PCA_NNET_breast_cancer_classification.R

vaishnavimyadam

2020-06-26

```
## Vaishu Myadam (vmyadam1208@gmail.com)
## June, 2020

## Using the Wisconsin Breast Cancer dataset

# Necessary libraries

library("e1071")
library(caret)
```

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

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

```
library(pROC)
```

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

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

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

```
library(gridExtra)
library(grid)
library(ggfortify)
library(purrr)
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:caret':
##
##     lift
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

```
## The following objects are masked from 'package:stats':
##
```

```
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(reshape2)
library(readr)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
require(foreach)
```

```
## Loading required package: foreach
##
## Attaching package: 'foreach'
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
require(iterators)
```

```
## Loading required package: iterators
```

```
require(parallel)
```

```
## Loading required package: parallel
```

```
library(nnet)
library(doParallel)
registerDoParallel()

# Reading in data

wisconsindata = read.csv("wisconsindata.csv", sep = ",")

# Data cleaning

str(wisconsindata) # Seeing a summary to decide what features to remove
```

```
## 'data.frame':    569 obs. of  33 variables:
##  $ id                      : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844
##  $ diagnosis               : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ 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 ...
```

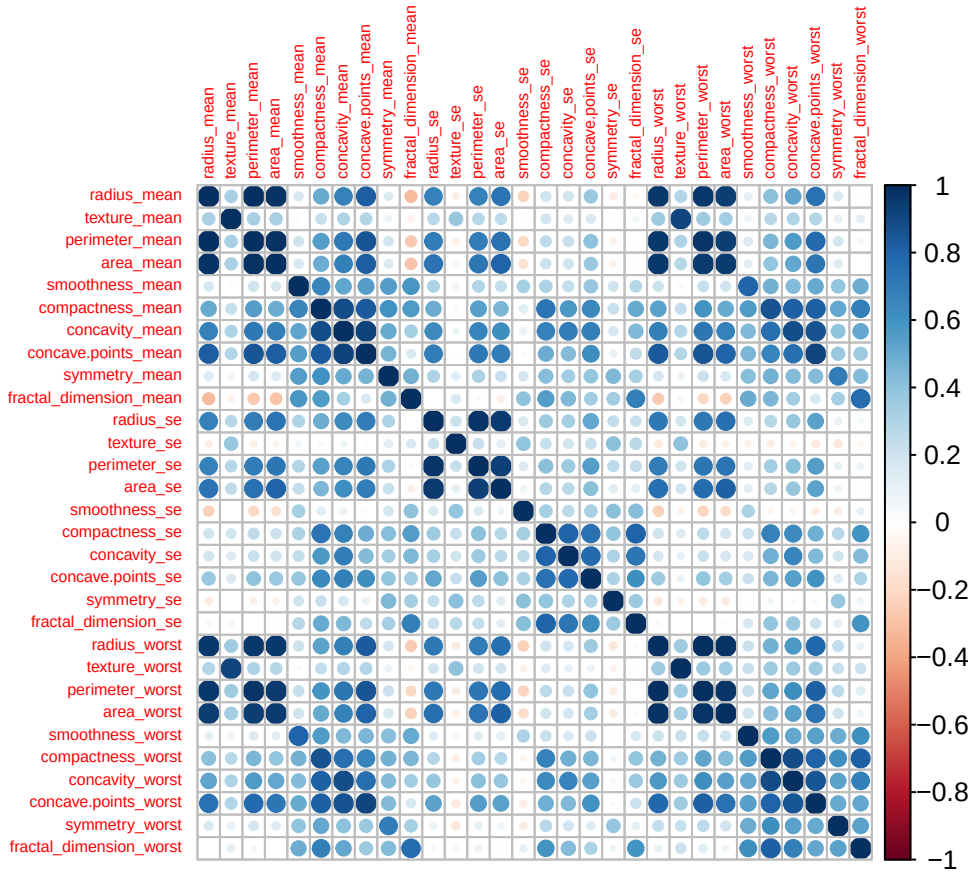
```
##  $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ 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 ...
```

```
cleaned_data = wisconsindata[,-c(0:1)] # Removing unnecessary ID column
cleaned_data = cleaned_data[, -32] # Removing useless last column
cleaned_data$diagnosis = as.factor(cleaned_data$diagnosis) # Tidying the dataset

summary(cleaned_data)
```

```
##  diagnosis  radius_mean      texture_mean   perimeter_mean     area_mean
##  B:357     Min.   : 6.981   Min.   : 9.71   Min.   : 43.79   Min.   : 143.5
##  M:212     1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17   1st Qu.: 420.3
##            Median :13.370   Median :18.84   Median : 86.24   Median : 551.1
##            Mean   :14.127   Mean   :19.29   Mean   : 91.97   Mean   : 654.9
##            3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10   3rd Qu.: 782.7
##            Max.   :28.110   Max.   :39.28   Max.   :188.50   Max.   :2501.0
##  smoothness_mean   compactness_mean  concavity_mean    concave.points_mean
##  Min.   :0.05263   Min.   :0.01938   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956   1st Qu.:0.02031
##  Median :0.09587   Median :0.09263   Median :0.06154   Median :0.03350
##  Mean   :0.09636   Mean   :0.10434   Mean   :0.08880   Mean   :0.04892
##  3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070   3rd Qu.:0.07400
##  Max.   :0.16340   Max.   :0.34540   Max.   :0.42680   Max.   :0.20120
##  symmetry_mean    fractal_dimension_mean   radius_se        texture_se
##  Min.   :0.1060   Min.   :0.04996        Min.   :0.1115   Min.   :0.3602
##  1st Qu.:0.1619   1st Qu.:0.05770        1st Qu.:0.2324   1st Qu.:0.8339
##  Median :0.1792   Median :0.06154        Median :0.3242   Median :1.1080
##  Mean   :0.1812   Mean   :0.06280        Mean   :0.4052   Mean   :1.2169
##  3rd Qu.:0.1957   3rd Qu.:0.06612        3rd Qu.:0.4789   3rd Qu.:1.4740
##  Max.   :0.3040   Max.   :0.09744        Max.   :2.8730   Max.   :4.8850
##   perimeter_se       area_se        smoothness_se      compactness_se
##  Min.   : 0.757   Min.   :  6.802   Min.   :0.001713   Min.   :0.002252
##  1st Qu.: 1.606   1st Qu.: 17.850   1st Qu.:0.005169   1st Qu.:0.013080
##  Median : 2.287   Median : 24.530   Median :0.006380   Median :0.020450
##  Mean   : 2.866   Mean   : 40.337   Mean   :0.007041   Mean   :0.025478
##  3rd Qu.: 3.357   3rd Qu.: 45.190   3rd Qu.:0.008146   3rd Qu.:0.032450
##  Max.   :21.980   Max.   :542.200   Max.   :0.031130   Max.   :0.135400
##   concavity_se     concave.points_se   symmetry_se       fractal_dimension_se
##  Min.   :0.00000   Min.   :0.000000   Min.   :0.007882   Min.   :0.0008948
```

```
##  1st Qu.:0.01509   1st Qu.:0.007638   1st Qu.:0.015160   1st Qu.:0.0022480
##  Median :0.02589   Median :0.010930   Median :0.018730   Median :0.0031870
##  Mean   :0.03189   Mean   :0.011796   Mean   :0.020542   Mean   :0.0037949
##  3rd Qu.:0.04205   3rd Qu.:0.014710   3rd Qu.:0.023480   3rd Qu.:0.0045580
##  Max.   :0.39600   Max.   :0.052790   Max.   :0.078950   Max.   :0.0298400
##   radius_worst   texture_worst   perimeter_worst    area_worst
##  Min.   : 7.93   Min.   :12.02   Min.   : 50.41   Min.   : 185.2
##  1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11   1st Qu.: 515.3
##  Median :14.97   Median :25.41   Median : 97.66   Median : 686.5
##  Mean   :16.27   Mean   :25.68   Mean   :107.26   Mean   : 880.6
##  3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40   3rd Qu.:1084.0
##  Max.   :36.04   Max.   :49.54   Max.   :251.20   Max.   :4254.0
##  smoothness_worst  compactness_worst concavity_worst  concave.points_worst
##  Min.   :0.07117   Min.   :0.02729   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.11660   1st Qu.:0.14720   1st Qu.:0.1145   1st Qu.:0.06493
##  Median :0.13130   Median :0.21190   Median :0.2267   Median :0.09993
##  Mean   :0.13237   Mean   :0.25427   Mean   :0.2722   Mean   :0.11461
##  3rd Qu.:0.14600   3rd Qu.:0.33910   3rd Qu.:0.3829   3rd Qu.:0.16140
##  Max.   :0.22260   Max.   :1.05800   Max.   :1.2520   Max.   :0.29100
##  symmetry_worst   fractal_dimension_worst
##  Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2822   Median :0.08004
##  Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :0.6638   Max.   :0.20750
```

```r
# Removing unnecessary predictors (bivariate multivariate analysis)

correlations = cor(cleaned_data[,2:31])
corrplot(correlations, order = "original", tl.cex = 0.5)
```

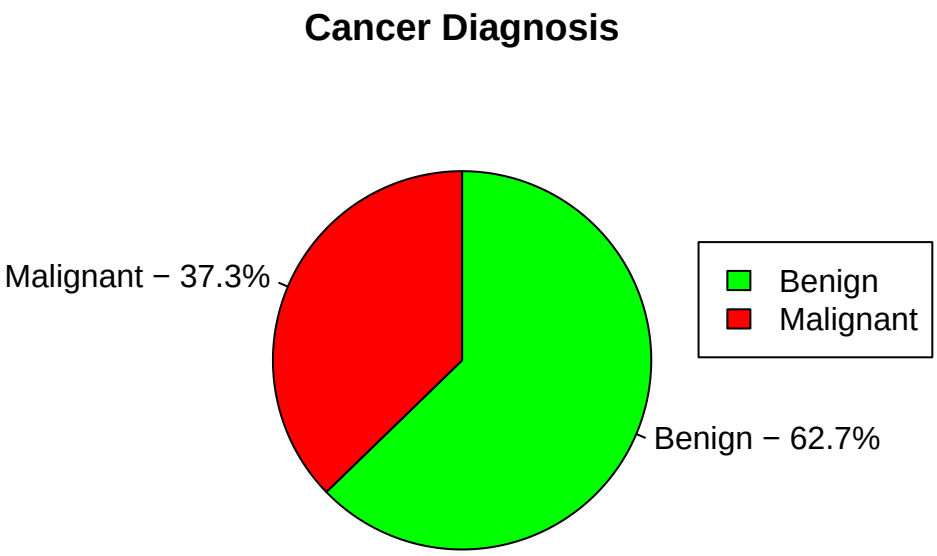



```
highly_correlated_features = colnames(cleaned_data)[findCorrelation(correlations, cutoff = 0.9, verbose
```

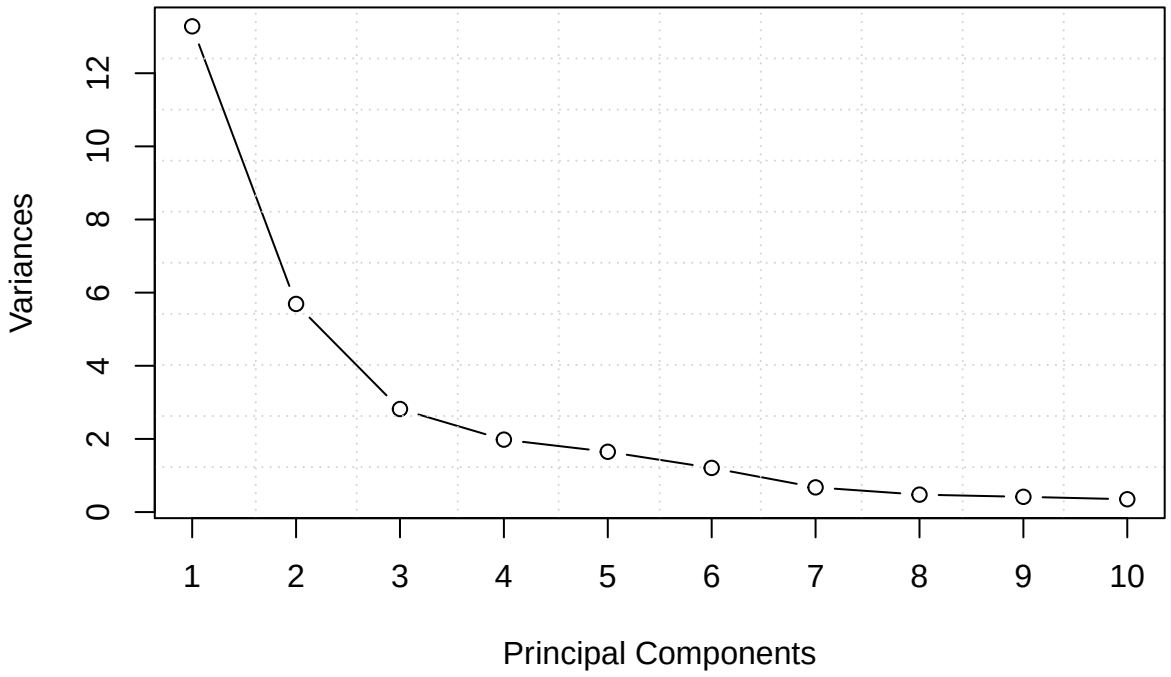
```
## Compare row 7  and column  8 with corr  0.921
##   Means:  0.571 vs 0.389 so flagging column 7
## Compare row 8  and column  28 with corr  0.91
##   Means:  0.542 vs 0.377 so flagging column 8
## Compare row 23  and column  21 with corr  0.994
##   Means:  0.48 vs 0.367 so flagging column 23
## Compare row 21  and column  3 with corr  0.969
##   Means:  0.446 vs 0.359 so flagging column 21
## Compare row 3  and column  24 with corr  0.942
##   Means:  0.414 vs 0.353 so flagging column 3
## Compare row 24  and column  1 with corr  0.941
##   Means:  0.39 vs 0.349 so flagging column 24
## Compare row 1  and column  4 with corr  0.987
##   Means:  0.35 vs 0.347 so flagging column 1
## Compare row 13  and column  11 with corr  0.973
##   Means:  0.372 vs 0.346 so flagging column 13
## Compare row 11  and column  14 with corr  0.952
##   Means:  0.323 vs 0.347 so flagging column 14
## Compare row 22  and column  2 with corr  0.912
##   Means:  0.224 vs 0.357 so flagging column 2
## All correlations <= 0.9
```

```
cleaned_data_cor = cleaned_data[, which(!colnames(cleaned_data) %in% highly_correlated_features)]

# Visualization
```

```
diagnosis_list = table(cleaned_data$diagnosis)
diagnosis_proportions = prop.table(diagnosis_list) * 100
pielabels = sprintf("%s - %3.1f%s", c("Benign", "Malignant"), diagnosis_proportions, "%")

pie(diagnosis_proportions,
    labels = pielabels,
    clockwise = TRUE,
    col= c("green", "red"),
    border="black",
    radius = 0.8,
    cex = 1,
    main="Cancer Diagnosis")
legend(1, .5, legend = c("Benign", "Malignant"), cex = 1, fill = c("green", "red"))
```




```
# Data preprocessing (principal component analysis)

pca_data = prcomp(cleaned_data[, 2:31], center=TRUE, scale=TRUE)
plot(pca_data, type="l", main='Principal Components Weight')
grid(nx = 10, ny = 10)
title(main = "Principal Components Weight", sub = NULL, xlab = "Principal Components")
box()
```




```
summary(pca_data) # To see the difference from non pca data
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172
## Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251
## Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759 0.91010
##                           PC8    PC9    PC10   PC11    PC12    PC13    PC14
## Standard deviation     0.69037 0.6457 0.59219 0.5421 0.51104 0.49128 0.39624
## Proportion of Variance 0.01589 0.0139 0.01169 0.0098 0.00871 0.00805 0.00523
## Cumulative Proportion  0.92598 0.9399 0.95157 0.9614 0.97007 0.97812 0.98335
##                           PC15    PC16    PC17    PC18    PC19    PC20   PC21
## Standard deviation     0.30681 0.28260 0.24372 0.22939 0.22244 0.17652 0.1731
## Proportion of Variance 0.00314 0.00266 0.00198 0.00175 0.00165 0.00104 0.0010
## Cumulative Proportion  0.98649 0.98915 0.99113 0.99288 0.99453 0.99557 0.9966
##                           PC22    PC23   PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.16565 0.15602 0.1344 0.12442 0.09043 0.08307 0.03987
## Proportion of Variance 0.00091 0.00081 0.0006 0.00052 0.00027 0.00023 0.00005
## Cumulative Proportion  0.99749 0.99830 0.9989 0.99942 0.99969 0.99992 0.99997
##                           PC29    PC30
## Standard deviation     0.02736 0.01153
## Proportion of Variance 0.00002 0.00000
## Cumulative Proportion  1.00000 1.00000
```

```
# Removing highly correlated features again
pca_data_cleaned = prcomp(cleaned_data_cor, center=TRUE, scale=TRUE)
summary(pca_data_cleaned)
```

```
## Importance of components:
##                           PC1    PC2   PC3     PC4     PC5     PC6     PC7
```

```
## Standard deviation     3.053 2.1105 1.456 1.21994 1.09673 0.75004 0.66893
## Proportion of Variance 0.444 0.2121 0.101 0.07087 0.05728 0.02679 0.02131
## Cumulative Proportion  0.444 0.6561 0.757 0.82791 0.88519 0.91197 0.93328
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.56454 0.53543 0.45639 0.41367 0.34423 0.26012 0.24137
## Proportion of Variance 0.01518 0.01365 0.00992 0.00815 0.00564 0.00322 0.00277
## Cumulative Proportion  0.94846 0.96211 0.97203 0.98018 0.98582 0.98904 0.99182
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.22045 0.20547 0.17791 0.15094 0.13695 0.08384 0.02885
## Proportion of Variance 0.00231 0.00201 0.00151 0.00108 0.00089 0.00033 0.00004
## Cumulative Proportion  0.99413 0.99614 0.99765 0.99873 0.99963 0.99996 1.00000
```

```
pca_df = as.data.frame(pca_data_cleaned$x)

# Splitting training and testing data
set.seed(1208)
complete_dataset = cbind(diagnosis = cleaned_data$diagnosis, cleaned_data_cor)
index = createDataPartition(complete_dataset$diagnosis, p = 0.7, list = FALSE)

training_set = complete_dataset[ index,]
testing_set = complete_dataset[-index,]

# Building the model

fitControl = trainControl(method="cv",
                          number = 5,
                          preProcOptions = list(thresh = 0.99),
                          classProbs = TRUE,
                          summaryFunction = twoClassSummary)

pca_nnet_model = train(diagnosis~.,
                       data = training_set,
                       method="nnet",
                       metric="ROC",
                       preProcess=c('center', 'scale', 'pca'),
                       tuneLength=10,
                       trace=FALSE,
                       trControl = fitControl)
# Presenting results

predicted_pca_nnet = predict(pca_nnet_model, testing_set)
confusion_matrix_pca_nnet = confusionMatrix(predicted_pca_nnet, testing_set$diagnosis, positive = "M")
confusion_matrix_pca_nnet
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 99  5
##          M  8 58
##
##                Accuracy : 0.9235
##                  95% CI : (0.8728, 0.9587)
##     No Information Rate : 0.6294
##     P-Value [Acc > NIR] : <2e-16
```

```
##
##                   Kappa : 0.8377
##
##  Mcnemar's Test P-Value : 0.5791
##
##             Sensitivity : 0.9206
##             Specificity : 0.9252
##          Pos Pred Value : 0.8788
##          Neg Pred Value : 0.9519
##              Prevalence : 0.3706
##          Detection Rate : 0.3412
##    Detection Prevalence : 0.3882
##       Balanced Accuracy : 0.9229
##
##        'Positive' Class : M
##
```

```r
confusion_table <- as.table(confusion_matrix_pca_nnet, nrow = 2, byrow = TRUE)
fourfoldplot(confusion_table, color = c("red", "green"),
             conf.level = 0, margin = 1, main = "Confusion Matrix")
```