STATS 762 Week 9

May 15, 2019

Preparation

First I loaded required packages and load datasets.

```
library(MASS)
library(klaR)
```

```
## Warning: package 'klaR' was built under R version 3.5.3
```

```
library(nnet)
library(reshape2)
library(ggplot2)

#Setup random numbers
set.seed(1e4)
```

Low birth weight

Logistic regression

```
#read the data and convert low as a nominal variable
Birth=read.csv(file='Birthwt.csv',header=TRUE)
Birth$low=as.factor(Birth$low)

#slides 6 and 7
#Fit a logistic regression
Birth.logistic <- multinom(low ~ ., data = Birth)
```

```
## # weights:  18 (17 variable)
## initial  value 131.004817
## iter  10 value 95.617337
## iter  20 value 93.026314
## iter  30 value 92.040264
## final  value 92.030490
## converged
```

```
#predict class
bp <- predict(Birth.logistic,newdata=Birth)

#coefficient estimates
coef(Birth.logistic)
```

```
## (Intercept)        age1        age2        age3        lwt1        lwt2
##  -1.6335668 -13.2366667 -21.7955955 -16.2522595  -7.0484965  -2.4012762
##        lwt3       white       black       smoke        ptl1       ptl2m
##  -4.7102126  -0.7322719   0.5423045   0.8732237   1.6780989  -0.3168657
##          ht          ui        ftv1        ftv2       ftv3m
##   2.1082575   0.8099013  -0.3932662  -0.1630313   0.7328330
```

```
#confusion matrix
table(bp,Birth$low)
```

```
##
## bp    0   1
##   0 114  32
##   1  16  27
```

Iris data preparation

```
#read iris data
iris=read.csv(file='iris.csv', header = TRUE)
n=dim(iris)[1]

#a train data contains 120 randomly selected samples
#train.iris is a list of indices of samples in the train data
train.iris=sample(n,120,replace=FALSE)
```

Multinomial regression

```
#slides 14-17
#fit a multinomial regression for the train data
iris.mn <- multinom(Species ~ ., data = iris,subset=train.iris)
```

```
## # weights:  18 (10 variable)
## initial  value 131.833475
## iter  10 value 22.126151
## iter  20 value 2.386804
## iter  30 value 1.423919
## iter  40 value 1.100270
## iter  50 value 1.046728
## iter  60 value 0.932282
## iter  70 value 0.582339
## iter  80 value 0.523135
## iter  90 value 0.356813
## iter 100 value 0.330983
## final  value 0.330983
## stopped after 100 iterations
```

```
#class membership probabilities
prob=fitted(iris.mn)
print(prob[1:5,])
```

```
##           setosa   versicolor     virginica
## 67 8.827678e-124 1.000000e+00  1.198569e-15
## 71 8.216716e-133 9.131764e-01  8.682360e-02
## 50  1.000000e+00 4.189651e-30 1.328473e-208
## 89 7.108254e-104 1.000000e+00  2.089928e-38
## 60 3.425184e-107 1.000000e+00  7.339474e-31
```

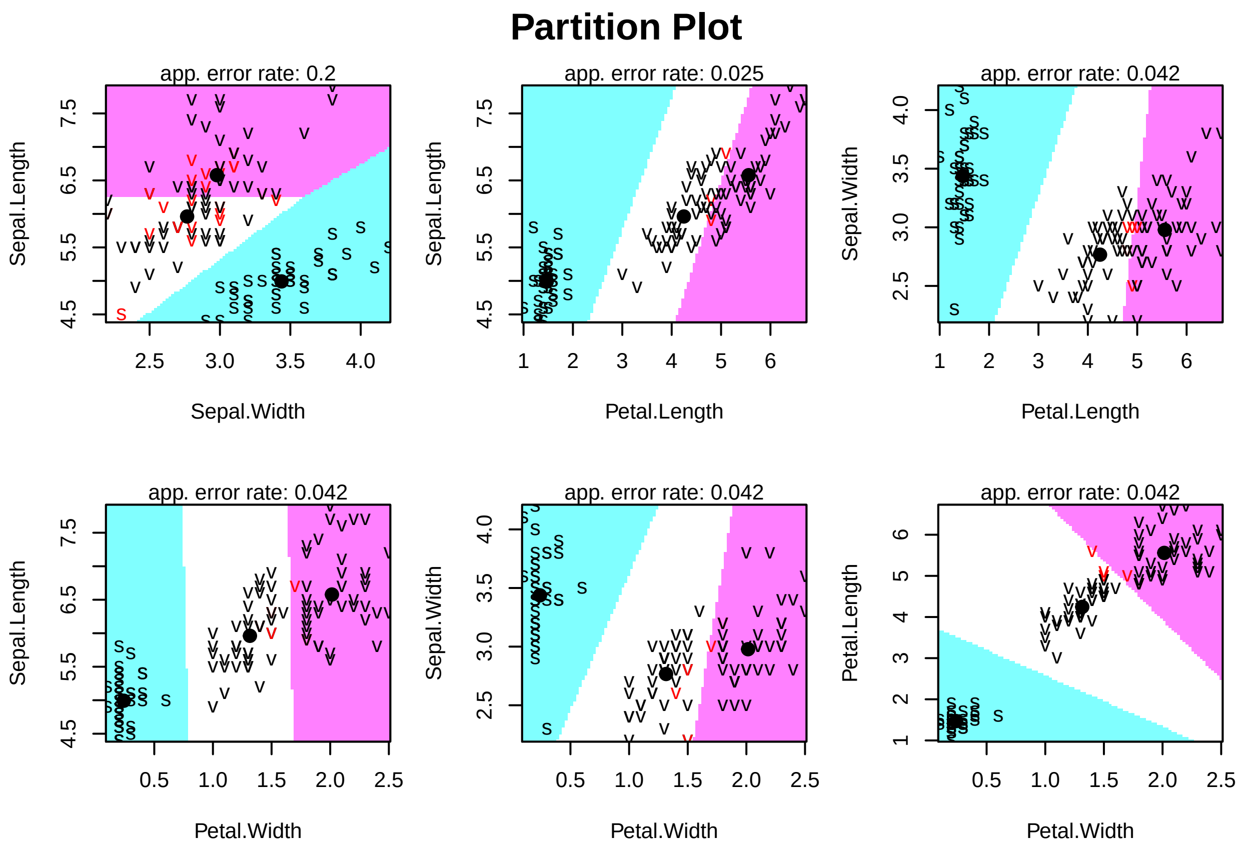
```
#prediction for the test data
pp <- predict(iris.mn,newdata=iris[-train.iris,])
```

```
#confusion matrix
table(pp,iris$Species[-train.iris])
```

```
##
## pp           setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         0
##   virginica       0          1         9
```

LDA

```
#slides 26-28
#Fit the LDA for the test data
lda.iris <- lda(Species ~ .,iris,subset=train.iris)

#Plot boundaries
partimat(Species ~ ., data=iris[train.iris,], method="lda")
```



```
#prediction
lda.pred=predict(lda.iris,iris[-train.iris,])

#print the class membership probability
lda.pred$posterior
```

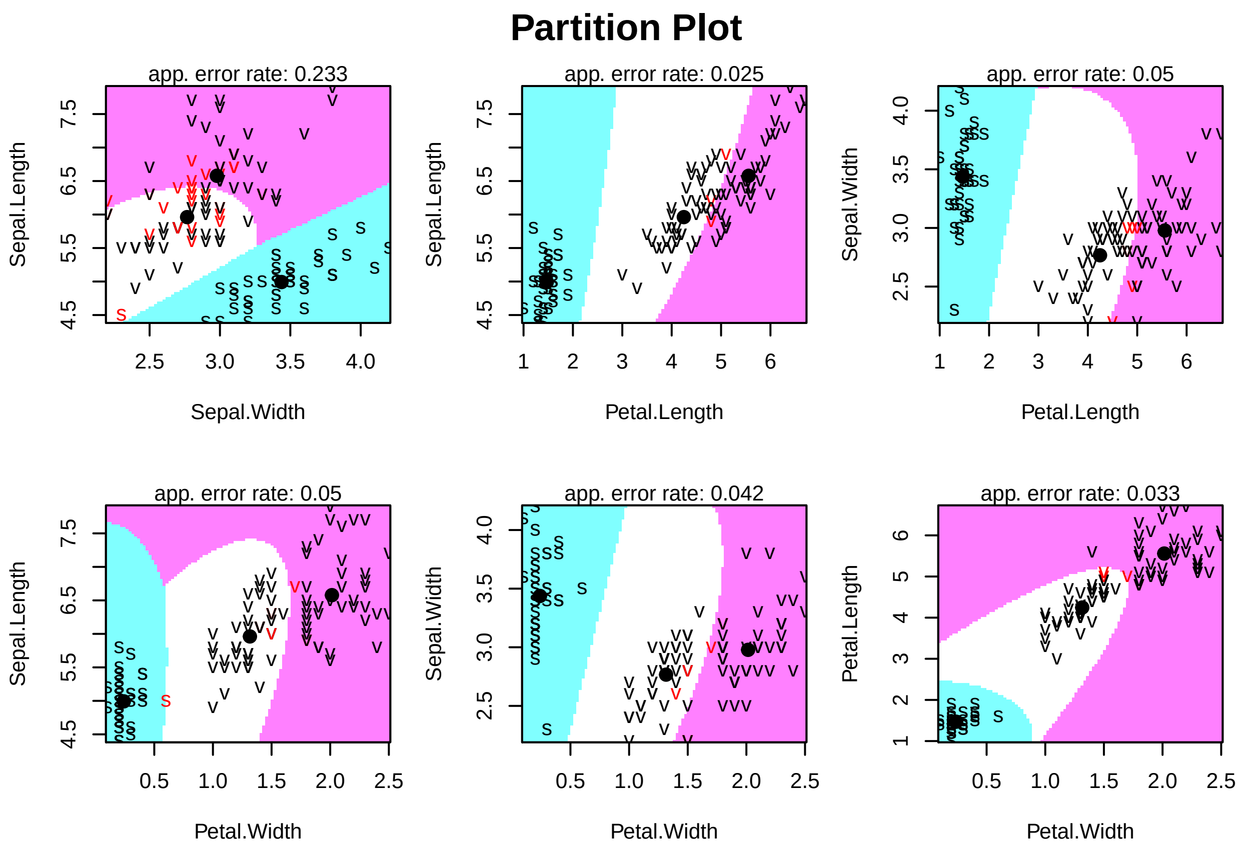
```
##            setosa   versicolor    virginica
## 10  1.000000e+00 4.573220e-20 1.520405e-41
## 13  1.000000e+00 6.477449e-20 1.848727e-41
## 14  1.000000e+00 2.736322e-21 3.939742e-43
## 16  1.000000e+00 2.551334e-30 3.312917e-53
## 17  1.000000e+00 3.541977e-27 1.436859e-49
## 22  1.000000e+00 1.866369e-22 3.122372e-43
## 24  1.000000e+00 5.570129e-16 1.019777e-34
## 26  1.000000e+00 1.881435e-17 6.534732e-38
## 37  1.000000e+00 2.263041e-26 1.212443e-49
## 46  1.000000e+00 1.012529e-17 5.821823e-38
## 51  2.206658e-20 9.999477e-01 5.232806e-05
## 52  1.831774e-21 9.995608e-01 4.392163e-04
## 56  5.720243e-25 9.985092e-01 1.490802e-03
## 61  1.652071e-20 9.999995e-01 5.315794e-07
## 62  9.303793e-22 9.995573e-01 4.427416e-04
## 84  2.402264e-35 7.217220e-02 9.278278e-01
## 85  1.579873e-26 9.515875e-01 4.841245e-02
## 86  3.037823e-22 9.950186e-01 4.981376e-03
## 88  4.285151e-26 9.997016e-01 2.984384e-04
## 94  5.776589e-16 1.000000e+00 3.081967e-08
## 95  4.443869e-23 9.997901e-01 2.098603e-04
## 107 3.268023e-36 2.220516e-02 9.777948e-01
## 111 8.028988e-35 8.167290e-03 9.918327e-01
## 113 1.196015e-42 8.321732e-05 9.999168e-01
## 119 3.005080e-65 1.261932e-10 1.000000e+00
## 121 1.720555e-46 2.028755e-06 9.999980e-01
## 124 1.776848e-34 7.380713e-02 9.261929e-01
## 130 4.135260e-36 4.849002e-02 9.515100e-01
## 141 7.725361e-49 3.508387e-07 9.999996e-01
## 145 4.949252e-50 6.320754e-08 9.999999e-01
```

```
#confusion matrix
table(lda.pred$class,iris$Species[-train.iris])
```

```
##
##                 setosa versicolor virginica
##   setosa            10          0         0
##   versicolor         0         10         0
##   virginica          0          1         9
```

QDA

```
#slides 30-34
#Fit the QDA for the train data
qda.iris <- qda(Species ~ .,iris,subset=train.iris)

#plot boundaries
partimat(Species ~ ., data=iris[train.iris,], method="qda")
```



```
#prediction
qda.pred=predict(qda.iris,iris[-train.iris,])

#class membership probability
qda.pred$posterior
```

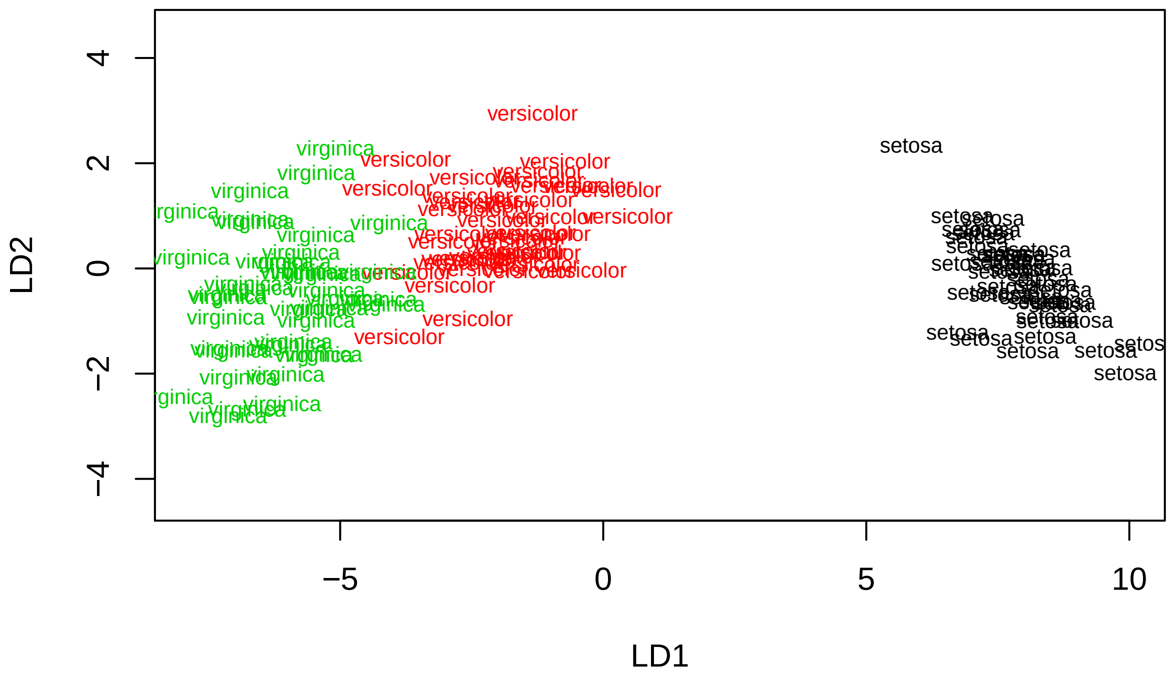
```
##            setosa   versicolor    virginica
## 10   1.000000e+00 1.939852e-20 3.074246e-42
## 13   1.000000e+00 1.156013e-19 7.636634e-42
## 14   1.000000e+00 8.221906e-20 5.391686e-42
## 16   1.000000e+00 1.073771e-40 1.572583e-63
## 17   1.000000e+00 2.634520e-32 4.772562e-56
## 22   1.000000e+00 9.096346e-26 2.522076e-47
## 24   1.000000e+00 2.449560e-16 3.147541e-36
## 26   1.000000e+00 2.388778e-17 1.058343e-38
## 37   1.000000e+00 3.365059e-29 7.275027e-55
## 46   1.000000e+00 1.036567e-17 3.051268e-39
## 51   1.873434e-93 9.999888e-01 1.117047e-05
## 52   1.063951e-85 9.999125e-01 8.746071e-05
## 56   2.547396e-79 9.952434e-01 4.756580e-03
## 61   3.532178e-42 9.999349e-01 6.510487e-05
## 62   5.872831e-75 9.996815e-01 3.184808e-04
## 84  1.448678e-117 8.278758e-02 9.172124e-01
## 85   1.765355e-84 9.502255e-01 4.977445e-02
## 86   9.249440e-87 9.984985e-01 1.501541e-03
## 88   3.475780e-84 9.986637e-01 1.336314e-03
## 94   4.644362e-35 9.999995e-01 5.488211e-07
## 95   3.405821e-69 9.995440e-01 4.559744e-04
## 107  1.910310e-97 8.392983e-04 9.991607e-01
## 111 2.217926e-135 1.141574e-02 9.885843e-01
## 113 8.516511e-164 1.614888e-04 9.998385e-01
## 119 9.516144e-269 1.134035e-09 1.000000e+00
## 121 1.457729e-185 1.879731e-06 9.999981e-01
## 124 2.051255e-119 4.590726e-02 9.540927e-01
## 130 9.112623e-158 2.328233e-02 9.767177e-01
## 141 2.337582e-188 1.006037e-08 1.000000e+00
## 145 2.033599e-199 1.393443e-09 1.000000e+00
```

```
#confusion matrix
table(qda.pred$class,iris$Species[-train.iris])
```

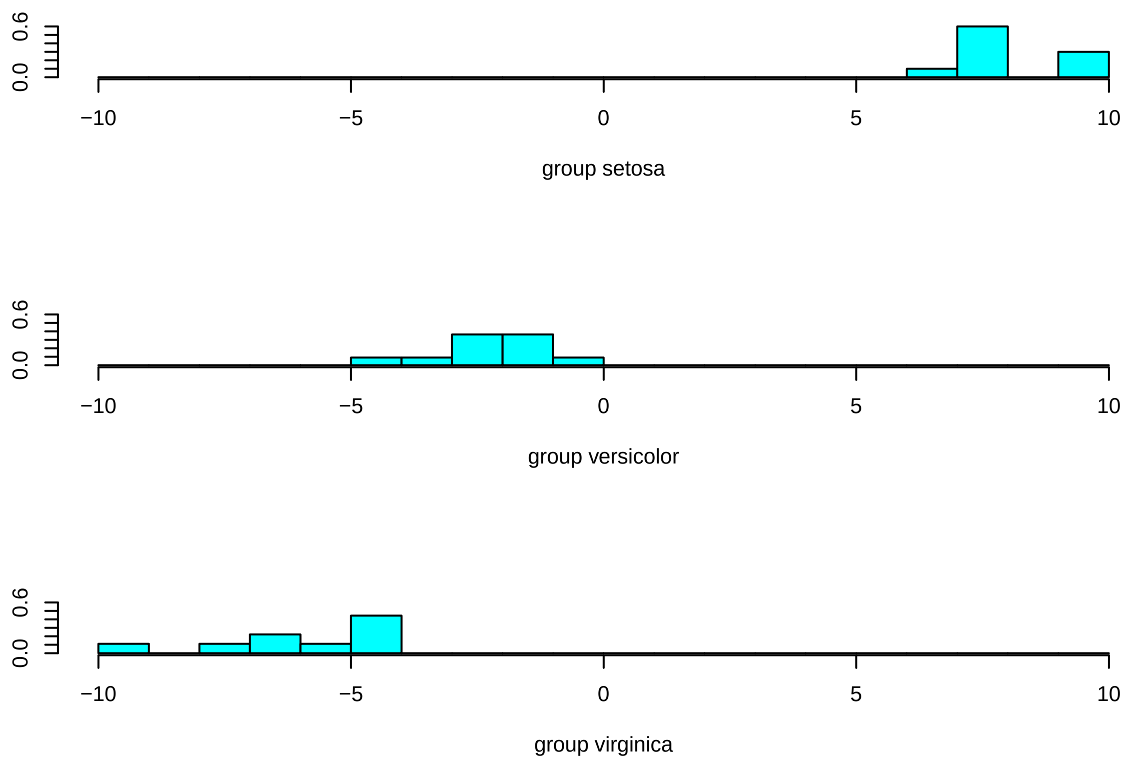
```
##
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         0
##   virginica       0          1         9
```

Discriminant function

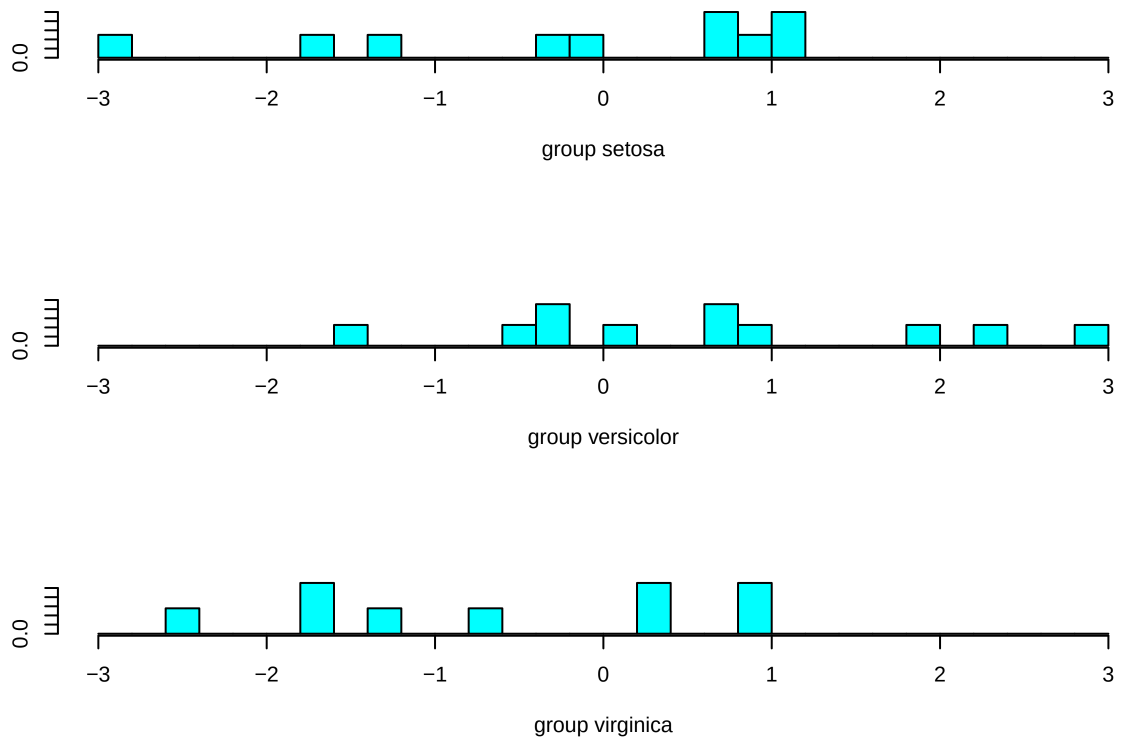
```
#slides 43-45
#recall lda fit
lda.iris <- lda(Species ~ .,iris,subset=train.iris)

#plot projected samples
plot(lda.iris, col = as.integer(iris$Species[train.iris]))
```



```
#histogram of projected samples using DA1
ldahist(lda.pred$x[,1],g=iris$Species[-train.iris])
```



```
#histogram of projected samples using DA2
ldahist(lda.pred$x[,2],g=iris$Species[-train.iris])
```



Glass data

```
#load the glass data
glass=read.csv(file='glass.csv',header=TRUE);
glass$Type=as.factor(glass$Type)
```

Multinomial regression

```
#slides 55-56
#Fit the multinomial regression
glass.mn <- multinom(Type ~ ., glass)
```

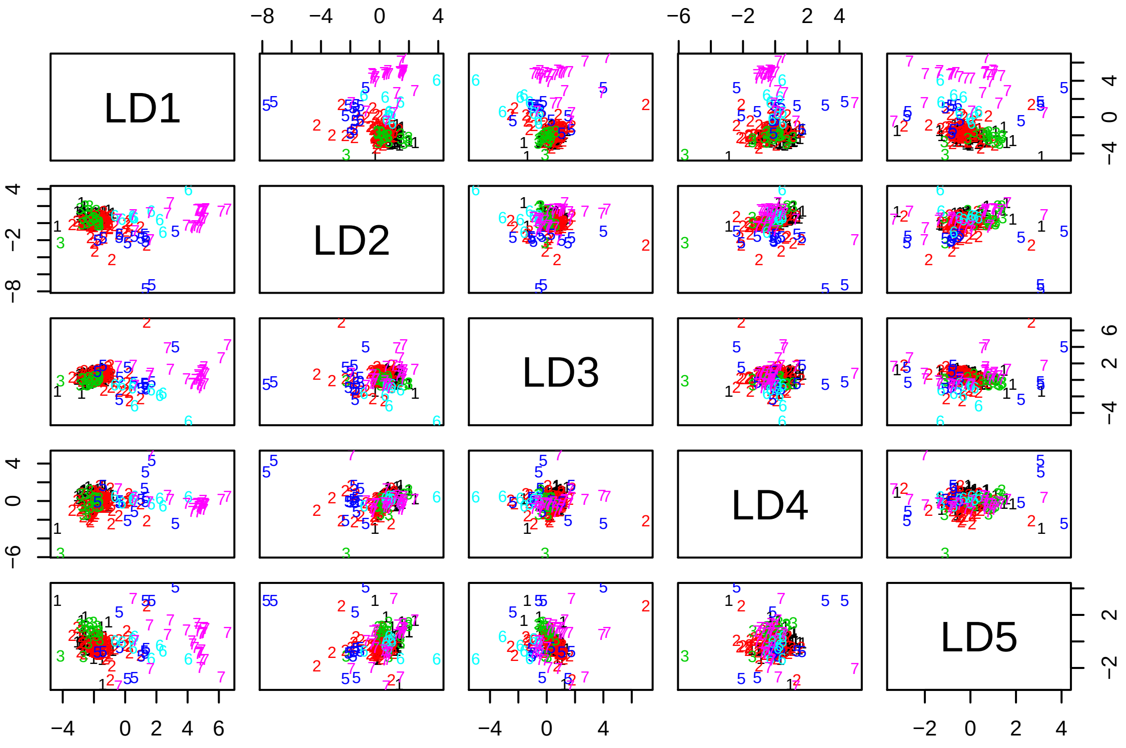
```
## # weights:  66 (50 variable)
## initial  value 383.436526
## iter  10 value 257.359885
## iter  20 value 181.634208
## iter  30 value 161.554088
## iter  40 value 157.912577
## iter  50 value 154.889493
## iter  60 value 153.706333
## iter  70 value 153.334999
## iter  80 value 152.219340
## iter  90 value 149.994098
## iter 100 value 149.743843
## final  value 149.743843
## stopped after 100 iterations
```

```
#Prediction
glass.mnpredict <- predict(glass.mn,glass)

#confusion matrix
table(glass.mnpredict,glass$Type)
```

```
##
## glass.mnpredict  1  2  3  5  6  7
##               1 52 19 10  0  0  0
##               2 18 54  7  3  0  2
##               3  0  0  0  0  0  0
##               5  0  1  0  9  0  0
##               6  0  0  0  0  9  0
##               7  0  2  0  1  0 27
```

LDA

```
#slides 57-58
#LDA
glass.lda <- lda(Type~.,glass)

#prediction
glass.ldapredict <- predict(glass.lda,glass)

#projected samples
plot(glass.lda, col = as.integer(glass$Type))
```



```r
#confusion matrix
table(glass.ldapredict$class,glass$Type)
```

```
##
##      1  2  3  5  6  7
##   1 52 17 11  0  1  1
##   2 15 54  6  5  2  2
##   3  3  0  0  0  0  0
##   5  0  3  0  7  0  1
##   6  0  2  0  0  6  0
##   7  0  0  0  1  0 25
```

QDA

If you fit the QDA naively, you will get an error message that the number of samples for Type=6 is too small. We exclude samples for Type=6 and fit the QDA.

```r
#slides 59
#glass.sub is the data excluding Type=6.
#reset the level.
glass.sub=glass[which(glass$Type!='6'),]; glass.sub$Type=factor(glass.sub$Type)

#QDA
glass.qda <- qda(Type~.,glass.sub)

#Prediction
```

```
glass.qdapredict <- predict(glass.qda,glass.sub)

#confusion matrix
table(glass.qdapredict$class,glass.sub$Type)
```

```
##
##      1  2  3  5  7
##   1 63 44  0  0  0
##   2  5 29  0  3  0
##   3  2  2 17  0  0
##   5  0  0  0 10  0
##   7  0  1  0  0 29
```

subset variables minimizing the AIc.

```
glass.aic=stepAIC(glass.mn,k=2,trace=FALSE)
```

```
## # weights:  60 (45 variable)
## initial  value 383.436526
## iter  10 value 257.349721
## iter  20 value 181.695198
## iter  30 value 161.838552
## iter  40 value 158.058376
## iter  50 value 155.239690
## iter  60 value 154.288425
## iter  70 value 154.073007
## iter  80 value 153.390949
## iter  90 value 149.937989
## iter 100 value 149.620303
## final  value 149.620303
## stopped after 100 iterations
## # weights:  60 (45 variable)
## initial  value 383.436526
## iter  10 value 236.986651
## iter  20 value 178.016939
## iter  30 value 164.899031
## iter  40 value 160.007638
## iter  50 value 158.495803
## iter  60 value 157.843305
## iter  70 value 156.521543
## iter  80 value 156.015282
## iter  90 value 155.598392
## iter 100 value 155.482188
## final  value 155.482188
## stopped after 100 iterations
## # weights:  60 (45 variable)
## initial  value 383.436526
## iter  10 value 264.430521
## iter  20 value 193.280510
## iter  30 value 179.643622
## iter  40 value 171.343282
## iter  50 value 164.576651
## iter  60 value 163.306037
## iter  70 value 160.440801
## iter  80 value 159.444204
## iter  90 value 157.097626
## iter 100 value 156.754334
## final  value 156.754334
## stopped after 100 iterations
## # weights:  60 (45 variable)
## initial  value 383.436526
## iter  10 value 259.284475
## iter  20 value 196.295853
## iter  30 value 176.863644
## iter  40 value 173.797462
## iter  50 value 168.316361
## iter  60 value 165.911050
## iter  70 value 165.085354
## iter  80 value 162.879571
## iter  90 value 160.938495
## iter 100 value 160.842664
## final  value 160.842664
## stopped after 100 iterations
## # weights:  60 (45 variable)
## initial  value 383.436526
## iter  10 value 238.842925
## iter  20 value 174.502379
## iter  30 value 160.983573
## iter  40 value 157.235258
## iter  50 value 155.703689
## iter  60 value 154.784159
## iter  70 value 154.476293
## iter  80 value 154.023801
## iter  90 value 153.676207
## iter 100 value 153.604653
## final  value 153.604653
## stopped after 100 iterations
## # weights:  60 (45 variable)
## initial  value 383.436526
## iter  10 value 256.596642
## iter  20 value 182.343568
## iter  30 value 170.541928
## iter  40 value 167.477130
## iter  50 value 162.576042
## iter  60 value 160.927386
## iter  70 value 160.371432
## iter  80 value 157.823834
## iter  90 value 157.271700
## iter 100 value 157.094780
## final  value 157.094780
## stopped after 100 iterations
## # weights:  60 (45 variable)
## initial  value 383.436526
## iter  10 value 244.344936
## iter  20 value 175.095645
## iter  30 value 164.636681
## iter  40 value 158.157793
## iter  50 value 155.923262
## iter  60 value 154.961461
## iter  70 value 154.386035
## iter  80 value 154.199698
## iter  90 value 153.403268
## iter 100 value 153.176983
## final  value 153.176983
## stopped after 100 iterations
## # weights:  60 (45 variable)
## initial  value 383.436526
## iter  10 value 260.094000
## iter  20 value 189.684568
## iter  30 value 168.244651
## iter  40 value 163.955662
## iter  50 value 158.623001
## iter  60 value 157.833853
## iter  70 value 157.099277
## iter  80 value 156.175928
## iter  90 value 156.076296
## iter 100 value 156.043060
## final  value 156.043060
## stopped after 100 iterations
## # weights:  60 (45 variable)
## initial  value 383.436526
## iter  10 value 257.510683
## iter  20 value 180.334712
## iter  30 value 165.005869
## iter  40 value 162.376464
## iter  50 value 157.854487
## iter  60 value 155.282529
## iter  70 value 153.541996
## iter  80 value 151.598016
## iter  90 value 151.492862
## iter 100 value 151.486234
## final  value 151.486234
## stopped after 100 iterations
## # weights:  60 (45 variable)
## initial  value 383.436526
## iter  10 value 257.349721
## iter  20 value 181.695198
## iter  30 value 161.838552
## iter  40 value 158.058376
## iter  50 value 155.239690
## iter  60 value 154.288425
## iter  70 value 154.073007
## iter  80 value 153.390949
## iter  90 value 149.937989
## iter 100 value 149.620303
## final  value 149.620303
## stopped after 100 iterations
## # weights:  54 (40 variable)
## initial  value 383.436526
## iter  10 value 236.942494
## iter  20 value 177.996974
## iter  30 value 165.312475
## iter  40 value 161.641421
## iter  50 value 159.939832
## iter  60 value 159.389039
## iter  70 value 158.194278
## iter  80 value 157.541561
## iter  90 value 157.518008
## iter 100 value 157.479441
## final  value 157.479441
## stopped after 100 iterations
## # weights:  54 (40 variable)
## initial  value 383.436526
## iter  10 value 264.276042
## iter  20 value 194.232654
## iter  30 value 182.390277
## iter  40 value 176.629161
## iter  50 value 170.598911
## iter  60 value 167.399980
## iter  70 value 163.337156
## iter  80 value 162.764303
## iter  90 value 162.674255
## iter 100 value 162.598302
## final  value 162.598302
## stopped after 100 iterations
## # weights:  54 (40 variable)
## initial  value 383.436526
## iter  10 value 259.261500
## iter  20 value 192.040430
## iter  30 value 176.655720
## iter  40 value 174.182935
## iter  50 value 171.308581
## iter  60 value 170.126554
## iter  70 value 168.760798
## iter  80 value 162.593332
## iter  90 value 161.812502
## iter 100 value 161.296293
## final  value 161.296293
## stopped after 100 iterations
## # weights:  54 (40 variable)
## initial  value 383.436526
## iter  10 value 239.728789
## iter  20 value 174.638904
## iter  30 value 161.324824
## iter  40 value 157.311534
## iter  50 value 155.724722
## iter  60 value 155.303127
## iter  70 value 155.083157
## iter  80 value 154.516979
## iter  90 value 154.392043
## iter 100 value 154.329870
## final  value 154.329870
## stopped after 100 iterations
## # weights:  54 (40 variable)
## initial  value 383.436526
## iter  10 value 256.599871
## iter  20 value 185.019455
## iter  30 value 170.789637
## iter  40 value 168.760325
## iter  50 value 165.879771
## iter  60 value 164.290428
## iter  70 value 163.683538
## iter  80 value 160.491039
## iter  90 value 160.219408
## iter 100 value 160.154358
## final  value 160.154358
## stopped after 100 iterations
## # weights:  54 (40 variable)
## initial  value 383.436526
## iter  10 value 244.324864
## iter  20 value 177.808459
## iter  30 value 167.118370
## iter  40 value 161.790811
## iter  50 value 158.353651
## iter  60 value 155.436943
## iter  70 value 153.981268
## iter  80 value 153.309727
## iter  90 value 153.275784
## iter 100 value 153.239626
## final  value 153.239626
## stopped after 100 iterations
## # weights:  54 (40 variable)
## initial  value 383.436526
## iter  10 value 260.077544
## iter  20 value 189.827199
## iter  30 value 168.497779
## iter  40 value 165.085345
## iter  50 value 161.521257
## iter  60 value 160.952117
## iter  70 value 160.030011
## iter  80 value 157.549209
## iter  90 value 157.307468
## iter 100 value 157.244713
## final  value 157.244713
## stopped after 100 iterations
## # weights:  54 (40 variable)
## initial  value 383.436526
## iter  10 value 257.500493
## iter  20 value 179.406446
## iter  30 value 165.025861
## iter  40 value 162.482847
## iter  50 value 158.650240
## iter  60 value 157.312179
## iter  70 value 154.522363
## iter  80 value 151.368199
## iter  90 value 151.214444
## iter 100 value 151.179754
## final  value 151.179754
## stopped after 100 iterations
## # weights:  54 (40 variable)
## initial  value 383.436526
## iter  10 value 257.500493
## iter  20 value 179.406446
## iter  30 value 165.025861
## iter  40 value 162.482847
## iter  50 value 158.650240
## iter  60 value 157.312179
## iter  70 value 154.522363
## iter  80 value 151.368199
## iter  90 value 151.214444
## iter 100 value 151.179754
## final  value 151.179754
## stopped after 100 iterations
## # weights:  48 (35 variable)
## initial  value 383.436526
## iter  10 value 237.364051
## iter  20 value 180.321055
## iter  30 value 169.973728
## iter  40 value 165.887098
## iter  50 value 163.758925
## iter  60 value 161.276168
## iter  70 value 160.997630
## iter  80 value 160.906631
## iter  90 value 160.845941
## iter 100 value 160.261020
## final  value 160.261020
## stopped after 100 iterations
## # weights:  48 (35 variable)
## initial  value 383.436526
## iter  10 value 254.580757
## iter  20 value 195.540549
## iter  30 value 186.974011
## iter  40 value 179.816633
## iter  50 value 168.577918
## iter  60 value 167.135947
## iter  70 value 166.274436
## iter  80 value 166.178520
## iter  90 value 166.088852
## iter 100 value 165.168752
## final  value 165.168752
## stopped after 100 iterations
## # weights:  48 (35 variable)
## initial  value 383.436526
## iter  10 value 259.417865
## iter  20 value 197.488110
## iter  30 value 179.274285
## iter  40 value 177.199068
## iter  50 value 172.825931
## iter  60 value 165.622981
## iter  70 value 164.200465
## iter  80 value 163.400191
## iter  90 value 163.159732
## iter 100 value 163.060075
## final  value 163.060075
## stopped after 100 iterations
## # weights:  48 (35 variable)
## initial  value 383.436526
## iter  10 value 239.870270
## iter  20 value 176.529089
## iter  30 value 165.367450
## iter  40 value 161.540879
## iter  50 value 159.640800
## iter  60 value 158.931856
## iter  70 value 158.426648
## iter  80 value 158.171220
## iter  90 value 158.007188
## iter 100 value 157.880522
## final  value 157.880522
## stopped after 100 iterations
## # weights:  48 (35 variable)
## initial  value 383.436526
## iter  10 value 256.769714
## iter  20 value 184.559069
## iter  30 value 174.392179
## iter  40 value 173.936784
## iter  50 value 170.308484
## iter  60 value 166.888437
## iter  70 value 166.538044
## iter  80 value 166.516214
## iter  90 value 166.496447
## iter 100 value 165.655600
## final  value 165.655600
## stopped after 100 iterations
## # weights:  48 (35 variable)
## initial  value 383.436526
## iter  10 value 244.612453
## iter  20 value 181.483349
## iter  30 value 172.513183
## iter  40 value 163.677551
## iter  50 value 160.071905
## iter  60 value 158.514370
## iter  70 value 156.943613
## iter  80 value 156.912767
## iter  90 value 156.848746
## iter 100 value 156.135643
## final  value 156.135643
## stopped after 100 iterations
## # weights:  48 (35 variable)
## initial  value 383.436526
## iter  10 value 260.219226
## iter  20 value 193.907623
## iter  30 value 172.565967
## iter  40 value 169.455792
## iter  50 value 164.281023
## iter  60 value 160.052528
## iter  70 value 159.780576
## iter  80 value 159.777897
## final  value 159.777887
## converged
## # weights:  48 (35 variable)
## initial  value 383.436526
## iter  10 value 244.612453
## iter  20 value 181.483349
## iter  30 value 172.513183
## iter  40 value 163.677551
## iter  50 value 160.071905
## iter  60 value 158.514370
## iter  70 value 156.943613
## iter  80 value 156.912767
## iter  90 value 156.848746
## iter 100 value 156.135643
## final  value 156.135643
## stopped after 100 iterations
## # weights:  42 (30 variable)
## initial  value 383.436526
## iter  10 value 251.528352
## iter  20 value 191.071085
## iter  30 value 182.391829
## iter  40 value 179.167258
## iter  50 value 177.016443
## iter  60 value 176.613152
## iter  70 value 176.473726
## iter  80 value 176.372525
## iter  90 value 175.839096
## iter 100 value 175.785520
## final  value 175.785520
## stopped after 100 iterations
## # weights:  42 (30 variable)
## initial  value 383.436526
## iter  10 value 240.615803
## iter  20 value 208.245548
## iter  30 value 194.575360
## iter  40 value 189.620960
## iter  50 value 187.819646
## iter  60 value 186.722870
## iter  70 value 186.593592
## iter  80 value 186.409019
## iter  90 value 185.219597
## iter 100 value 184.347238
## final  value 184.347238
## stopped after 100 iterations
## # weights:  42 (30 variable)
## initial  value 383.436526
## iter  10 value 250.712432
## iter  20 value 196.833244
## iter  30 value 189.213322
## iter  40 value 183.604240
## iter  50 value 178.373186
## iter  60 value 177.658780
## iter  70 value 177.569900
## iter  80 value 177.478787
## iter  90 value 176.675996
## iter 100 value 176.537863
## final  value 176.537863
## stopped after 100 iterations
## # weights:  42 (30 variable)
## initial  value 383.436526
## iter  10 value 245.762421
## iter  20 value 181.579304
## iter  30 value 172.817025
## iter  40 value 168.577441
## iter  50 value 165.847989
## iter  60 value 165.536103
## iter  70 value 165.258165
## iter  80 value 165.025810
## iter  90 value 164.477250
## iter 100 value 164.133520
## final  value 164.133520
## stopped after 100 iterations
## # weights:  42 (30 variable)
## initial  value 383.436526
## iter  10 value 247.036386
## iter  20 value 186.651670
## iter  30 value 182.968354
## iter  40 value 177.662597
## iter  50 value 173.899002
## iter  60 value 172.358001
## iter  70 value 172.246764
## iter  80 value 172.142245
## iter  90 value 171.198279
## iter 100 value 170.391960
## final  value 170.391960
## stopped after 100 iterations
## # weights:  42 (30 variable)
## initial  value 383.436526
## iter  10 value 251.377023
## iter  20 value 190.534312
## iter  30 value 180.988049
## iter  40 value 171.889819
## iter  50 value 169.622979
## iter  60 value 168.528857
## iter  70 value 168.326426
## iter  80 value 168.312778
## iter  90 value 167.997594
## iter 100 value 167.708767
## final  value 167.708767
## stopped after 100 iterations
```

```
glass.aic$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## Type ~ RI + Na + Mg + Al + Si + K + Ca + Ba + Fe
##
```

```
## Final Model:
## Type ~ Na + Mg + Al + Si + K + Ba
##
##
##   Step Df  Deviance Resid. Df Resid. Dev      AIC
## 1                         164   299.4877 399.4877
## 2 - RI  5 0.2470805       169   299.2406 389.2406
## 3 - Fe  5 3.1189018       174   302.3595 382.3595
## 4 - Ca  5 9.9117784       179   312.2713 382.2713
```

```
glass.mn.aic <- multinom(Type ~ Na+Mg+Al+Si+K+Ba, glass)
```

```
## # weights:  48 (35 variable)
## initial  value 383.436526
## iter  10 value 244.612453
## iter  20 value 181.483349
## iter  30 value 172.513183
## iter  40 value 163.677551
## iter  50 value 160.071905
## iter  60 value 158.514370
## iter  70 value 156.943613
## iter  80 value 156.912767
## iter  90 value 156.848746
## iter 100 value 156.135643
## final  value 156.135643
## stopped after 100 iterations
```

```
#Prediction
glass.mnpredict.aic <- predict(glass.mn.aic,glass)

#confusion matrix
table(glass.mnpredict.aic,glass$Type)
```

```
##
## glass.mnpredict.aic  1  2  3  5  6  7
##                   1 51 21 10  0  0  0
##                   2 19 52  7  3  0  2
##                   3  0  0  0  0  0  0
##                   5  0  1  0  9  0  0
##                   6  0  0  0  0  9  0
##                   7  0  2  0  1  0 27
```

Questions

1. We want to predict the green tea quality (levels 1-5). For each tea, 5 chemical contains are measured. Write how you will predict the tea quality given 5 chemical contains using multinomial regression, LDA and QDA.

2. The spreadsheet binary.csv (see Week 9) contains postgraduate applications. If an applicant is accepted to the postgraduate programme, admit=1. Otherwise admit=0. We want to predict the postgraduate school entry given gre and gpa scores and, the rank of highschool. The train data contains 70% of samples and the test data contains the rest samples. Predict the admit state using (a) a logistic regression, (b) LDA and (c) QDA. Compare the predictivity of the three classification methods.