Project 4

Introduction

This is Project 4 for STAT 557 2018 Spring by Meridith Bartley and Fei Jiang. The aim of this project is to practice EM algorithm, MDA algrithm and compare MDA with QDA.

Description of Data - EM Algorithm

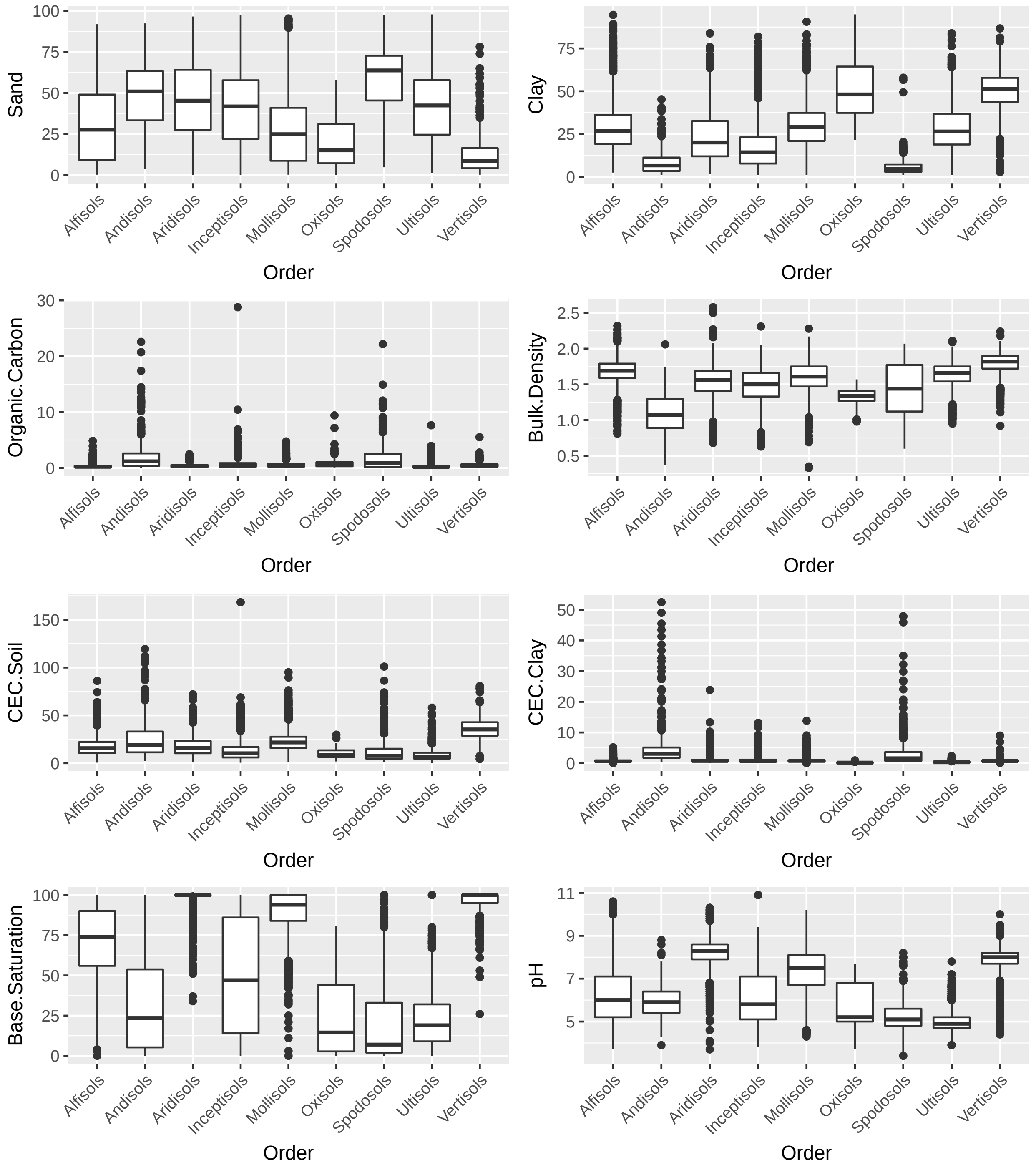
This dataset contains Wisconsin breast cancer data over the US downloaded from UCI Machine Learning Data Repository. With no missing data there are around 570 records, each of which includes ten real-valued features are computed for each cell nucleus (thickness,size.unif, shape.unif, adhesion, size.cell, nuclei, chromatin, nucleoli, mitoses) and the corresponding diagnosis (malignant or benign)

Description of Data - MDA

This dataset contains soil sample data over the US downloaded from Natural Resources Conservation Service (NRCS). After removing the incomplete data records and excluding the data records with impossible values, there are around 14,000 records left, each of which includes physical and chemical properties of soil samples (sand, silt, clay, organic carbon, bulk density, CEC soil, CEC clay, base saturation, and pH) and the corresponding soil classification group (soil order).

Boxplots for each physical and chemical property used as explanitory variables in the subsequent classification models are included below. This EDA allows for early indication of which variables may possibly be ommitted during dimention reduction. That is, what properties do not differ significantly between soil Orders.

Exploritory Data Analysis

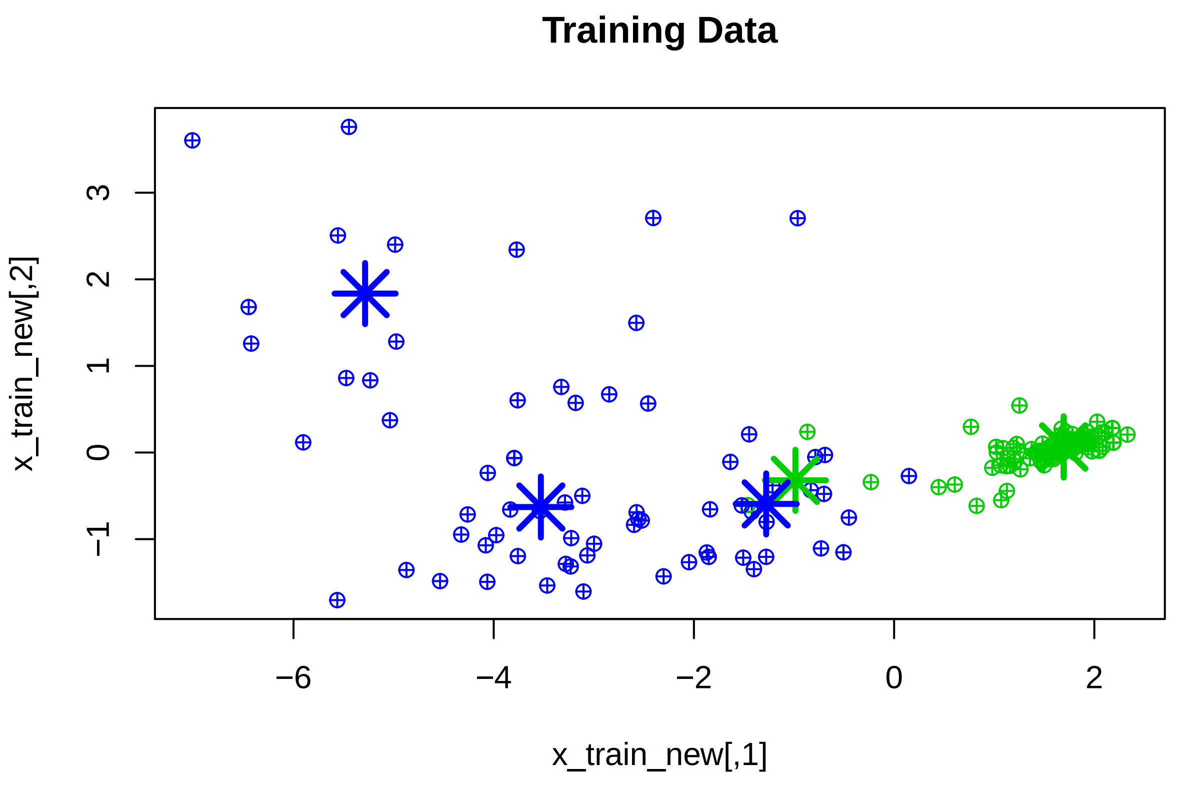


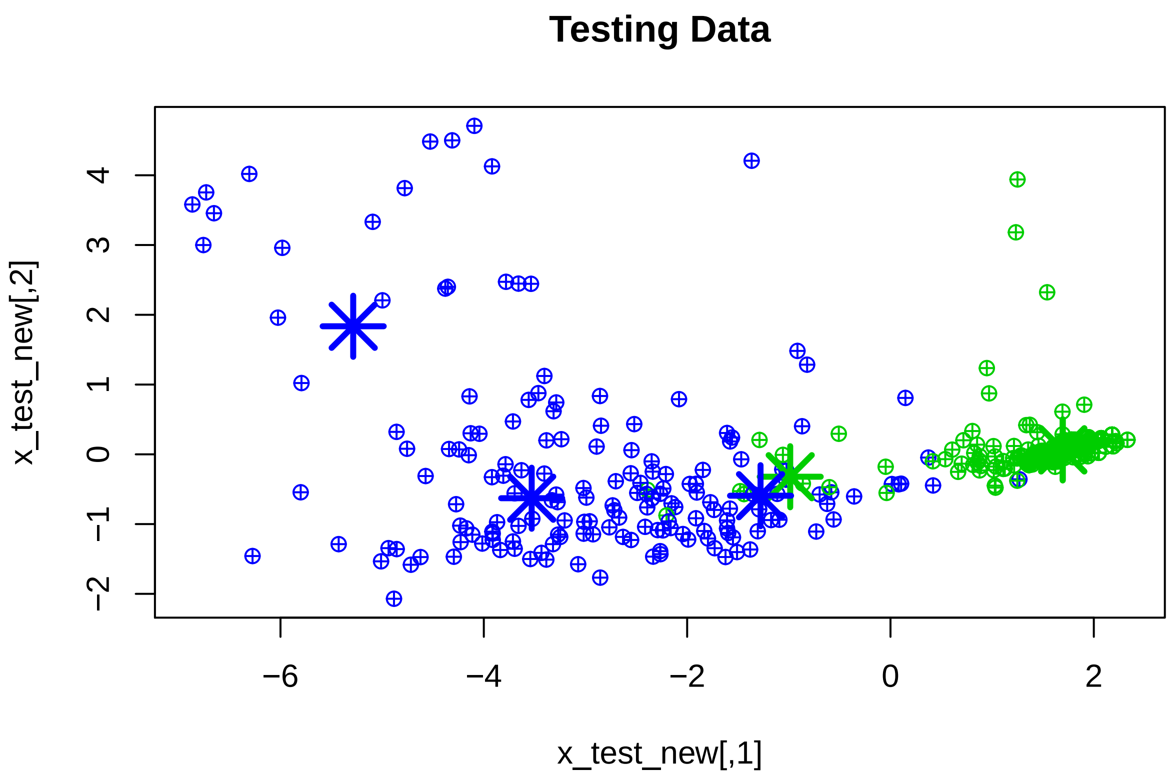
Analysis

EM Algorithm for Mixture of Gaussians

In this application of an EM algorithm for a mixutre of Gaussians we randomly divided our dataset to training data (70%) and test data (30%). We applied the EM algorithm to training data to get the model and then evaluated its accuracy in test data. Resulting plots and precition confusion table are included below.

```
## Warning in plot.window(...): "alpha" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "alpha" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "alpha" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "alpha" is not
## a graphical parameter

## Warning in box(...): "alpha" is not a graphical parameter

## Warning in title(...): "alpha" is not a graphical parameter
```





	benign	malignant
benign	298	6
malignant	11	164
## Comapriso	n between	MDA and Quadratic Discriminant Analysis (QDA)

In this part of the analysis, we randomly divided our dataset to traning data (80%) and test data (20%). We applied MDA in training data to get the model and then evaluated its accuracy in test data. In order to test the effects of the number of components in model performance and elaborate the dilemma between model complexity and prediction accuracy, we conducted MDA with the number of components included ranging from 2 to 8. The plot below shows the increasing prediction accuracy with the increasing number of components included. Along with the number of components increasing from 2 to 8, the predicting accuracy increased from 50% to 58%. From a perspective of balancing model complexity and prediction accuracy, we chose our final MDA model with four components included and used that model to conduct following comparison with QDA model.

In this analysis, we conducted MDA and QDA analysis based on the same training and test dataset and the same first four components. The prediction accuracy for each soil types and overall accuracy of these two methods are shown in the table below. For the table we can see that the overall accuracy of MDA and QDA are very similar (55% for MDA and 54% for QDA). However, their ability in correctly predicting individual soil groups differed. For instance, MDA is better at Inceptisols and Mollisols while QDA is better at Aridisols and Vertisols.

	Alfisols	Andisols	Aridisols	Inceptisols	Mollisols	Oxisols	Spodosols	Ultisols	Vertisols	Overall
MDA	0.53	0.34	0.42	0.24	0.72	0.47	0.41	0.69	0.55	0.55

	Alfisols	Andisols	Aridisols	Inceptisols	Mollisols	Oxisols	Spodosols	Ultisols	Vertisols	Overall
QDA	0.51	0.31	0.65	0.05	0.58	0.35	0.56	0.76	0.68	0.54

Conclusion

In this project, we finished up two part.

In the first part, we coded a EM algorithm for mixed Gaussians and applied it to a breast cancer dataset. Code for the EM functions are included in the Appendix below.

In the second part, we conducted MDA on our dataset and explored the increasing prediction accuracy with the increasing model complexity. Through comparison with QDA on the same dataset using same predictors, we found out the comparably acceptable performance of these two methods.

Contributions

The different tasks required to complete this project were equally divided between Meridith and Fei. Both members of this group contributed to the presentation slides and this report.

Appendix

```
library(ggplot2)
library(DAAG)
library(dplyr)
library(magrittr)
library(MASS)
library(caret)
library(nnet)
library(scales)
library(klaR)
library(stats)
library(grid)
library(gridExtra)
library(mclust)
library(mvtnorm)
library(mda)


### split data into test and training sets
# df:         data set
# train_amt:  proportion of data to use for training
# resp_idx:   which column the class variable is (for breast cancer data this is 10)
split_data <- function(df, train_amt, resp_idx) {
  N <- floor(nrow(df) * train_amt)
  ind <- sample.int(nrow(df), size = N)
  x_train <- as.matrix(df[ind,-resp_idx])
  x_test <- as.matrix(df[-ind,-resp_idx])
  class_train <- df[ind,resp_idx]
  class_test <- df[-ind,resp_idx]
```

```r
  return(list(x_train = x_train,
              x_test = x_test,
              class_train = class_train,
              class_test = class_test))
}

### EM algorithm
# x:      covariates (as a matrix, one row = one observation)
# y:      response (as a factor)
# R:      number of subclasses for each class
# tol:    condition for convergence
# maxit:  maximum number of iterations for EM to run
EM <- function(x, y, R, tol = 1e-6, maxit = 10) {
  labs <- levels(y)
  K <- length(labs)
  M <- ncol(x)
  N <- nrow(x)
  pi <- array(0, dim = c(K, R))
  mu <- array(0, dim = c(K, M, R))
  sigma <- diag(M)
  y <- as.numeric(y)
  ### come up with starting points using kmeans
  for(k in 1:K) {
    ind <- which(y == k)
    mu[k, , ] <- t(kmeans(x[ind, ], R)$centers)
    pi[k, ] <- rep(1/R, length = R)
  }
  converged <- F
  ### main EM loop
  # while(!converged) {
  for(it in 1:maxit) {
    # E step
    p <- array(0, dim = c(N, R))
    for(i in 1:N) {
      k <- y[i]
      for(r in 1:R) {
        p[i,r] <- pi[k,r] * dmvnorm(x[i, ], mu[k, ,r], sigma)
      }
      p[i, ] <- p[i, ]/sum(p[i, ])
    }
    # M step
    for(k in 1:K) {
      ind <- which(y == k)
      for(r in 1:R) {
        pi[k,r] <- sum(p[ind,r])/length(ind)
        mu[k, ,r] <- colSums(x[ind, ]*p[ind,r])/sum(p[ind,r])
      }
    }
    tally <- 0
    for(i in 1:N) {
      z <- y[i]
      for(r in 1:R) {
        tmp <- x[i, ] - mu[z, ,r]
        tally <- tally + p[i,r] * (tmp %o% tmp)
      }
    }
    sigma <- tally/N
    # check convergence

  }
  return(list(pi = pi,
              mu = mu,
              sigma = sigma,
              class_labels = labs))
}

### predict classes
# em: object from the result of EM function above
# x: new data (as a matrix, one row = one observation)
em_predict <- function(em, x) {
  pi <- em$pi
  mu <- em$mu
  sigma <- em$sigma
  labs <- em$class_labels
  K <- dim(mu)[1]
  R <- dim(mu)[3]
  classes <- rep(NA, nrow(x))

  for(i in 1:nrow(x)) {
    probs <- rep(0, K)
    for(k in 1:K) {
      for(r in 1:R) {
        probs[k] <- probs[k] + pi[k,r] * dmvnorm(x[i, ], mean = mu[k, ,r], sigma = sigma)
      }
    }
    classes[i] <- which.max(probs)
  }
  return(factor(classes, labels = labs))
}
#EM data
# read data in and format it
data <- read.csv('http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breas
names(data) <- c('id','thickness','size.unif','shape.unif','adhesion','size.cell','nuclei','chromatin',
data$class <- as.factor(data$class)
levels(data$class) <- c('benign','malignant')
data <- data[-which(is.na(data$nuclei)),-1] # leave ID column out and remove NAs


# Basic clearning of our dataset
project1data <- read.csv(file = "project1data.csv")
project1data <- project1data[,-c(1,2,4)] #remove the ID and Horizon columns which are useless,Silt is r
project1data <- project1data[sample(nrow(project1data),nrow(project1data)),] ###shuffle rows



varlist <- names(project1data)[-9]
```

```r
customPlot <- function(varName) {

project1data %>%
group_by_("Order") %>%
select_("Order",varName) %>%
ggplot(aes_string("Order",varName)) + geom_boxplot() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))

}

grid.arrange(grobs = lapply(varlist[1:4],customPlot))
grid.arrange(grobs = lapply(varlist[5:8],customPlot))


# pairs(project1data)
# cor(project1data[, -9])



# Create new dataset with PCA
pca = prcomp(project1data[,-9])
# print(pca$rotation)
summary.pca = summary(pca)
# print(summary.pca)
project1data.new = data.frame(pca=pca$x[,1:8],Order=project1data$Order)

#table of the variance and coefficients
#knitr::kable(summary.pca$importance)
#knitr::kable(pca$rotation[, 1:4])


#data partition
train_id <- caret::createDataPartition(y=project1data.new$Order, p=0.8,list = FALSE)
train <- project1data.new[train_id,]
test <- project1data.new[-train_id,]
# Mixture Discriminant Analysis (MDA)






R <- 3 # number of subclasses
training_amount <- 0.3

X <- split_data(data, training_amount, 10)
x_train <- X$x_train
x_test <- X$x_test
class_train <- X$class_train
class_test <- X$class_test

# run MDA
fit <- EM(x_train, class_train, R)
```

```r
# classify test data
fit_pred <- em_predict(fit, x_test)
# resulting confusion matrix and error rate
# confusion(fit_pred, class_test)

# try PCA
x <- rbind(x_train, x_test)
x_scaled <- apply(x, 2, function(x) (x - mean(x))/sd(x))
eig <- eigen(cov(x_scaled))
# eig$values/sum(eig$values)
V <- eig$vectors[ ,1:2]
x_new <- as.matrix(x_scaled %*% V)
x_train_new <- x_new[1:nrow(x_train), ]
x_test_new <- x_new[-(1:nrow(x_train)), ]

fit2 <- EM(x_train_new, class_train, R)
fit_pred2 <- em_predict(fit2, x_test_new)
# confusion(fit_pred2, class_test)

#training plot
plot(x_train_new, pch = 10, alpha = 0.5, col = as.numeric(class_train)+2,
      main = "Training Data")
points(t(fit2$mu[1, , ]), pch = 8, cex = 3, lwd = 3, col = 3)
points(t(fit2$mu[2, , ]), pch = 8, cex = 3, lwd = 3, col = 4)
#testing plot
plot(x_test_new, pch = 10, col = as.numeric(class_test)+2,
     main = "Testing Data")
points(t(fit2$mu[1, , ]), pch = 8, cex = 3, lwd = 3, col = 3)
points(t(fit2$mu[2, , ]), pch = 8, cex = 3, lwd = 3, col = 4)



knitr::kable(table(fit_pred2, class_test))


#mda - 2 component
mda.2 <- mda(Order ~ pca.PC1 + pca.PC2 , data = train)
#predict test data
mda2.predict = predict(mda.2, newdata=test)
# Assess the total accuracy of test data
ct2.mda <- table(test$Order, mda2.predict)
totcorrect2.mda <- sum(diag(prop.table(ct2.mda)))
# Assess the accuracy of the prediction for each soil class (Order) in test data
#cc2.mda <- diag(prop.table(ct2.mda, 1))

#mda - 3 component
mda.3 <- mda(Order ~ pca.PC1 + pca.PC2+ pca.PC3 , data = train)
#predict test data
mda3.predict = predict(mda.3, newdata=test)
# Assess the total accuracy of test data
ct3.mda <- table(test$Order, mda3.predict)
totcorrect3.mda <- sum(diag(prop.table(ct3.mda)))
```

```
#mda - 4 component
mda.4 <- mda(Order ~ pca.PC1 + pca.PC2+ pca.PC3+ pca.PC4 , data = train)
#predict test data
mda4.predict = predict(mda.4, newdata=test)
# Assess the total accuracy of test data
ct4.mda <- table(test$Order, mda4.predict)
totcorrect4.mda <- sum(diag(prop.table(ct4.mda)))

#mda - 5 component
mda.5 <- mda(Order ~ pca.PC1 + pca.PC2+ pca.PC3+ pca.PC4 + pca.PC5, data = train)
#predict test data
mda5.predict = predict(mda.5, newdata=test)
# Assess the total accuracy of test data
ct5.mda <- table(test$Order, mda5.predict)
totcorrect5.mda <- sum(diag(prop.table(ct5.mda)))

#mda - 6 component
mda.6 <- mda(Order ~ pca.PC1 + pca.PC2+ pca.PC3+ pca.PC4 + pca.PC5+ pca.PC6, data = train)
#predict test data
mda6.predict = predict(mda.6, newdata=test)
# Assess the total accuracy of test data
ct6.mda <- table(test$Order, mda6.predict)
totcorrect6.mda <- sum(diag(prop.table(ct6.mda)))

#mda - 7 component
mda.7 <- mda(Order ~ pca.PC1 + pca.PC2+ pca.PC3+ pca.PC4 + pca.PC5+ pca.PC6+ pca.PC7, data = train)
#predict test data
mda7.predict = predict(mda.7, newdata=test)
# Assess the total accuracy of test data
ct7.mda <- table(test$Order, mda7.predict)
totcorrect7.mda <- sum(diag(prop.table(ct7.mda)))

#mda - 8 component
mda.8 <- mda(Order ~ pca.PC1 + pca.PC2+ pca.PC3+ pca.PC4 + pca.PC5+ pca.PC6+ pca.PC7+ pca.PC8, data = t
#predict test data
mda8.predict = predict(mda.8, newdata=test)
# Assess the total accuracy of test data
ct8.mda <- table(test$Order, mda8.predict)
totcorrect8.mda <- sum(diag(prop.table(ct8.mda)))


#mda - 4 component
mda <- mda(Order ~ pca.PC1 + pca.PC2+ pca.PC3+ pca.PC4 , data = train)
#predict test data
mda.predict = predict(mda, newdata=test)
# Assess the accuracy of the prediction for each soil class (Order) in test data
ct.mda <- table(test$Order, mda.predict)
cc.mda <- diag(prop.table(ct.mda, 1))
# Assess the total accuracy of test data
totcorrect.mda <- sum(diag(prop.table(ct4.mda)))


#qda
```

```r
qda.fei <- MASS::qda(Order ~pca.PC1 + pca.PC2+ pca.PC3+ pca.PC4, data = train)

#predict test data
qda.predict = predict(qda.fei, newdata=test)

# Assess the accuracy of the prediction for each soil class (Order) in test data
ct.qda <- table(test$Order, qda.predict$class)
cc.qda <- diag(prop.table(ct.qda, 1))


### comparison
#mda
correct.mda <- diag(prop.table(ct.mda, 1))
# total percent correct
totcorrect.mda <- sum(diag(prop.table(ct.mda)))

#qda
correct.qda <- diag(prop.table(ct.qda, 1))
# total percent correct
totcorrect.qda <- sum(diag(prop.table(ct.qda)))


##table

model_comparison <- data.frame(rbind(correct.mda, correct.qda))
model_comparison$Overall=c(totcorrect.mda,totcorrect.qda)

model_comparison <-model_comparison %>%set_rownames(c("MDA", "QDA"))
knitr::kable(round(model_comparison, digits = 2))

#p <- ggplot(test, aes(x = Sand, y = Clay, color = mda.predict)) + geom_point() + stat_contour(aes(x = 
#p
```