Project_Breast_Cancer

Leonardo Tamashiro

18/01/2020

#This document shows data Breast Cancer machine learning algorithms

First we need to import libraries

```
library(matrixStats)
```

```
## Warning: package 'matrixStats' was built under R version 3.6.2
```

```
library(tidyverse)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.2
```

```
library(dslabs)
```

```
## Warning: package 'dslabs' was built under R version 3.6.2
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.6.2
```

```
data(brca)
```

Read and check data

```
dim(brca$x)
```

```
## [1] 569  30
```

```
length(brca$y)
```

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

```
head(brca$x)
```

```
##      radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## [1,]      13.540        14.36          87.46     566.3         0.09779
## [2,]      13.080        15.71          85.63     520.0         0.10750
## [3,]       9.504        12.44          60.34     273.9         0.10240
## [4,]      13.030        18.42          82.61     523.8         0.08983
## [5,]       8.196        16.84          51.71     201.9         0.08600
## [6,]      12.050        14.63          78.04     449.3         0.10310
##      compactness_mean concavity_mean concave_pts_mean symmetry_mean
## [1,]          0.08129        0.06664         0.047810        0.1885
## [2,]          0.12700        0.04568         0.031100        0.1967
## [3,]          0.06492        0.02956         0.020760        0.1815
## [4,]          0.03766        0.02562         0.029230        0.1467
## [5,]          0.05943        0.01588         0.005917        0.1769
## [6,]          0.09092        0.06592         0.027490        0.1675
##      fractal_dim_mean radius_se texture_se perimeter_se area_se smoothness_se
## [1,]          0.05766    0.2699     0.7886        2.058  23.560      0.008462
## [2,]          0.06811    0.1852     0.7477        1.383  14.670      0.004097
## [3,]          0.06905    0.2773     0.9768        1.909  15.700      0.009606
## [4,]          0.05863    0.1839     2.3420        1.170  14.160      0.004352
## [5,]          0.06503    0.1563     0.9567        1.094   8.205      0.008968
## [6,]          0.06043    0.2636     0.7294        1.848  19.870      0.005488
##      compactness_se concavity_se concave_pts_se symmetry_se fractal_dim_se
## [1,]       0.014600      0.02387       0.013150     0.01980       0.002300
## [2,]       0.018980      0.01698       0.006490     0.01678       0.002425
## [3,]       0.014320      0.01985       0.014210     0.02027       0.002968
## [4,]       0.004899      0.01343       0.011640     0.02671       0.001777
## [5,]       0.016460      0.01588       0.005917     0.02574       0.002582
## [6,]       0.014270      0.02322       0.005660     0.01428       0.002422
##      radius_worst texture_worst perimeter_worst area_worst smoothness_worst
## [1,]       15.110         19.26           99.70      711.2           0.14400
## [2,]       14.500         20.49           96.09      630.5           0.13120
## [3,]       10.230         15.66           65.13      314.9           0.13240
## [4,]       13.300         22.81           84.46      545.9           0.09701
## [5,]        8.964         21.96           57.26      242.2           0.12970
## [6,]       13.760         20.70           89.88      582.6           0.14940
##      compactness_worst concavity_worst concave_pts_worst symmetry_worst
## [1,]           0.17730         0.23900           0.12880         0.2977
## [2,]           0.27760         0.18900           0.07283         0.3184
## [3,]           0.11480         0.08867           0.06227         0.2450
## [4,]           0.04619         0.04833           0.05013         0.1987
## [5,]           0.13570         0.06880           0.02564         0.3105
## [6,]           0.21560         0.30500           0.06548         0.2747
##      fractal_dim_worst
## [1,]           0.07259
## [2,]           0.08183
## [3,]           0.07773
```

```
## [4,]           0.06169
## [5,]           0.07409
## [6,]           0.08301
```

```
head(brca$y)
```

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

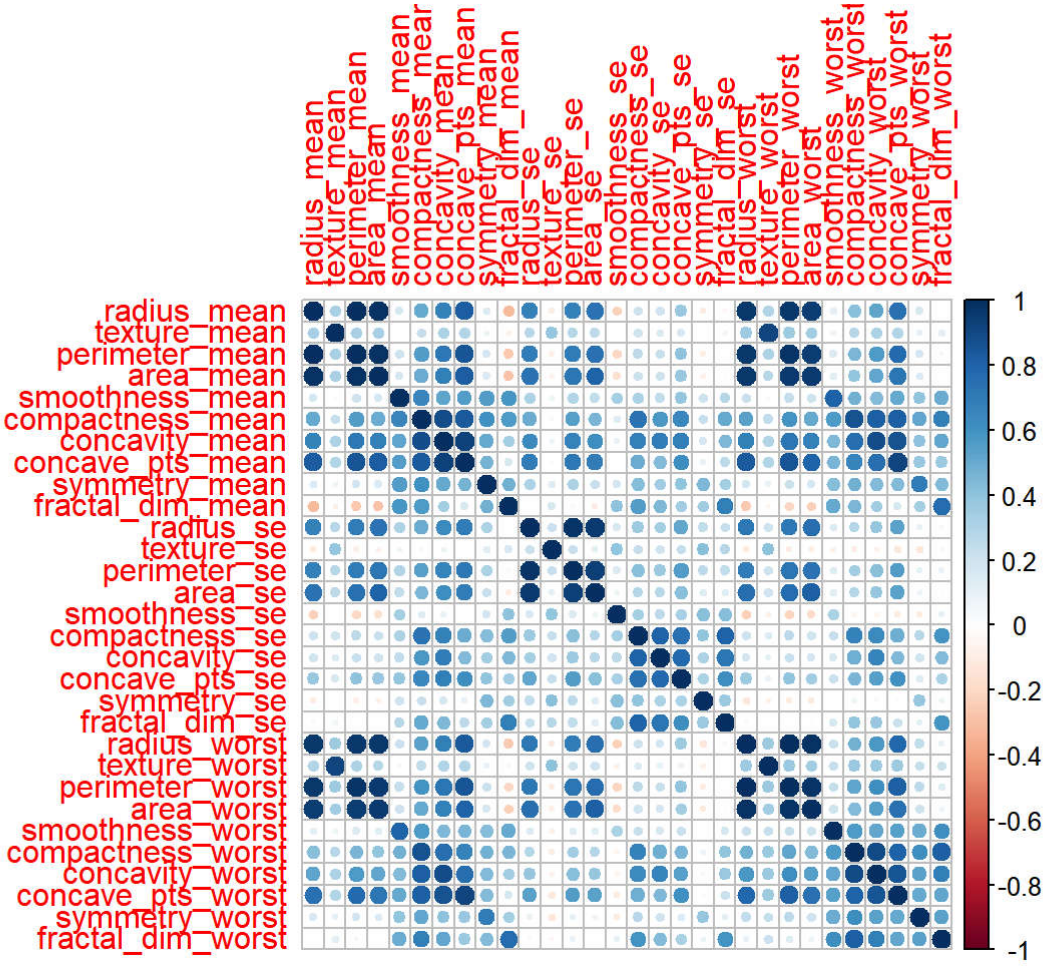
```
summary(brca$y)
```

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

```
# proportion
prop.table(table(brca$y))
```

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

```
## some correlation
corr_mat <- cor(brca$x)
corrplot(corr_mat)
```




```
# scaling the matrix

x_centered <- sweep(brca$x,2,colMeans(brca$x))
x_scaled <- sweep(x_centered,2, colSds(brca$x),FUN="/")

# after scaling standar deviation is 1 for all columns
colSds(x_scaled)
```

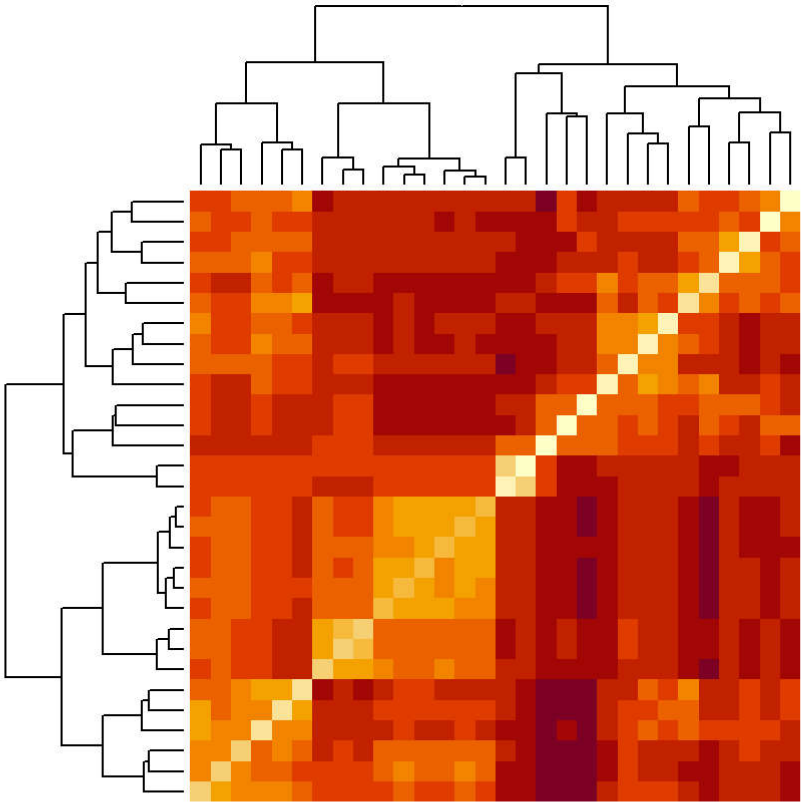
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
summary(x_scaled)
```

```
##   radius_mean       texture_mean     perimeter_mean      area_mean
##  Min.   :-2.0279   Min.   :-2.2273   Min.   :-1.9828   Min.   :-1.4532
##  1st Qu.:-0.6888   1st Qu.:-0.7253   1st Qu.:-0.6913   1st Qu.:-0.6666
##  Median :-0.2149   Median :-0.1045   Median :-0.2358   Median :-0.2949
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.4690   3rd Qu.: 0.5837   3rd Qu.: 0.4992   3rd Qu.: 0.3632
##  Max.   : 3.9678   Max.   : 4.6478   Max.   : 3.9726   Max.   : 5.2459
##  smoothness_mean    compactness_mean  concavity_mean    concave_pts_mean
##  Min.   :-3.10935   Min.   :-1.6087   Min.   :-1.1139   Min.   :-1.2607
##  1st Qu.:-0.71034   1st Qu.:-0.7464   1st Qu.:-0.7431   1st Qu.:-0.7373
##  Median :-0.03486   Median :-0.2217   Median :-0.3419   Median :-0.3974
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.63564   3rd Qu.: 0.4934   3rd Qu.: 0.5256   3rd Qu.: 0.6464
##  Max.   : 4.76672   Max.   : 4.5644   Max.   : 4.2399   Max.   : 3.9245
##  symmetry_mean      fractal_dim_mean    radius_se         texture_se
##  Min.   :-2.74171   Min.   :-1.8183   Min.   :-1.0590   Min.   :-1.5529
##  1st Qu.:-0.70262   1st Qu.:-0.7220   1st Qu.:-0.6230   1st Qu.:-0.6942
##  Median :-0.07156   Median :-0.1781   Median :-0.2920   Median :-0.1973
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.53031   3rd Qu.: 0.4706   3rd Qu.: 0.2659   3rd Qu.: 0.4661
##  Max.   : 4.48081   Max.   : 4.9066   Max.   : 8.8991   Max.   : 6.6494
##   perimeter_se        area_se        smoothness_se     compactness_se
##  Min.   :-1.0431   Min.   :-0.7372   Min.   :-1.7745   Min.   :-1.2970
##  1st Qu.:-0.6232   1st Qu.:-0.4943   1st Qu.:-0.6235   1st Qu.:-0.6923
##  Median :-0.2864   Median :-0.3475   Median :-0.2201   Median :-0.2808
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.2428   3rd Qu.: 0.1067   3rd Qu.: 0.3680   3rd Qu.: 0.3893
##  Max.   : 9.4537   Max.   :11.0321   Max.   : 8.0229   Max.   : 6.1381
##   concavity_se     concave_pts_se     symmetry_se      fractal_dim_se
##  Min.   :-1.0566   Min.   :-1.9118   Min.   :-1.5315   Min.   :-1.0960
##  1st Qu.:-0.5567   1st Qu.:-0.6739   1st Qu.:-0.6511   1st Qu.:-0.5846
##  Median :-0.1989   Median :-0.1404   Median :-0.2192   Median :-0.2297
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.3365   3rd Qu.: 0.4722   3rd Qu.: 0.3554   3rd Qu.: 0.2884
##  Max.   :12.0621   Max.   : 6.6438   Max.   : 7.0657   Max.   : 9.8429
##   radius_worst     texture_worst      perimeter_worst     area_worst
##  Min.   :-1.7254   Min.   :-2.22204   Min.   :-1.6919   Min.   :-1.2213
##  1st Qu.:-0.6743   1st Qu.:-0.74797   1st Qu.:-0.6890   1st Qu.:-0.6416
##  Median :-0.2688   Median :-0.04348   Median :-0.2857   Median :-0.3409
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5216   3rd Qu.: 0.65776   3rd Qu.: 0.5398   3rd Qu.: 0.3573
##  Max.   : 4.0906   Max.   : 3.88249   Max.   : 4.2836   Max.   : 5.9250
##  smoothness_worst  compactness_worst concavity_worst   concave_pts_worst
##  Min.   :-2.6803   Min.   :-1.4426   Min.   :-1.3047   Min.   :-1.7435
##  1st Qu.:-0.6906   1st Qu.:-0.6805   1st Qu.:-0.7558   1st Qu.:-0.7557
##  Median :-0.0468   Median :-0.2693   Median :-0.2180   Median :-0.2233
```

```
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5970   3rd Qu.: 0.5392   3rd Qu.: 0.5307   3rd Qu.: 0.7119
##  Max.   : 3.9519   Max.   : 5.1084   Max.   : 4.6965   Max.   : 2.6835
##  symmetry_worst    fractal_dim_worst
##  Min.   :-2.1591   Min.   :-1.6004
##  1st Qu.:-0.6413   1st Qu.:-0.6913
##  Median :-0.1273   Median :-0.2163
##  Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.4497   3rd Qu.: 0.4504
##  Max.   : 6.0407   Max.   : 6.8408
```

```
# heatmap of the relationship between features using the scaled matrix
d_features <- dist(t(x_scaled))
heatmap(as.matrix(d_features),labRow=NA,labCol=NA)
```



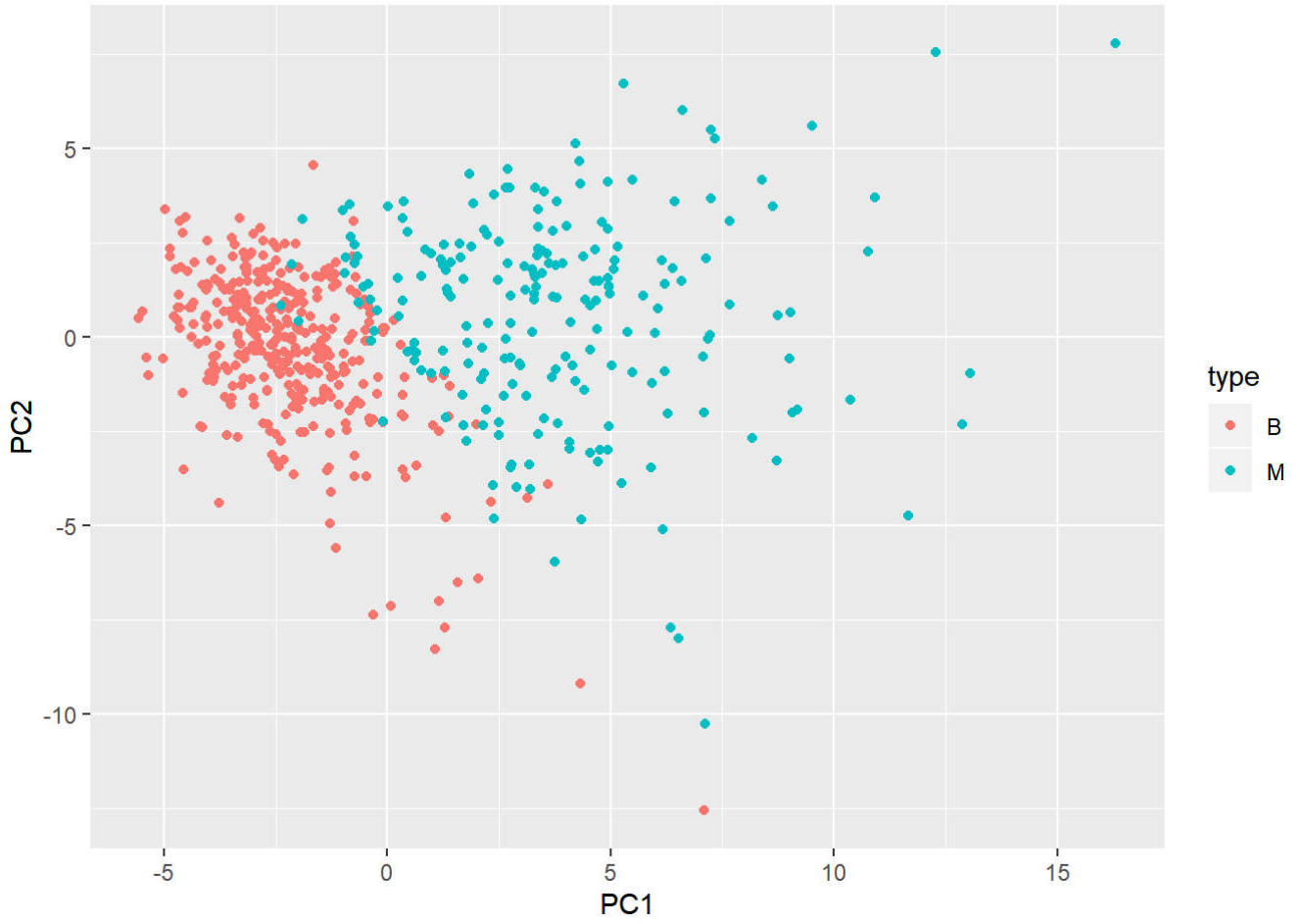
```
# clustering
h <- hclust(d_features)
groups <- cutree(h,k=5)
split(names(groups),groups)
```

```
## $`1`
##  [1] "radius_mean"       "perimeter_mean"    "area_mean"
##  [4] "concavity_mean"    "concave_pts_mean"  "radius_se"
##  [7] "perimeter_se"      "area_se"           "radius_worst"
## [10] "perimeter_worst"   "area_worst"        "concave_pts_worst"
##
## $`2`
## [1] "texture_mean"  "texture_worst"
##
## $`3`
## [1] "smoothness_mean"   "compactness_mean"  "symmetry_mean"
## [4] "fractal_dim_mean"  "smoothness_worst"  "compactness_worst"
## [7] "concavity_worst"   "symmetry_worst"    "fractal_dim_worst"
##
## $`4`
## [1] "texture_se"    "smoothness_se" "symmetry_se"
##
## $`5`
## [1] "compactness_se" "concavity_se"   "concave_pts_se" "fractal_dim_se"
```

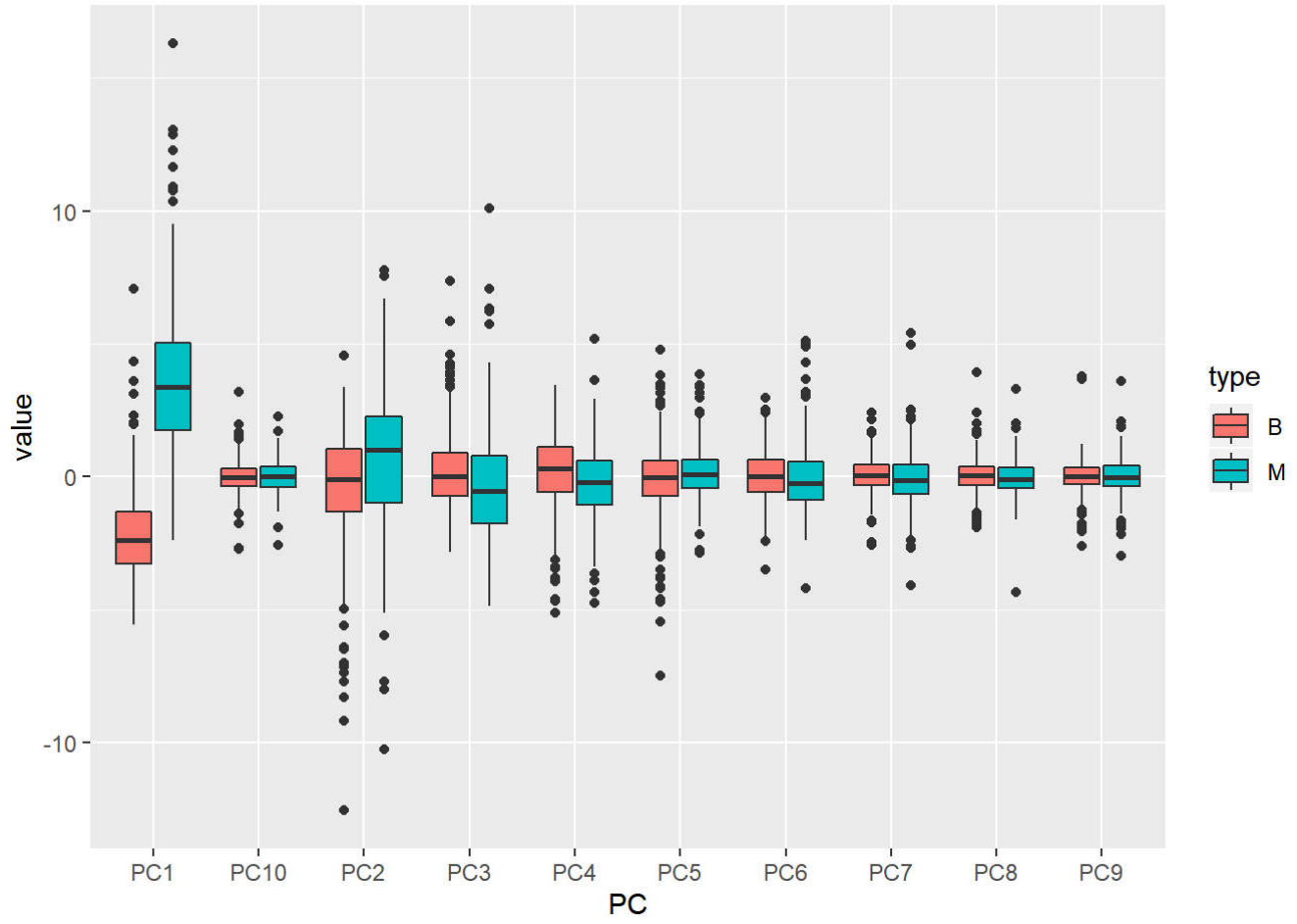
```
# PCA
pca <- prcomp(x_scaled)
summary(pca)
```

```
## 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
```

```
# Plotting PCs, we can see the benign tumors tend to have smaller values of PC1 and
# higher values for malignant tumors
data.frame(pca$x[,1:2],type = brca$y) %>%
  ggplot(aes(PC1,PC2,color=type)) +
  geom_point()
```




```
# Plotting PCs, boxplot.  We can see PC1 is significantly different from others
data.frame(type = brca$y ,pca$x[,1:10]) %>%
  gather(key = "PC",value="value", -type) %>%
  ggplot(aes(PC,value,fill = type)) +
  geom_boxplot()
```




```
  geom_point()
```

```
## geom_point: na.rm = FALSE
## stat_identity: na.rm = FALSE
## position_identity
```

```
# Creating data partition
set.seed(1,sample.kind = "Rounding")
test_index <- createDataPartition(brca$y, time=1, p=0.2,list=FALSE)
test_x <- x_scaled[test_index,]
test_y <- brca$y[test_index]
train_x <- x_scaled[-test_index,]
train_y <- brca$y[-test_index]
```

```
# We can see train and test sets have similar proportions
# proportion test
prop.table(table(test_y))
```

```
## test_y
##         B         M
## 0.626087 0.373913
```

```
# proportion train
prop.table(table(train_y))
```

```
## train_y
##          B          M
## 0.6277533 0.3722467
```

```
# K-means Clustering
predict_kmeans <- function(x, k) {
  centers <- k$centers
  distances <- sapply(1:nrow(x), function(i){
    apply(centers, 1, function(y) dist(rbind(x[i,], y)))
  })
  max.col(-t(distances))
}

set.seed(3,sample.kind = "Rounding")
k <- kmeans(train_x, centers = 2)
kmeans_preds <- ifelse(predict_kmeans(test_x, k) == 1, "B", "M")

# K-means overall accuracy
mean(kmeans_preds == test_y)
```

```
## [1] 0.9217391
```

```
# Logistic Regression
train_glm <- train(train_x,train_y,method="glm")
glm_preds <- predict(train_glm,test_x)

# Logistic Regression overall accuracy
mean(glm_preds==test_y)
```

```
## [1] 0.9565217
```

```
# LDA and QDA models
train_lda <- train(train_x,train_y,method="lda")
lda_preds <- predict(train_lda,test_x)
mean(lda_preds==test_y)
```

```
## [1] 0.9913043
```

```
train_qda <- train(train_x,train_y,method="qda")
qda_preds <- predict(train_qda,test_x)
mean(qda_preds==test_y)
```

```
## [1] 0.9565217
```

```
# Loess model
set.seed(5, sample.kind = "Rounding")
train_loess <- train(train_x,train_y,method="gamLoess")

loess_preds <- predict(train_loess,test_x)
mean(loess_preds==test_y)
```

```
## [1] 0.9826087
```

```
# K-nearest neighbors
set.seed(7, sample.kind = "Rounding")
tuning <- data.frame(k=seq(3,21,2))
train_knn <- train(train_x,train_y,method="knn",tuneGrid = tuning)
train_knn$bestTune
```

```
##    k
## 10 21
```

```
knn_preds <- predict(train_knn,test_x)
mean(knn_preds == test_y)
```

```
## [1] 0.9478261
```

```
# Random Forest Model
set.seed(9, sample.kind = "Rounding")
tuning <- data.frame(mtry=c(3,5,7,9))
train_rf <- train(train_x,train_y, method="rf",tuneGrid = tuning, importance = TRUE
  )
train_rf$bestTune
```

```
##   mtry
## 1    3
```

```
rf_preds <- predict(train_rf, test_x)
mean(rf_preds == test_y)
```

```
## [1] 0.973913
```

```
# Ensemble
ensemble <- cbind(glm=glm_preds=="B",lda=lda_preds=="B",qda=qda_preds=="B",loess=lo
  ess_preds=="B",
                  rf=rf_preds=="B",knn=knn_preds=="B",kmeans=kmeans_preds=="B")

ensemble_preds <- ifelse(rowMeans(ensemble) >0.5,"B","M")
mean(ensemble_preds==test_y)
```

```
## [1] 0.9826087
```

```
models <- c("K means", "Logistic regression", "LDA", "QDA", "Loess", "K nearest nei
  ghbors", "Random fore
st", "Ensemble")
accuracy <- c(mean(kmeans_preds == test_y),
              mean(glm_preds == test_y),
              mean(lda_preds == test_y),
              mean(qda_preds == test_y),
              mean(loess_preds == test_y),
              mean(knn_preds == test_y),
              mean(rf_preds == test_y),
              mean(ensemble_preds == test_y))
data.frame(Model = models, Accuracy = accuracy)
```

```
##                 Model  Accuracy
## 1             K means 0.9217391
## 2 Logistic regression 0.9565217
## 3                 LDA 0.9913043
## 4                 QDA 0.9565217
## 5               Loess 0.9826087
## 6 K nearest neighbors 0.9478261
## 7     Random fore\nst 0.9739130
## 8            Ensemble 0.9826087
```

Conclusion LDA Model has the highest accuracy