Lab 4: Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA)

Makesh Srinivasan

3/6/2022

Registration number: 19BCE1717 Faculty: Prof. Parvathi R Slot: L55 + L56 Course code: CSE3020

PCA LDA Lab exercise

Question

- Part 1: Use Breast Cancer Wisconsin data set from the UCI Machine learning repo to plot the PCA analysis. Use the 'prcomp' function runs PCA on the data.
  - You want to explain difference between malignant and benign tumors using Visualisation and add the response variable (diagnosis) to the plot
  - Construct some kind of model using the first 6 principal components to predict whether a tumor is benign or malignant and then compare it to a model using the original 30 variables.
- Part 2: Use the built-in iris dataset in R to plot the LDA analysis. Use the lda function of the MASS package in R Project the LDA visual output and Compare the LDA and PCA 2D Projection of Iris dataset

---

Sections:

- Part 1: PCA & LDA using Breast Cancer Wisconsin data set from the UCI Machine learning repo
  - Visualisation of benign and malignant
  - Model to classify benign and malignant
- Part 2: IRIS dataset

PART 1

PCA & LDA using Breast Cancer Wisconsin data set from the UCI Machine learning repo

Load necessary libraries:

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.2
```

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

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

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

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

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

```r
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.0.2
```

```r
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 4.0.2

## Loading required package: usethis

## Warning: package 'usethis' was built under R version 4.0.2
```

```r
# NOTE: I have imported other libraries as and when needed, theses are the ones I needed prior to perfo
```

Load the dataset:

```r
data <- read.csv("lab5data.csv")
head(data)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean concavity_mean concave_points_mean
## 1         0.11840          0.27760         0.3001             0.14710
## 2         0.08474          0.07864         0.0869             0.07017
## 3         0.10960          0.15990         0.1974             0.12790
## 4         0.14250          0.28390         0.2414             0.10520
## 5         0.10030          0.13280         0.1980             0.10430
## 6         0.12780          0.17000         0.1578             0.08089
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.2419                0.07871    1.0950     0.9053        8.589
## 2        0.1812                0.05667    0.5435     0.7339        3.398
## 3        0.2069                0.05999    0.7456     0.7869        4.585
## 4        0.2597                0.09744    0.4956     1.1560        3.445
## 5        0.1809                0.05883    0.7572     0.7813        5.438
## 6        0.2087                0.07613    0.3345     0.8902        2.217
##   area_se smoothness_se compactness_se concavity_se concave_points_se
## 1  153.40      0.006399        0.04904      0.05373           0.01587
## 2   74.08      0.005225        0.01308      0.01860           0.01340
## 3   94.03      0.006150        0.04006      0.03832           0.02058
## 4   27.23      0.009110        0.07458      0.05661           0.01867
## 5   94.44      0.011490        0.02461      0.05688           0.01885
```

```
## 6   27.19      0.007510        0.03345      0.03672           0.01137
##   symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 1     0.03003             0.006193        25.38         17.33          184.60
## 2     0.01389             0.003532        24.99         23.41          158.80
## 3     0.02250             0.004571        23.57         25.53          152.50
## 4     0.05963             0.009208        14.91         26.50           98.87
## 5     0.01756             0.005115        22.54         16.67          152.20
## 6     0.02165             0.005082        15.47         23.75          103.40
##   area_worst smoothness_worst compactness_worst concavity_worst
## 1     2019.0           0.1622            0.6656          0.7119
## 2     1956.0           0.1238            0.1866          0.2416
## 3     1709.0           0.1444            0.4245          0.4504
## 4      567.7           0.2098            0.8663          0.6869
## 5     1575.0           0.1374            0.2050          0.4000
## 6      741.6           0.1791            0.5249          0.5355
##   concave_points_worst symmetry_worst fractal_dimension_worst
## 1               0.2654         0.4601                 0.11890
## 2               0.1860         0.2750                 0.08902
## 3               0.2430         0.3613                 0.08758
## 4               0.2575         0.6638                 0.17300
## 5               0.1625         0.2364                 0.07678
## 6               0.1741         0.3985                 0.12440
```

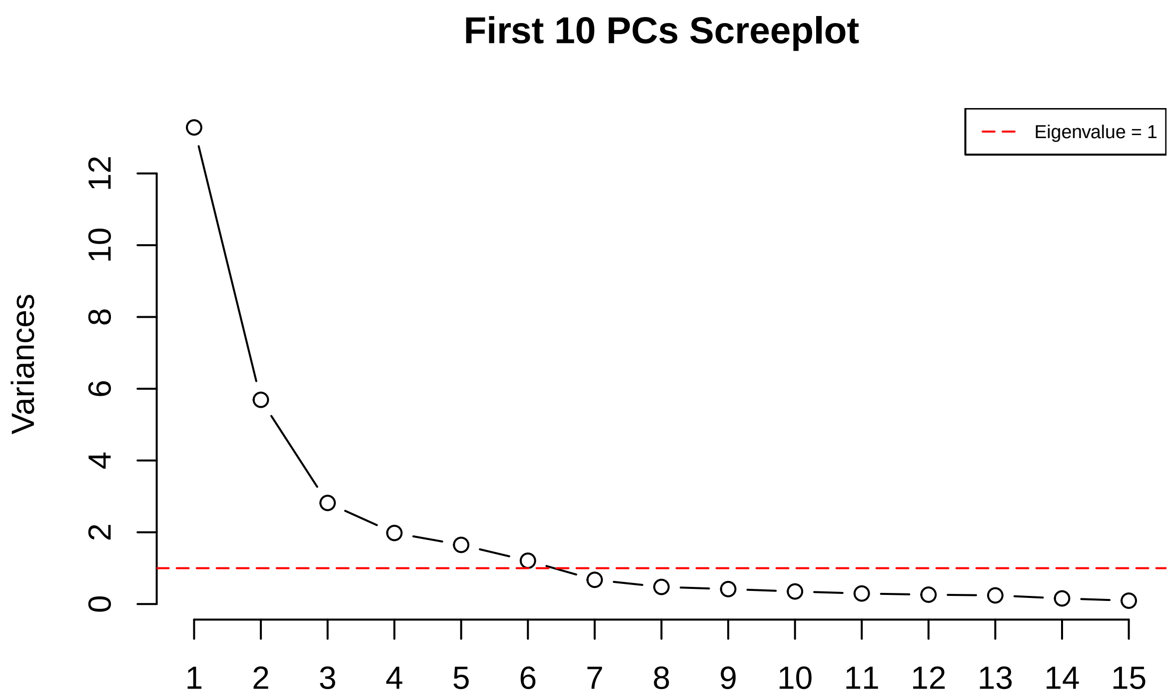
Run prcomp function and summarise the data:

```r
# Center and scale the data as well
data.pr <- prcomp(data[c(3:32)], center = TRUE, scale = TRUE)
summary(data.pr)
```

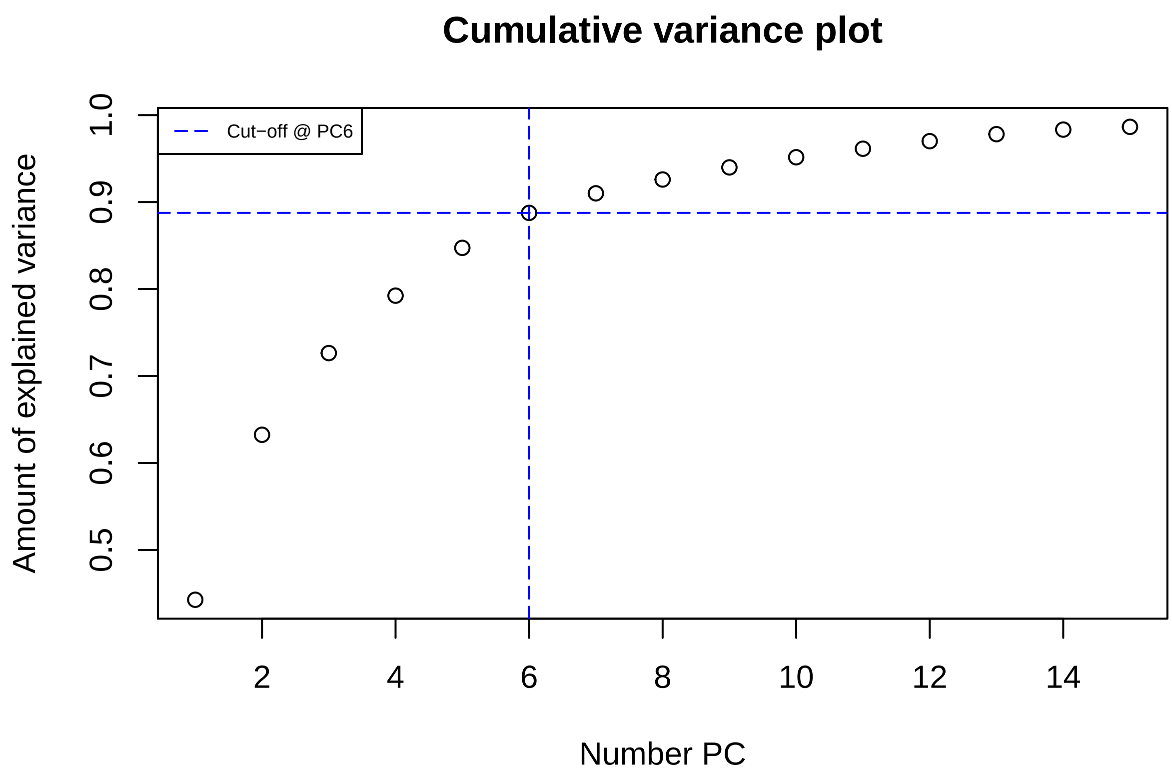
```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172
## Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251
## Cumulative Proportion  0.4427 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
```

The data is now standardised

```r
screeplot(data.pr, type = "l", npcs = 15, main = "First 10 PCs Screeplot")
abline(h = 1, col="red", lty=5)
legend("topright", legend=c("Eigenvalue = 1"), col=c("red"), lty=5, cex=0.6)
```

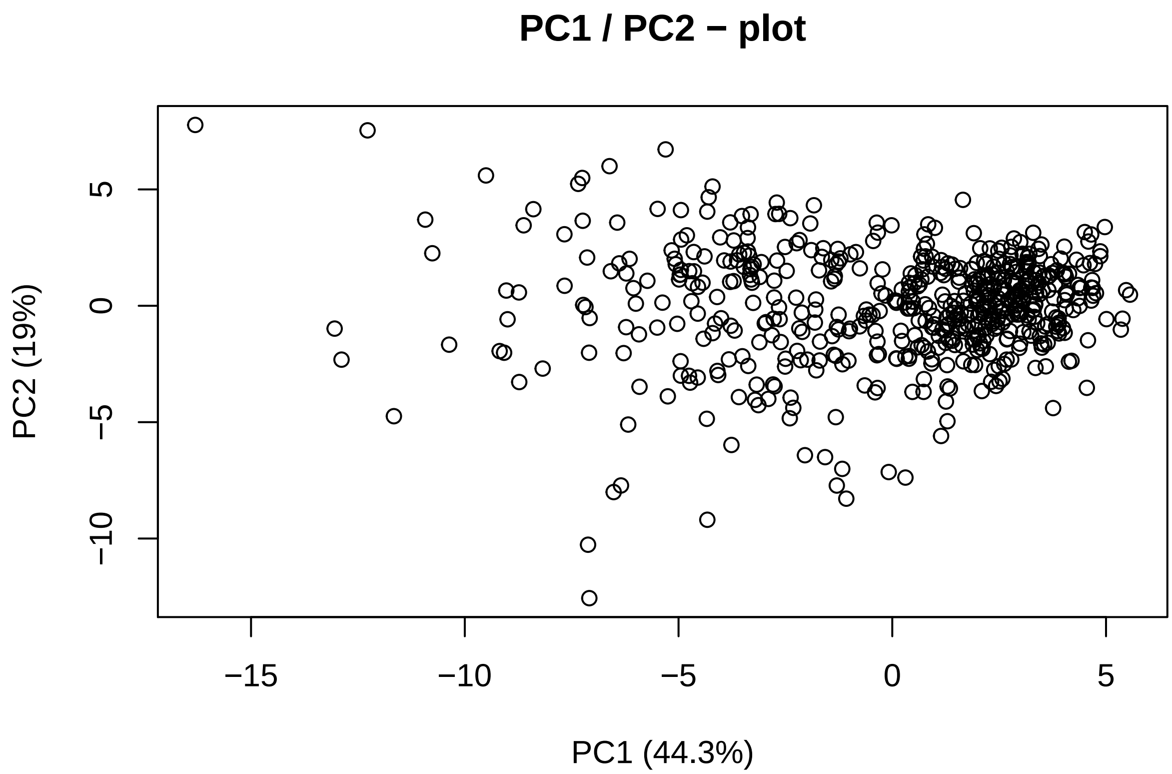



```
cumpro <- cumsum(data.pr$sdev^2 / sum(data.pr$sdev^2))
plot(cumpro[0:15], xlab = "Number PC", ylab = "Amount of explained variance", main = "Cumulative varian
abline(v = 6, col="blue", lty=5)
abline(h = 0.88759, col="blue", lty=5)
legend("topleft", legend=c("Cut-off @ PC6"), col=c("blue"), lty=5, cex=0.6)
```



INFERENCE: the first 6 components has an Eigenvalue > 1 and explains almost 90% of variance Hence, we can reduce the dimension from 30 to 6

```r
plot(data.pr$x[,1],data.pr$x[,2], xlab="PC1 (44.3%)", ylab = "PC2 (19%)", main = "PC1 / PC2 - plot")
```



INFERENCE: Here, we observe the first two components that explain 60% of the total variance

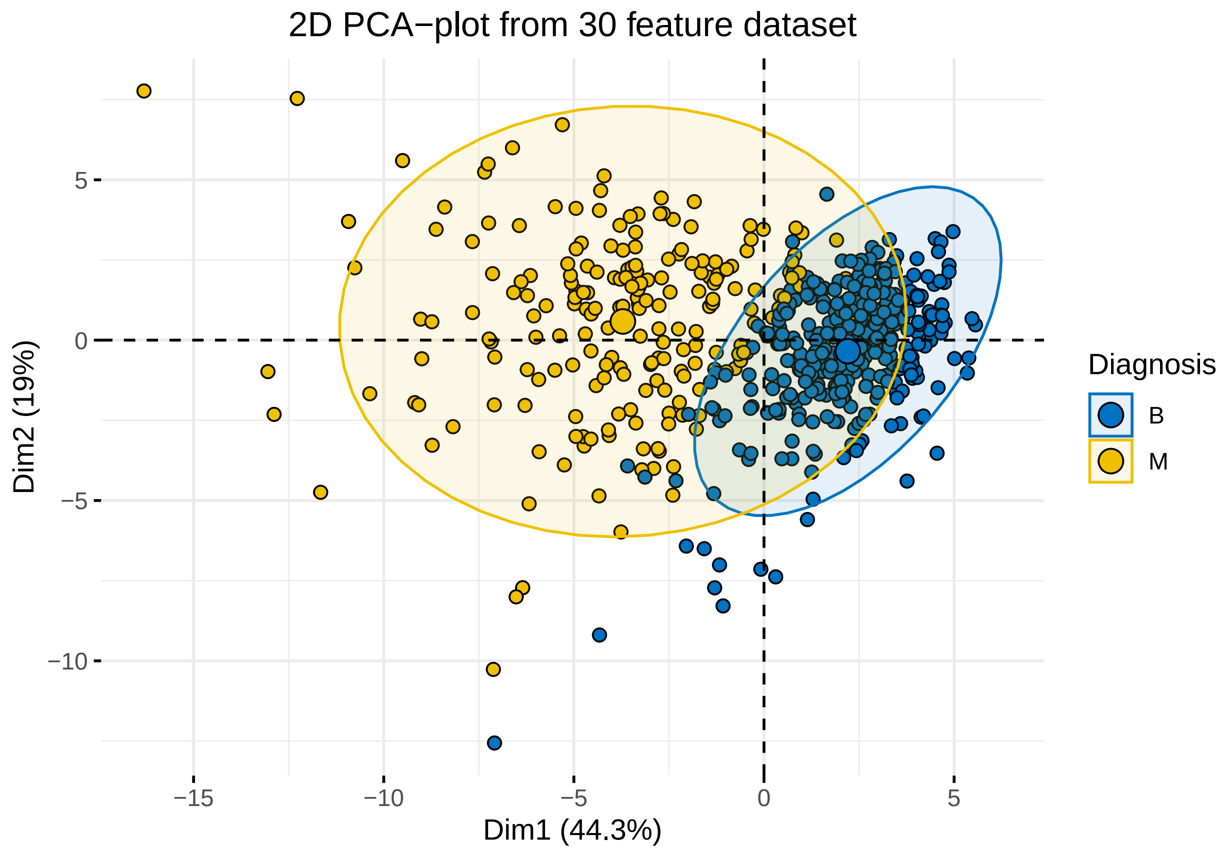
```r
# visualisation library
library("factoextra")
```

```
## Warning: package 'factoextra' was built under R version 4.0.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.2

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

Visualisation of benign and malignant

(i) You want to explain difference between malignant and benign tumors using Visualisation and add the response variable (diagnosis) to the plot

```r
fviz_pca_ind(data.pr, geom.ind = "point", pointshape = 21, pointsize = 2, fill.ind = data$diagnosis, co
  ggtitle("2D PCA-plot from 30 feature dataset") +
  theme(plot.title = element_text(hjust = 0.5))
```



INFERENCE: The same when plotted with the response variable (diagnosis) we see a clear difference between the benign and malignant tumours. Thus, we can run classification algorithms on this. We have visualised and explained the difference between the benign (blue B) and malignant (yellow M) on the plot above along with the response variable diagnosis. Now, we can move to part 2.

Check for missing values:

```
data %>%
  summarise_all(funs(sum(is.na(.))))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))

##   id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1  0         0           0            0              0         0
##   smoothness_mean compactness_mean concavity_mean concave_points_mean
## 1               0                0              0                   0
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1             0                      0         0          0            0
##   area_se smoothness_se compactness_se concavity_se concave_points_se
```

```
## 1       0              0               0             0                 0
##   symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 1           0                    0            0             0               0
##   area_worst smoothness_worst compactness_worst concavity_worst
## 1          0                0                 0               0
##   concave_points_worst symmetry_worst fractal_dimension_worst
## 1                    0              0                       0
```

Model to classify benign and malignant

(ii) Construct some kind of model using the first 6 principal components to predict whether a tumor is benign or malignant and then compare it to a model using the original 30 variables.

```
# Conver the dataset into matix
data.data <- as.matrix(data[,c(3:32)])
row.names(data.data) <- data$id
data_raw <- cbind(data.data, as.numeric(as.factor(data$diagnosis))-1)
colnames(data_raw)[31] <- "diagnosis"

# 75/25 split of our data using the sample()
s_size_raw <- floor(0.75 * nrow(data_raw))
t_raw <- sample(nrow(data_raw), size = s_size_raw)
train_raw.data <- as.data.frame(data_raw[t_raw, ])
test_raw.data <- as.data.frame(data_raw[-t_raw, ])
f <- paste(names(train_raw.data)[31], "~", paste(names(train_raw.data)[-31], collapse=" + "))
data_raw.lda <- lda(as.formula(paste(f)), data = train_raw.data)

# Predictions
data_raw.lda.predict <- predict(data_raw.lda, newdata = test_raw.data)
data_raw.lda.predict$class
```

```
##   [1] 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 1 0
##  [38] 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1
## [112] 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1
## Levels: 0 1
```

```
data_raw.lda.predict$posterior
```

```
##                      0            1
## 84300903  8.991221e-06 9.999910e-01
## 843786    4.123639e-02 9.587636e-01
## 845636    4.159865e-01 5.840135e-01
## 84610002  2.001809e-03 9.979982e-01
## 84667401  4.129095e-02 9.587090e-01
## 848406    6.096502e-03 9.939035e-01
## 8510824   9.999943e-01 5.690670e-06
## 8511133   1.625861e-02 9.837414e-01
## 852552    5.978116e-06 9.999940e-01
## 855167    9.753195e-01 2.468048e-02
## 855625    6.432032e-07 9.999994e-01
## 856106    1.006695e-01 8.993305e-01
## 857155    9.992917e-01 7.083355e-04
## 857373    9.997079e-01 2.920803e-04
```

```
## 857392    8.233463e-02 9.176654e-01
## 85759902  9.963011e-01 3.698924e-03
## 858970    9.999841e-01 1.591357e-05
## 858986    9.863938e-04 9.990136e-01
## 859464    9.998918e-01 1.082269e-04
## 859465    9.988646e-01 1.135375e-03
## 861648    9.932195e-01 6.780514e-03
## 862261    9.999582e-01 4.183352e-05
## 862722    1.000000e+00 1.575984e-08
## 862980    9.998327e-01 1.672789e-04
## 863031    9.717026e-01 2.829737e-02
## 864033    9.996956e-01 3.044382e-04
## 864292    9.999962e-01 3.849143e-06
## 864877    4.225345e-06 9.999958e-01
## 86517     9.996640e-04 9.990003e-01
## 865468    9.997878e-01 2.121810e-04
## 866083    3.018401e-01 6.981599e-01
## 866714    9.996183e-01 3.816707e-04
## 86730502  3.087375e-01 6.912625e-01
## 868202    9.100062e-01 8.999384e-02
## 868999    9.999996e-01 4.276457e-07
## 869104    1.388029e-01 8.611971e-01
## 869476    9.998160e-01 1.840281e-04
## 871001501 9.963978e-01 3.602247e-03
## 8710441   9.999998e-01 1.886357e-07
## 8711003   9.985576e-01 1.442438e-03
## 8711202   1.543636e-02 9.845636e-01
## 8712289   2.036046e-05 9.999796e-01
## 87163     1.348687e-01 8.651313e-01
## 872608    9.998626e-01 1.373545e-04
## 873357    9.999723e-01 2.766399e-05
## 874662    9.998732e-01 1.267771e-04
## 875878    9.995924e-01 4.075714e-04
## 878796    2.671726e-07 9.999997e-01
## 87930     9.367999e-01 6.320013e-02
## 881046502 4.634196e-04 9.995366e-01
## 8810987   7.548316e-01 2.451684e-01
## 8813129   9.879078e-01 1.209223e-02
## 88143502  9.255153e-01 7.448471e-02
## 881972    7.133126e-04 9.992867e-01
## 88199202  9.999749e-01 2.508937e-05
## 883263    3.945876e-03 9.960541e-01
## 88350402  9.960543e-01 3.945675e-03
## 884437    9.490318e-01 5.096817e-02
## 884448    9.996692e-01 3.308180e-04
## 884626    9.785912e-01 2.140882e-02
## 884689    9.997149e-01 2.851438e-04
## 88518501  9.995471e-01 4.528881e-04
## 8860702   1.158524e-02 9.884148e-01
## 888570    1.663806e-02 9.833619e-01
## 889719    1.130160e-03 9.988698e-01
## 8910506   9.995263e-01 4.736671e-04
## 8910720   9.999890e-01 1.096421e-05
## 8911800   9.995715e-01 4.284673e-04
```

```
## 8911834   9.986101e-01 1.389908e-03
## 8912521   9.999734e-01 2.661839e-05
## 8913      9.999835e-01 1.648992e-05
## 891716    9.999950e-01 5.008158e-06
## 891923    9.999563e-01 4.370481e-05
## 892189    9.991401e-01 8.599387e-04
## 892214    9.979281e-01 2.071904e-03
## 893988    9.999990e-01 9.919843e-07
## 894329    9.999917e-01 8.274583e-06
## 894604    9.997606e-01 2.394069e-04
## 89524     9.994984e-01 5.016483e-04
## 8953902   9.881245e-03 9.901188e-01
## 896864    9.973703e-01 2.629657e-03
## 897374    9.997030e-01 2.969798e-04
## 89742801  2.416886e-04 9.997583e-01
## 897880    9.999737e-01 2.629172e-05
## 898143    9.999709e-01 2.914775e-05
## 89827     9.997540e-01 2.460290e-04
## 898431    6.824358e-06 9.999932e-01
## 899187    9.997495e-01 2.504724e-04
## 9010259   9.902040e-01 9.795963e-03
## 9010333   9.999865e-01 1.349050e-05
## 901034301 9.998212e-01 1.787687e-04
## 9010598   9.997974e-01 2.025843e-04
## 9012000   1.068753e-05 9.999893e-01
## 9012568   9.995919e-01 4.081174e-04
## 901288    1.961808e-04 9.998038e-01
## 901303    9.962591e-01 3.740910e-03
## 9013594   9.950711e-01 4.928917e-03
## 901549    9.935638e-01 6.436233e-03
## 90251     9.980682e-01 1.931776e-03
## 90291     1.814239e-01 8.185761e-01
## 902975    9.998119e-01 1.881267e-04
## 90317302  9.999974e-01 2.584421e-06
## 90401601  9.876037e-01 1.239631e-02
## 90401602  9.999075e-01 9.253010e-05
## 904357    9.997364e-01 2.636139e-04
## 904647    9.997367e-01 2.633175e-04
## 905520    9.999304e-01 6.964227e-05
## 905978    9.968602e-01 3.139809e-03
## 906564    9.886223e-01 1.137770e-02
## 906616    9.998557e-01 1.442702e-04
## 908445    8.404531e-04 9.991595e-01
## 908916    9.989063e-01 1.093663e-03
## 909410    9.997914e-01 2.086145e-04
## 90944601  9.999624e-01 3.762122e-05
## 9110127   5.328172e-01 4.671828e-01
## 9110720   9.806887e-01 1.931129e-02
## 9110732   4.800372e-05 9.999520e-01
## 9111805   7.376928e-04 9.992623e-01
## 9112712   9.999631e-01 3.689365e-05
## 911320502 9.946997e-01 5.300342e-03
## 9113514   9.999084e-01 9.159384e-05
## 9113538   2.683380e-04 9.997317e-01
```

```
## 9113778   9.998314e-01 1.685669e-04
## 911391    9.999753e-01 2.471276e-05
## 911654    9.916282e-01 8.371850e-03
## 912519    9.974912e-01 2.508758e-03
## 913505    9.699904e-06 9.999903e-01
## 913535    8.550119e-01 1.449881e-01
## 914862    9.990388e-01 9.612205e-04
## 915276    9.998550e-01 1.450250e-04
## 915452    8.752674e-01 1.247326e-01
## 916221    9.996041e-01 3.959154e-04
## 916838    9.809319e-04 9.990191e-01
## 917062    9.997031e-01 2.969290e-04
## 91789     9.999961e-01 3.892011e-06
## 91903901  9.989556e-01 1.044404e-03
## 919812    9.885069e-01 1.149314e-02
## 921362    9.999976e-01 2.387447e-06
## 921385    9.999940e-01 5.998938e-06
## 923465    9.982527e-01 1.747327e-03
## 924632    9.832329e-01 1.676705e-02
## 925292    9.434940e-01 5.650604e-02
## 925622    1.043746e-06 9.999990e-01
```

```
data_raw.lda.predict.posteriors <-as.data.frame(data_raw.lda.predict$posterior)
data_raw.lda.predict.posteriors
```

```
##                      0            1
## 84300903  8.991221e-06 9.999910e-01
## 843786    4.123639e-02 9.587636e-01
## 845636    4.159865e-01 5.840135e-01
## 84610002  2.001809e-03 9.979982e-01
## 84667401  4.129095e-02 9.587090e-01
## 848406    6.096502e-03 9.939035e-01
## 8510824   9.999943e-01 5.690670e-06
## 8511133   1.625861e-02 9.837414e-01
## 852552    5.978116e-06 9.999940e-01
## 855167    9.753195e-01 2.468048e-02
## 855625    6.432032e-07 9.999994e-01
## 856106    1.006695e-01 8.993305e-01
## 857155    9.992917e-01 7.083355e-04
## 857373    9.997079e-01 2.920803e-04
## 857392    8.233463e-02 9.176654e-01
## 85759902  9.963011e-01 3.698924e-03
## 858970    9.999841e-01 1.591357e-05
## 858986    9.863938e-04 9.990136e-01
## 859464    9.998918e-01 1.082269e-04
## 859465    9.988646e-01 1.135375e-03
## 861648    9.932195e-01 6.780514e-03
## 862261    9.999582e-01 4.183352e-05
## 862722    1.000000e+00 1.575984e-08
## 862980    9.998327e-01 1.672789e-04
## 863031    9.717026e-01 2.829737e-02
## 864033    9.996956e-01 3.044382e-04
## 864292    9.999962e-01 3.849143e-06
## 864877    4.225345e-06 9.999958e-01
## 86517     9.996640e-04 9.990003e-01
```

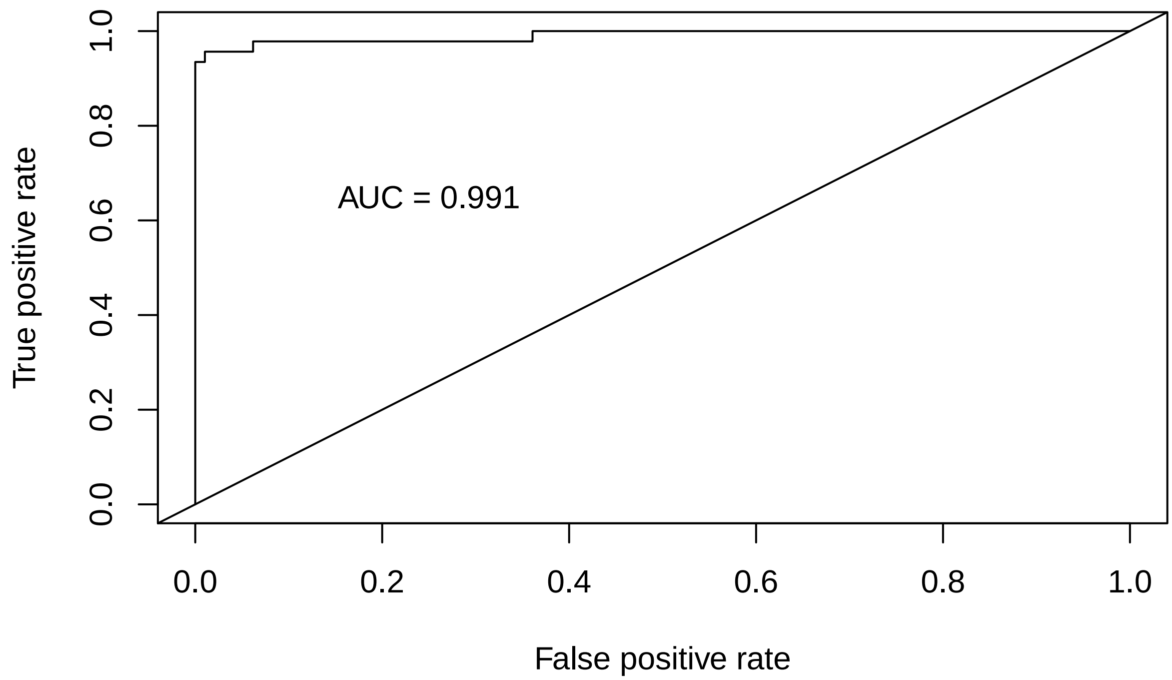
```
## 865468    9.997878e-01 2.121810e-04
## 866083    3.018401e-01 6.981599e-01
## 866714    9.996183e-01 3.816707e-04
## 86730502  3.087375e-01 6.912625e-01
## 868202    9.100062e-01 8.999384e-02
## 868999    9.999996e-01 4.276457e-07
## 869104    1.388029e-01 8.611971e-01
## 869476    9.998160e-01 1.840281e-04
## 871001501 9.963978e-01 3.602247e-03
## 8710441   9.999998e-01 1.886357e-07
## 8711003   9.985576e-01 1.442438e-03
## 8711202   1.543636e-02 9.845636e-01
## 8712289   2.036046e-05 9.999796e-01
## 87163     1.348687e-01 8.651313e-01
## 872608    9.998626e-01 1.373545e-04
## 873357    9.999723e-01 2.766399e-05
## 874662    9.998732e-01 1.267771e-04
## 875878    9.995924e-01 4.075714e-04
## 878796    2.671726e-07 9.999997e-01
## 87930     9.367999e-01 6.320013e-02
## 881046502 4.634196e-04 9.995366e-01
## 8810987   7.548316e-01 2.451684e-01
## 8813129   9.879078e-01 1.209223e-02
## 88143502  9.255153e-01 7.448471e-02
## 881972    7.133126e-04 9.992867e-01
## 88199202  9.999749e-01 2.508937e-05
## 883263    3.945876e-03 9.960541e-01
## 88350402  9.960543e-01 3.945675e-03
## 884437    9.490318e-01 5.096817e-02
## 884448    9.996692e-01 3.308180e-04
## 884626    9.785912e-01 2.140882e-02
## 884689    9.997149e-01 2.851438e-04
## 88518501  9.995471e-01 4.528881e-04
## 8860702   1.158524e-02 9.884148e-01
## 888570    1.663806e-02 9.833619e-01
## 889719    1.130160e-03 9.988698e-01
## 8910506   9.995263e-01 4.736671e-04
## 8910720   9.999890e-01 1.096421e-05
## 8911800   9.995715e-01 4.284673e-04
## 8911834   9.986101e-01 1.389908e-03
## 8912521   9.999734e-01 2.661839e-05
## 8913      9.999835e-01 1.648992e-05
## 891716    9.999950e-01 5.008158e-06
## 891923    9.999563e-01 4.370481e-05
## 892189    9.991401e-01 8.599387e-04
## 892214    9.979281e-01 2.071904e-03
## 893988    9.999990e-01 9.919843e-07
## 894329    9.999917e-01 8.274583e-06
## 894604    9.997606e-01 2.394069e-04
## 89524     9.994984e-01 5.016483e-04
## 8953902   9.881245e-03 9.901188e-01
## 896864    9.973703e-01 2.629657e-03
## 897374    9.997030e-01 2.969798e-04
## 89742801  2.416886e-04 9.997583e-01
```

```
## 897880    9.999737e-01 2.629172e-05
## 898143    9.999709e-01 2.914775e-05
## 89827     9.997540e-01 2.460290e-04
## 898431    6.824358e-06 9.999932e-01
## 899187    9.997495e-01 2.504724e-04
## 9010259   9.902040e-01 9.795963e-03
## 9010333   9.999865e-01 1.349050e-05
## 901034301 9.998212e-01 1.787687e-04
## 9010598   9.997974e-01 2.025843e-04
## 9012000   1.068753e-05 9.999893e-01
## 9012568   9.995919e-01 4.081174e-04
## 901288    1.961808e-04 9.998038e-01
## 901303    9.962591e-01 3.740910e-03
## 9013594   9.950711e-01 4.928917e-03
## 901549    9.935638e-01 6.436233e-03
## 90251     9.980682e-01 1.931776e-03
## 90291     1.814239e-01 8.185761e-01
## 902975    9.998119e-01 1.881267e-04
## 90317302  9.999974e-01 2.584421e-06
## 90401601  9.876037e-01 1.239631e-02
## 90401602  9.999075e-01 9.253010e-05
## 904357    9.997364e-01 2.636139e-04
## 904647    9.997367e-01 2.633175e-04
## 905520    9.999304e-01 6.964227e-05
## 905978    9.968602e-01 3.139809e-03
## 906564    9.886223e-01 1.137770e-02
## 906616    9.998557e-01 1.442702e-04
## 908445    8.404531e-04 9.991595e-01
## 908916    9.989063e-01 1.093663e-03
## 909410    9.997914e-01 2.086145e-04
## 90944601  9.999624e-01 3.762122e-05
## 9110127   5.328172e-01 4.671828e-01
## 9110720   9.806887e-01 1.931129e-02
## 9110732   4.800372e-05 9.999520e-01
## 9111805   7.376928e-04 9.992623e-01
## 9112712   9.999631e-01 3.689365e-05
## 911320502 9.946997e-01 5.300342e-03
## 9113514   9.999084e-01 9.159384e-05
## 9113538   2.683380e-04 9.997317e-01
## 9113778   9.998314e-01 1.685669e-04
## 911391    9.999753e-01 2.471276e-05
## 911654    9.916282e-01 8.371850e-03
## 912519    9.974912e-01 2.508758e-03
## 913505    9.699904e-06 9.999903e-01
## 913535    8.550119e-01 1.449881e-01
## 914862    9.990388e-01 9.612205e-04
## 915276    9.998550e-01 1.450250e-04
## 915452    8.752674e-01 1.247326e-01
## 916221    9.996041e-01 3.959154e-04
## 916838    9.809319e-04 9.990191e-01
## 917062    9.997031e-01 2.969290e-04
## 91789     9.999961e-01 3.892011e-06
## 91903901  9.989556e-01 1.044404e-03
## 919812    9.885069e-01 1.149314e-02
```

```
## 921362     9.999976e-01 2.387447e-06
## 921385     9.999940e-01 5.998938e-06
## 923465     9.982527e-01 1.747327e-03
## 924632     9.832329e-01 1.676705e-02
## 925292     9.434940e-01 5.650604e-02
## 925622     1.043746e-06 9.999990e-01
```

Evaluation of the model

```
pred <- prediction(data_raw.lda.predict.posteriors[,2], test_raw.data$diagnosis)
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred, measure = "auc")
auc.train <- auc.train@y.values
plot(roc.perf)
abline(a=0, b= 1)
text(x = .25, y = .65 ,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```

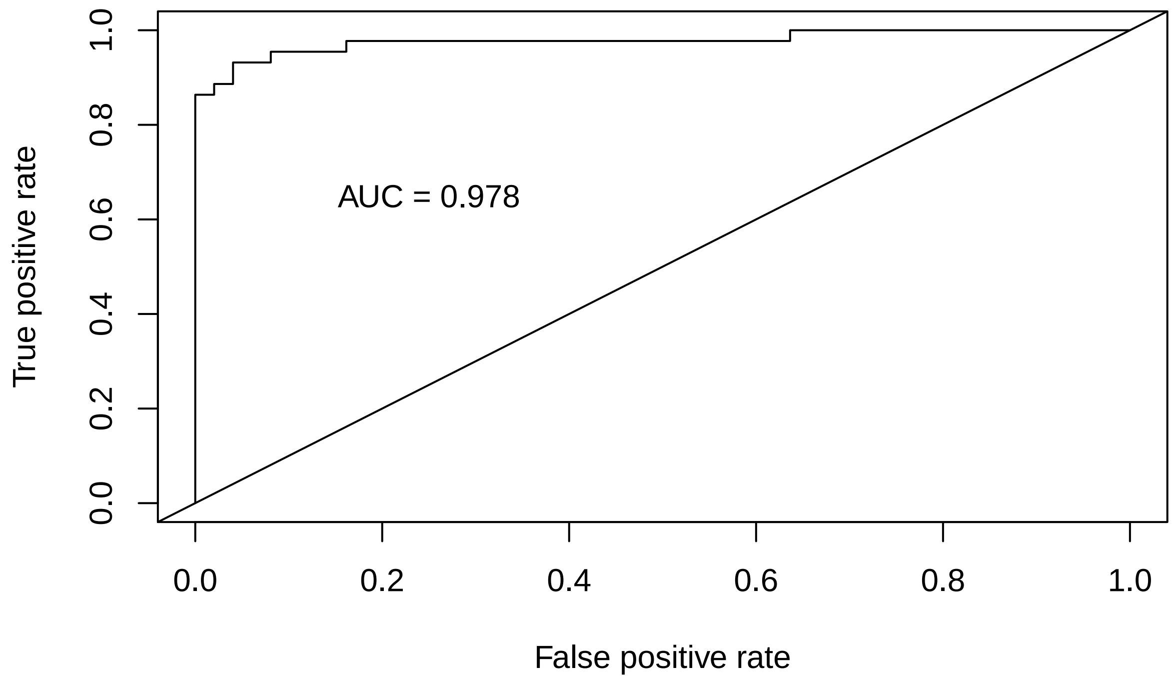



```
data.pcst <- data.pr$x[,1:6]
data.pcst <- cbind(data.pcst, as.numeric(as.factor(data$diagnosis))-1)
colnames(data.pcst)[7] <- "diagnosis"
colnames(data.pcst)
```

```
## [1] "PC1"       "PC2"       "PC3"       "PC4"       "PC5"       "PC6"
## [7] "diagnosis"
```

```
smp_size <- floor(0.75 * nrow(data.pcst))
train_ind <- sample(nrow(data.pcst), size = smp_size)
train_raw2.data <- as.data.frame(data.pcst[train_ind, ])
test_raw2.data <- as.data.frame(data.pcst[-train_ind, ])
data.lda <- lda(diagnosis ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6, data = train_raw2.data)
data.lda.predict <- predict(data.lda, newdata = test_raw2.data)
f2 <- paste(names(train_raw2.data)[7], "~", paste(names(train_raw2.data)[-7], collapse=" + "))
data_raw.lda2 <- lda(as.formula(paste(f2)), data = train_raw2.data)
data_raw.lda.predict2 <- predict(data_raw.lda2, newdata = test_raw2.data)
```

```r
data_raw.lda.predict.posteriors2 <- as.data.frame(data_raw.lda.predict2$posterior)
pred2 <- prediction(data_raw.lda.predict.posteriors2[,2], test_raw2.data$diagnosis)
roc.perf2 = performance(pred2, measure = "tpr", x.measure = "fpr")
auc.train2 <- performance(pred2, measure = "auc")
auc.train2 <- auc.train2@y.values
# Plot
plot(roc.perf2)
abline(a=0, b= 1)
text(x = .25, y = .65 ,paste("AUC = ", round(auc.train2[[1]],3), sep = ""))
```




```r
data.pcst <- data.pr$x[,1:6]
data.pcst <- cbind(data.pcst, as.numeric(as.factor(data$diagnosis))-1)
colnames(data.pcst)[7] <- "diagnosis"
smp_size <- floor(0.75 * nrow(data.pcst))
train_ind <- sample(nrow(data.pcst), size = smp_size)
train.data <- as.data.frame(data.pcst[train_ind, ])
test.data <- as.data.frame(data.pcst[-train_ind, ])
data.lda <- lda(diagnosis ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6, data = train.data)
data.lda.predict <- predict(data.lda, newdata = test.data)
```

Conclusion

We notice that the Principal Component model clearly does better than the model with original 30 variables.

PART 2

Use the built-in iris dataset in R to plot the LDA analysis. Use the lda function of the MASS package in R Project the LDA visual output and Compare the LDA and PCA 2D Projection of Iris dataset

```r
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

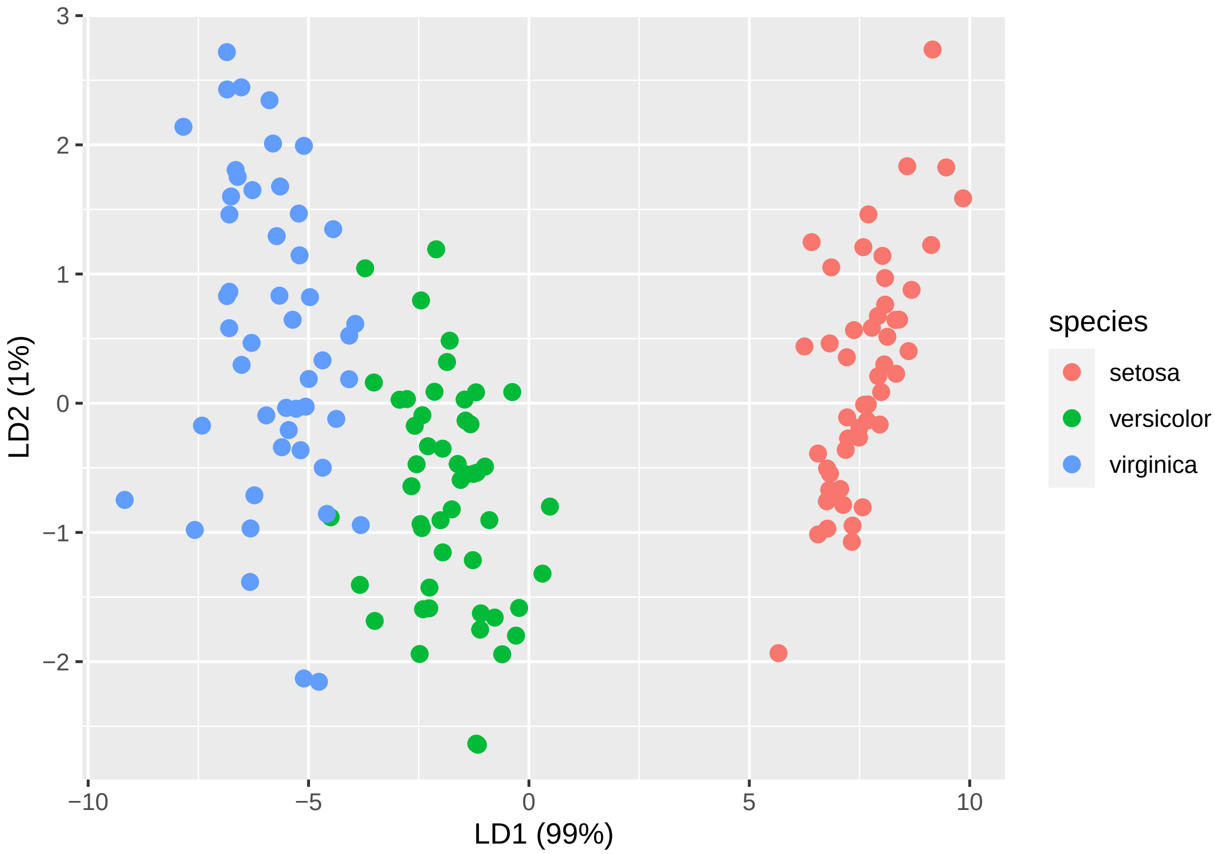
```r
require(MASS)
require(ggplot2)
require(scales)
```

```
## Loading required package: scales
## Warning: package 'scales' was built under R version 4.0.2
```

```r
 pca <- prcomp(iris[,-5],
             center = TRUE,
             scale. = TRUE)
prop.pca = pca$sdev^2/sum(pca$sdev^2)
lda <- lda(Species ~ .,
           iris,
           prior = c(1,1,1)/3)
r <- lda(formula = Species ~ .,
         data = iris,
         prior = c(1,1,1)/3)
prop.lda = r$svd^2/sum(r$svd^2)
plda <- predict(object = lda,
                newdata = iris)
dataset = data.frame(species = iris[,"Species"],
                     pca = pca$x, lda = plda$x)
```

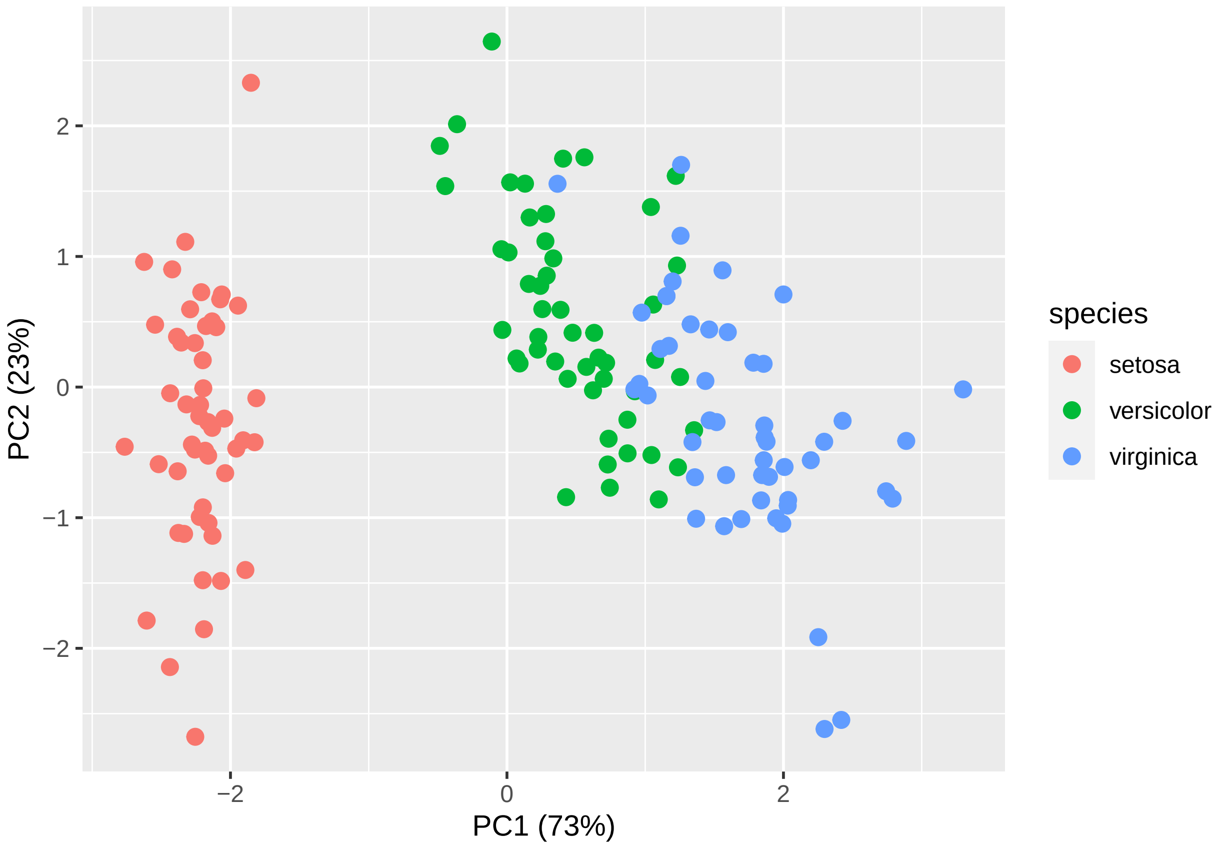
Plotting LDA:

```r
ggplot(dataset) + geom_point(aes(lda.LD1, lda.LD2, colour = species,), size = 2.5) + labs(x = paste("LD
```



Plotting PCA

```
ggplot(dataset) + geom_point(aes(pca.PC1, pca.PC2, colour = species), size = 2.5) + labs(x = paste("PC1
```



2
1
0
−1
−2
PC2 (23%)
−2
0
2
PC1 (73%)
species
setosa
versicolor
virginica