PCA and LDA Data Reduction

Yazan Abughazaleh YFA190000 Umaid Ahmed UXA180002
Dinesh Angadipeta DXA190032 Waheed Anwar WXA200000
Manny Asante EFA190000

Load in the data

In this part, we are loading in the mushroom data set from https://archive.ics.uci.edu/ml/datasets/Mushroom which is a classification data set. The target is whether the mushroom is edible or poisonous and there are 22 available predictors.

```
msm <- read.csv("mushrooms.csv",header = TRUE)
library(caret)
```

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

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

```
library(dplyr)
```

```
##
## 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
```

```
msm2 <- msm
for(i in 1:ncol(msm)){
  if(is.character(msm[,i])){
    msm[,i] <- as.factor(msm[,i])
    msm2[,i] <- as.factor(msm[,i])
    msm2[,i] <- as.integer(msm[,i])
  }
}
str(msm)
```

```
## 'data.frame':    8124 obs. of  23 variables:
##  $ class                   : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...
##  $ cap.shape               : Factor w/ 6 levels "b","c","f","k",..: 6 6 1 6 6 6 1 1 6 1 ...
```

```
##  $ cap.surface             : Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4 3 4 4 3 ...
##  $ cap.color               : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9 4 10 9 9 9 10 ...
##  $ bruises                 : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "a","c","f","l",..: 7 1 4 7 6 1 1 4 7 1 ...
##  $ gill.attachment         : Factor w/ 2 levels "a","f": 2 2 2 2 2 2 2 2 2 2 ...
##  $ gill.spacing            : Factor w/ 2 levels "c","w": 1 1 1 1 2 1 1 1 1 1 ...
##  $ gill.size               : Factor w/ 2 levels "b","n": 2 1 1 2 1 1 1 1 2 1 ...
##  $ gill.color              : Factor w/ 12 levels "b","e","g","h",..: 5 5 6 6 5 6 3 6 8 3 ...
##  $ stalk.shape             : Factor w/ 2 levels "e","t": 1 1 1 1 2 1 1 1 1 1 ...
##  $ stalk.root              : Factor w/ 5 levels "?","b","c","e",..: 4 3 3 4 4 3 3 3 4 3 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ veil.type               : Factor w/ 1 level "p": 1 1 1 1 1 1 1 1 1 1 ...
##  $ veil.color              : Factor w/ 4 levels "n","o","w","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ ring.number             : Factor w/ 3 levels "n","o","t": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ring.type               : Factor w/ 5 levels "e","f","l","n",..: 5 5 5 5 1 5 5 5 5 5 ...
##  $ spore.print.color       : Factor w/ 9 levels "b","h","k","n",..: 3 4 4 3 4 3 3 4 3 3 ...
##  $ population              : Factor w/ 6 levels "a","c","n","s",..: 4 3 3 4 1 3 3 4 5 4 ...
##  $ habitat                 : Factor w/ 7 levels "d","g","l","m",..: 6 2 4 6 2 2 4 4 2 4 ...
```

```
str(msm2)
```

```
## 'data.frame':    8124 obs. of  23 variables:
##  $ class                   : int  2 1 1 2 1 1 1 1 2 1 ...
##  $ cap.shape               : int  6 6 1 6 6 6 1 1 6 1 ...
##  $ cap.surface             : int  3 3 3 4 3 4 3 4 4 3 ...
##  $ cap.color               : int  5 10 9 9 4 10 9 9 9 10 ...
##  $ bruises                 : int  2 2 2 2 1 2 2 2 2 2 ...
##  $ odor                    : int  7 1 4 7 6 1 1 4 7 1 ...
##  $ gill.attachment         : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ gill.spacing            : int  1 1 1 1 2 1 1 1 1 1 ...
##  $ gill.size               : int  2 1 1 2 1 1 1 1 2 1 ...
##  $ gill.color              : int  5 5 6 6 5 6 3 6 8 3 ...
##  $ stalk.shape             : int  1 1 1 1 2 1 1 1 1 1 ...
##  $ stalk.root              : int  4 3 3 4 4 3 3 3 4 3 ...
##  $ stalk.surface.above.ring: int  3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: int  3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ veil.type               : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ veil.color              : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ ring.number             : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ ring.type               : int  5 5 5 5 1 5 5 5 5 5 ...
##  $ spore.print.color       : int  3 4 4 3 4 3 3 4 3 3 ...
##  $ population              : int  4 3 3 4 1 3 3 4 5 4 ...
##  $ habitat                 : int  6 2 4 6 2 2 4 4 2 4 ...
```

Now we would like to split our data into a train and test set, and we will use an 80/20 split.

```
set.seed(3)
i <- sample(1:nrow(msm), nrow(msm) * 0.80,replace=FALSE)
```

```
train <-msm[i,]
test <- msm[-i,]
train2 <-msm2[i,]
test2 <- msm2[-i,]
pca_train <- train2[,2:23]
pca_test <- test2[,2:23]
```

Principal Component Analysis (PCA)

Now we can perform PCA on our train data set and evaluate the results. We first want to check is we have any NA values in our data set and remove them.

```
sapply(msm, function(x) sum(is.na(x)==TRUE))
```

```
##                     class                 cap.shape               cap.surface
##                         0                         0                         0
##                cap.color                   bruises                      odor
##                         0                         0                         0
##          gill.attachment              gill.spacing                 gill.size
##                         0                         0                         0
##               gill.color               stalk.shape               stalk.root
##                         0                         0                         0
## stalk.surface.above.ring stalk.surface.below.ring   stalk.color.above.ring
##                         0                         0                         0
##   stalk.color.below.ring                 veil.type                veil.color
##                         0                         0                         0
##              ring.number                 ring.type        spore.print.color
##                         0                         0                         0
##               population                   habitat
##                         0                         0
```

Since there are no NA's in the data set, we do not have to do anything to our data set and can jump into performing PCA.

```
pca_out <- preProcess(train[,2:23],method=c("center","scale","pca"))
```

```
## Warning in pre_process_options(method, column_types): The following
## pre-processing methods were eliminated: 'center', 'scale', 'pca'
```

```
pca_out
```

```
## Created from 6499 samples and 22 variables
##
## Pre-processing:
##   - ignored (22)
```

In this attempt at PCA, we can see that all 22 predictors were ignored. This is because PCA can only work on contiguous variables. To work around this issue, categorical variables can be converted to integers, or a technique called Multiple Correspondence Analysis (MCA) will be used.

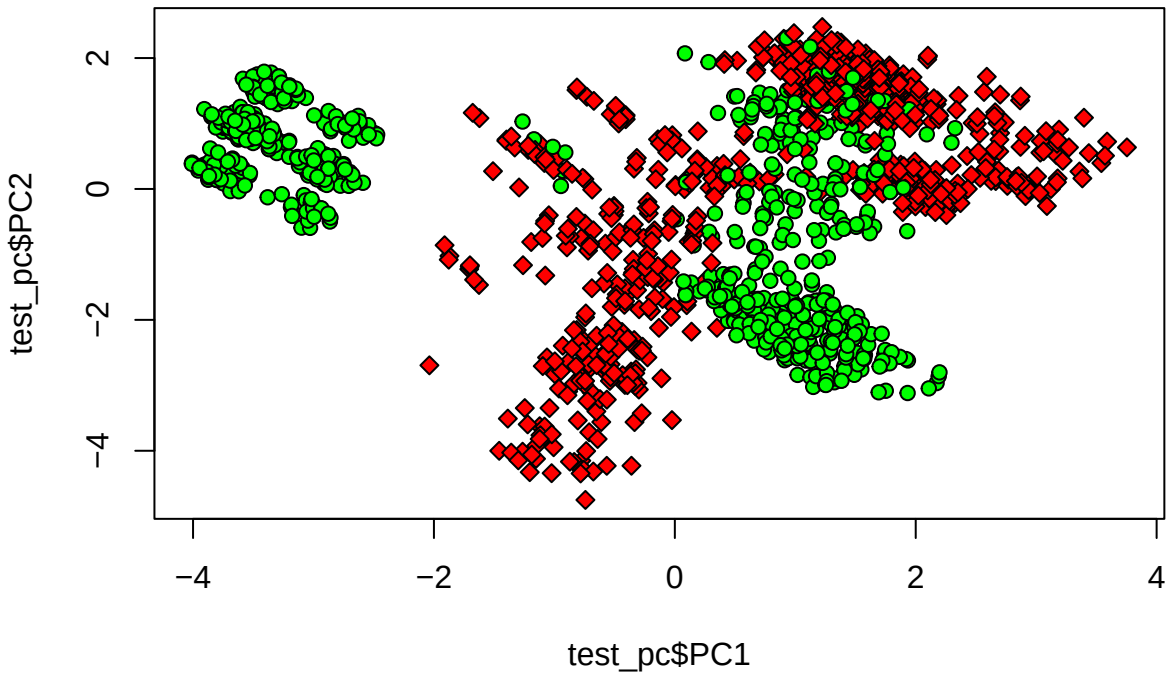
```
vals <- sapply(pca_train, function(v) var(v, na.rm=TRUE)!=0)
pca_out <- preProcess(pca_train[,vals],method=c("center","scale","pca"))
pca_out
```

```
## Created from 6499 samples and 21 variables
##
## Pre-processing:
##   - centered (21)
##   - ignored (0)
##   - principal component signal extraction (21)
##   - scaled (21)
##
## PCA needed 15 components to capture 95 percent of the variance
```

We can see here that converting to integer values has allowed a PCA reduction to 15 predictors.

PCA Plot

```
train_pc <- predict(pca_out, train2[, 2:23])
test_pc <- predict(pca_out, test2[,])
plot(test_pc$PC1, test_pc$PC2, pch=c(23,21,22)[unclass(test_pc$class)], bg=c("red","green")[unclass(tes
```



Comparing Models

```
train_cl <- train2[,c(1)]

train_pcN <- cbind(train_pc,train_cl)
train_pcN <- train_pcN %>% rename("class" = "train_cl")
```

Here we have reattached the class labels to the PCA data set to be able to perform classification. We would like to compare the results of the reduced data set to the original so we will make two classification models.

```
glm1 <- glm(class ~., data=train[, sapply(train, nlevels) > 1], family = 'binomial')
```

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

```
summary(glm1)
```

```
##
## Call:
## glm(formula = class ~ ., family = "binomial", data = train[,
##     sapply(train, nlevels) > 1])
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -2.409e-06  -2.409e-06  -2.409e-06   2.409e-06   2.409e-06
##
## Coefficients: (10 not defined because of singularities)
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -2.657e+01  3.543e+05       0        1
## cap.shapec                -9.003e-08  2.370e+05       0        1
## cap.shapef                 1.806e-10  2.620e+04       0        1
## cap.shapek                -5.571e-10  2.837e+04       0        1
## cap.shapes                 1.458e-10  8.974e+04       0        1
## cap.shapex                 1.225e-10  2.514e+04       0        1
## cap.surfaceg              -1.149e-07  2.730e+05       0        1
## cap.surfaces               4.212e-11  1.498e+04       0        1
## cap.surfacey               5.720e-11  1.258e+04       0        1
## cap.colorc                 5.257e-09  8.897e+04       0        1
## cap.colore                 5.915e-09  4.056e+04       0        1
## cap.colorg                 6.598e-09  3.882e+04       0        1
## cap.colorn                 6.256e-09  3.971e+04       0        1
## cap.colorp                -2.060e-09  5.041e+04       0        1
## cap.colorr                 7.364e-09  1.511e+05       0        1
## cap.coloru                 7.226e-09  1.538e+05       0        1
## cap.colorw                 7.307e-09  3.913e+04       0        1
## cap.colory                 7.056e-09  4.165e+04       0        1
## bruisest                   2.657e+01  1.267e+05       0        1
## odorc                      2.657e+01  1.546e+05       0        1
## odorf                     -2.657e+01  3.785e+05       0        1
## odorl                      3.923e-11  2.849e+04       0        1
## odorm                      1.328e+02  5.919e+05       0        1
## odorn                     -7.970e+01  3.638e+05       0        1
```

```
## odorp                     -5.313e+01  4.493e+05       0        1
## odors                     -2.657e+01  3.792e+05       0        1
## odory                     -2.657e+01  3.792e+05       0        1
## gill.attachmentf          -1.939e-07  1.524e+05       0        1
## gill.spacingw             -3.512e-08  5.692e+04       0        1
## gill.sizen                -7.970e+01  3.386e+05       0        1
## gill.colore               -2.657e+01  2.358e+05       0        1
## gill.colorg               -2.657e+01  2.328e+05       0        1
## gill.colorh               -2.657e+01  2.324e+05       0        1
## gill.colork               -2.657e+01  2.332e+05       0        1
## gill.colorn               -2.657e+01  2.325e+05       0        1
## gill.coloro               -2.657e+01  2.413e+05       0        1
## gill.colorp               -2.657e+01  2.321e+05       0        1
## gill.colorr               -2.657e+01  2.518e+05       0        1
## gill.coloru               -2.657e+01  2.332e+05       0        1
## gill.colorw               -2.657e+01  2.316e+05       0        1
## gill.colory               -2.657e+01  2.399e+05       0        1
## stalk.shapet              -5.313e+01  2.201e+05       0        1
## stalk.rootb               -2.657e+01  1.475e+05       0        1
## stalk.rootc               -1.594e+02  6.845e+05       0        1
## stalk.roote                2.657e+01  1.345e+05       0        1
## stalk.rootr               -1.860e+02  5.953e+05       0        1
## stalk.surface.above.ringk -2.764e-11  2.984e+04       0        1
## stalk.surface.above.rings -9.919e-12  2.408e+04       0        1
## stalk.surface.above.ringy  1.070e-07  3.113e+05       0        1
## stalk.surface.below.ringk -2.520e-10  2.984e+04       0        1
## stalk.surface.below.rings  4.570e-11  2.407e+04       0        1
## stalk.surface.below.ringy  2.657e+01  1.946e+05       0        1
## stalk.color.above.ringc           NA         NA      NA       NA
## stalk.color.above.ringe    2.800e-09  6.638e+04       0        1
## stalk.color.above.ringg    2.777e-09  3.471e+04       0        1
## stalk.color.above.ringn    8.503e-10  2.705e+04       0        1
## stalk.color.above.ringo   -5.313e+01  3.000e+05       0        1
## stalk.color.above.ringp    2.785e-09  2.706e+04       0        1
## stalk.color.above.ringw    2.758e-09  3.081e+04       0        1
## stalk.color.above.ringy    1.594e+02  5.736e+05       0        1
## stalk.color.below.ringc           NA         NA      NA       NA
## stalk.color.below.ringe   -2.014e-08  6.644e+04       0        1
## stalk.color.below.ringg   -2.016e-08  3.475e+04       0        1
## stalk.color.below.ringn   -2.024e-08  2.723e+04       0        1
## stalk.color.below.ringo           NA         NA      NA       NA
## stalk.color.below.ringp   -2.015e-08  2.723e+04       0        1
## stalk.color.below.ringw   -2.016e-08  3.100e+04       0        1
## stalk.color.below.ringy    4.491e-06  1.428e+05       0        1
## veil.coloro                2.231e-12  5.712e+04       0        1
## veil.colorw                       NA         NA      NA       NA
## veil.colory                       NA         NA      NA       NA
## ring.numbero               1.328e+02  5.363e+05       0        1
## ring.numbert                      NA         NA      NA       NA
## ring.typef                 5.313e+01  2.783e+05       0        1
## ring.typel                        NA         NA      NA       NA
## ring.typen                        NA         NA      NA       NA
## ring.typep                 2.657e+01  1.139e+05       0        1
## spore.print.colorh                NA         NA      NA       NA
```

```
## spore.print.colork        -5.660e-11  7.941e+04       0        1
## spore.print.colorn        -2.735e-11  7.833e+04       0        1
## spore.print.coloro        -4.157e-11  8.165e+04       0        1
## spore.print.colorr         1.328e+02  3.631e+05       0        1
## spore.print.coloru        -2.623e-11  1.129e+05       0        1
## spore.print.colorw         7.970e+01  3.470e+05       0        1
## spore.print.colory        -4.169e-11  7.922e+04       0        1
## populationc               -2.791e-09  6.912e+04       0        1
## populationn                6.324e-11  4.034e+04       0        1
## populations                4.624e-11  2.872e+04       0        1
## populationv               -2.786e-09  3.904e+04       0        1
## populationy               -2.478e-09  4.048e+04       0        1
## habitatg                  -5.490e-09  2.390e+04       0        1
## habitatl                   8.589e-10  2.200e+04       0        1
## habitatm                  -5.473e-09  4.093e+04       0        1
## habitatp                  -1.072e-09  1.744e+04       0        1
## habitatu                  -5.057e-09  4.177e+04       0        1
## habitatw                          NA         NA      NA       NA
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9.0032e+03  on 6498  degrees of freedom
## Residual deviance: 3.7704e-08  on 6413  degrees of freedom
## AIC: 172
##
## Number of Fisher Scoring iterations: 25
```

```
probs <- predict(glm1,newdata=test, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
pred <- ifelse(probs>0.5,1,0)
pred <- as.factor(pred)
levels(pred) <- list("e"="0","p"="1")
acc <- mean(as.integer(pred)==as.integer(test$class))
print(paste("glm1 accuracy: ", acc))
```

```
## [1] "glm1 accuracy:  1"
```

Next, we will look at running classification with the PCA reduced data set.

```
train_pcN$class <- as.factor(train_pcN$class)
test_pc$class <- as.factor(test_pc$class)
str(train_pc)
```

```
## 'data.frame':    6499 obs. of  16 variables:
##  $ veil.type: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ PC1      : num  1.52 1.96 1.93 1.67 -3.25 ...
##  $ PC2      : num  1.7178 1.527 0.0653 -2.3758 1.3457 ...
##  $ PC3      : num  -0.3931 0.1278 -1.3641 1.6941 -0.0482 ...
##  $ PC4      : num  -1.003 -1.163 1.12 -0.932 -0.27 ...
```

```
##  $ PC5      : num  0.7083 1.4191 -1.8896 0.0815 -0.0437 ...
##  $ PC6      : num  -0.0838 1.1595 -1.0374 0.2611 -0.5384 ...
##  $ PC7      : num  0.386 -0.672 -1.795 1.288 1.168 ...
##  $ PC8      : num  -0.773 0.617 0.566 1.242 0.197 ...
##  $ PC9      : num  1.2041 -0.9412 -0.5524 -0.0291 -0.1547 ...
##  $ PC10     : num  -0.28944 0.26592 0.39452 -0.00317 0.61695 ...
##  $ PC11     : num  0.295 0.109 0.346 0.39 0.335 ...
##  $ PC12     : num  0.135 0.786 -0.435 0.518 -0.105 ...
##  $ PC13     : num  0.0912 -1.3112 -0.1695 0.3042 0.1167 ...
##  $ PC14     : num  -0.215 -0.225 -0.774 -0.583 -0.471 ...
##  $ PC15     : num  -0.19192 -0.28651 -0.00327 -0.29488 0.20336 ...
```

```
glm2 <- glm(class ~., data=train_pcN, family = 'binomial')
summary(glm2)
```

```
##
## Call:
## glm(formula = class ~ ., family = "binomial", data = train_pcN)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2469  -0.3658  -0.0300   0.2222   2.7410
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.14415    0.06021   2.394   0.0167 *
## veil.type         NA         NA      NA       NA
## PC1         -1.07480    0.04162 -25.827  < 2e-16 ***
## PC2         -0.18058    0.03653  -4.943 7.68e-07 ***
## PC3          1.39309    0.04758  29.277  < 2e-16 ***
## PC4         -0.66262    0.04399 -15.065  < 2e-16 ***
## PC5         -1.22214    0.04941 -24.734  < 2e-16 ***
## PC6         -0.73770    0.04380 -16.842  < 2e-16 ***
## PC7          0.06790    0.04459   1.523   0.1278
## PC8          0.37706    0.04720   7.989 1.36e-15 ***
## PC9         -0.13417    0.05600  -2.396   0.0166 *
## PC10        -0.73578    0.05767 -12.759  < 2e-16 ***
## PC11        -0.60256    0.07168  -8.407  < 2e-16 ***
## PC12         0.55842    0.06960   8.023 1.03e-15 ***
## PC13         0.32296    0.07238   4.462 8.12e-06 ***
## PC14         0.28419    0.06971   4.077 4.57e-05 ***
## PC15        -0.62929    0.07144  -8.808  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9003.2  on 6498  degrees of freedom
## Residual deviance: 2946.3  on 6483  degrees of freedom
## AIC: 2978.3
##
## Number of Fisher Scoring iterations: 7
```

```
probs_PCA <- predict(glm2,newdata=test_pc, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
pred_PCA <- ifelse(probs_PCA>0.5,1,0)
pred_PCA <- as.factor(pred_PCA)
acc2 <- mean(as.integer(pred_PCA)==test_pc$class)
print(paste("glm2 accuracy: ", acc2))
```

```
## [1] "glm2 accuracy:  0.913230769230769"
```

Here we see a reduction in accuracy of approximately 9%. The original accuracy shown is at 100%, indicating that the initial logistic regression model was able to perfectly predict the class a mushroom was in with all the given predictors. With the reduced data set, there is a 91% accuracy in predictions which is excellent, but the predictors were reduced.

Linear Discriminant Analysis

With PCA, the focus is on reducing the number of predictors without regards to the target and class. PCA is a form of unsupervised learning that aims to simplify the data set, however other approaches may be more effective to use. LDA is another data reduction technique that does consider the class of the data when performing a reduction. To start with this, we need to load in the MASS library.

```
library(MASS)
```

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

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

```
lda1 <- lda(class~.,data=train[, sapply(train, nlevels) > 1])
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
writeLines("\n")
```

```
lda1$means
```

```
##    cap.shapec cap.shapef cap.shapek  cap.shapes cap.shapex cap.surfaceg
## e 0.000000000  0.3742167  0.0552074 0.007758878  0.4685169  0.000000000
## p 0.000952986  0.4008895  0.1588310 0.000000000  0.4269377  0.000952986
##   cap.surfaces cap.surfacey  cap.colorc cap.colore cap.colorg cap.colorn
## e    0.2736497    0.3518353 0.007460460  0.1527902  0.2464936  0.2969263
## p    0.3649936    0.4431385 0.002858958  0.2255400  0.2055273  0.2627065
##   cap.colorp  cap.colorr  cap.coloru cap.colorw cap.colory  bruisest      odorc
## e 0.01223515 0.003879439 0.003581021 0.17308266 0.09280812 0.6523426 0.00000000
```

```
## p 0.02223634 0.000000000 0.000000000 0.07909784 0.16899619 0.1581957 0.04987294
##       odorf      odorl       odorm      odorn     odorp     odors     odory
## e 0.0000000 0.09310654 0.000000000 0.81319009 0.0000000 0.0000000 0.0000000
## p 0.5492376 0.00000000 0.007306226 0.03113088 0.0660737 0.1486658 0.1477128
##   gill.attachmentf gill.spacingw gill.sizen gill.colore gill.colorg gill.colorh
## e        0.9531483    0.28648165 0.06714414  0.02118771  0.05610266  0.05073113
## p        0.9955527    0.02890724 0.57465057  0.00000000  0.13119441  0.13437103
##   gill.colork gill.colorn gill.coloro gill.colorp gill.colorr gill.coloru
## e  0.08176664  0.22918532  0.01521934   0.2023277 0.000000000  0.10593853
## p  0.01620076  0.02890724  0.00000000   0.1585133 0.006670902  0.01143583
##   gill.colorw gill.colory stalk.shapet stalk.rootb stalk.rootc stalk.roote
## e  0.22262011 0.014920919    0.6195166   0.4592659  0.12026261   0.2065055
## p  0.06162643 0.005082592    0.5174714   0.4701398  0.00952986   0.0660737
##   stalk.rootr stalk.surface.above.ringk stalk.surface.above.rings
## e  0.04356908                0.03581021                 0.8627275
## p  0.00000000                0.56925032                 0.3910419
##   stalk.surface.above.ringy stalk.surface.below.ringk stalk.surface.below.rings
## e               0.003879439                0.03491495                 0.8081170
## p               0.002223634                0.55304956                 0.3945362
##   stalk.surface.below.ringy stalk.color.above.ringc stalk.color.above.ringe
## e                0.04744852             0.000000000              0.02238138
## p                0.01778907             0.007306226              0.00000000
##   stalk.color.above.ringg stalk.color.above.ringn stalk.color.above.ringo
## e               0.1330946              0.003879439              0.04685169
## p               0.0000000              0.107687421              0.00000000
##   stalk.color.above.ringp stalk.color.above.ringw stalk.color.above.ringy
## e               0.1417487                0.6520442             0.000000000
## p               0.3329098                0.4367853             0.002223634
##   stalk.color.below.ringc stalk.color.below.ringe stalk.color.below.ringg
## e             0.000000000               0.02178454                0.1402566
## p             0.007306226               0.00000000                0.0000000
##   stalk.color.below.ringn stalk.color.below.ringo stalk.color.below.ringp
## e               0.0146225               0.04685169                0.1333930
## p               0.1140407               0.00000000                0.3367217
##   stalk.color.below.ringw stalk.color.below.ringy veil.coloro veil.colorw
## e               0.6430916             0.000000000  0.02357505   0.9531483
## p               0.4275731             0.006670902  0.00000000   0.9977764
##   veil.colory ring.numbero ring.numbert ring.typef ring.typel  ring.typen
## e 0.000000000    0.8743659    0.1256341 0.01074306  0.0000000 0.000000000
## p 0.002223634    0.9742694    0.0184244 0.00000000  0.3281449 0.007306226
##   ring.typep spore.print.colorh spore.print.colork spore.print.colorn
## e  0.7508207         0.01074306         0.39510594         0.41211579
## p  0.2080686         0.39961881         0.05876747         0.05717916
##   spore.print.coloro spore.print.colorr spore.print.coloru spore.print.colorw
## e         0.01044464          0.0000000         0.01104148          0.1363772
## p         0.00000000          0.0184244         0.00000000          0.4660102
##   spore.print.colory populationc populationn populations populationv
## e         0.01163832  0.06744255  0.09609072  0.20531185   0.2829006
## p         0.00000000  0.01175349  0.00000000  0.09498094   0.7309403
##   populationy  habitatg   habitatl    habitatm   habitatp   habitatu  habitatw
## e   0.2551477 0.3333333 0.05759475 0.061474187 0.03282602 0.02267980 0.0438675
## p   0.1623253 0.1861499 0.15279543 0.009847522 0.25571792 0.06956798 0.0000000
```

After performing LDA, we now would like to test the results.

```
lda_pred <- predict(lda1, newdata=test, type="class")
lda_pred$class
```

```
##    [1] e p e e e e e e e e e e e e e e e e e e e e p p e e e e e e e e e e e e e
##   [38] e e e e e p e e p e e p e e e e e e p e e e e e e p e e e e p e e e e e e
##   [75] e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e
##  [112] e e p e e e e e p e e e e e e e e e e p e e e e e e e e e e e e e e e e p
##  [149] e p e e e e e e e p e e e p e e p e p e p e e e e e e e e e e e e e e e e
##  [186] e e e p p e e e p e e e e e e e e e e e e e e e e e e e e e e e e e e e p
##  [223] p e e e p e e e e p e e e e e p e e e e e e e e e e e e e e e p e e e e e
##  [260] e e p e e e e e e e e e e e e e e e p e e p e e e e e e e e p e p e e e e
##  [297] e e p e p e e e e e e e e p e e e e e e e p e e p e e e e e e e e e e p e
##  [334] p e e e e e e e e e e e e e e e e p e e e e e e e e e e e e p e e e e e e
##  [371] e p e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e p e e
##  [408] p e e e e e e e e e e e e e e e e e e e e e e e e e e p e e e e e e e e e
##  [445] e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e p e e e e e
##  [482] e e e e e e e e p e e e e p e e e e e p e e e e e p e e e e e e p e e e e
##  [519] e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e e
##  [556] e e e e e e e e e e e e e e e p e e e e e e e e e e e e e e e e e e e e e
##  [593] e e e e e e e e p e e e e e e p e e e e p e p e p e p e e p e e p e p e e
##  [630] e e e e p e e p p e p p p e e p p e e e p p p p p e e e e e e e e e e p e
##  [667] p e e p p p e p e e e e e e e p e p e p e e p e p e e e e p e e e e e e e
##  [704] p e p e e e p p e e e p e p p p p e p p e p p p p e e e e e p e e e e e e
##  [741] p e e p e e p e e e e e p p e e p e p e e p p e p e e e e e p p p e e e p
##  [778] e e e p e p p e p e p e p p p e p p e e p p p p p e p p p e p p p p p e e
##  [815] e p e e p p p e p p p e p p p e e p e p p p p p p p p e p p p p p p p p p
##  [852] e p p p p p p p p e p p p p p p p p p p p p p p p p p p p p p p p p p p p
##  [889] p p p p p p p p e p p p p e p p p p e p p e e p p p p p p p p p e p p p p
##  [926] p p p p p p p p p e e p p p p p p p p p p p p p p p e p p p p p p p p p p
##  [963] p e e p p p p p p p p p p e p p e p e p p p p p p p p p p p p p p p p p p
## [1000] p p p e e p p p p p p p e p e p p p p p p p p p p p p p e p p p e p e p p
## [1037] p p e p e p p p p p e p e p p e p p p p e p e p e p p p p p p p p p p p p
## [1074] p p p p p p p e p p p e p p e p e e p p p e e p p p p p p p p p p p p p p
## [1111] p p p p e e p e p p p e e p p e p p p e e e e p e p p e e p p p p e e p p
## [1148] e p p e p e p p p e p p p e p p e p e p p e p p p e p p e p e p p p p e e
## [1185] e p e p p e p p p p e p p p p p e e e p p p p e p p p p p e p p p p p e e
## [1222] p p p p p p p p p p p p p p p p p p p p e p p p p p p p p p p p p p p p p
## [1259] p p p p e p p p p p p p p p p p e p p p e p p p p p e p p p p p p p p p p
## [1296] p p p p p e p p e p p p p p e p p p p p p e p p p p p p p p p p p p p p p
## [1333] p p p p p p p p p p p p p p e p p p p p p p p p p p p p p p p p p p p p p
## [1370] p p p p p p p p p p p p p p e e p p e p p p p p p p p p p e p p p p p p p
## [1407] p p p p p p p e p p p p p p p p e p p p p e p p p e p e p p e p p p p e p
## [1444] p p p e p p e p p p p p e p p p p e p e p p e p e p e p p e p e e p e p p
## [1481] p p e p p p e e p e p e p e p e e e p p e p p e e p e p p p p p e p e p p
## [1518] e e p e p e p e p e p p e p p e e p p p p p p p e p p e e p p p e p e p p
## [1555] e p p e p p e p e p e e e e p p p p p p e p e p p p p p p p e p e p e e p e
## [1592] e p e e e e e e p p e p p e p p e p p e p e p e p e p e e e e e p e
## Levels: e p
```

```
mean(lda_pred$class==test$class)
```

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

We can see the results produced by LDA predictions are identical to the results of the original predictions in terms of accuracy.