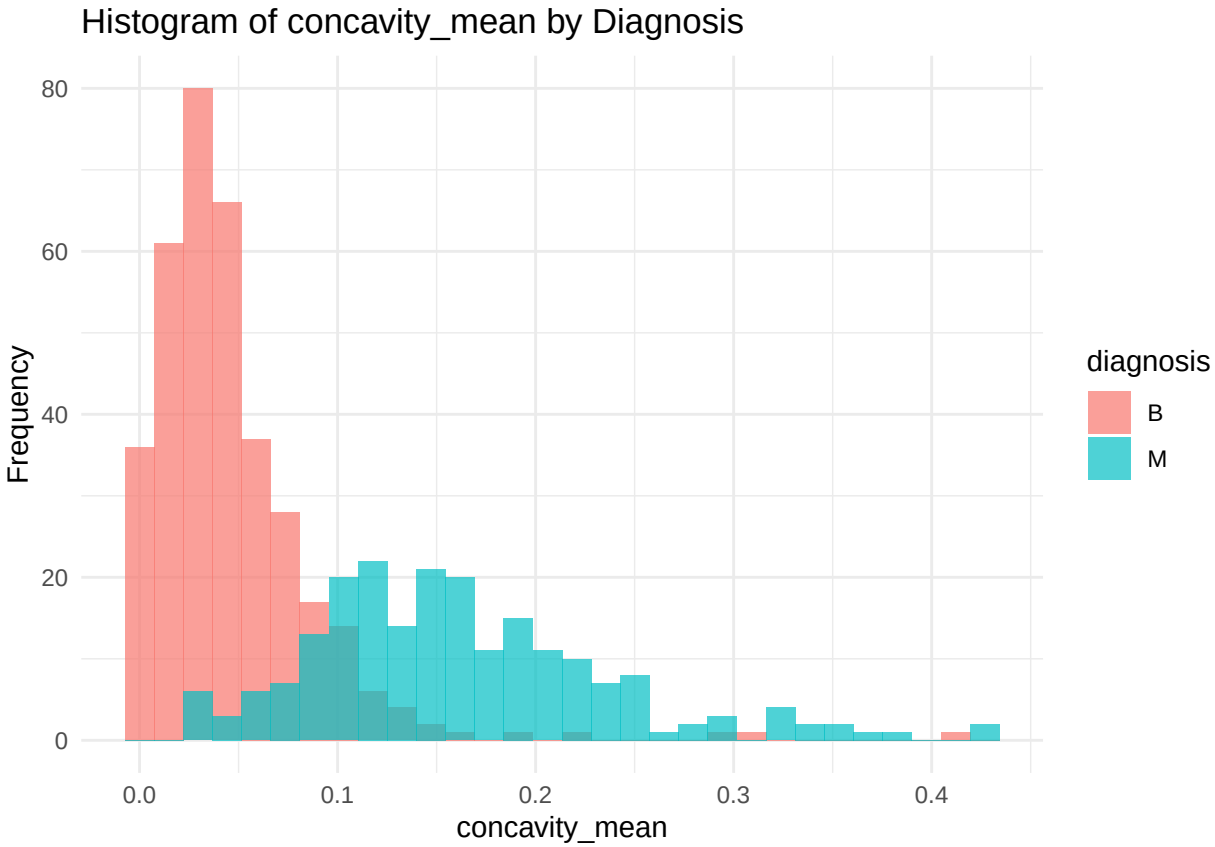

Histogram of texture_mean by Diagnosis
Frequency
0
10
20
30
40
10
20
30
40
texture_mean
diagnosis
B
M

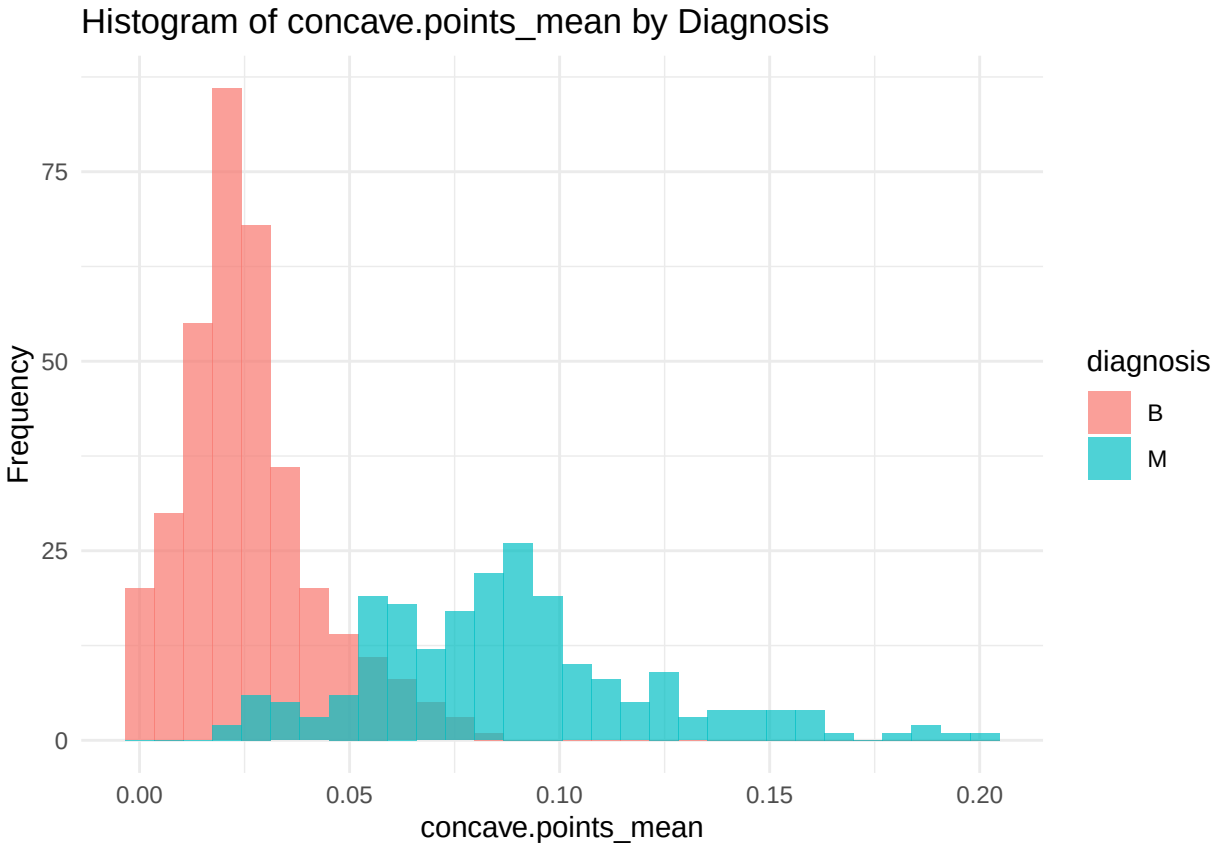

Histogram of perimeter_mean by Diagnosis
Frequency
60
40
20
0
40
80
120
160
perimeter_mean
diagnosis
B
M

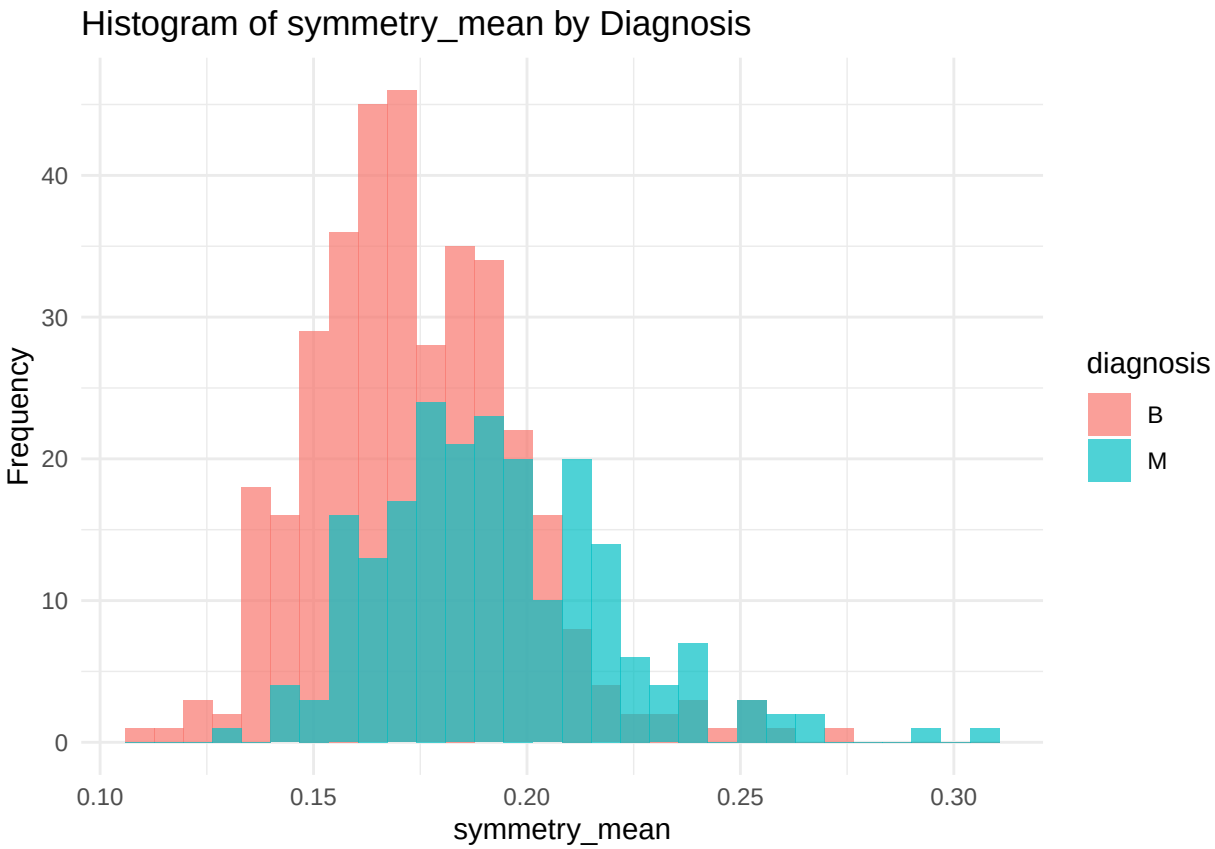

Histogram of area_mean by Diagnosis
Frequency
75
50
25
0
0
500
1000
1500
2000
2500
area_mean
diagnosis
B
M

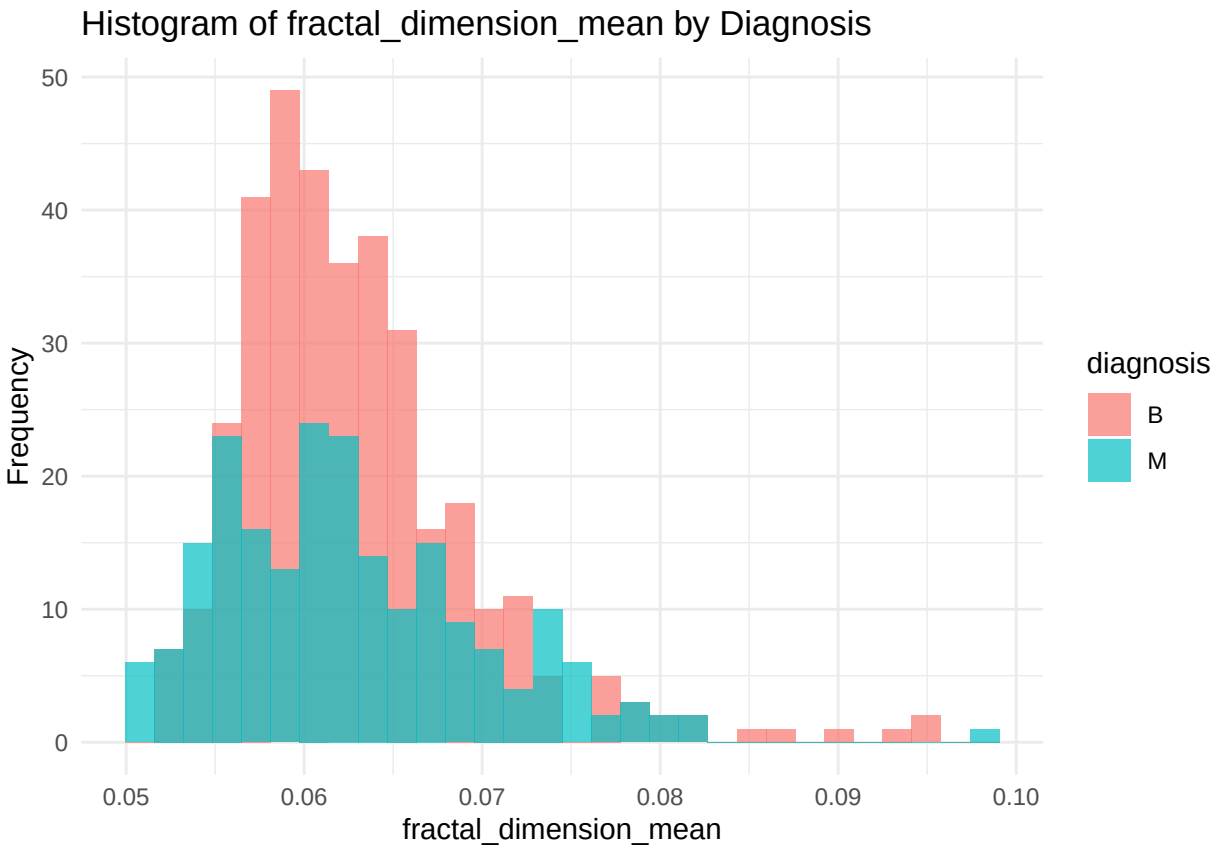

Histogram of smoothness_mean by Diagnosis
Frequency
40
30
20
10
0
0.050
0.075
0.100
0.125
0.150
smoothness_mean
diagnosis
B
M


Histogram of compactness_mean by Diagnosis
Frequency
60
40
20
0
0.0
0.1
0.2
0.3
compactness_mean
diagnosis
B
M


Histogram of concavity_mean by Diagnosis
Frequency
80
60
40
20
0
0.0
0.1
0.2
0.3
0.4
concavity_mean
diagnosis
B
M


Histogram of concave.points_mean by Diagnosis
Frequency
75
50
25
0
0.00
0.05
0.10
0.15
0.20
concave.points_mean
diagnosis
B
M


Histogram of symmetry_mean by Diagnosis
Frequency
40
30
20
10
0
0.10
0.15
0.20
0.25
0.30
symmetry_mean
diagnosis
B
M

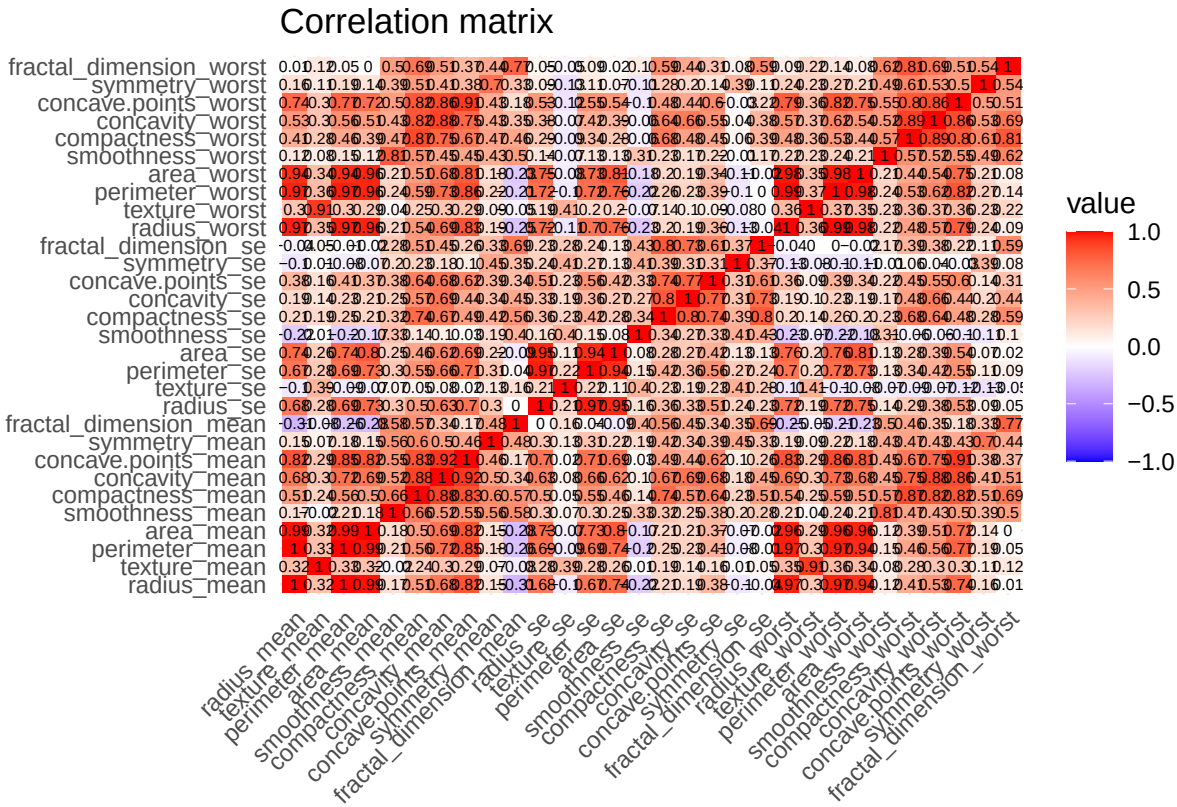


Codificación d ela variable objetivo (Diagnosis)

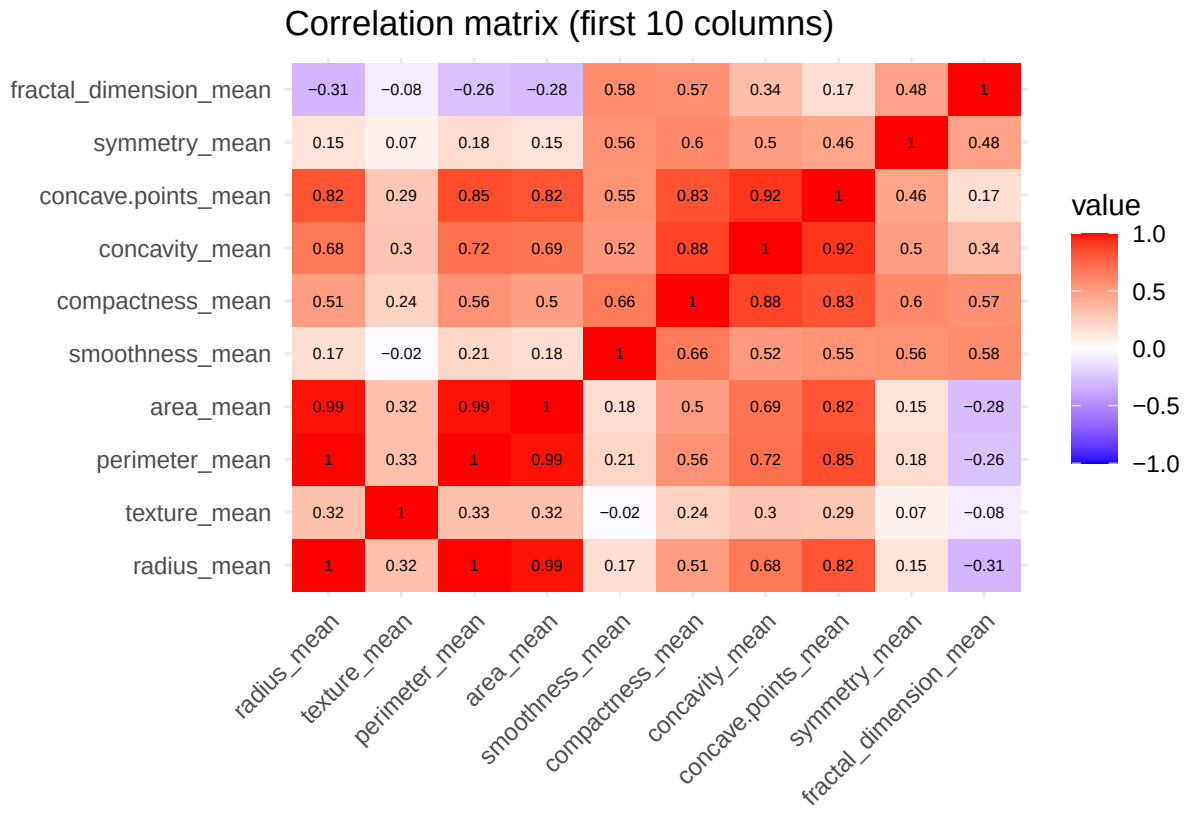
```
# Codificación de etiquetas
df <- df %>%
  mutate(diagnosis = ifelse(diagnosis == "M", 1, 0))

dff <- df[, -c(1)]
```

Matriz de correlación

```
# Calcular la matriz de correlación
correlation_matrix <- cor(dff)

# Convertir la matriz de correlación en formato de datos largo para ggplot2
cor_df <- reshape2::melt(correlation_matrix)

# Crear el gráfico de correlación con ggplot2
ggplot(cor_df, aes(x = Var1, y = Var2, fill = value, label = round(value, 2))) +
  geom_tile() +
  geom_text(color = "black", size = 2) +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0, limits=c(-1,1)) +
  labs(title = "Correlation matrix", x = "", y = "") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```




```r
# Seleccionar las 10 primeras columnas después de la variable "diagnosis"
df_subset <- df[, 2:11]

# Calcular la matriz de correlación
correlation_matrix_subset <- cor(df_subset)

# Convertir la matriz de correlación en formato de datos largo para ggplot2
cor_df_subset <- reshape2::melt(correlation_matrix_subset)

# Crear el gráfico de correlación con ggplot2
ggplot(cor_df_subset, aes(x = Var1, y = Var2, fill = value, label = round(value, 2))) +
  geom_tile() +
  geom_text(color = "black", size = 2) +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0, limits=c(-1,1)) +
  labs(title = "Correlation matrix (first 10 columns)", x = "", y = "") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



Generación de los conjuntos de entrenamiento y de test

```r
# Separar la variable independiente (X) y la variable dependiente (y)
X <- df[, !(names(df) %in% c("diagnosis"))]
y <- df$diagnosis

# Dividir el conjunto de datos en train y test
set.seed(123)
trainIndex <- createDataPartition(y, p = 0.8, list = FALSE)
x_train <- X[trainIndex, ]
y_train <- y[trainIndex]
x_test <- X[-trainIndex, ]
y_test <- y[-trainIndex]

# Para la normalización (Min_Max Scaling)
min_max <- preProcess(x_train, method = c("range"))
x_train_normalized <- predict(min_max, newdata = x_train)
x_test_normalized <- predict(min_max, newdata = x_test)

x_train_normalized_df <- as.data.frame(x_train_normalized)
x_test_normalized_df <- as.data.frame(x_test_normalized)
y_train_df <- as.data.frame(y_train)
y_test_df <- as.data.frame(y_test)
```

Regresión Logística

```
# Logistic Regression
logreg <- glm(y_train ~ ., family = binomial(link = "logit"), data = x_train_normalized_df)
```

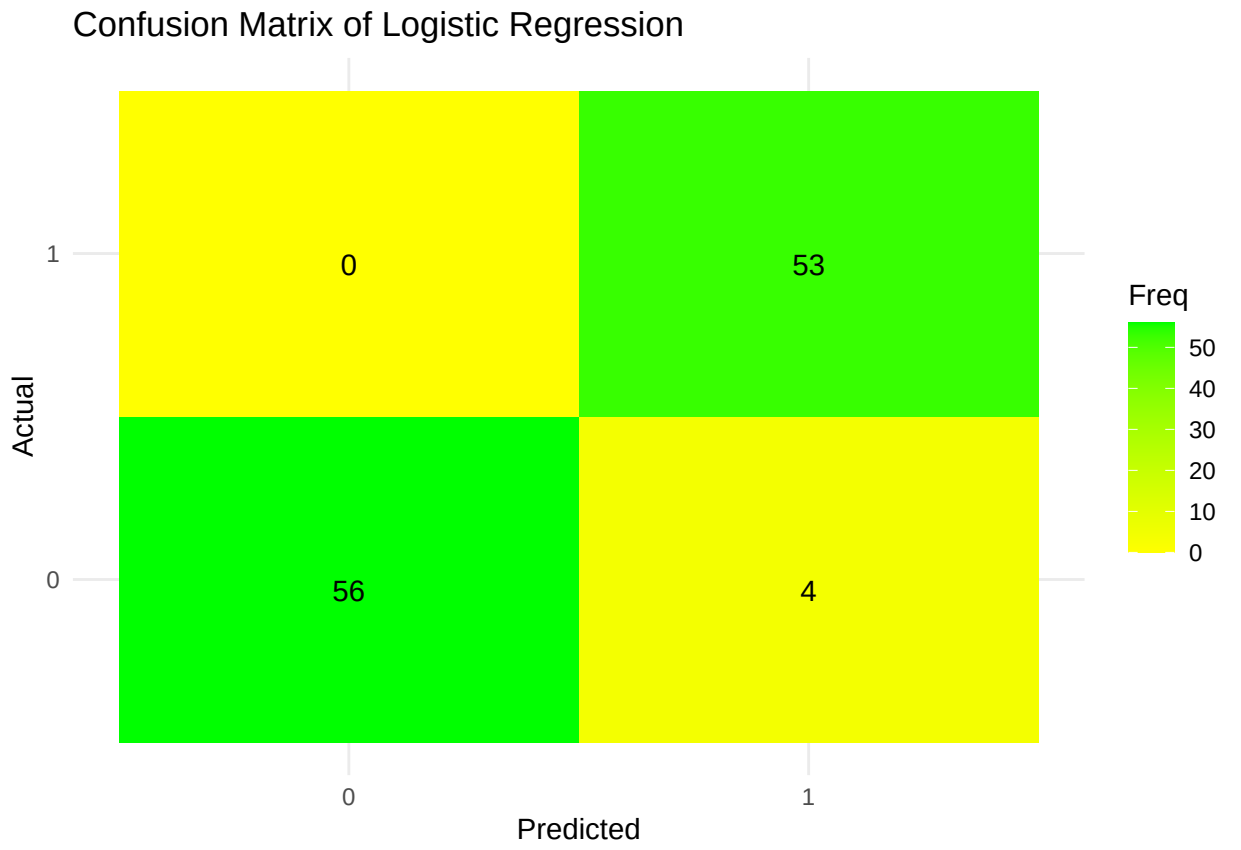
```
## Warning: glm.fit: algorithm did not converge
```

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

```
# Predicciones
y_pred_logreg <- predict(logreg, newdata = x_test_normalized_df, type = "response")

# Redondear las probabilidades a 0 o 1
y_pred_rounded <- ifelse(y_pred_logreg >= 0.5, 1, 0)

# Crear un dataframe con las predicciones redondeadas y las etiquetas verdaderas
predictions_df_logreg <- data.frame(Predicted = y_pred_rounded, Actual = y_test_df)

# Convertir las columnas a factores
predictions_df_logreg$Predicted <- as.factor(predictions_df_logreg$Predicted)
predictions_df_logreg$y_test <- as.factor(predictions_df_logreg$y_test)

# Crear la matriz de confusión
conf_matrix_logreg <- confusionMatrix(predictions_df_logreg$Predicted, predictions_df_logreg$y_test)

conf_matrix_logreg
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 56  0
##          1  4 53
##
##                Accuracy : 0.9646
##                  95% CI : (0.9118, 0.9903)
##     No Information Rate : 0.531
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.9292
##
##  Mcnemar's Test P-Value : 0.1336
##
##             Sensitivity : 0.9333
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9298
##              Prevalence : 0.5310
##          Detection Rate : 0.4956
##    Detection Prevalence : 0.4956
##       Balanced Accuracy : 0.9667
##
##        'Positive' Class : 0
##
```

```r
# Convertir la matriz de confusión a un dataframe
conf_matrix_df_logreg <- as.data.frame(conf_matrix_logreg$table)

# Crear el gráfico de la matriz de confusión
ggplot(data = conf_matrix_df_logreg, aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = 1) +
  scale_fill_gradient(low = "yellow", high = "green") +
  labs(title = "Confusion Matrix of Logistic Regression",
       x = "Predicted",
       y = "Actual") +
  theme_minimal()
```



Random Forest

```r
# Random Forest
rf <- randomForest(x = x_train_normalized_df, y = y_train)
```

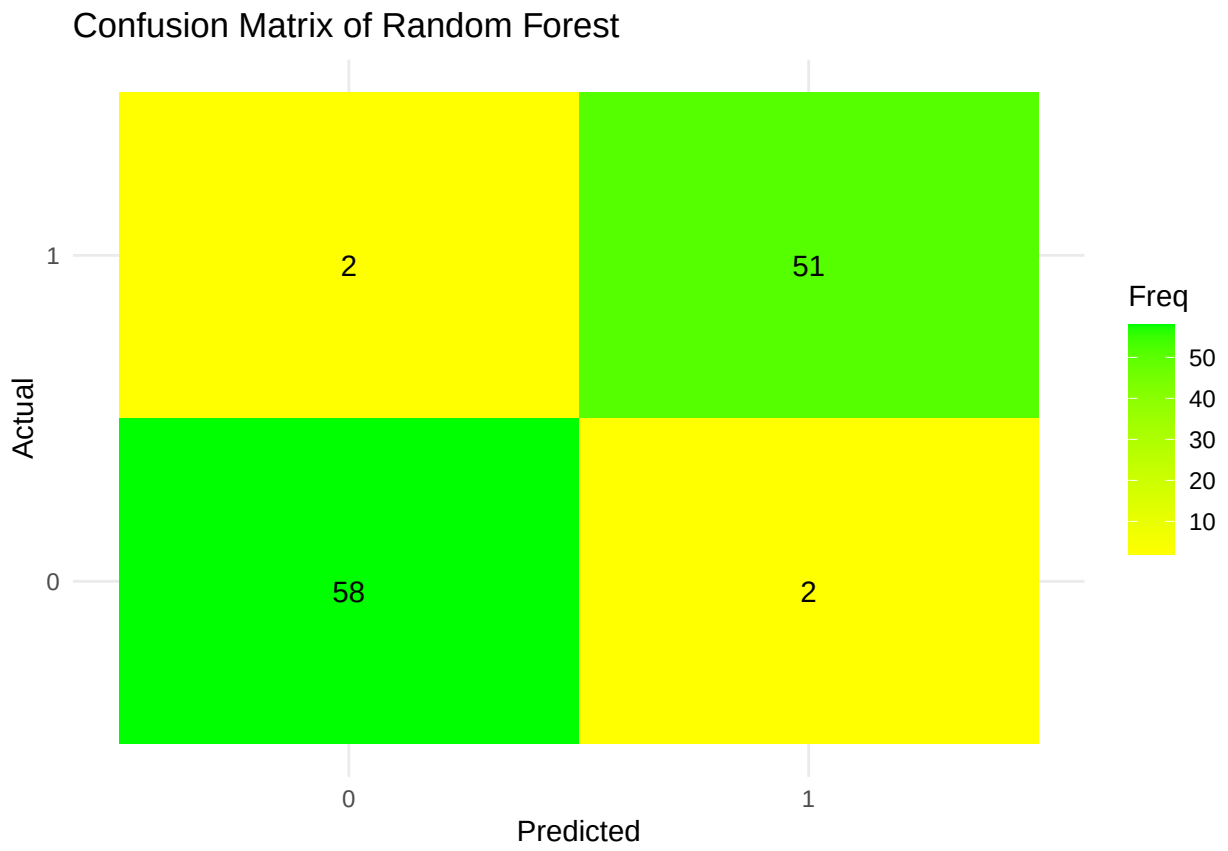
```
## Warning in randomForest.default(x = x_train_normalized_df, y = y_train): The
## response has five or fewer unique values.  Are you sure you want to do
## regression?
```

```r
# Predicciones
y_pred_rf <- predict(rf, newdata = x_test_normalized_df, type = "response")

# Redondear las probabilidades a 0 o 1
```

```r
y_pred_rounded <- ifelse(y_pred_rf >= 0.5, 1, 0)

# Crear un dataframe con las predicciones redondeadas y las etiquetas verdaderas
predictions_df_rf <- data.frame(Predicted = y_pred_rounded, Actual = y_test_df)

# Convertir las columnas a factores
predictions_df_rf$Predicted <- as.factor(predictions_df_rf$Predicted)
predictions_df_rf$y_test <- as.factor(predictions_df_rf$y_test)

# Crear la matriz de confusión
conf_matrix_rf <- confusionMatrix(predictions_df_rf$Predicted, predictions_df_rf$y_test)

conf_matrix_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 58  2
##          1  2 51
##
##                Accuracy : 0.9646
##                  95% CI : (0.9118, 0.9903)
##     No Information Rate : 0.531
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.9289
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9667
##             Specificity : 0.9623
##          Pos Pred Value : 0.9667
##          Neg Pred Value : 0.9623
##              Prevalence : 0.5310
##          Detection Rate : 0.5133
##    Detection Prevalence : 0.5310
##       Balanced Accuracy : 0.9645
##
##        'Positive' Class : 0
##
```

```r
# Convertir la matriz de confusión a un dataframe
conf_matrix_df_rf <- as.data.frame(conf_matrix_rf$table)

# Crear el gráfico de la matriz de confusión
ggplot(data = conf_matrix_df_rf, aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = 1) +
  scale_fill_gradient(low = "yellow", high = "green") +
  labs(title = "Confusion Matrix of Random Forest",
       x = "Predicted",
       y = "Actual") +
  theme_minimal()
```



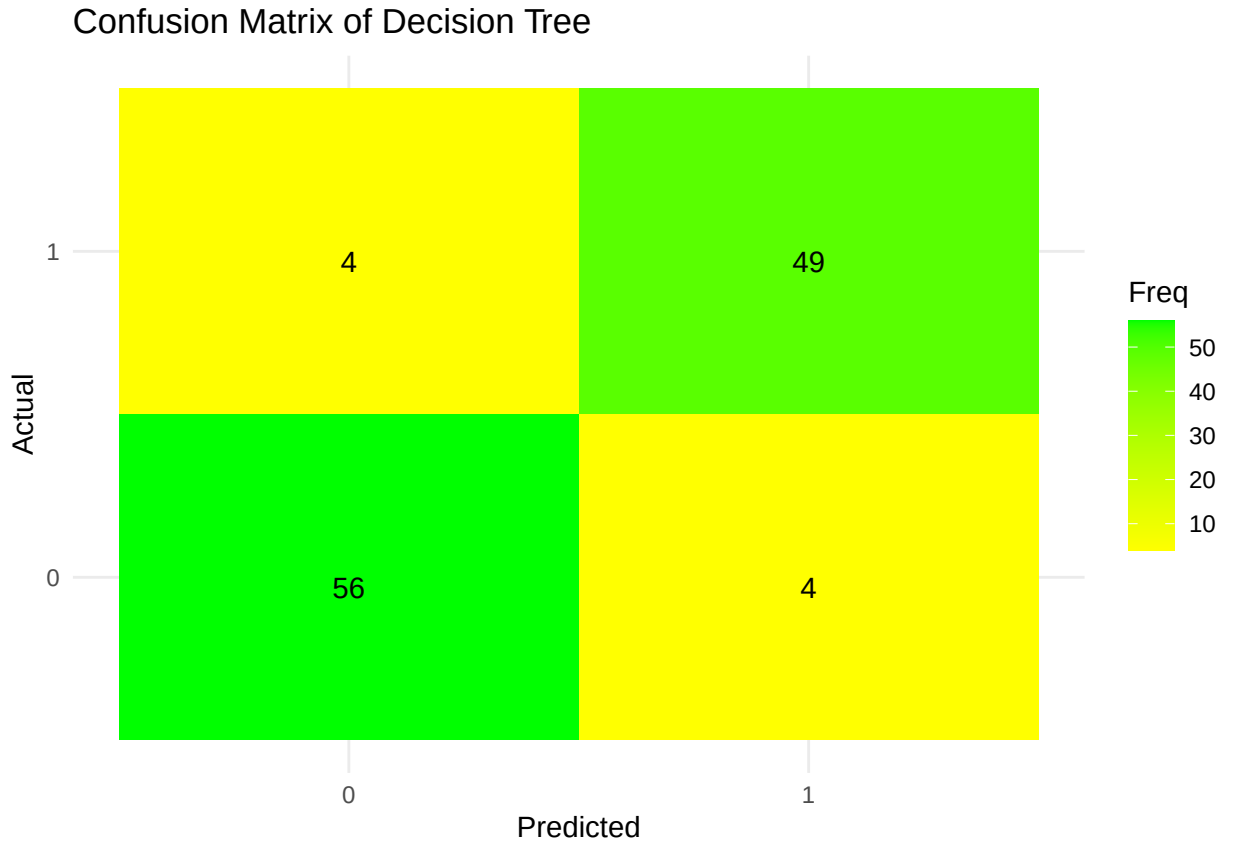
Árbol de decisión

```
# Árbol de decisión
dt <- rpart(y_train ~ ., data = x_train_normalized_df, method = "class")

# Predicciones
y_pred_dt <- predict(dt, newdata = x_test_normalized_df, type = "class")

# Crear un dataframe con las predicciones redondeadas y las etiquetas verdaderas
predictions_df_dt <- data.frame(Predicted = y_pred_dt, Actual = y_test)

# Convertir las columnas a factores
predictions_df_dt$Predicted <- as.factor(predictions_df_dt$Predicted)
predictions_df_dt$Actual <- as.factor(predictions_df_dt$Actual)

# Crear la matriz de confusión
conf_matrix_dt <- confusionMatrix(predictions_df_dt$Predicted, predictions_df_dt$Actual)

conf_matrix_dt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 56  4
```

```
##          1  4 49
##
##                Accuracy : 0.9292
##                  95% CI : (0.8653, 0.9689)
##     No Information Rate : 0.531
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.8579
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9333
##             Specificity : 0.9245
##          Pos Pred Value : 0.9333
##          Neg Pred Value : 0.9245
##              Prevalence : 0.5310
##          Detection Rate : 0.4956
##    Detection Prevalence : 0.5310
##       Balanced Accuracy : 0.9289
##
##        'Positive' Class : 0
##
```

```r
# Convertir la matriz de confusión a un dataframe
conf_matrix_df_dt <- as.data.frame(conf_matrix_dt$table)

# Crear el gráfico de la matriz de confusión
ggplot(data = conf_matrix_df_dt, aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = 1) +
  scale_fill_gradient(low = "yellow", high = "green") +
  labs(title = "Confusion Matrix of Decision Tree",
       x = "Predicted",
       y = "Actual") +
  theme_minimal()
```



k-NN

```
# k-NN
# Realiza la validación cruzada
ctrl <- trainControl(method="repeatedcv", number=10, repeats=3)

knn_model <- train(x = x_train_normalized_df, y = y_train, method = "knn", trControl = ctrl, tuneLength
```

```
## Warning in train.default(x = x_train_normalized_df, y = y_train, method =
## "knn", : You are trying to do regression and your outcome only has two possible
## values Are you trying to do classification? If so, use a 2 level factor as your
## outcome column.
```

```
# Muestra los resultados
print(knn_model)
```

```
## k-Nearest Neighbors
##
## 456 samples
##  30 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 410, 410, 410, 411, 410, 410, ...
## Resampling results across tuning parameters:
##
```

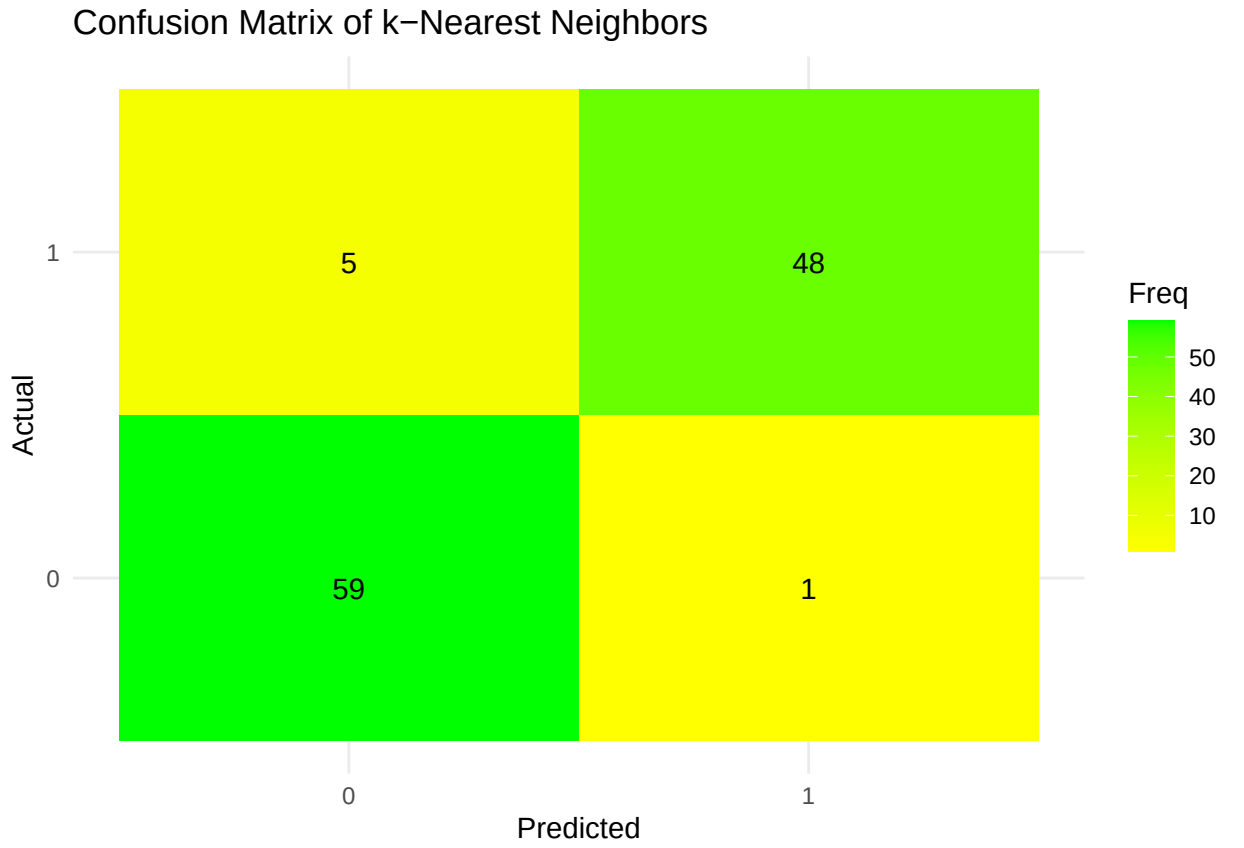
```
##   k   RMSE       Rsquared   MAE
##    5  0.1613821  0.8753196  0.05751369
##    7  0.1682515  0.8678884  0.06445997
##    9  0.1707046  0.8649690  0.06809304
##   11  0.1718982  0.8629840  0.07155504
##   13  0.1739042  0.8605567  0.07399604
##   15  0.1785590  0.8542993  0.07772437
##   17  0.1817801  0.8495293  0.07984234
##   19  0.1834582  0.8459766  0.08115044
##   21  0.1844933  0.8450195  0.08174276
##   23  0.1863032  0.8429119  0.08350565
##   25  0.1873747  0.8417286  0.08470886
##   27  0.1886738  0.8400539  0.08641490
##   29  0.1896439  0.8389536  0.08745599
##   31  0.1904614  0.8381346  0.08845031
##   33  0.1913051  0.8370930  0.08973296
##   35  0.1928557  0.8352542  0.09114848
##   37  0.1954182  0.8314844  0.09327240
##   39  0.1972237  0.8290981  0.09510488
##   41  0.1984787  0.8277111  0.09636451
##   43  0.1993448  0.8268164  0.09760211
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 5.
```

```
# k-NN
knn <- knn(x_train_normalized_df, x_test_normalized_df, y_train, k = 5)

# Crear un dataframe con las predicciones y las etiquetas verdaderas
predictions_df_knn <- data.frame(Predicted = knn, Actual = y_test_df)

# Convertir las columnas a factores
predictions_df_knn$Predicted <- as.factor(predictions_df_knn$Predicted)
predictions_df_knn$y_test <- as.factor(predictions_df_knn$y_test)

# Crear la matriz de confusión
conf_matrix_knn <- confusionMatrix(predictions_df_knn$Predicted, predictions_df_knn$y_test)

conf_matrix_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 59  5
##          1  1 48
##
##                Accuracy : 0.9469
##                  95% CI : (0.888, 0.9803)
##     No Information Rate : 0.531
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.8929
##
##  Mcnemar's Test P-Value : 0.2207
```

```
##
##             Sensitivity : 0.9833
##             Specificity : 0.9057
##          Pos Pred Value : 0.9219
##          Neg Pred Value : 0.9796
##              Prevalence : 0.5310
##          Detection Rate : 0.5221
##    Detection Prevalence : 0.5664
##       Balanced Accuracy : 0.9445
##
##        'Positive' Class : 0
##
```

```r
# Convertir la matriz de confusión a un dataframe
conf_matrix_df_knn <- as.data.frame(conf_matrix_knn$table)

# Crear el gráfico de la matriz de confusión
ggplot(data = conf_matrix_df_knn, aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = 1) +
  scale_fill_gradient(low = "yellow", high = "green") +
  labs(title = "Confusion Matrix of k-Nearest Neighbors",
       x = "Predicted",
       y = "Actual") +
  theme_minimal()
```



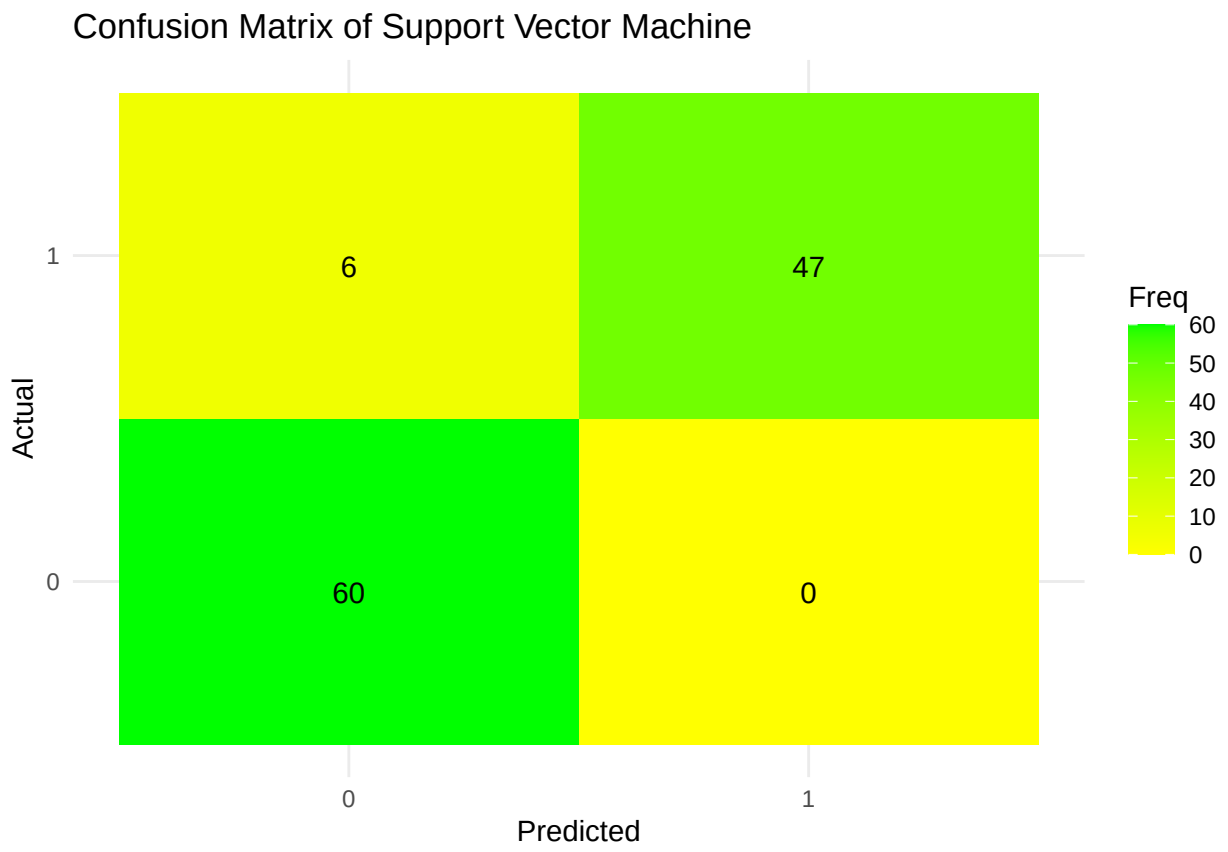
SVM

```
# SVM
svm_model <- svm(y_train ~ ., data = x_train_normalized_df)

# Predicciones
y_pred_svc <- predict(svm_model, newdata = x_test_normalized_df)

# Redondear las probabilidades a 0 o 1
y_pred_rounded <- ifelse(y_pred_svc >= 0.5, 1, 0)

# Crear un dataframe con las predicciones redondeadas y las etiquetas verdaderas
predictions_df_svc <- data.frame(Predicted = y_pred_rounded, Actual = y_test_df)

# Convertir las columnas a factores
predictions_df_svc$Predicted <- as.factor(predictions_df_svc$Predicted)
predictions_df_svc$y_test <- as.factor(predictions_df_svc$y_test)

# Crear la matriz de confusión
conf_matrix_svc <- confusionMatrix(predictions_df_svc$Predicted, predictions_df_svc$y_test)

conf_matrix_svc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 60  6
##          1  0 47
##
##                Accuracy : 0.9469
##                  95% CI : (0.888, 0.9803)
##     No Information Rate : 0.531
##     P-Value [Acc > NIR] : < 0.0000000000000002
##
##                   Kappa : 0.8927
##
##  Mcnemar's Test P-Value : 0.04123
##
##             Sensitivity : 1.0000
##             Specificity : 0.8868
##          Pos Pred Value : 0.9091
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5310
##          Detection Rate : 0.5310
##    Detection Prevalence : 0.5841
##       Balanced Accuracy : 0.9434
##
##        'Positive' Class : 0
##
```

```
# Convertir la matriz de confusión a un dataframe
conf_matrix_df_svc <- as.data.frame(conf_matrix_svc$table)

# Crear el gráfico de la matriz de confusión
```

```
ggplot(data = conf_matrix_df_svc, aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = 1) +
  scale_fill_gradient(low = "yellow", high = "green") +
  labs(title = "Confusion Matrix of Support Vector Machine",
       x = "Predicted",
       y = "Actual") +
  theme_minimal()
```



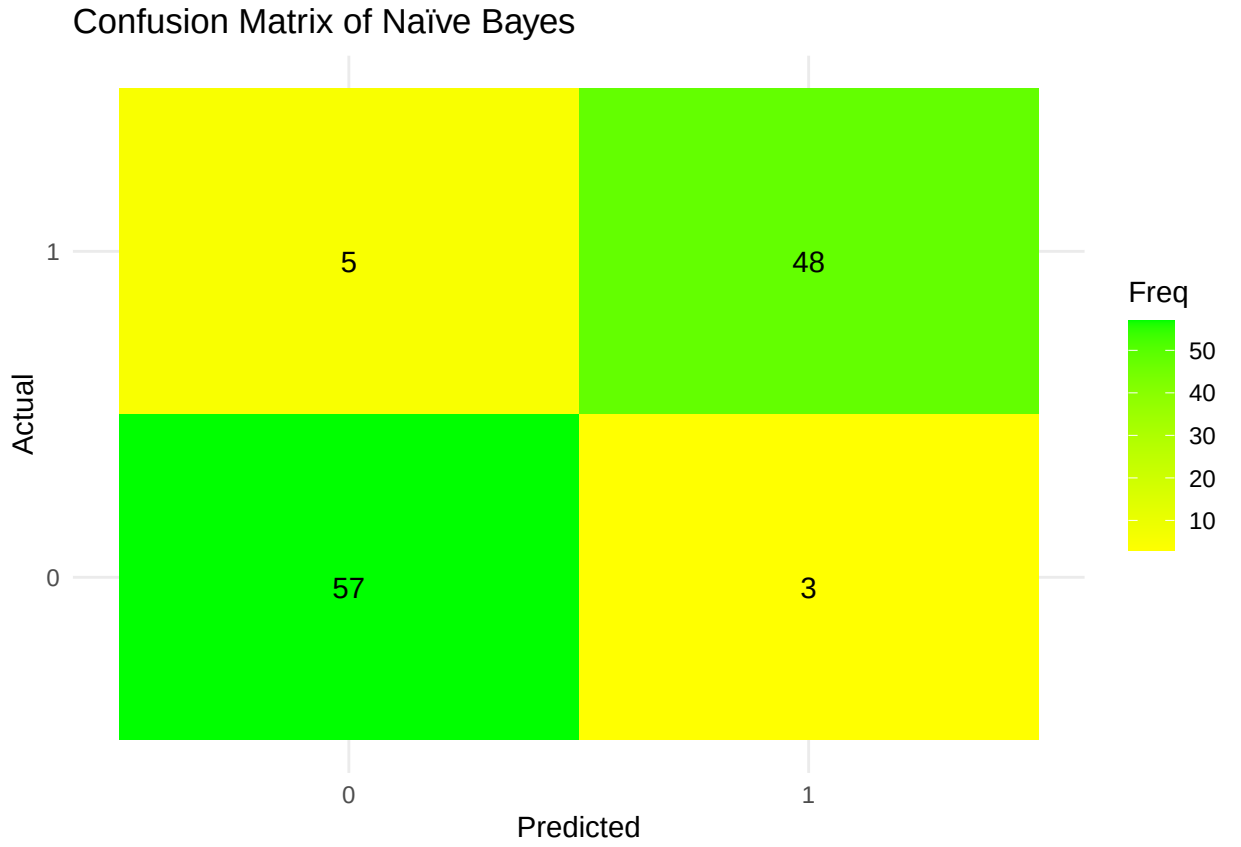
Naïve Bayes

```
# Naive Bayes
nb_model <- naiveBayes(x_train_normalized_df, y_train)

# Predicciones
y_pred_nb <- predict(nb_model, newdata = x_test_normalized_df)

# Crear un dataframe con las predicciones y las etiquetas verdaderas
predictions_df_nb <- data.frame(Predicted = y_pred_nb, Actual = y_test_df)

# Convertir las columnas a factores
predictions_df_nb$Predicted <- as.factor(predictions_df_nb$Predicted)
predictions_df_nb$y_test <- as.factor(predictions_df_nb$y_test)

# Crear la matriz de confusión
```

```r
conf_matrix_nb <- confusionMatrix(predictions_df_nb$Predicted, predictions_df_nb$y_test)

conf_matrix_nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 57  5
##          1  3 48
##
##                Accuracy : 0.9292
##                  95% CI : (0.8653, 0.9689)
##     No Information Rate : 0.531
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.8575
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9500
##             Specificity : 0.9057
##          Pos Pred Value : 0.9194
##          Neg Pred Value : 0.9412
##              Prevalence : 0.5310
##          Detection Rate : 0.5044
##    Detection Prevalence : 0.5487
##       Balanced Accuracy : 0.9278
##
##        'Positive' Class : 0
##
```

```r
# Convertir la matriz de confusión a un dataframe
conf_matrix_df_nb <- as.data.frame(conf_matrix_nb$table)

# Crear el gráfico de la matriz de confusión
ggplot(data = conf_matrix_df_nb, aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = 1) +
  scale_fill_gradient(low = "yellow", high = "green") +
  labs(title = "Confusion Matrix of Naïve Bayes",
       x = "Predicted",
       y = "Actual") +
  theme_minimal()
```



Resultados

```
# Crear un dataframe para almacenar todas las métricas
results <- data.frame(
  Model = c("Logistic Regression", "Random Forest", "Decision Tree", "k-NN", "SVM", "Naive Bayes"),
  Accuracy = c(conf_matrix_logreg$overall["Accuracy"], conf_matrix_rf$overall["Accuracy"],
               conf_matrix_dt$overall["Accuracy"], conf_matrix_knn$overall["Accuracy"],
               conf_matrix_svc$overall["Accuracy"], conf_matrix_nb$overall["Accuracy"]),
  Sensitivity = c(conf_matrix_logreg$byClass["Sensitivity"], conf_matrix_rf$byClass["Sensitivity"],
                  conf_matrix_dt$byClass["Sensitivity"], conf_matrix_knn$byClass["Sensitivity"],
                  conf_matrix_svc$byClass["Sensitivity"], conf_matrix_nb$byClass["Sensitivity"]),
  Specificity = c(conf_matrix_logreg$byClass["Specificity"], conf_matrix_rf$byClass["Specificity"],
                  conf_matrix_dt$byClass["Specificity"], conf_matrix_knn$byClass["Specificity"],
                  conf_matrix_svc$byClass["Specificity"], conf_matrix_nb$byClass["Specificity"]),
  Precision = c(conf_matrix_logreg$byClass["Precision"], conf_matrix_rf$byClass["Precision"],
                conf_matrix_dt$byClass["Precision"], conf_matrix_knn$byClass["Precision"],
                conf_matrix_svc$byClass["Precision"], conf_matrix_nb$byClass["Precision"])
)

results
```

```
##                 Model  Accuracy Sensitivity Specificity Precision
## 1 Logistic Regression 0.9646018   0.9333333   1.0000000 1.0000000
## 2       Random Forest 0.9646018   0.9666667   0.9622642 0.9666667
## 3       Decision Tree 0.9292035   0.9333333   0.9245283 0.9333333
## 4               k-NN 0.9469027   0.9833333   0.9056604 0.9218750
```

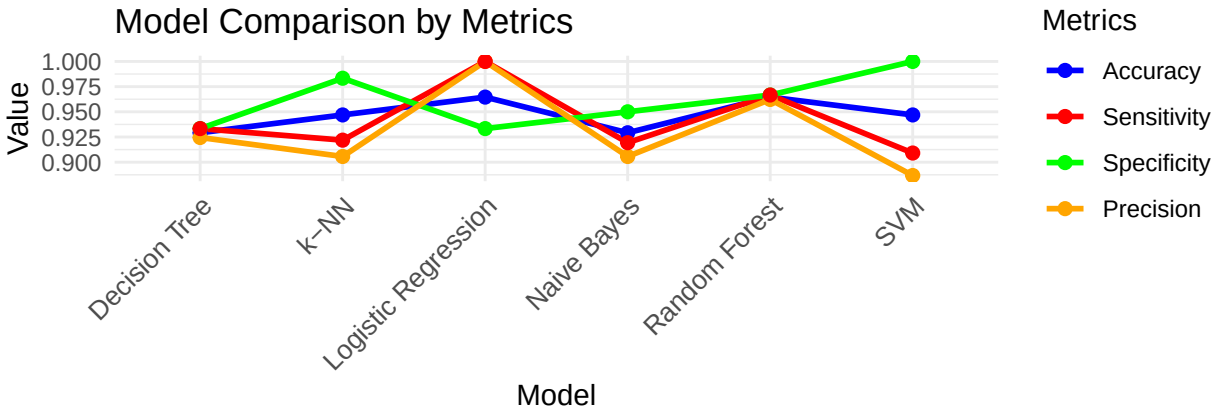
```
## 5                 SVM 0.9469027   1.0000000   0.8867925 0.9090909
## 6         Naive Bayes 0.9292035   0.9500000   0.9056604 0.9193548
```

Visualización de resultados

```
# Agrupar fila por métrica y modelo
results_line <- tidyr::gather(results, Metric, Value, -Model)

# Crear el gráfico de líneas
line_plot <- ggplot(results_line, aes(x = Model, y = Value, group = Metric, color = Metric)) +
  geom_line(size=1) +
  geom_point(size=2) +
  labs(title = "Model Comparison by Metrics",
       y = "Value",
       color = "Metrics") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 10)) +
  scale_color_manual(values = c("Accuracy" = "blue", "Sensitivity" = "green",
                                "Specificity" = "orange", "Precision" = "red"),
                     labels = c("Accuracy", "Sensitivity", "Specificity", "Precision"))
```

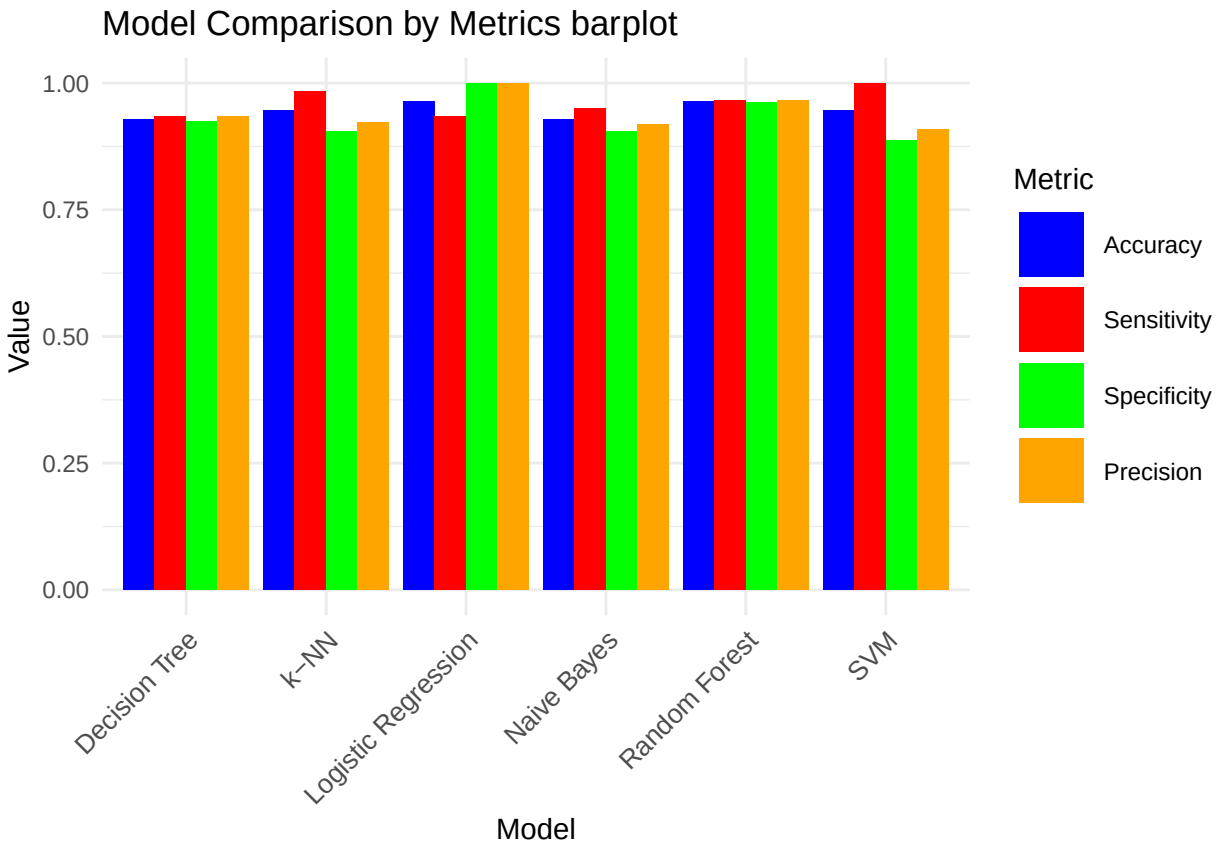
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# Crear la tabla de resumen
results_table <- tableGrob(results, rows = NULL)

# Combinar gráfico
grid = grid.arrange(line_plot, results_table, ncol = 1, heights = c(1, 1))
```



Model	Accuracy	Sensitivity	Specificity	Precision
Logistic Regression	0.9646018	0.9333333	1.0000000	1.0000000
Random Forest	0.9646018	0.9666667	0.9622642	0.9666667
Decision Tree	0.9292035	0.9333333	0.9245283	0.9333333
k−NN	0.9469027	0.9833333	0.9056604	0.9218750
SVM	0.9469027	1.0000000	0.8867925	0.9090909
Naive Bayes	0.9292035	0.9500000	0.9056604	0.9193548

```r
# Reordenar el dataframe para que las métricas aparezcan en el orden del gráfico de líneas
results_line$Metric <- factor(results_line$Metric, levels = c("Accuracy", "Sensitivity", "Specificity",

# Gráfico de barras
ggplot(results_line, aes(x = Model, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = "dodge", size = 10) +
  labs(title = "Model Comparison by Metrics barplot",
       y = "Value",
       x = "Model",
       fill = "Metric") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 10)) +
  scale_fill_manual(values = c("Accuracy" = "blue", "Sensitivity" = "red",
                               "Specificity" = "green", "Precision" = "orange"))
```


Model Comparison by Metrics barplot
1.00
0.75
0.50
0.25
0.00
Value
Decision Tree
k−NN
Logistic Regression
Naive Bayes
Random Forest
SVM
Model
Metric
Accuracy
Sensitivity
Specificity
Precision