Breast Cancer Analysis

Thiago Silva
04/26/2021

Breast cancer is the most common malignancy among women, accounting for nearly 1 in 3 cancers diagnosed among women in the United States, and is the second leading cause of cancer death among women. Breast cancer occurs as a result of the abnormal growth of cells in the breast tissue, commonly referred to as a tumor. A tumor does not mean cancer - tumors can be benign (non-cancerous), pre-malignant (precancerous) or malignant (cancerous). Tests such as MRI, mammography, ultrasound and biopsy are commonly used to diagnose breast cancer.
This dataset was created by Dr. William H. Wolberg, a physician at the University of Wisconsin Hospital in Madison, Wisconsin, USA. To create the dataset, Dr. Wolberg used fluid samples taken from patients with solid breast masses and an easy-to-use computer graphics program called Xcyt, which is capable of performing analysis of cytological features based on a digital scan. The program uses a curve-fitting algorithm to calculate ten features from each of the sample cells, then calculates the mean value, extreme value and standard error of each feature for the image, returning a vector of 30 reais.
Attribute Information:
ID number 2) Diagnosis (M = malignant, B = benign) 3-32) Ten real value features are calculated for each cell nucleus:
radius (average of distances from the center to perimeter points) texture (standard deviation of grayscale values) perimeter area smoothness (local variation in radius lengths) compactness (perimeter ^ 2 / area - 1.0) concavity (gravity of concave portions of the contour) concave points (number of concave portions of the contour) symmetry fractal dimension (“coastal approach” - 1)
Purpose: This analysis aims to look at which features are most useful in predicting malignant or benign cancer and to see general trends that can help us in model selection and hyperparameter selection. The objective is to classify whether breast cancer is benign or malignant. To achieve this, we use machine learning classification methods to fit a function that can predict the discrete class of new input data.
This database is found in the “mclust” package and can be loaded into R using the code below:

```
library(mclust)
```

```
## Package 'mclust' version 5.4.7
## Type 'citation("mclust")' for citing this R package in publications.
```

```
BD <- wdbc
```

Demonstrating the data table

```
head(BD, 10)
```

```
##          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
## 7    844359         M       18.25        19.98         119.60    1040.0
## 8  84458202         M       13.71        20.83          90.20     577.9
## 9    844981         M       13.00        21.82          87.50     519.8
## 10 84501001         M       12.46        24.04          83.97     475.9
##    Smoothness_mean Compactness_mean Concavity_mean Nconcave_mean Symmetry_mean
## 1          0.11840          0.27760        0.30010       0.14710        0.2419
## 2          0.08474          0.07864        0.08690       0.07017        0.1812
## 3          0.10960          0.15990        0.19740       0.12790        0.2069
## 4          0.14250          0.28390        0.24140       0.10520        0.2597
## 5          0.10030          0.13280        0.19800       0.10430        0.1809
## 6          0.12780          0.17000        0.15780       0.08089        0.2087
## 7          0.09463          0.10900        0.11270       0.07400        0.1794
## 8          0.11890          0.16450        0.09366       0.05985        0.2196
## 9          0.12730          0.19320        0.18590       0.09353        0.2350
## 10         0.11860          0.23960        0.22730       0.08543        0.2030
##    Fractaldim_mean Radius_se Texture_se Perimeter_se Area_se Smoothness_se
## 1          0.07871    1.0950     0.9053        8.589  153.40      0.006399
## 2          0.05667    0.5435     0.7339        3.398   74.08      0.005225
## 3          0.05999    0.7456     0.7869        4.585   94.03      0.006150
## 4          0.09744    0.4956     1.1560        3.445   27.23      0.009110
## 5          0.05883    0.7572     0.7813        5.438   94.44      0.011490
## 6          0.07613    0.3345     0.8902        2.217   27.19      0.007510
## 7          0.05742    0.4467     0.7732        3.180   53.91      0.004314
## 8          0.07451    0.5835     1.3770        3.856   50.96      0.008805
## 9          0.07389    0.3063     1.0020        2.406   24.32      0.005731
## 10         0.08243    0.2976     1.5990        2.039   23.94      0.007149
##    Compactness_se Concavity_se Nconcave_se Symmetry_se Fractaldim_se
## 1         0.04904      0.05373     0.01587     0.03003      0.006193
## 2         0.01308      0.01860     0.01340     0.01389      0.003532
## 3         0.04006      0.03832     0.02058     0.02250      0.004571
## 4         0.07458      0.05661     0.01867     0.05963      0.009208
## 5         0.02461      0.05688     0.01885     0.01756      0.005115
## 6         0.03345      0.03672     0.01137     0.02165      0.005082
## 7         0.01382      0.02254     0.01039     0.01369      0.002179
## 8         0.03029      0.02488     0.01448     0.01486      0.005412
## 9         0.03502      0.03553     0.01226     0.02143      0.003749
## 10        0.07217      0.07743     0.01432     0.01789      0.010080
##    Radius_extreme Texture_extreme Perimeter_extreme Area_extreme
## 1           25.38           17.33            184.60       2019.0
## 2           24.99           23.41            158.80       1956.0
## 3           23.57           25.53            152.50       1709.0
## 4           14.91           26.50             98.87        567.7
## 5           22.54           16.67            152.20       1575.0
## 6           15.47           23.75            103.40        741.6
## 7           22.88           27.66            153.20       1606.0
## 8           17.06           28.14            110.60        897.0
## 9           15.49           30.73            106.20        739.3
## 10          15.09           40.68             97.65        711.4
##    Smoothness_extreme Compactness_extreme Concavity_extreme Nconcave_extreme
## 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
## 7             0.1442             0.2576           0.3784          0.1932
## 8             0.1654             0.3682           0.2678          0.1556
## 9             0.1703             0.5401           0.5390          0.2060
## 10            0.1853             1.0580           1.1050          0.2210
##    Symmetry_extreme Fractaldim_extreme
## 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
## 7            0.3063            0.08368
## 8            0.3196            0.11510
## 9            0.4378            0.10720
## 10           0.4366            0.20750
```

Checking dataframe dimensions

```
dim(BD)
```

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

In this case, we have 569 rows and 32 columns.

Checking proportion of benign and malignant data

```
table(BD$Diagnosis)
```

```
##
##   B   M
## 357 212
```

```
prop.table(table(BD$Diagnosis))
```

```
##
##         B         M
## 0.6274165 0.3725835
```

The distribution of breast cancer in this database is divided into 62% benign cases and 37% malignant cases. To ensure the quality of the analyses, we check the percentage of missing values in each column.

```
anyNA(BD)
```

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

```
NAS <- round(colSums(is.na(BD))*100/nrow(BD), 2)
 NAS
```

```
##                  ID           Diagnosis         Radius_mean        Texture_mean
##                   0                   0                   0                   0
##      Perimeter_mean           Area_mean     Smoothness_mean    Compactness_mean
##                   0                   0                   0                   0
##      Concavity_mean       Nconcave_mean       Symmetry_mean     Fractaldim_mean
##                   0                   0                   0                   0
##           Radius_se          Texture_se        Perimeter_se             Area_se
##                   0                   0                   0                   0
##       Smoothness_se      Compactness_se        Concavity_se         Nconcave_se
##                   0                   0                   0                   0
##         Symmetry_se       Fractaldim_se      Radius_extreme     Texture_extreme
##                   0                   0                   0                   0
##   Perimeter_extreme        Area_extreme  Smoothness_extreme Compactness_extreme
##                   0                   0                   0                   0
##   Concavity_extreme    Nconcave_extreme    Symmetry_extreme  Fractaldim_extreme
##                   0                   0                   0                   0
```

As there was no missing data in our analysis, we continued the analysis.

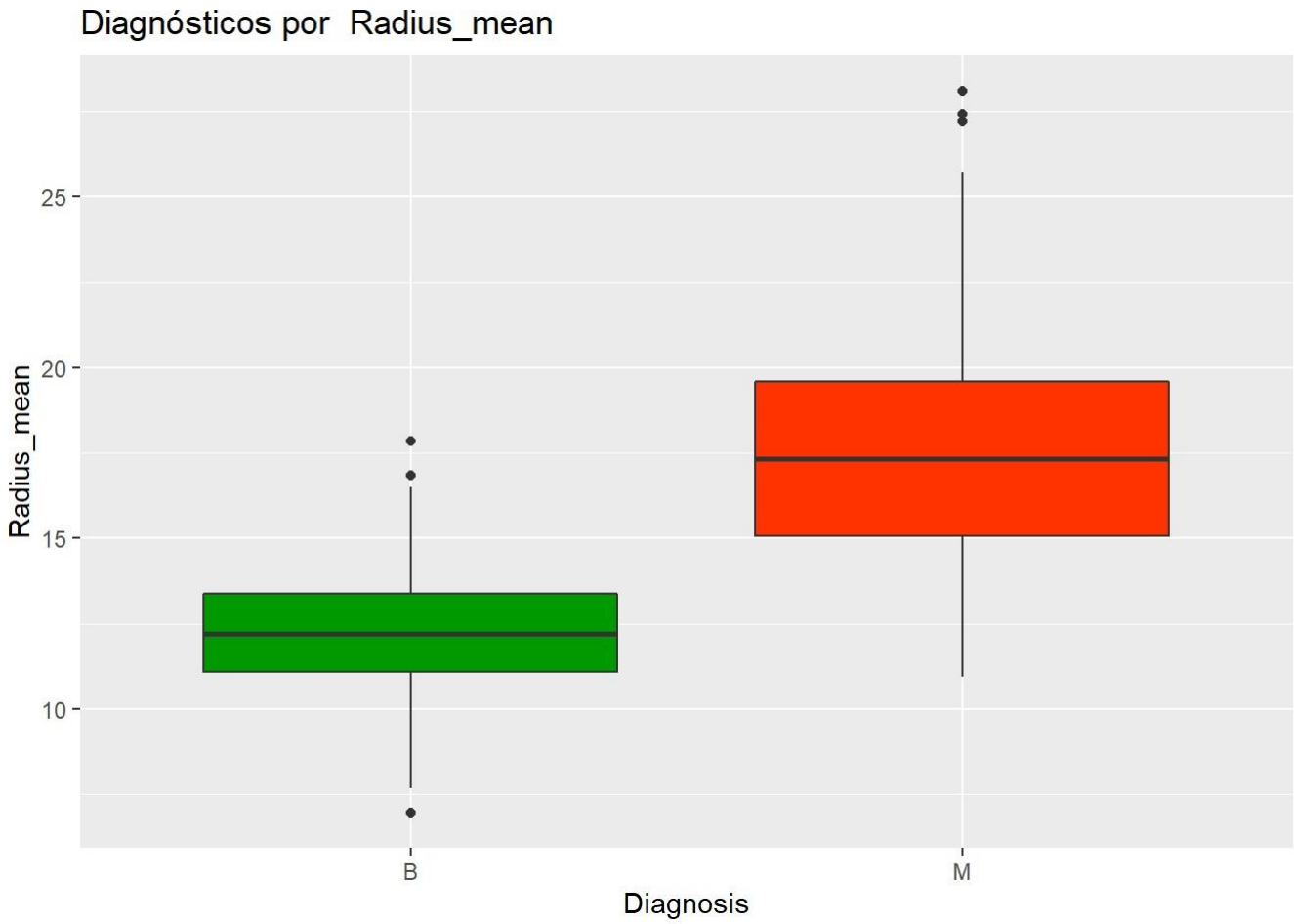
In our base there is a Variable ID that will not be useful to us, so we removed it.

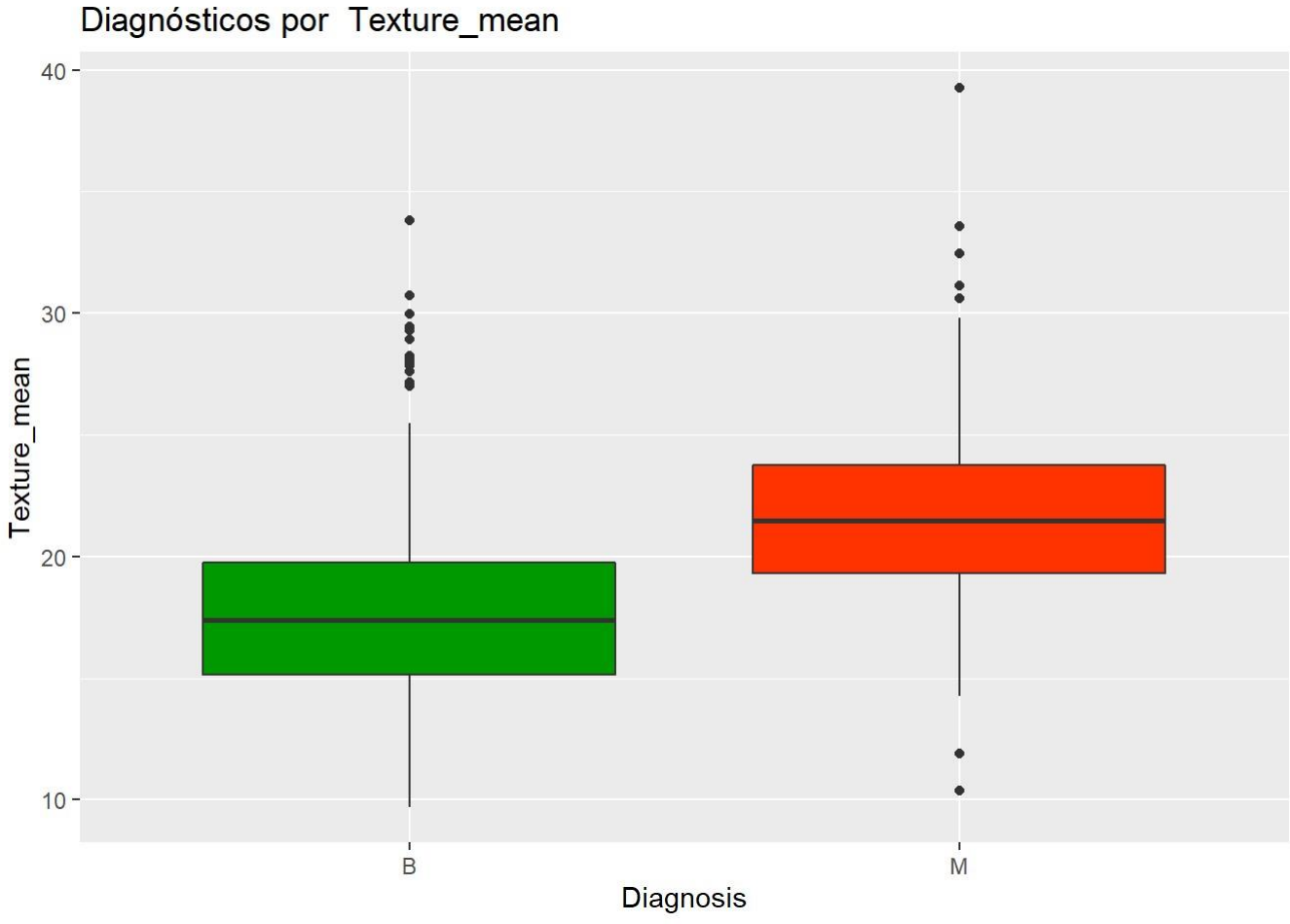
```
BD <- BD[,-1]
```

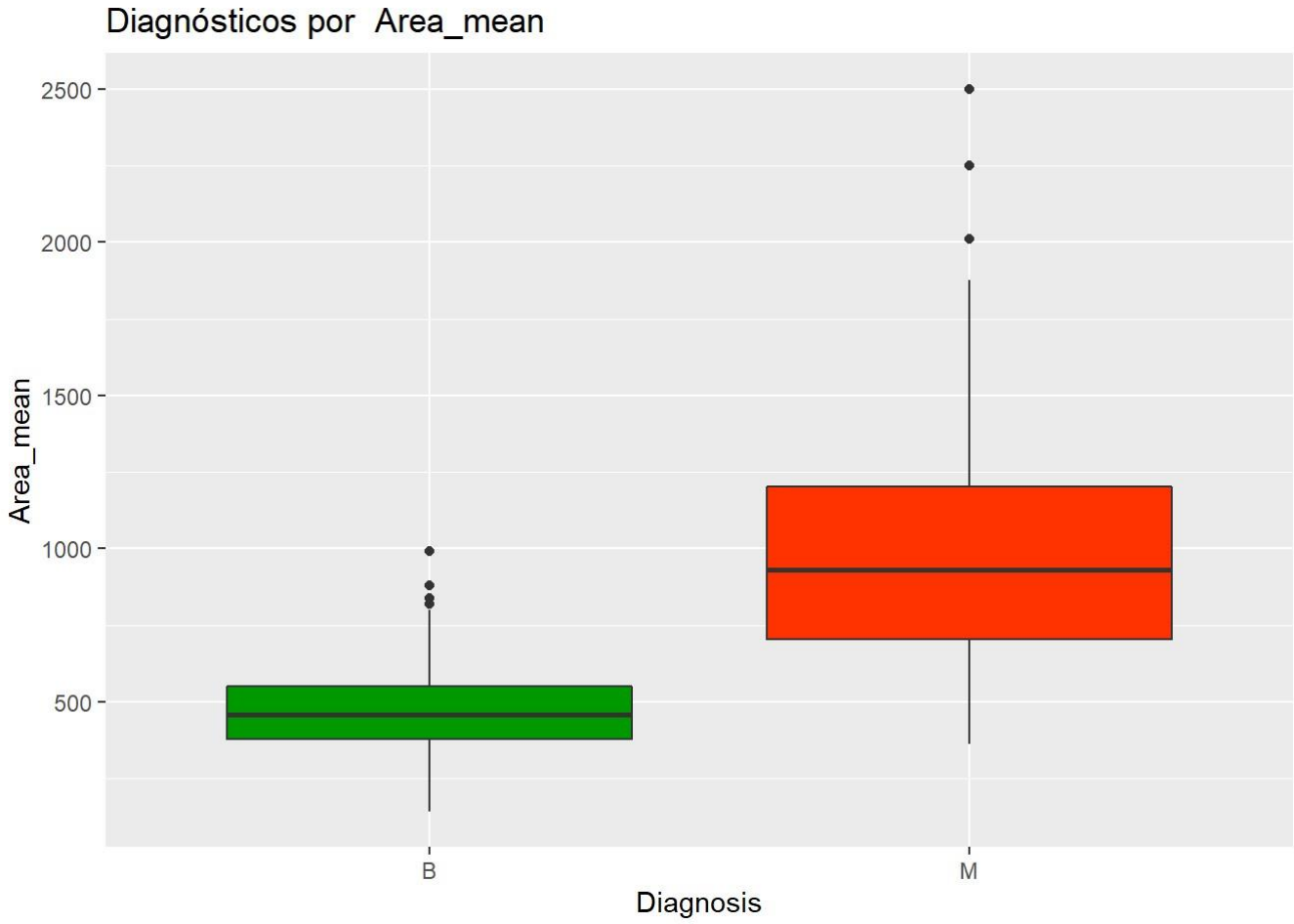
To understand a little more about our base, we visualize how the diagnoses are distributed when crossed with the other variables through boxplots.

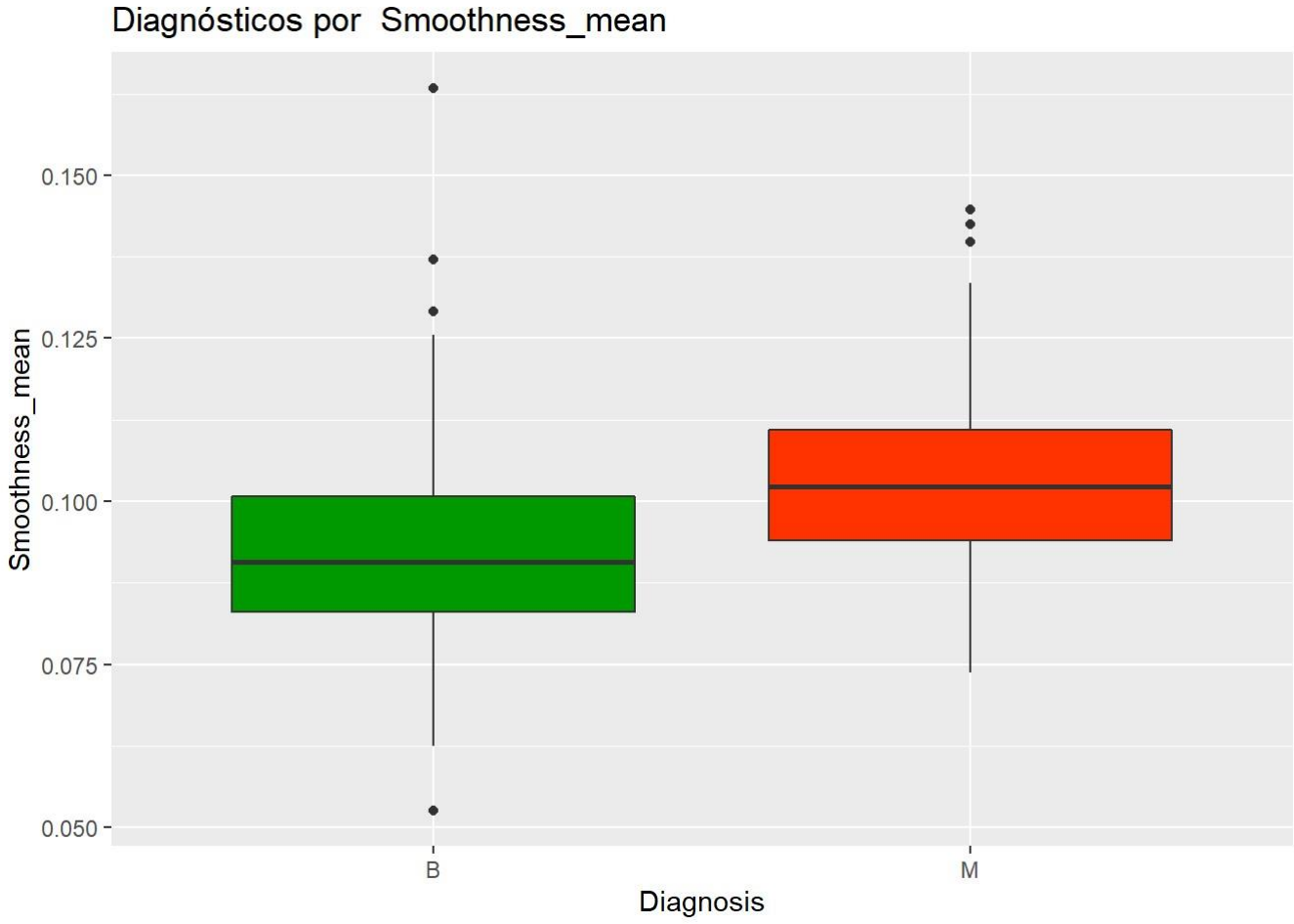
```
library(ggplot2)

box.plot <- function(x){
  if(is.numeric(BD[, x])){
    ggplot(BD, aes_string('Diagnosis', x)) +
      geom_boxplot( fill=c('#009900','#FF3300')) +
      ggtitle(paste('Diagnósticos por ', x))
  }
}

col.names = names(BD)
sapply(col.names, box.plot)
```

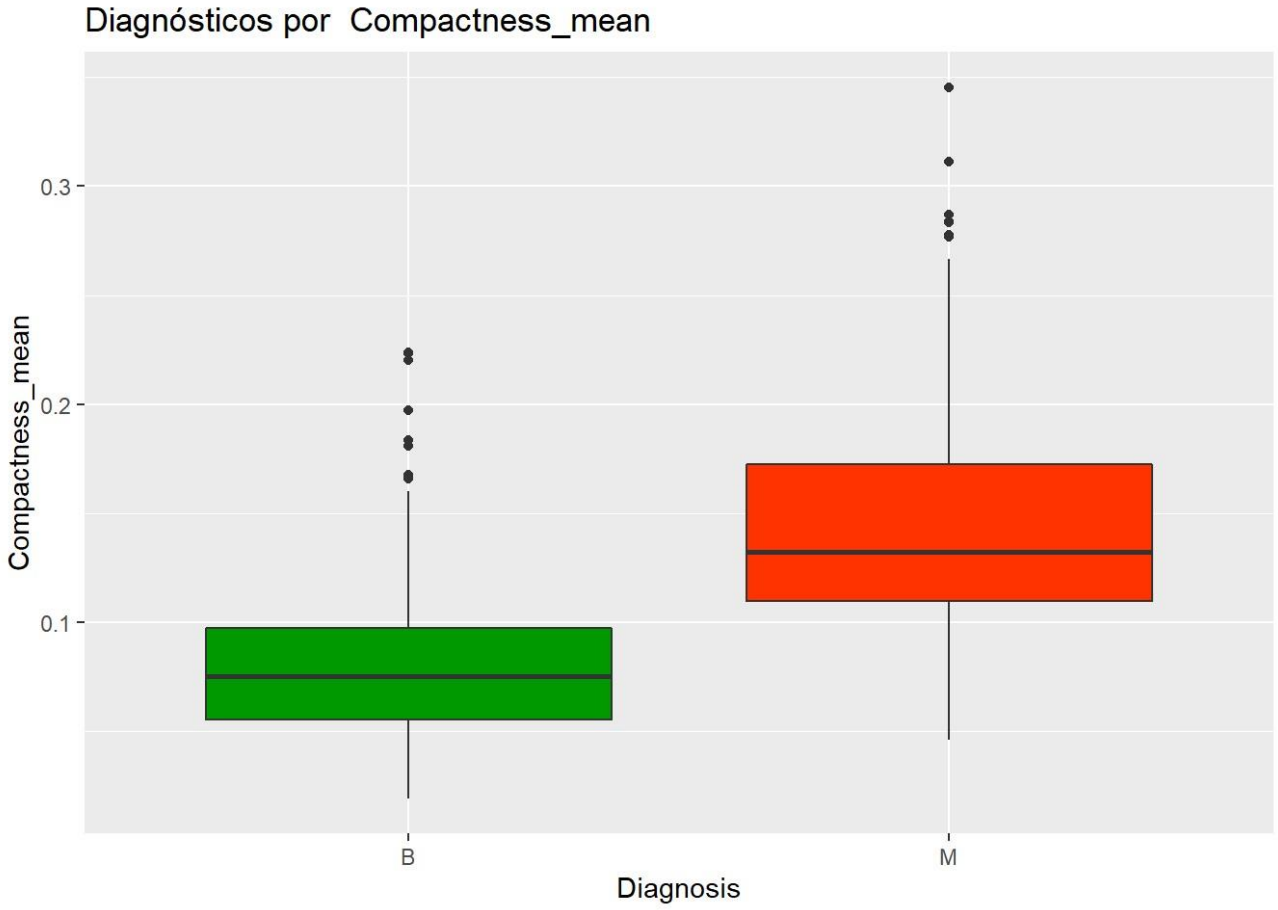
```
## $Diagnosis
## NULL
##
## $Radius_mean
```

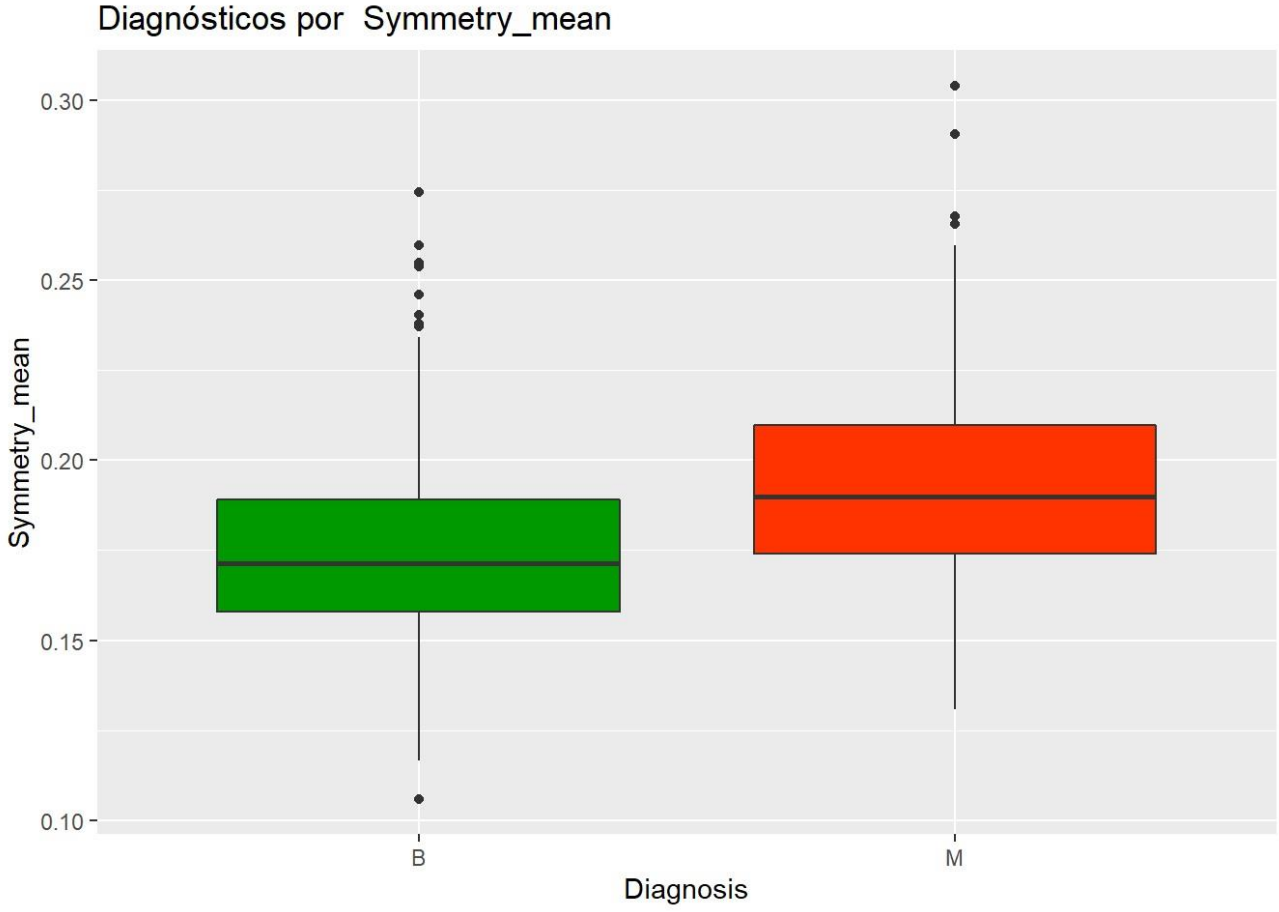

Diagnósticos por Radius_mean
Radius_mean
25
20
15
10
B
M
Diagnosis

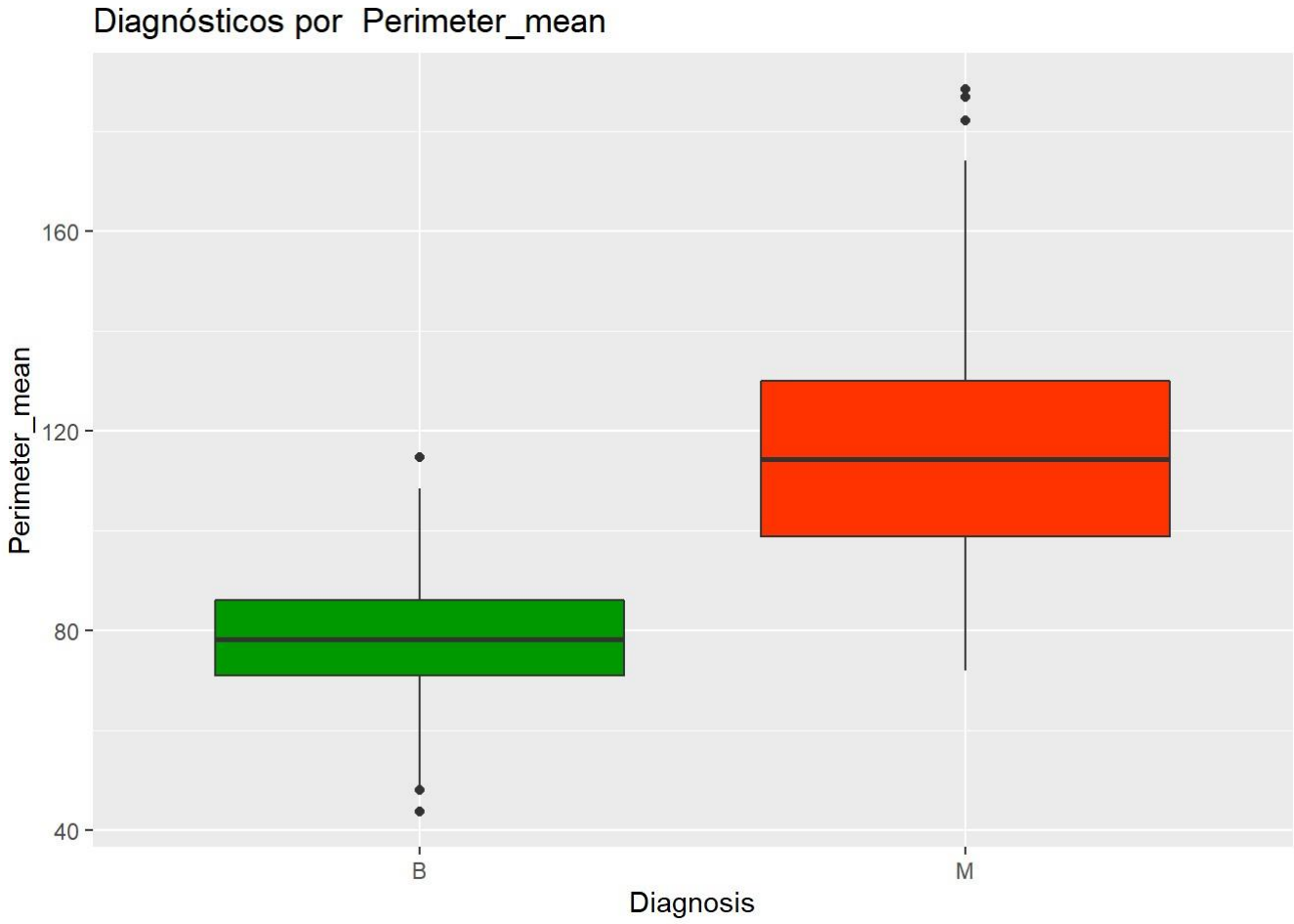

Diagnósticos por Texture_mean
Texture_mean
40
30
20
10
B
M
Diagnosis

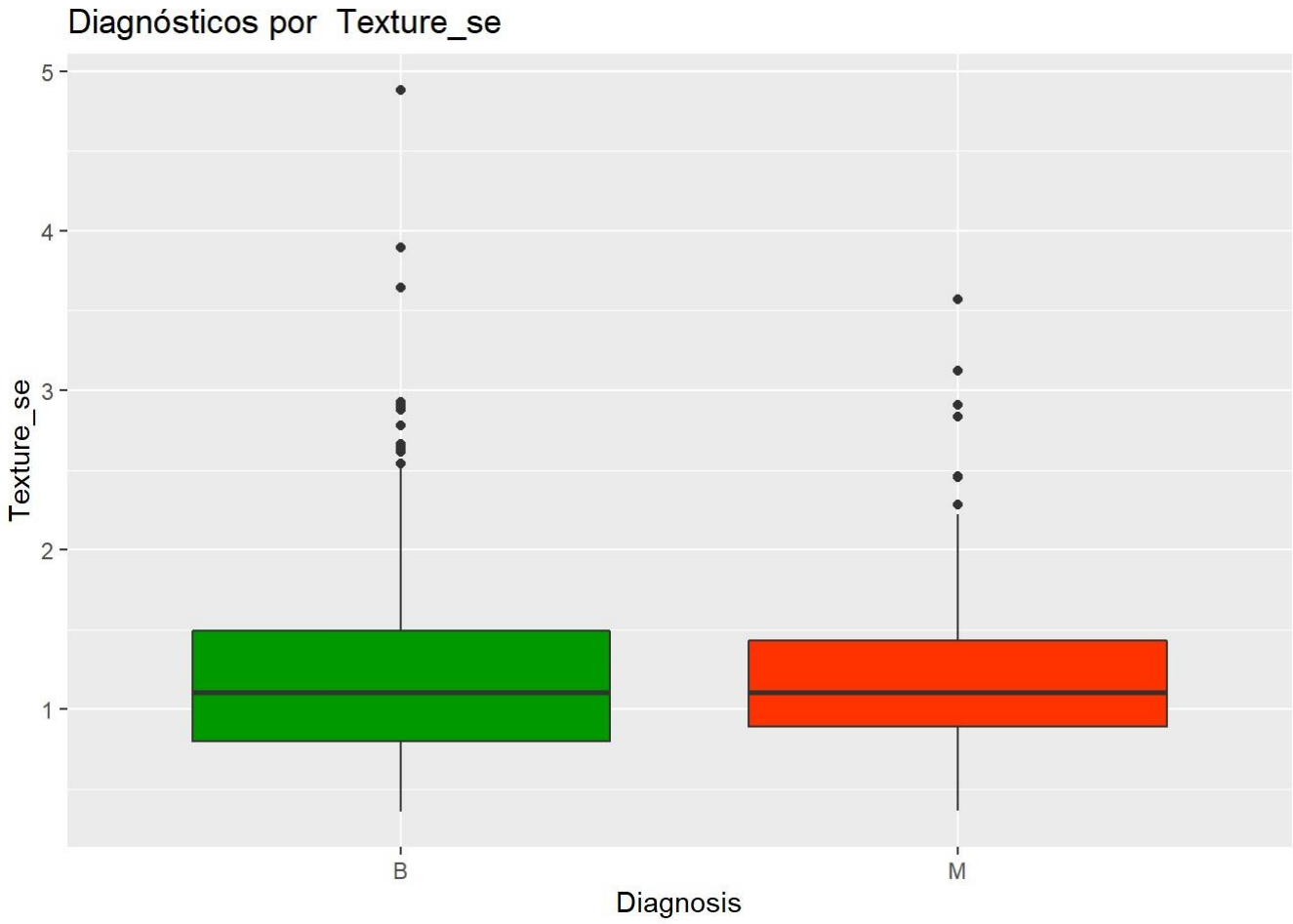

Diagnósticos por Area_mean
2500
2000
1500
1000
500
Area_mean
B
M
Diagnosis

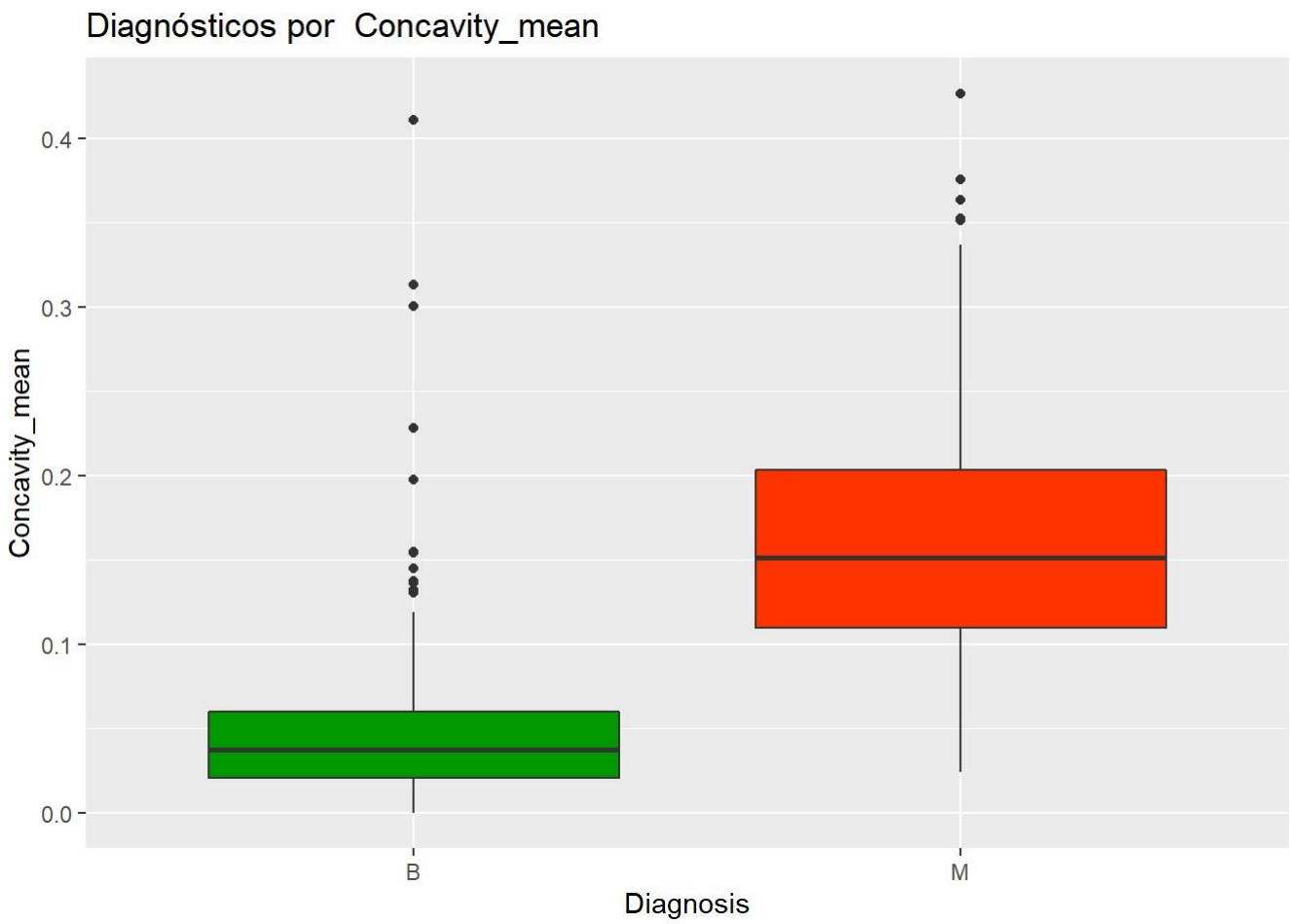

Diagnósticos por Smoothness_mean
0.150
0.125
0.100
0.075
0.050
Smoothness_mean
B
M
Diagnosis

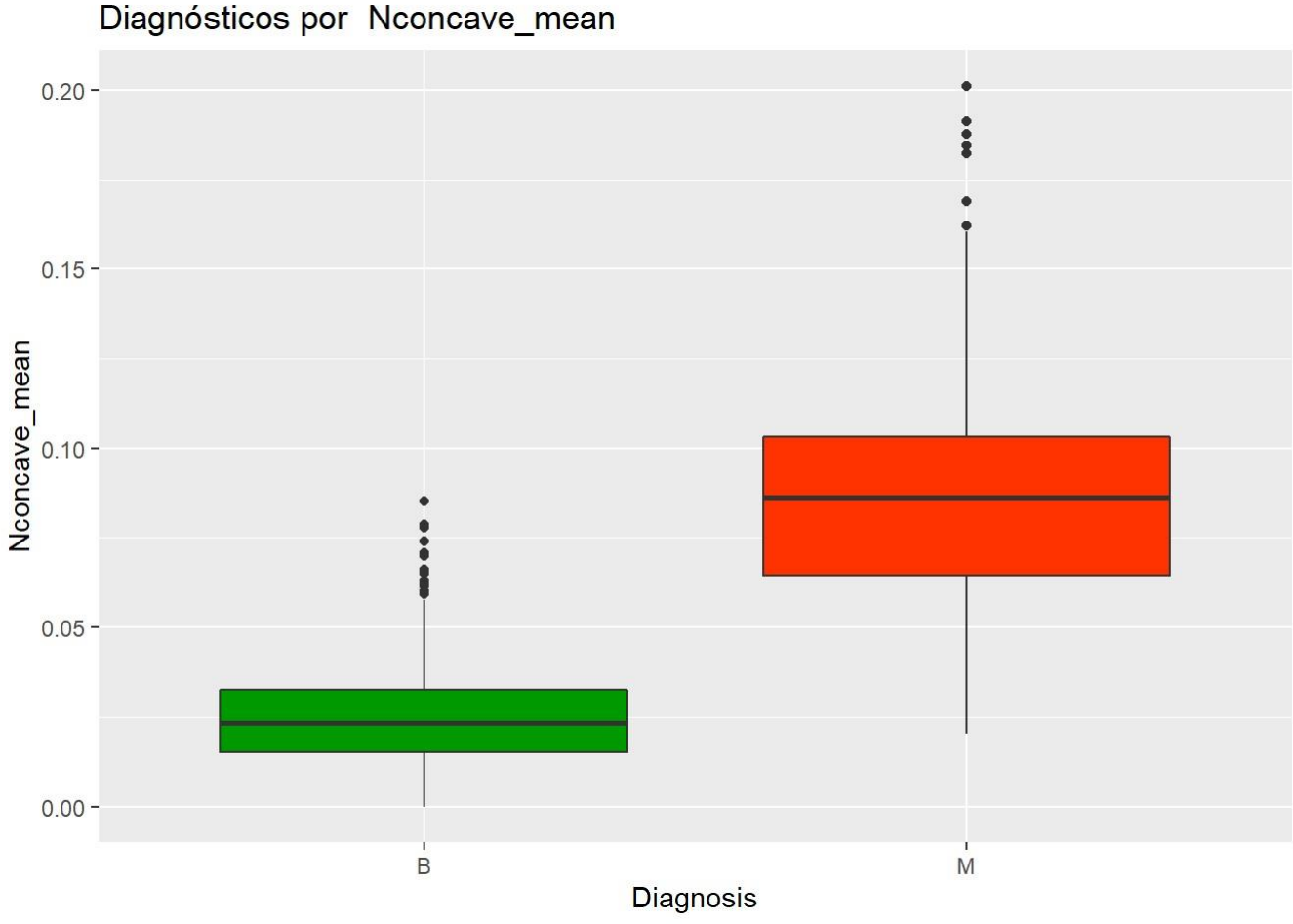

Diagnósticos por Compactness_mean
Compactness_mean
0.3
0.2
0.1
B
M
Diagnosis

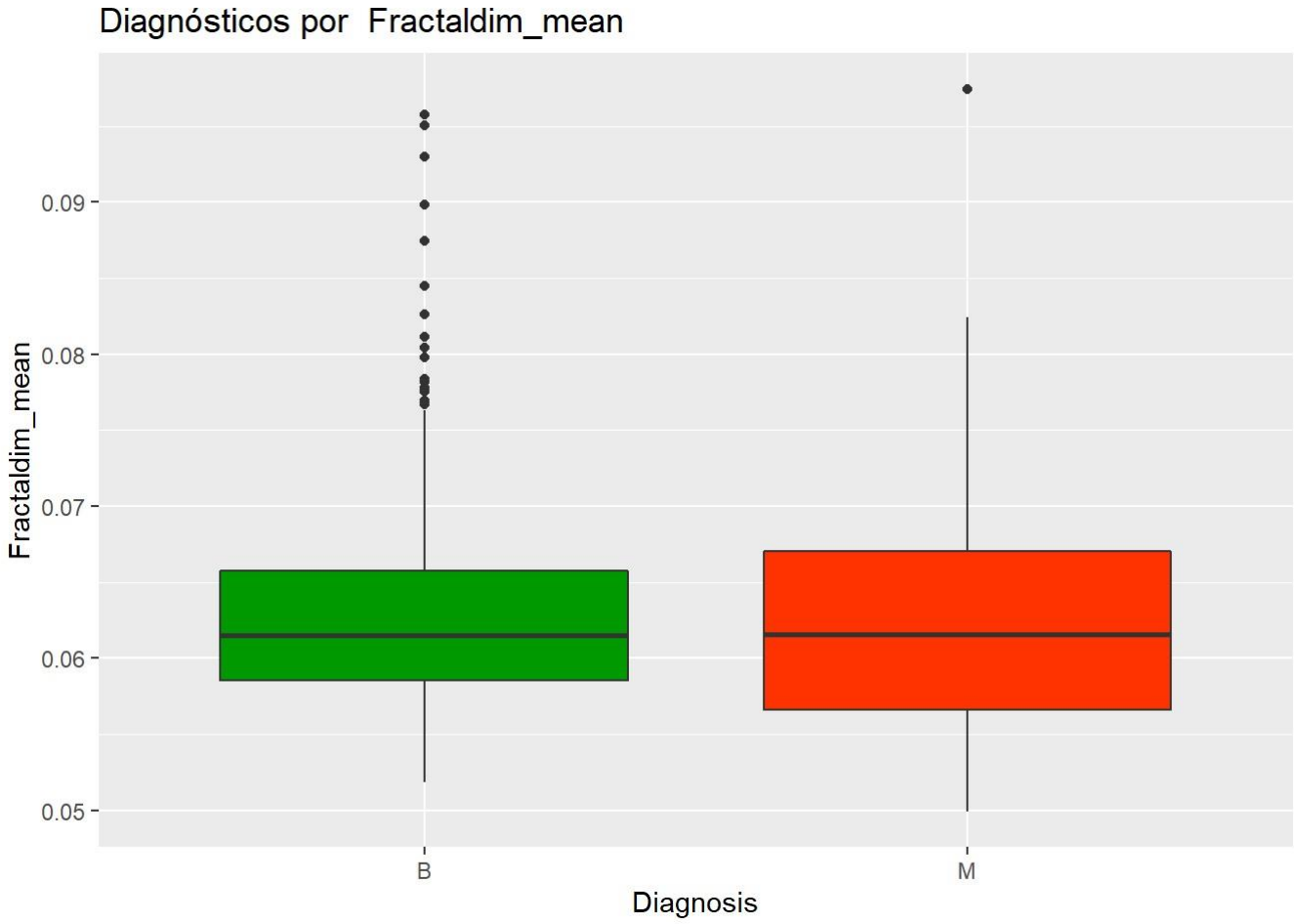

Diagnósticos por Symmetry_mean
Symmetry_mean
0.30
0.25
0.20
0.15
0.10
B
M
Diagnosis

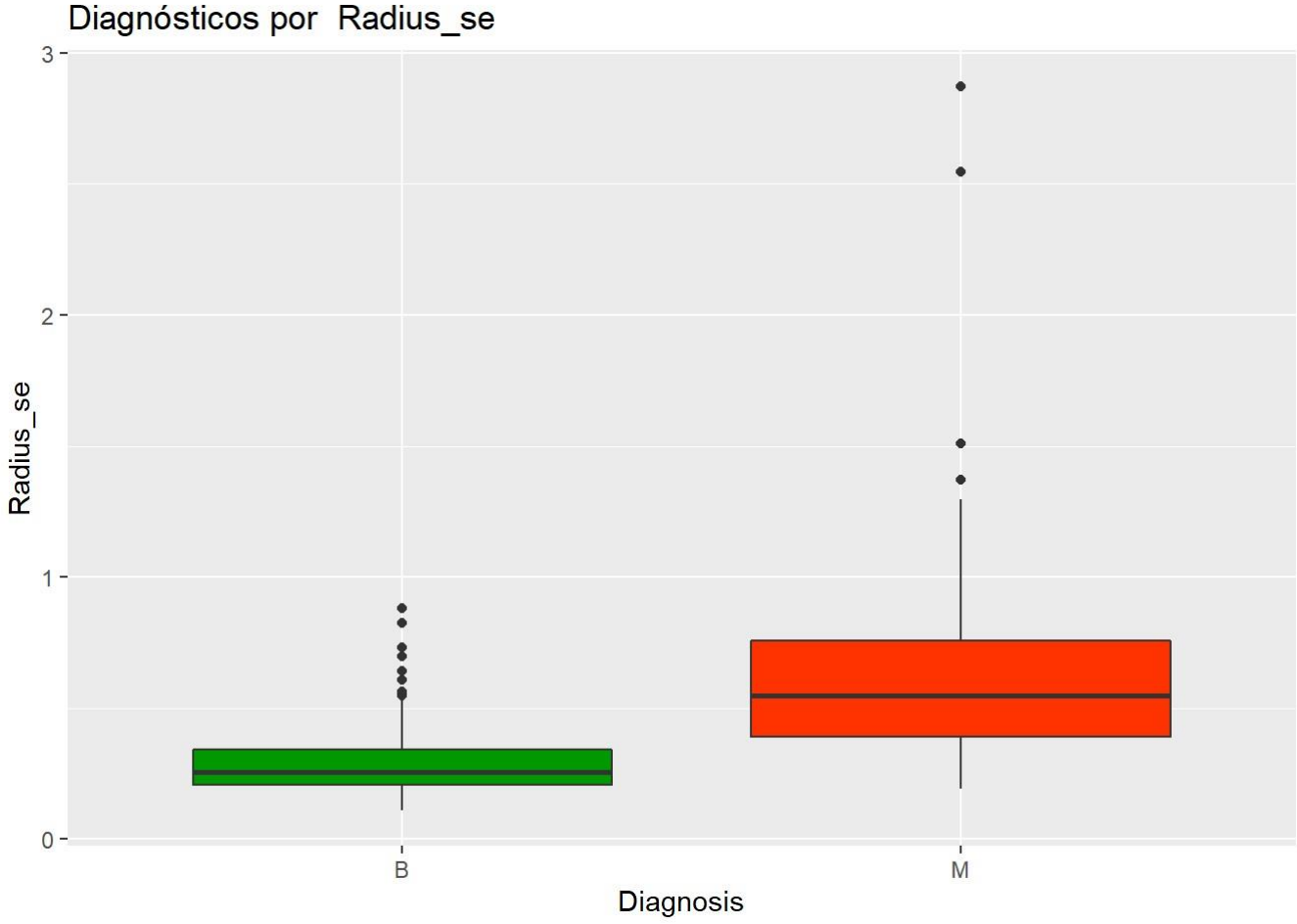

Diagnósticos por Perimeter_mean
Perimeter_mean
160
120
80
40
B
M
Diagnosis

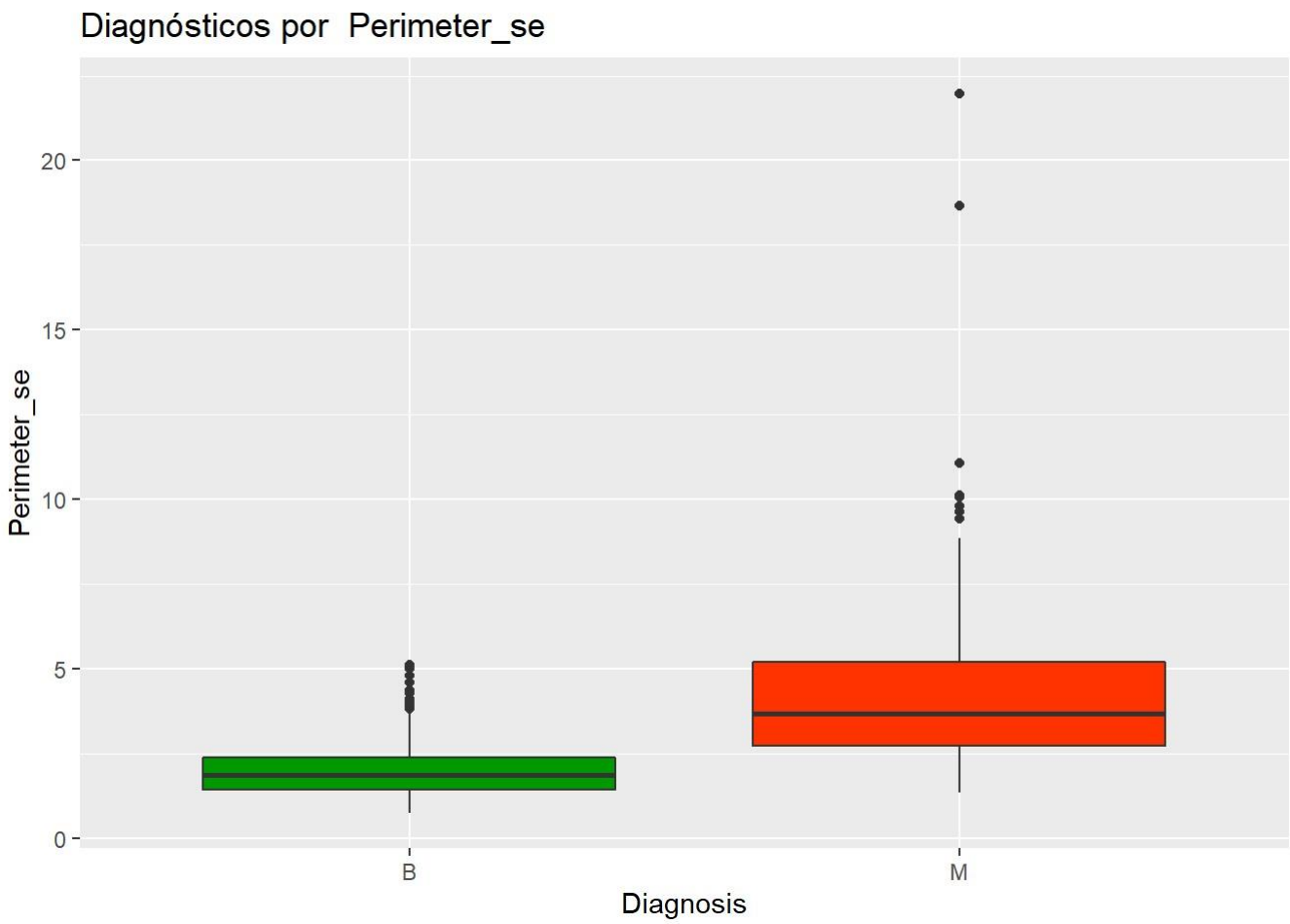

Diagnósticos por Texture_se
Texture_se
5
4
3
2
1
B
M
Diagnosis

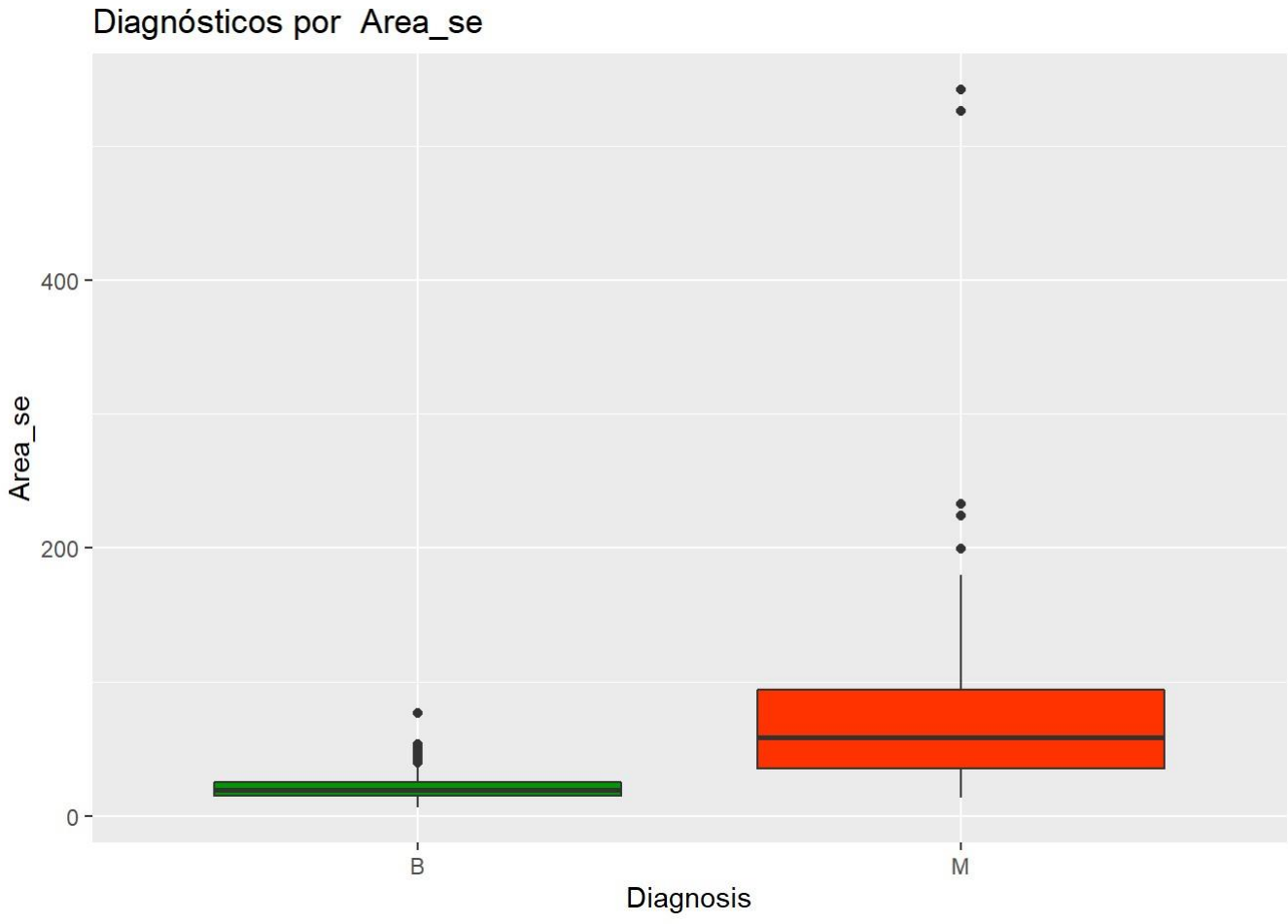

Diagnósticos por Concavity_mean
Concavity_mean
0.4
0.3
0.2
0.1
0.0
B
M
Diagnosis

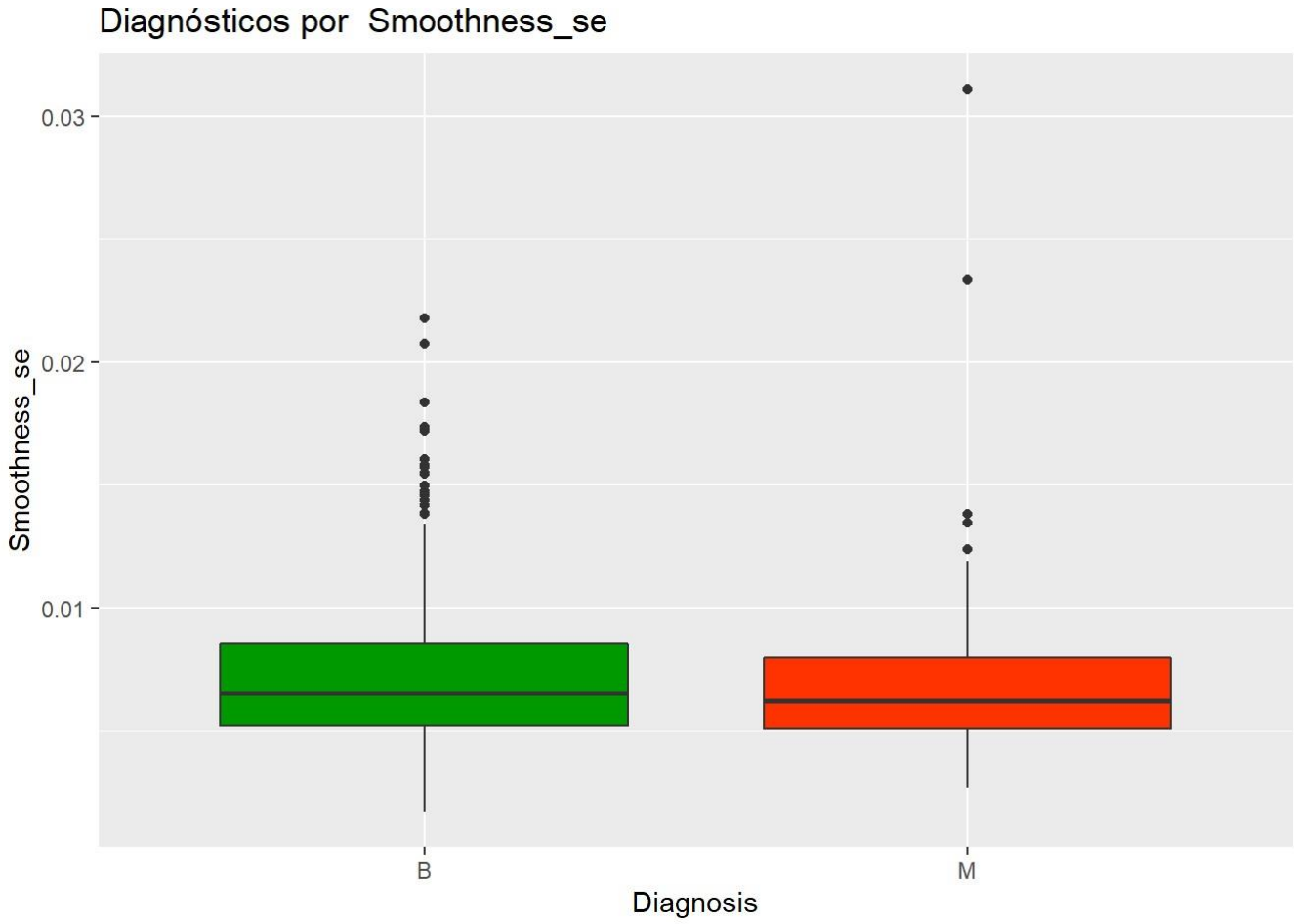

Diagnósticos por Nconcave_mean
Nconcave_mean
0.20
0.15
0.10
0.05
0.00
B
M
Diagnosis

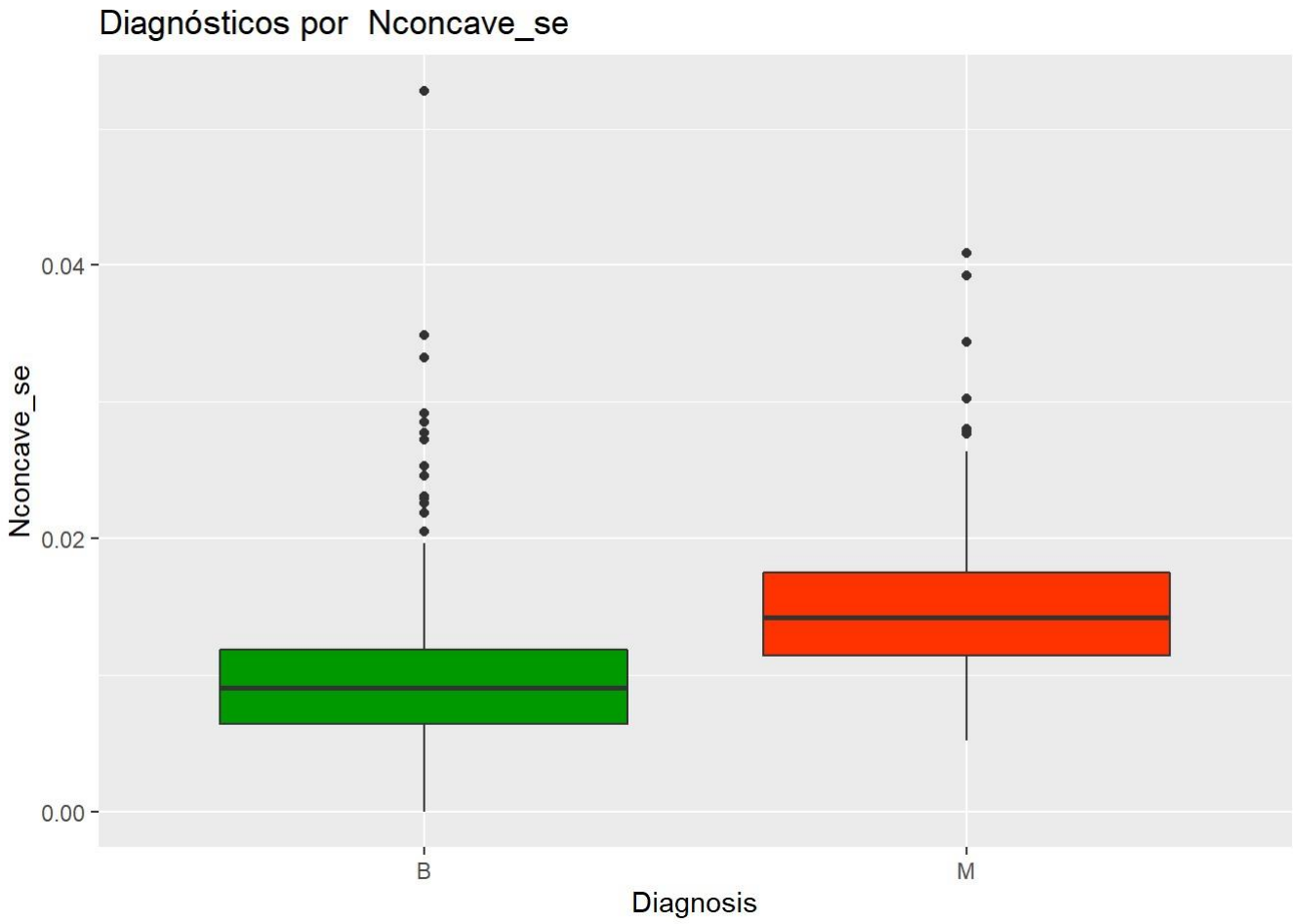

Diagnósticos por Fractaldim_mean
Fractaldim_mean
0.09
0.08
0.07
0.06
0.05
B
M
Diagnosis

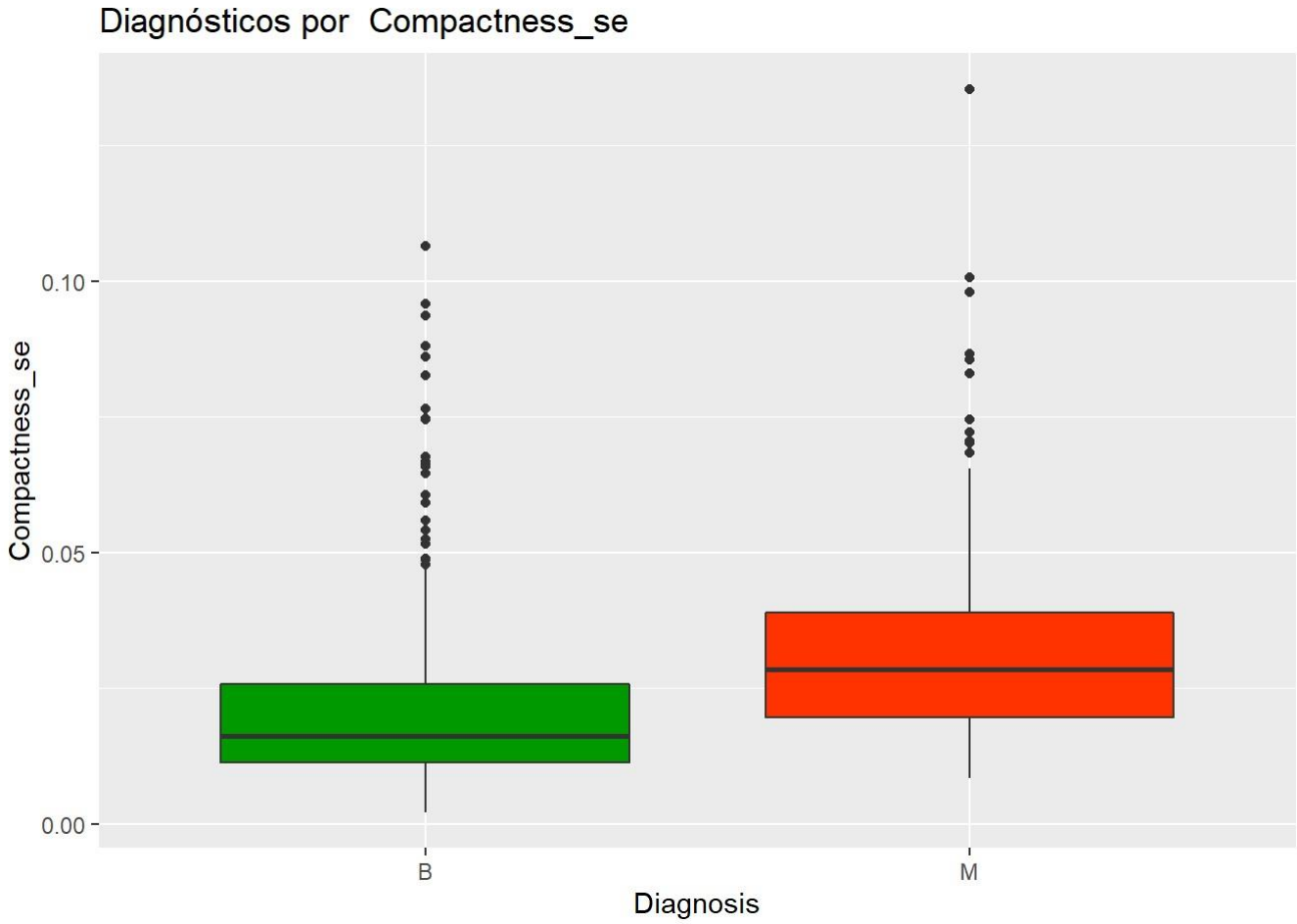

Diagnósticos por Radius_se
Radius_se
3
2
1
0
B
M
Diagnosis


Diagnósticos por Perimeter_se
Perimeter_se
20
15
10
5
0
B
M
Diagnosis

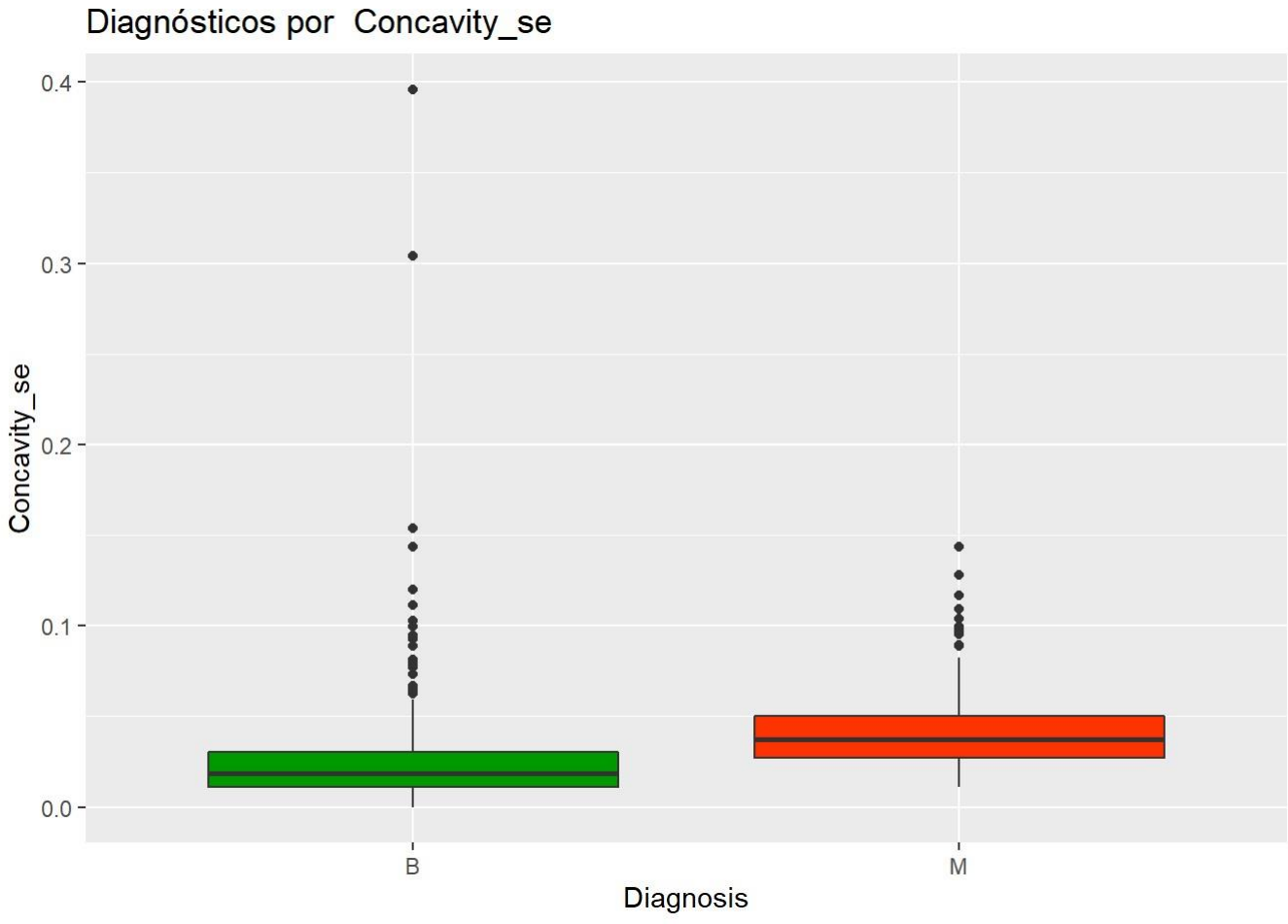

Diagnósticos por Area_se
Area_se
400
200
0
B
M
Diagnosis

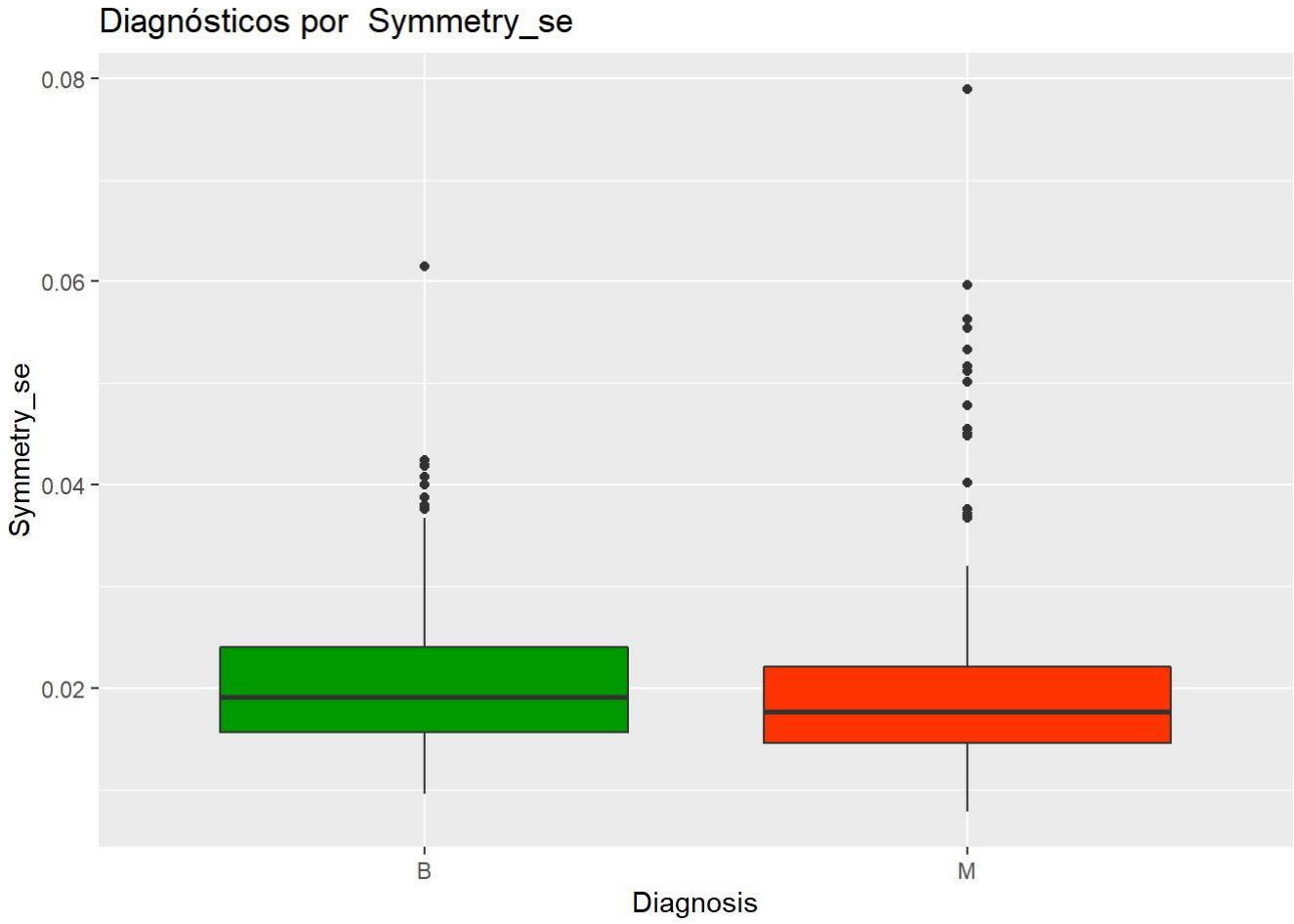

Diagnósticos por Smoothness_se
Smoothness_se
0.03
0.02
0.01
B
M
Diagnosis

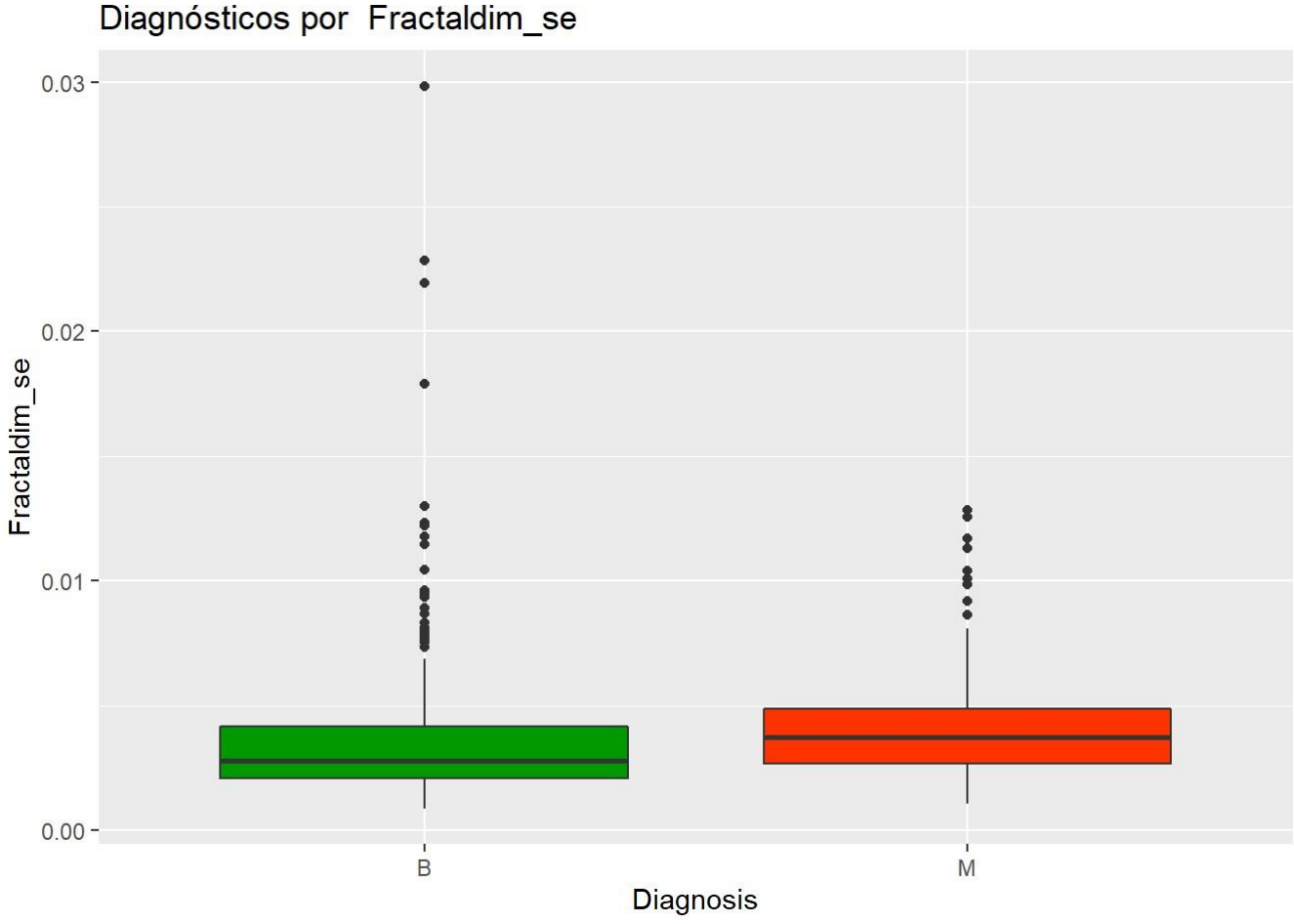

Diagnósticos por Nconcave_se
Nconcave_se
0.04
0.02
0.00
B
M
Diagnosis



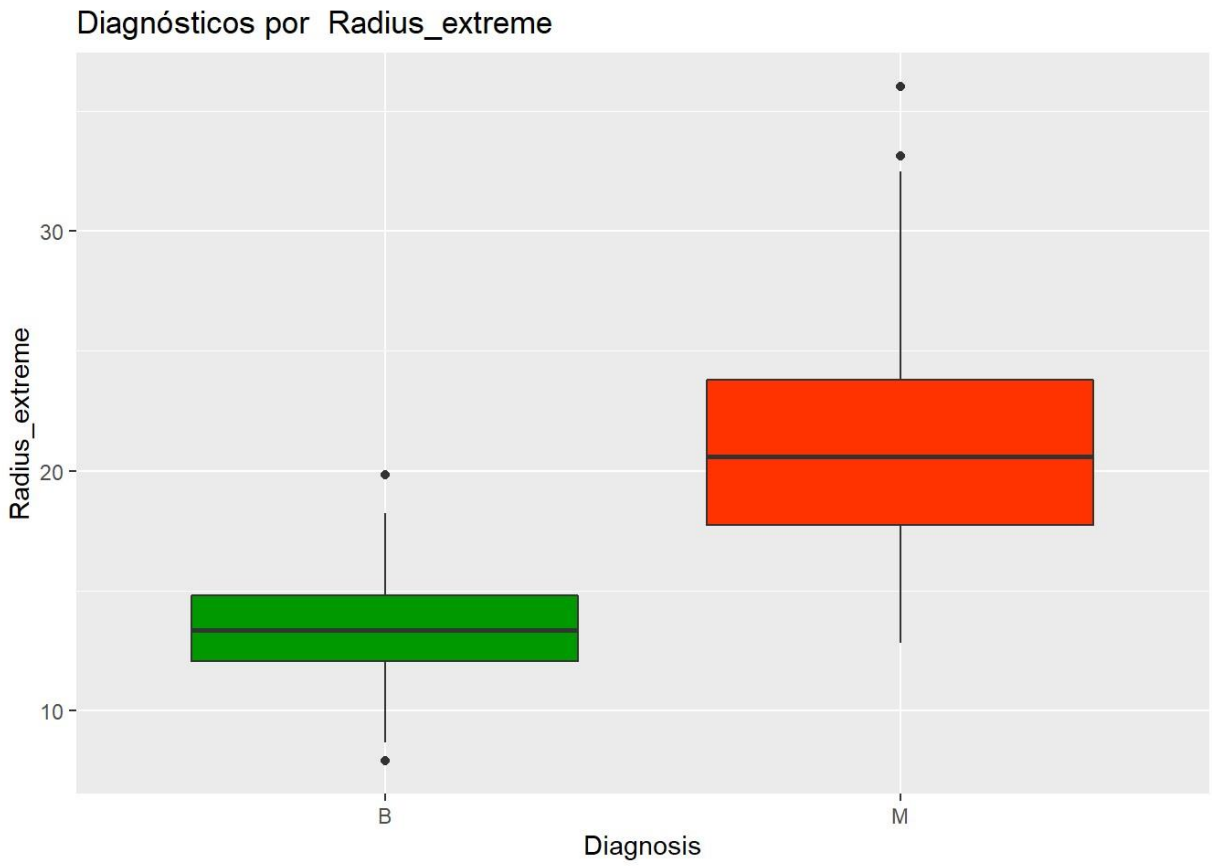

```
##
## $Concavity_se
```

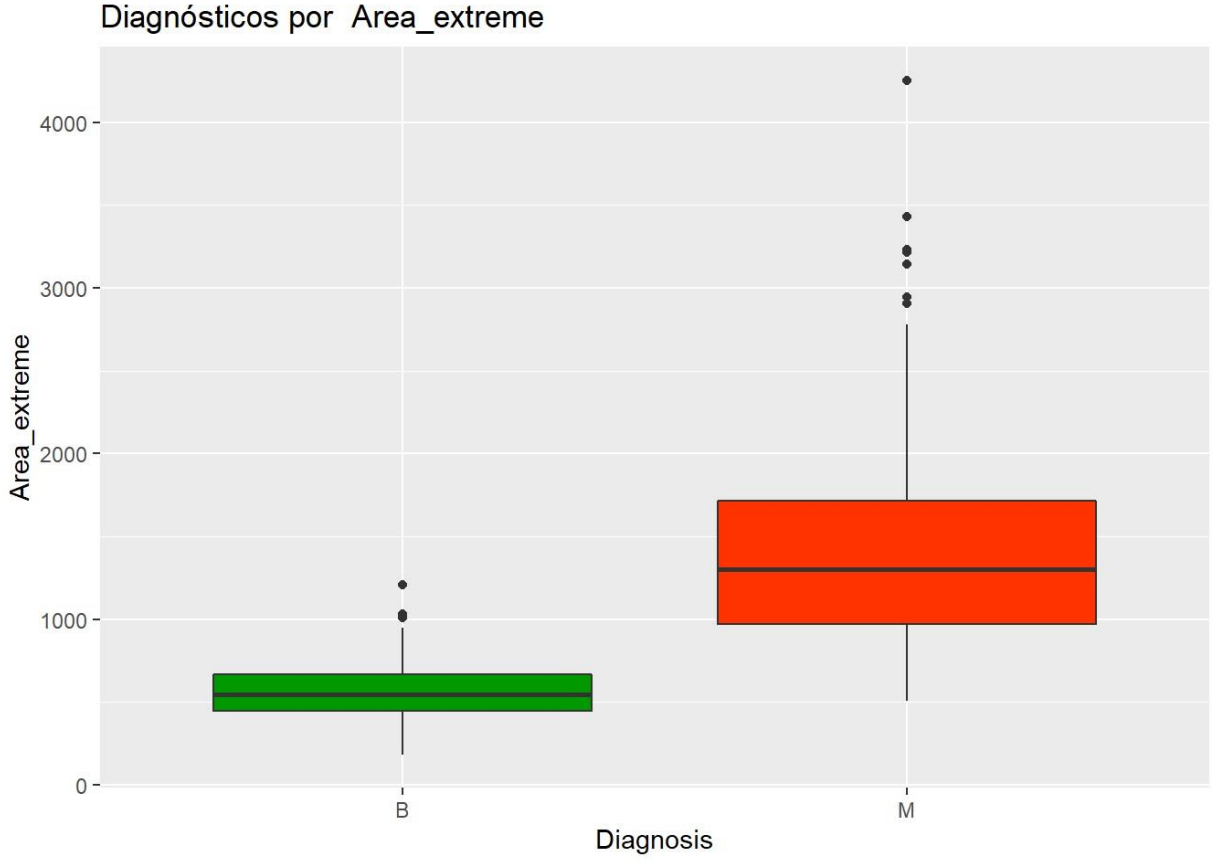


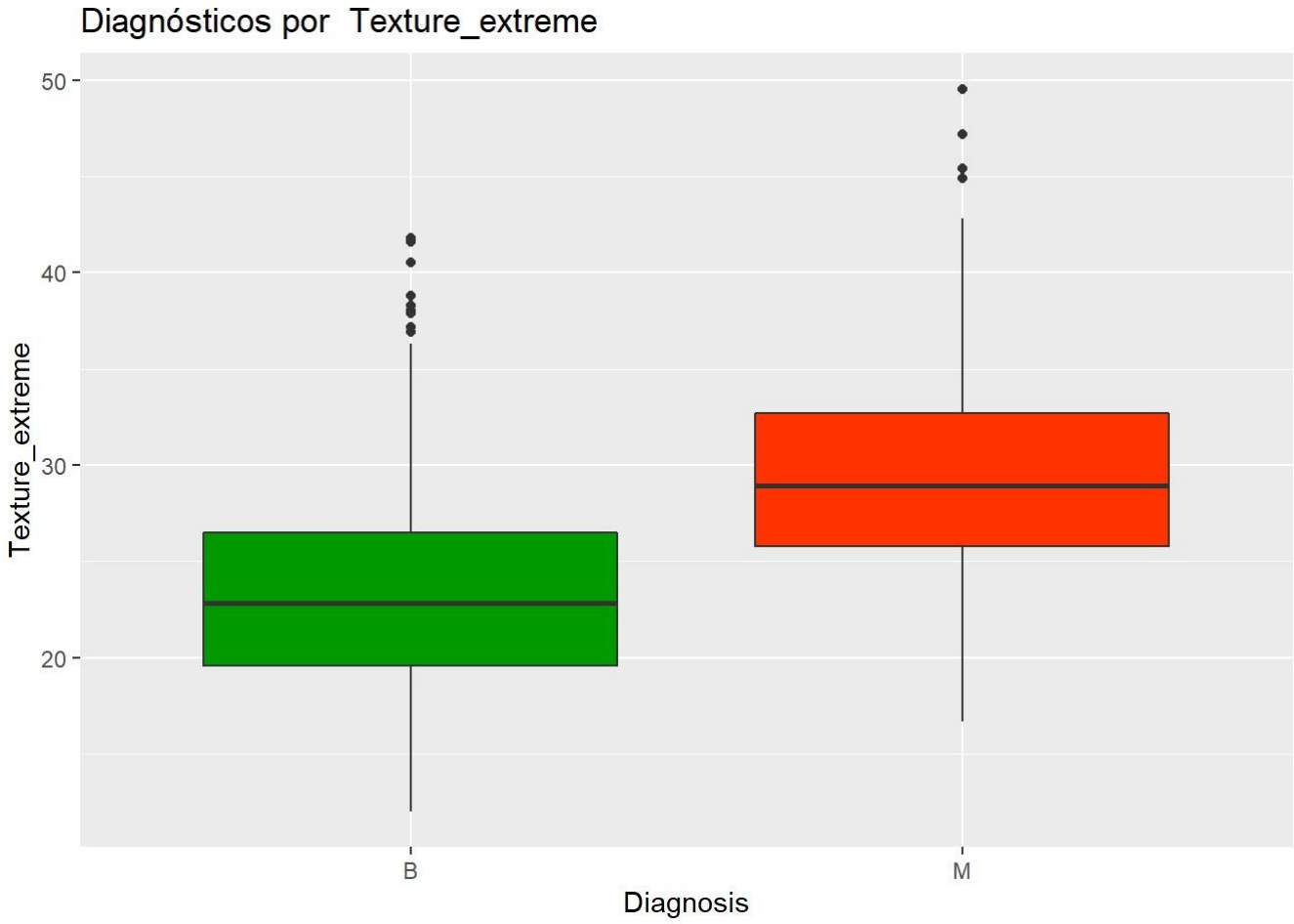

Diagnósticos por Symmetry_se
0.08
0.06
0.04
0.02
Symmetry_se
B
M
Diagnosis

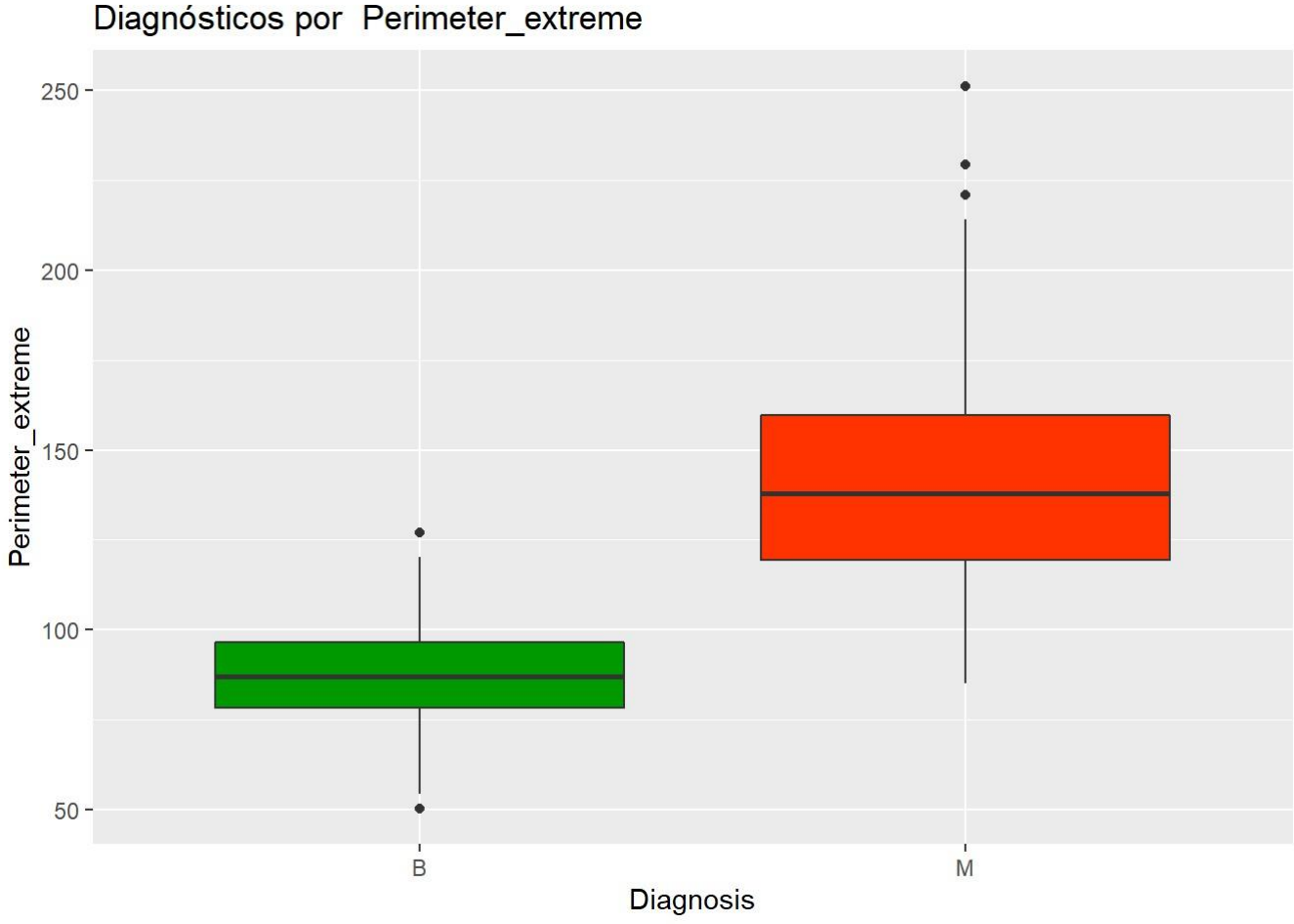

Diagnósticos por Fractaldim_se
0.03
0.02
0.01
0.00
Fractaldim_se
B
M
Diagnosis

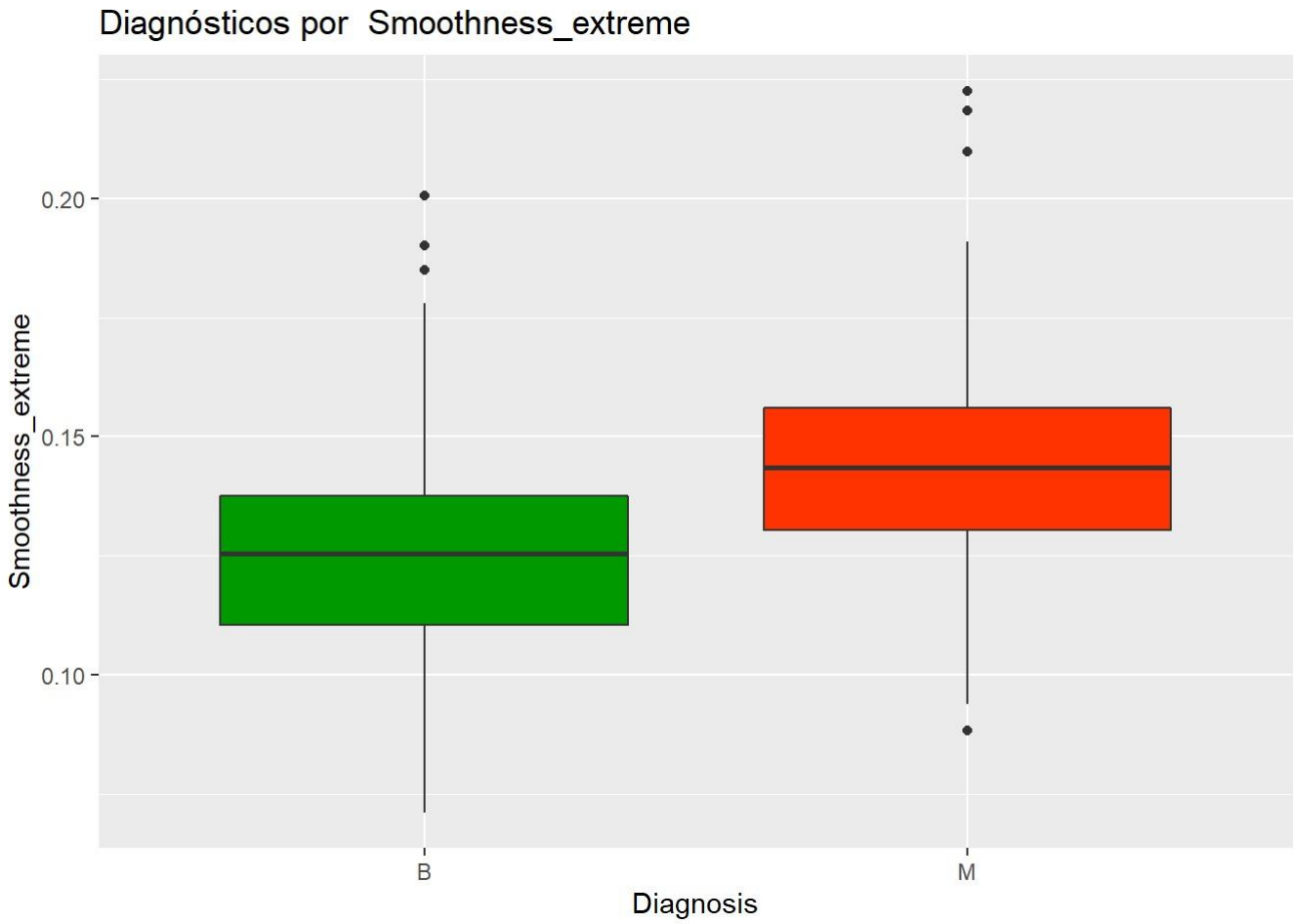
```
## $Radius_extreme
```

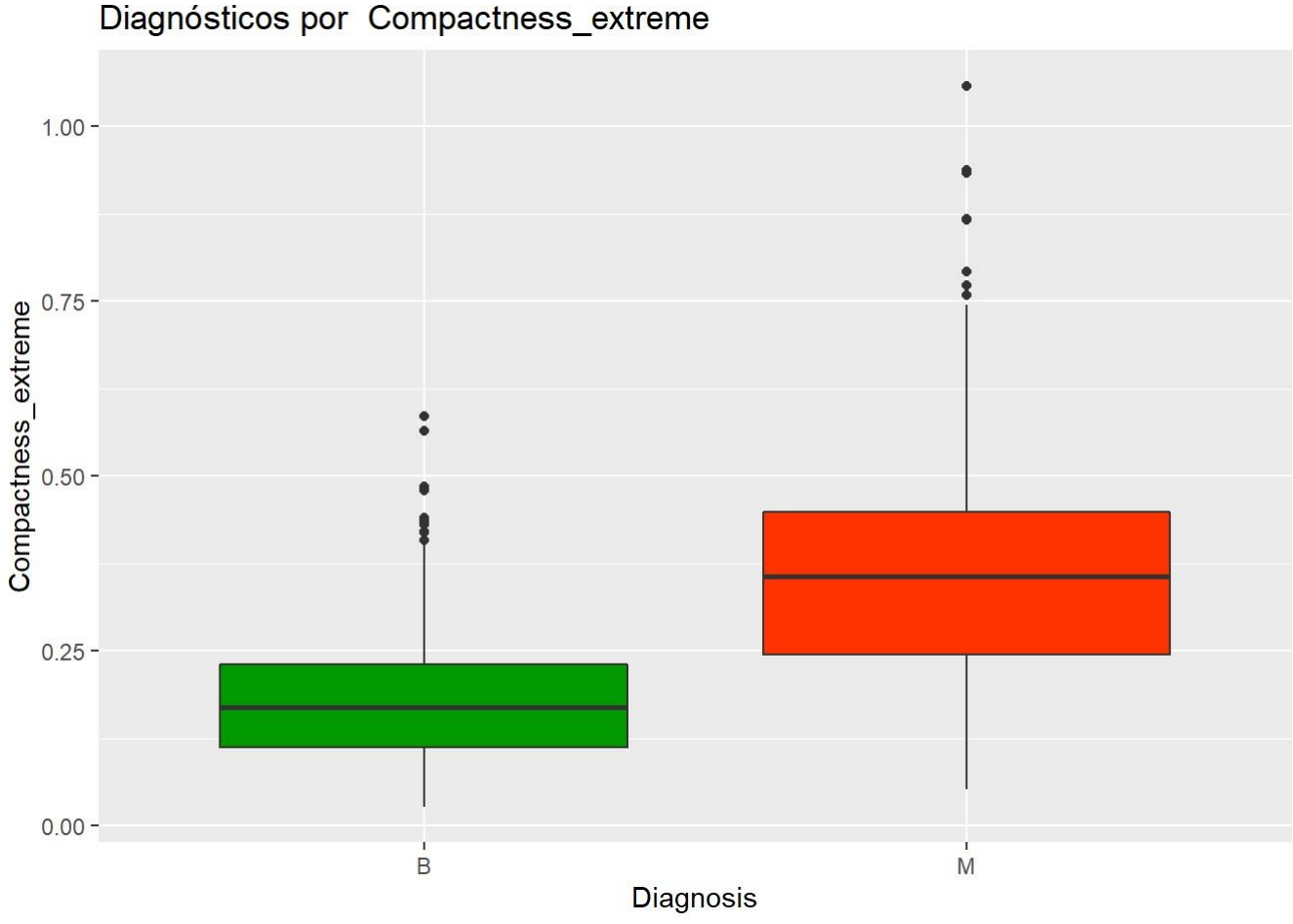


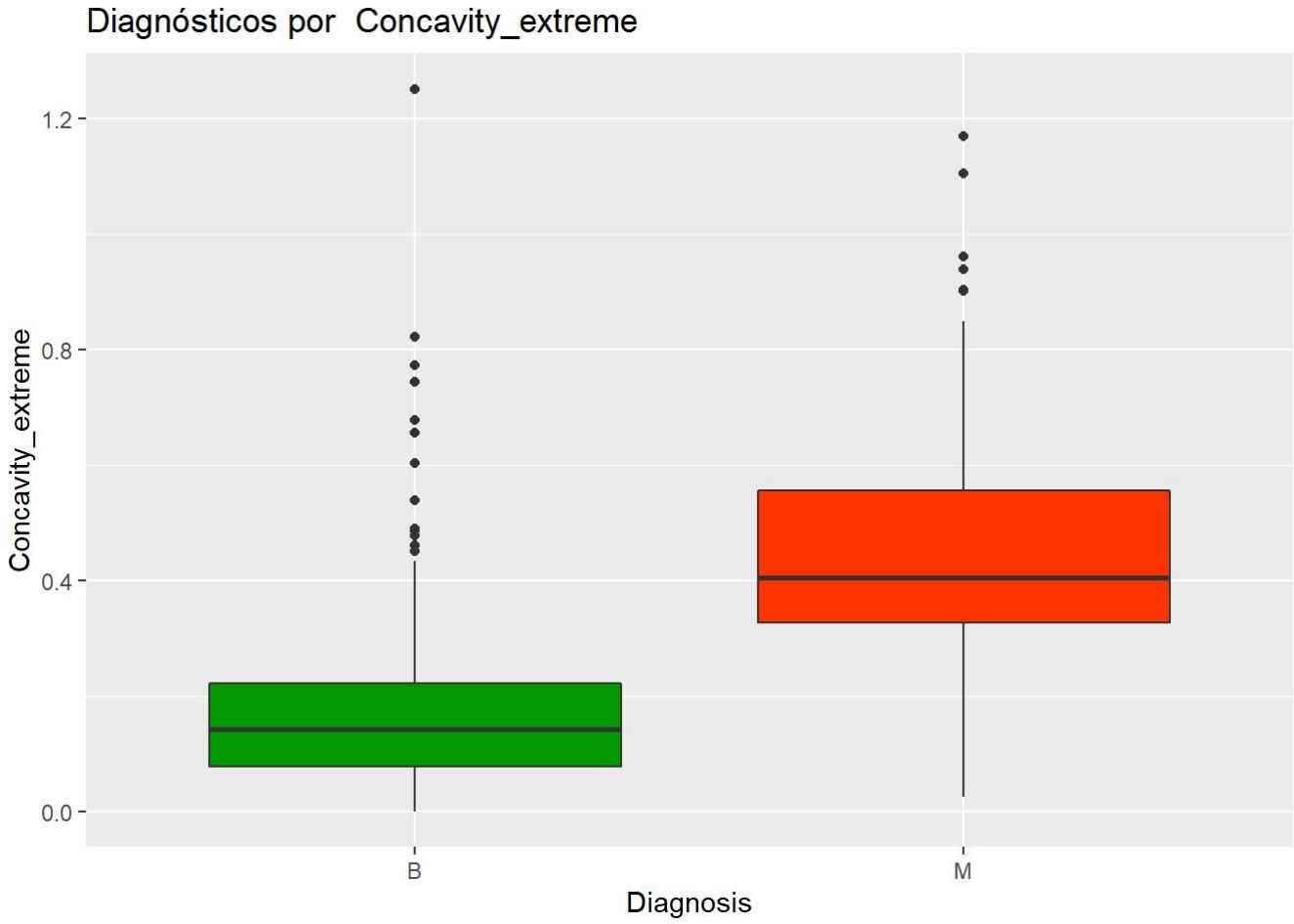


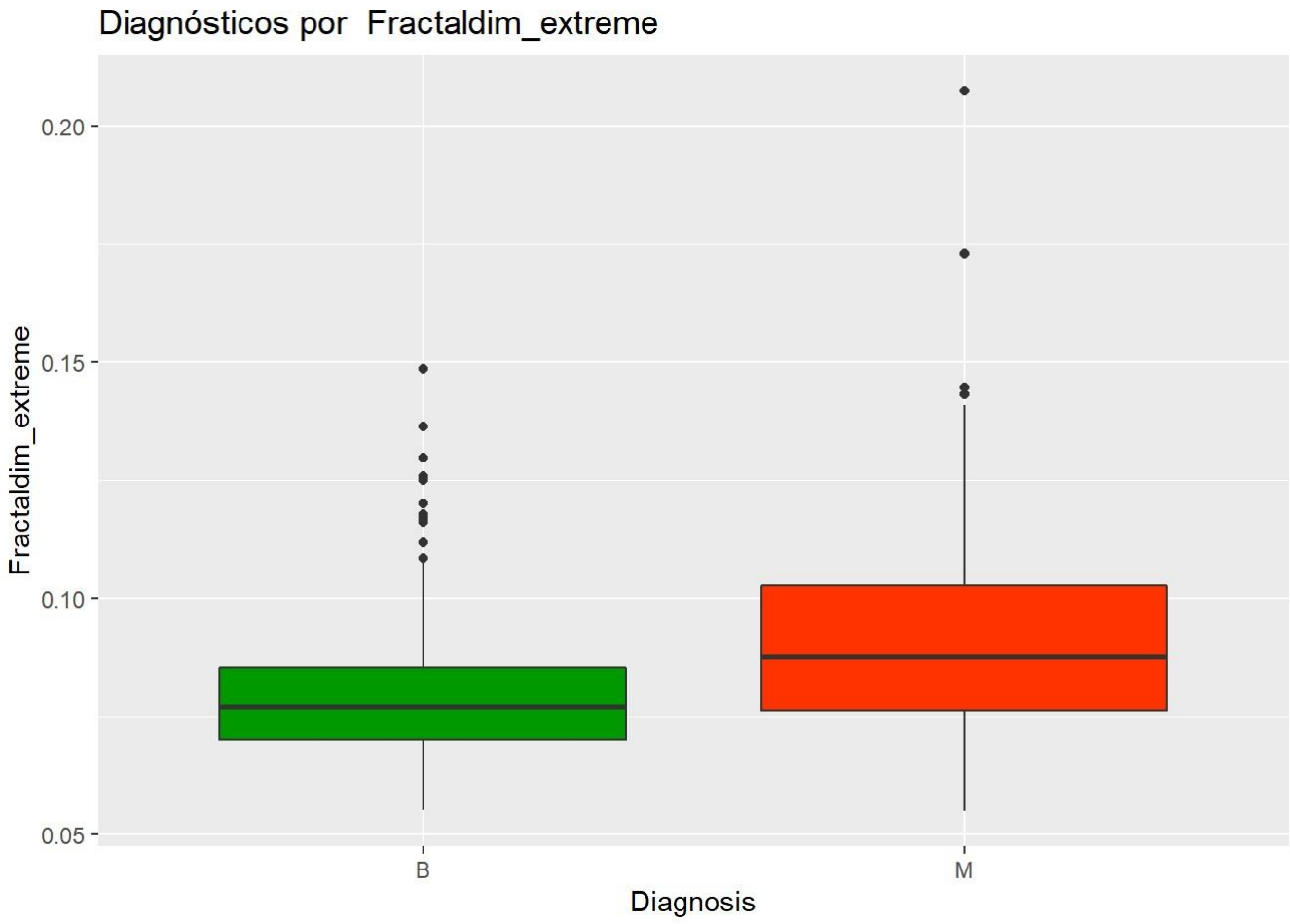

Diagnósticos por Texture_extreme
Texture_extreme
50
40
30
20
B
M
Diagnosis

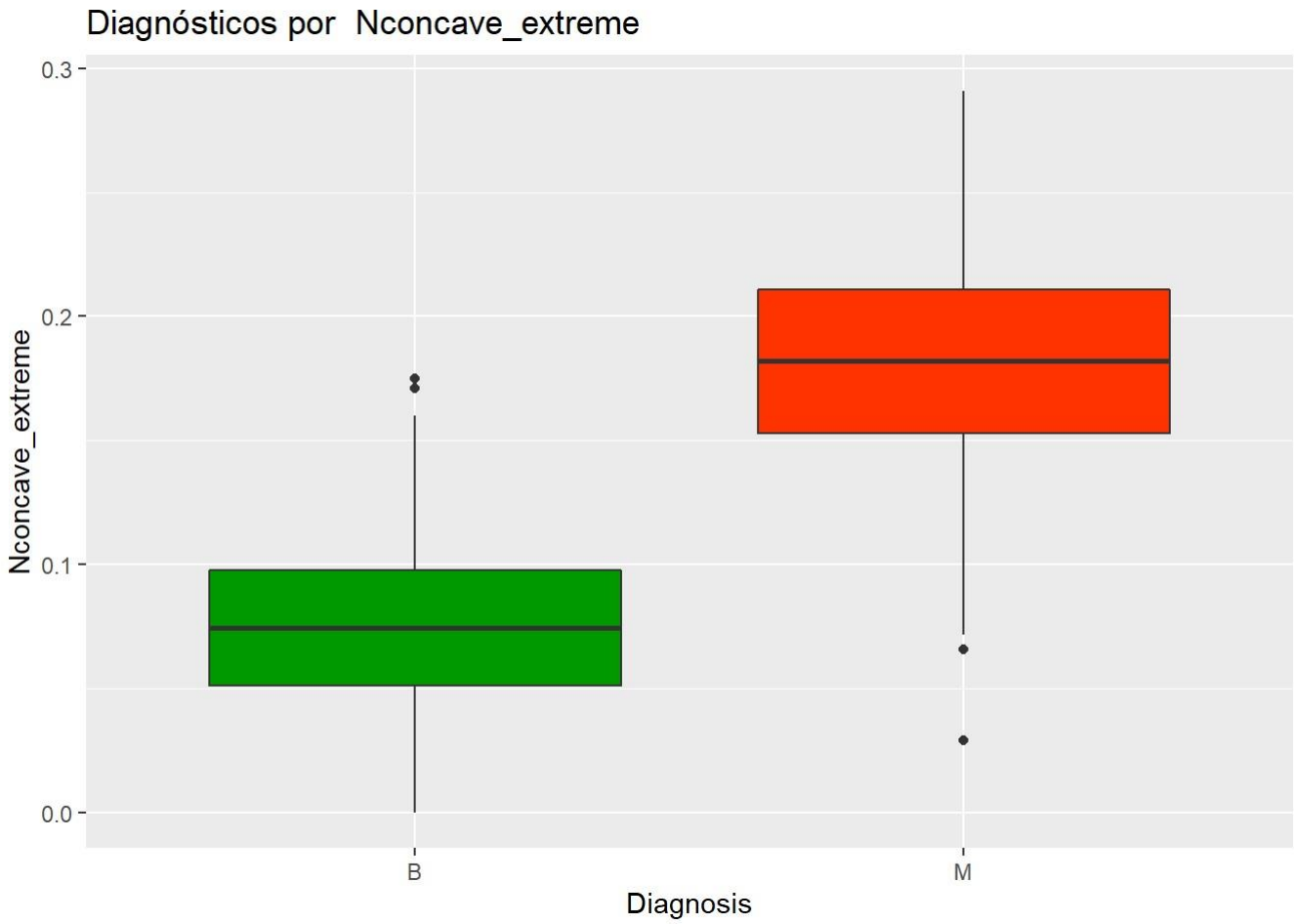

Diagnósticos por Perimeter_extreme
Perimeter_extreme
250
200
150
100
50
B
M
Diagnosis

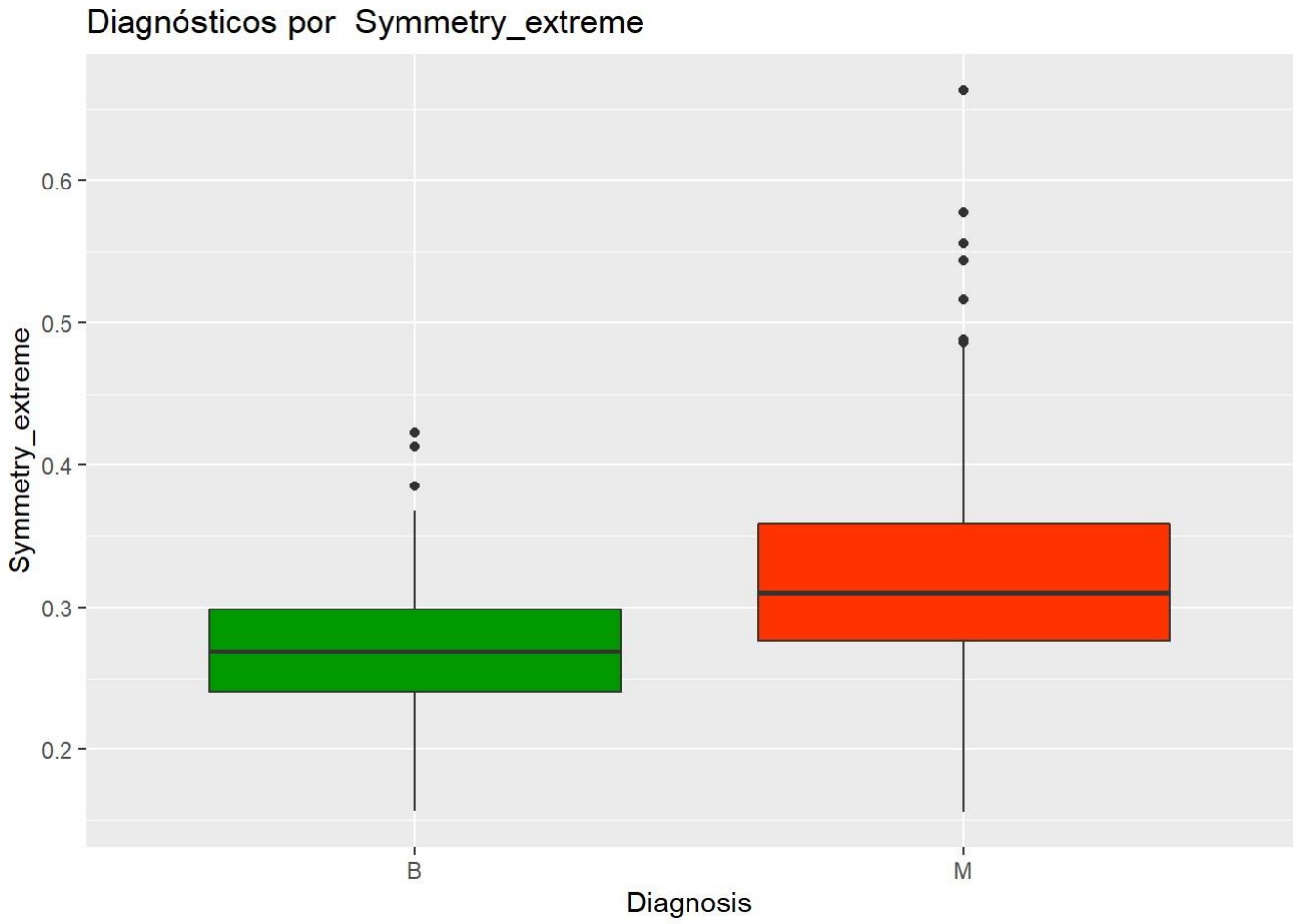

Diagnósticos por Smoothness_extreme
Smoothness_extreme
0.20
0.15
0.10
B
M
Diagnosis


Diagnósticos por Compactness_extreme
Compactness_extreme
1.00
0.75
0.50
0.25
0.00
B
M
Diagnosis


Diagnósticos por Concavity_extreme
Concavity_extreme
1.2
0.8
0.4
0.0
B
M
Diagnosis


Diagnósticos por Fractaldim_extreme
Fractaldim_extreme
0.20
0.15
0.10
0.05
B
M
Diagnosis


Diagnósticos por Nconcave_extreme
Nconcave_extreme
0.3
0.2
0.1
0.0
B
M
Diagnosis


Diagnósticos por Symmetry_extreme
Symmetry_extreme
0.6
0.5
0.4
0.3
0.2
B
M
Diagnosis

We separate our database into 70% for training and 30% for testing

```
library(caret)
```

```
## Loading required package: lattice
```

```
#Criando matriz com linhas dos dados de treino - 70%
set.seed(1)
filtro <- createDataPartition(y=BD$Diagnosis, p=0.7, list=FALSE)

#dados de treino e teste
treino <- BD[filtro,]
teste <- BD[-filtro,]
```

We created a logistic regression model to predict the diagnostic variable based on all the other variables, in this case to validate the model we will use the 5-level cross-validation, where 5 groups of data will be separated so that we can guarantee a minimum efficiency of the model.

```
#Criando o modelo
set.seed(1)
modelo <- train(Diagnosis ~ ., data=treino, method="glmnet", tuneLenght=4, trControl = trainC
ontrol(method="cv", number = 5))
```

We see below that after testing 5 different groups of training data, we have an accuracy of at least 97.46% and a maximum accuracy of 98.75%

```
#Acuracia de cada uma das 5 separacoes de dados
modelo$resample$Accuracy
```

```
## [1] 0.9750000 0.9875000 0.9750000 0.9746835 0.9875000
```

```
#Precisao no modelo de treino
mean(modelo$resample$Accuracy)
```

```
## [1] 0.9799367
```

Averaging the 5 groups, we have the average accuracy that the model delivers in the training data.

```
#Precisao no modelo de treino
mean(modelo$resample$Accuracy)
```

```
## [1] 0.9799367
```

So we can apply the model to the test data.

```
#Prevendo dados no modelo de teste
Prev <- predict(modelo, teste)
head(data.frame(teste$Diagnosis, Prev))
```

```
##   teste.Diagnosis Prev
## 1               M    M
## 2               M    M
## 3               M    M
## 4               M    M
## 5               M    M
## 6               M    M
```

Visualizing the first 6 lines of the test base with its predicted data.

```
head(data.frame(teste$Diagnosis, Prev))
```

```
##   teste.Diagnosis Prev
## 1               M    M
## 2               M    M
## 3               M    M
## 4               M    M
## 5               M    M
## 6               M    M
```

We can better visualize the result through a confusion matrix.

```
library(gmodels)

#Visualizando matriz de confusao
CrossTable(Prev, teste$Diagnosis, dnn = c("Previsto", "Real"), prop.chisq = FALSE, prop.t = F
ALSE, prop.r = FALSE, prop.c = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |-------------------------|
##
##
## Total Observations in Table:  170
##
##
##              | Real
##     Previsto |         B |         M | Row Total |
## -------------|-----------|-----------|-----------|
##            B |       106 |         4 |       110 |
## -------------|-----------|-----------|-----------|
##            M |         1 |        59 |        60 |
## -------------|-----------|-----------|-----------|
## Column Total |       107 |        63 |       170 |
## -------------|-----------|-----------|-----------|
##
##
```

We can visualize the other statistics such as sensitivity and specificity as well.

```
#Caret
#Verificando acuracia
confusionMatrix(Prev, teste$Diagnosis, dnn = c("Previsto", "Real"))
```

```
## Confusion Matrix and Statistics
##
##          Real
## Previsto   B   M
##        B 106   4
##        M   1  59
##
##                Accuracy : 0.9706
##                  95% CI : (0.9327, 0.9904)
##     No Information Rate : 0.6294
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9363
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9907
##             Specificity : 0.9365
##          Pos Pred Value : 0.9636
##          Neg Pred Value : 0.9833
##              Prevalence : 0.6294
##          Detection Rate : 0.6235
##    Detection Prevalence : 0.6471
##       Balanced Accuracy : 0.9636
##
##        'Positive' Class : B
##
```

In this case, we visualize a point of attention, our model classified malignant tumors as benign 4 times, which would be a big mistake for the health area, we do not want any patient to be informed that they have a benign cancer when in fact they have a malignant cancer , so we will work to improve this point later.

We will use the ROC curve to visualize how the model classifications behaved, as well as the best combinations of sensitivity and specificity.

To do this, we store the probabilities of each record having benign or malignant cancer and round it to 2 decimal places.

```
#Calculando Probabilidades de ser benigno ou maligno
PrevProb <- predict(modelo, teste, type="prob")
#Visualizando probabilidades
head(round(PrevProb, 2))
```

```
##      B    M
## 1 0.00 1.00
## 2 0.00 1.00
## 3 0.00 1.00
## 4 0.15 0.85
## 5 0.00 1.00
## 6 0.00 1.00
```

Note that the second column is just a complement of the probability of the previous column, so we need only one of these columns to make the substitutions, so we delete one of the columns.

```
#Precisa-se armazenar apenas uma das colunas de vetores, neste caso as probabilidades dos ben
ignos
PrevProb <- PrevProb$B
```

Next, we will use the ROC curve to visualize the best combinations of sensitivity and specificity that deliver the best model accuracy.

```
library(pROC)
```

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

```
##
## Attaching package: 'pROC'
```

```
## The following object is masked from 'package:gmodels':
##
##     ci
```

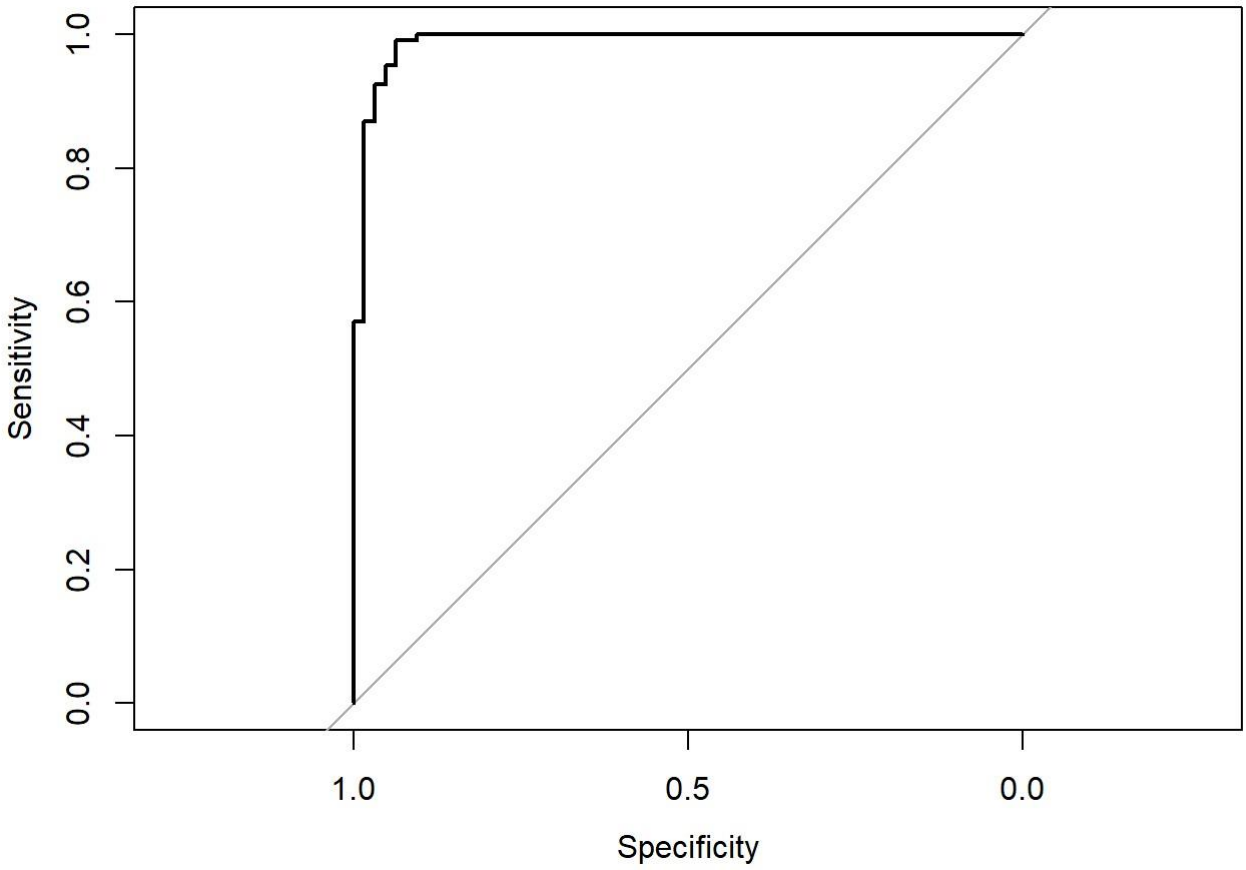
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
#Curva ROC
#Calculando os valores com base nos dados de teste em função da probabilidade
ROC <- roc(teste$Diagnosis ~ PrevProb, levels= c("M", "B"))
```

```
## Setting direction: controls < cases
```

Visualizing the curve.

```
plot(ROC)
```



Visualizing the area under the curve.

```
ROC$auc
```

```
## Area under the curve: 0.9889
```

In this case, the area under the curve is 0.9889, which is very close to 1, which makes the model ratings acceptable.

As we have noticed before, we need to correct or mitigate the number of malignant cases that were classified as benign in this model, for this we will explore the combinations of sensitivity and specificity, in order to find a solution that reduces the effect of a person with malignant cancer receiving a diagnosis. of benign cancer. For this we created a table with the combinations of sensitivity, specificity and thresholds.

```
head(round(data.frame(ROC$sensitivities, ROC$specificities, ROC$thresholds),2))
```

```
##   ROC.sensitivities ROC.specificities ROC.thresholds
## 1                 1              0.00          -Inf
## 2                 1              0.05             0
## 3                 1              0.06             0
## 4                 1              0.08             0
## 5                 1              0.10             0
## 6                 1              0.11             0
```

Sorting by specificity, we were able to visualize the combination that would eliminate the false positives, which would be present in line 111, but the model would lose a lot of sensitivity, remaining around only 56%.

```
#Ordenando por especificidade conseguimos visualizar a combinacao que zeraria os falsos posit
ivos,
#porem o modelo perderia muita sensibilidade ficando em torno de apenas 56%
ROC$thresholds[111]
```

```
## [1] 0.9998995
```

We store the probability vector in another variable, as it will be manipulated manually.

```
#Armazenando o vetor de probabilidades em outra variavel
PrevT <- PrevProb
```

Thus, we classify as benign only those records that have more than a 99.98% chance of being benign cancer, that is, we will classify as benign only when we are practically sure that this record will be.

```
#Substituindo as probabilidades acima de 99,45% por "B" (tumores benignos)
#Dessa forma, um tumor sera considerado benigno apenas se apresentar probabilidade
superior a 99,45%
PrevT[PrevT>ROC$thresholds[111]] <- "B"
#Todas as probabilidades que nao foram identificadas como "B" (benigno) serao "M"
(maligno)
PrevT[PrevT != "B"] <- "M"
#Convertendo para fator
PrevT <- as.factor(PrevT)
```

We can visualize the confusion matrix after these manipulations.

```
#Resultado do modelo com thereshold ajustado
confusionMatrix(PrevT, teste$Diagnosis, dnn = c("Previsto", "Real"))
```

```
## Confusion Matrix and Statistics
##
##          Real
## Previsto  B  M
##        B 58  0
##        M 49 63
##
##                Accuracy : 0.7118
##                  95% CI : (0.6374, 0.7785)
##     No Information Rate : 0.6294
##     P-Value [Acc > NIR] : 0.01486
##
##                   Kappa : 0.4673
##
##  Mcnemar's Test P-Value : 7.025e-12
##
##             Sensitivity : 0.5421
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.5625
##              Prevalence : 0.6294
##          Detection Rate : 0.3412
##    Detection Prevalence : 0.3412
##       Balanced Accuracy : 0.7710
##
##        'Positive' Class : B
##
```

Note that the accuracy of the model dropped by 24%, but we reduced the number of malignant cancer accusations that could be classified as benign as much as possible. In this case, we will choose to make 1 false accusation of benign cancer, in exchange for a more accurate model.

We look again at the table with the specificity and sensitivity combinations, this time we opted for a sensitivity of at least 89% and selected the best combination specificity

```
#Filtrando dados com sensibilidade acima de 0.89 e a maior combinacao possivel de
especificid ade head(round(data.frame(ROC$sensitivities, ROC$specificities,
ROC$thresholds),2))
```

```
##   ROC.sensitivities ROC.specificities ROC.thresholds
## 1                 1              0.00           -Inf
## 2                 1              0.05              0
## 3                 1              0.06              0
## 4                 1              0.08              0
## 5                 1              0.10              0
## 6                 1              0.11              0
```

This was on line 75, so we substituted it again in the odds table.

```
#Encontramos o vetor 75, com sensibilidade de 0.89 e especificidade de 0.98
#Substituindo no modelo temos
ROC$thresholds[75]
```

```
## [1] 0.9869348
```

```
PrevT <- PrevProb
PrevT[PrevT>ROC$thresholds[75]] <- "B"
PrevT[PrevT != "B"] <- "M"
PrevT <- as.factor(PrevT)
```

And we can see the result of this new model.

```
#Resultado do modelo com thereshold ajustado
confusionMatrix(PrevT, teste$Diagnosis, dnn = c("Previsto", "Real"))
```

```
## Confusion Matrix and Statistics
##
##          Real
## Previsto  B  M
##        B 93  1
##        M 14 62
##
##                Accuracy : 0.9118
##                  95% CI : (0.8586, 0.9498)
##     No Information Rate : 0.6294
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8186
##
##  Mcnemar's Test P-Value : 0.001946
##
##             Sensitivity : 0.8692
##             Specificity : 0.9841
##          Pos Pred Value : 0.9894
##          Neg Pred Value : 0.8158
##              Prevalence : 0.6294
##          Detection Rate : 0.5471
##    Detection Prevalence : 0.5529
##       Balanced Accuracy : 0.9266
##
##        'Positive' Class : B
##
```

We managed to improve the accuracy of the model by 20%, accepting 1 case of false positive for benign cancer, we will accept this model with an accuracy of 91%.