APPLIED MULTIVARIATE ANALYSIS

Classification of Vowel Sounds with Multivariate Analysis Methods

Ruixie Fang
002422276

Introduction

This project aims to study the classification of eleven vowel sounds with multivariate analysis methods. The datasets used here are "*vowel-train*" and "*vowel-test*". Both the training data and test data contain 11 classes and 10 predictors. The classes correspond to 11 vowel sounds, each contained in 11 different words. Here are the words, preceded by the symbols that represent them:

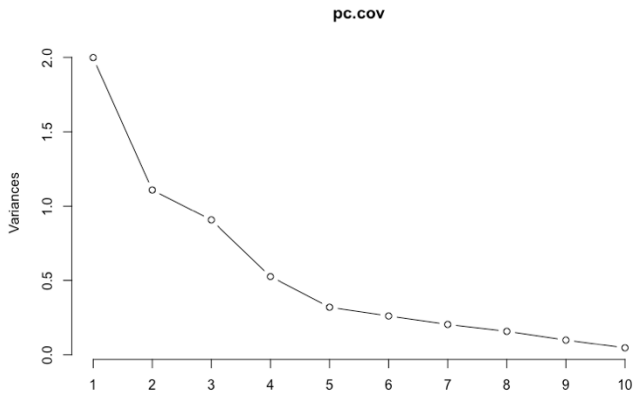
Vowel	Word	Vowel	Word	Vowel	Word	Vowel	Word
i:	heed	O	hod	I	hid	C:	hoard
E	head	U	hood	A	had	u:	Who'd
a:	hard	3:	heard	Y	hud		

In the training data set, each of eight speakers spoke each word six times. There are thus 528 training observations. In the test data set, each of seven speakers spoke each word six times. There are thus 462 test observations. The ten predictors (x.1,…,x.10) are derived from the digitized speech in a rather complicated way, but standard in the speech recognition world. The variable y is the class index for each observation. In this project, We will conduct a principal component analysis (PCA), linear discriminant analysis (LDA), quadratic discriminant analysis (QDA) and clustering analysis.

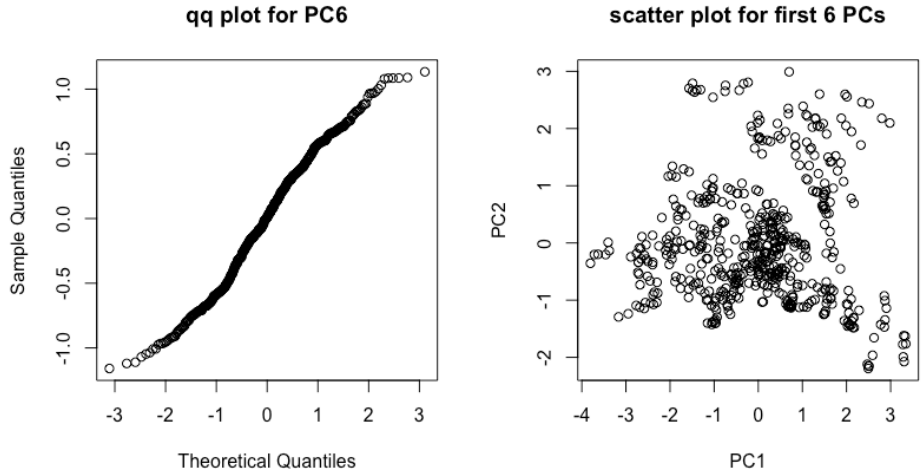
Q1 PCA

```
> tr=read.csv("/Users/balloon_n/Documents/F/study/class/Applied Multi/Final proj/vowel-train.txt")
> ts=read.csv("/Users/balloon_n/Documents/F/study/class/Applied Multi/Final proj/vowel-test.txt")
> training=tr[,3:12]
> testing=ts[,3:12]
> s<-cov(training);print(s,digit=3)
         x.1     x.2      x.3     x.4      x.5     x.6      x.7     x.8      x.9     x.10
x.1   0.9177 -0.5725 -0.30617  0.0138 -0.11690  0.1503 -0.02311  0.1143  0.01096 -0.05099
x.2  -0.5725  1.3479  0.06227 -0.3832 -0.29450 -0.3319  0.12837  0.1065  0.06296 -0.08411
x.3  -0.3062  0.0623  0.54962  0.0778 -0.00621 -0.2573 -0.10049 -0.0570  0.08187  0.12861
x.4   0.0138 -0.3832  0.07777  0.5919 -0.04438  0.0607 -0.20565 -0.0256  0.13113  0.08685
x.5  -0.1169 -0.2945 -0.00621 -0.0444  0.52130  0.0541  0.00255 -0.2067 -0.23992  0.02131
x.6   0.1503 -0.3319 -0.25728  0.0607  0.05412  0.4206  0.01074  0.0878 -0.05878 -0.10927
x.7  -0.0231  0.1284 -0.10049 -0.2057  0.00255  0.0107  0.22968 -0.0184 -0.06696 -0.08250
x.8   0.1143  0.1065 -0.05700 -0.0256 -0.20670  0.0878 -0.01844  0.3547  0.05159 -0.10031
x.9   0.0110  0.0630  0.08187  0.1311 -0.23992 -0.0588 -0.06696  0.0516  0.38388 -0.00177
x.10 -0.0510 -0.0841  0.12861  0.0869  0.02131 -0.1093 -0.08250 -0.1003 -0.00177  0.31396
> print(eigen(s),digit=3)
eigen() decomposition
$values
 [1] 1.9987 1.1085 0.9068 0.5263 0.3202 0.2614 0.2045 0.1583 0.0997 0.0469
$vectors
         [,1]     [,2]    [,3]     [,4]      [,5]      [,6]     [,7]   [,8]   [,9]   [,10]
 [1,]  0.4802  0.48519  0.1859  0.51249  0.101286  0.095045 -0.23565  0.046  0.162 -0.3654
 [2,] -0.7788  0.27818  0.0399 -0.04738  0.254244  0.178779 -0.18279  0.109  0.172 -0.3756
 [3,] -0.1627 -0.49115  0.2207  0.31756 -0.598890 -0.021712 -0.03082 -0.125  0.299 -0.3433
 [4,]  0.2229 -0.35393  0.4304 -0.42648  0.342367  0.175486 -0.34300 -0.309 -0.120 -0.2867
 [5,]  0.1301 -0.30142 -0.5916 -0.00467 -0.000172  0.000453 -0.32765  0.454 -0.276 -0.3905
 [6,]  0.2481  0.18408 -0.1383 -0.57109 -0.162966 -0.006532  0.39353  0.110  0.472 -0.3783
 [7,] -0.0766  0.18715 -0.2320  0.05203 -0.040510 -0.356838  0.22154 -0.609 -0.429 -0.4112
 [8,] -0.0193  0.29748  0.2415 -0.20251 -0.547539  0.451892 -0.00166  0.157 -0.531 -0.0405
 [9,] -0.0431  0.00138  0.4915 -0.04946  0.055632 -0.655530  0.12402  0.489 -0.216 -0.1435
[10,]  0.0139 -0.26915  0.0835  0.28293  0.343691  0.407964  0.68247  0.148 -0.172 -0.1963
> pc.cov<-prcomp(training);summary(pc.cov)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7    PC8     PC9    PC10
Standard deviation     1.4138 1.0529 0.9523 0.72544 0.56583 0.51127 0.45226 0.3978 0.31576 0.21647
Proportion of Variance 0.3549 0.1968 0.1610 0.09345 0.05686 0.04642 0.03632 0.0281 0.01771 0.00832
Cumulative Proportion  0.3549 0.5518 0.7128 0.80627 0.86313 0.90955 0.94587 0.9740 0.99168 1.00000
```

```
> plot(pc.cov,type="lines")
```




```
> par(mfrow=c(1,2))
> qqnorm(pc.cov$x[,6],main="qq plot for PC6")
> plot(pc.cov$x[,1:6],main="scatter plot for first 6 PCs")
```



Firstly, a principal component analysis is conducted based on the covariance matrix of the training data set. (Since these variables are measured on same unit and similar scales, we choose covariance matrix rather than correlation matrix). The corresponding percentage that each eigenvalue contributes to the total sample variance are shown below:

eigenvalues	1.9987	1.1085	0.9068	0.5263	0.3202	0.2614	0.2045	0.1583	0.0997	0.0469
percentage	35.49%	19.68%	16.10%	9.35%	5.69%	4.64%	3.63%	2.81%	1.77%	0.83%

The 1st column in eigenvectors output shows coefficients of linear combination that defines PC1 and so on. Example principal component function for PC6 is shown as below:

$$y_6 = 0.095x_1 + 0.179x_2 - 0.022x_3 + 0.175x_4 + 0.0005x_5 - 0.007x_6 - 0.357x_7 + 0.452x_8 - 0.656x_9 + 0.408x_{10}$$

Using the scree plot and the proportion of variance explained, it appears as if 6 components should be retained. These components explain about 90% of the total sample variability in the training data set, and seems a reasonable number given the scree plot. And we check the normality for K=6, the plots show that it's normally distributed with no suspect observation.

Q2 PC-LDA

In this step, we decide to use 6 principal components to summarize the original training dataset. We get the scores of these 6 components for each observation and then use these scores to conduct the linear discriminant analysis, and apply the obtained linear discriminant rule to testing dataset.

```
> library(MASS)
> pctr_score=as.data.frame(pc.cov$x[,1:6])
> labeltr_pc=cbind(pctr_score,tr[,2]);colnames(labeltr_pc)[7] <- "y"
> eig=as.matrix(pc.cov$rotation)
> #predict(pc.cov,testing)
> ts_pc=as.matrix(testing)-matrix(rep(colMeans(training),each=nrow(testing)),nrow=nrow(testing))
> pcts_score=as.data.frame((ts_pc%*%eig)[,1:6])
> labelts_pc=cbind(pcts_score,ts[,2]);colnames(labelts_pc)[7] <- "y"
> ##PC LDA
> vlpc.lda <- lda(y ~.,labeltr_pc)
> pred.train.pc <- predict(vlpc.lda,labeltr_pc)$class
> pred.test.pc <- predict(vlpc.lda,labelts_pc)$class
> mean(pred.train.pc != labeltr_pc$y)
[1] 0.405303
> mean(pred.test.pc != labelts_pc$y)
[1] 0.5909091
```

For linear discriminant analysis, the misclassification error rate based on the training data set is 0.405303, and the error rate for testing data set is 0.5909091.

Q3 PC-QDA

Here, we repeat the work in Q2 and do the quadratic discriminant analysis.

```
> vlpc.qda <- qda(y ~.,labeltr_pc)
> pred.tr.pc <- predict(vlpc.qda,labeltr_pc)$class
> pred.ts.pc <- predict(vlpc.qda,labelts_pc)$class
> mean(pred.tr.pc != labeltr_pc$y)
[1] 0.1231061
> mean(pred.ts.pc != labelts_pc$y)
[1] 0.4458874
```

For quadratic discriminant analysis, the misclassification error rate based on the training data set is 0.1231061, and the error rate for testing data set is 0.4458874. Comparing with LDA, QDA gives the lower testing error rate.

Q4 Original-LDA QDA

Next, we conduct LDA and QDA for the original training data set.

```
> labeltr=tr[,2:12];labelts=ts[,2:12]
> ##LDA
> vl.lda <- lda(y ~., labeltr)
> pred.train <- predict(vl.lda,labeltr)$class
> pred.test <- predict(vl.lda,labelts)$class
> mean(pred.train != labeltr$y)
[1] 0.3162879
> mean(pred.test != labelts$y)
[1] 0.5562771
> ##QDA
> vl.qda <- qda(y ~., labeltr)
> pred.tr <- predict(vl.qda,labeltr)$class
> pred.ts <- predict(vl.qda,labelts)$class
> mean(pred.tr != labeltr$y)
[1] 0.01136364
> mean(pred.ts != labelts$y)
[1] 0.5281385
```

The following table summarizes these error rates with those from Q2 and Q3:

		PC data	Original data
Training	LDA	0.4053030	0.3162879
	QDA	0.1231061	0.0113636
Testing	LDA	0.5909091	0.5562771
	QDA	**0.4458874**	**0.5281385**

From this table, we can see that:

- In the testing QDA, PC data has lower error rate (higher accuracy) than original data, because using the PCA before the LDA may filter out some noises.
- For all training and testing data sets, QDA has lower error rate (higher accuracy) than LDA. It may because LDA assumes a common covariance matrix while QDA assumes that each class has its own covariance matrix. In this case, QDA might be more appropriate.

Q5 Classes Distinguish

Then, we distinguish the first two most difficult classes in eight datasets (*pc-lda-train, pc-lda-test, pc-qda-train, pc-qda-test, orig-lda-train, orig-lda-test, orig-qda-train, orig-qda-test)* by using confusion matrix, get the most frequent two by comparing each measure, and then remove them. Also, repeat work and compare the changes of error rates for LDA and QDA.

```
> library(caret)
> #PCdata
> p1=confusionMatrix(pred.train.pc,as.factor(labeltr$y))$byClass;print(p1,digit = 3)
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision Recall    F1 Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1       0.646       0.969          0.674          0.965     0.674  0.646 0.660     0.0909         0.0587              0.0871            0.807
Class: 2       0.292       0.969          0.483          0.932     0.483  0.292 0.364     0.0909         0.0265              0.0549            0.630
Class: 3       0.917       0.956          0.677          0.991     0.677  0.917 0.779     0.0909         0.0833              0.1231            0.936
Class: 4       0.708       0.965          0.667          0.971     0.667  0.708 0.687     0.0909         0.0644              0.0966            0.836
Class: 5       0.396       0.952          0.452          0.940     0.452  0.396 0.422     0.0909         0.0360              0.0795            0.674
Class: 6       0.375       0.948          0.419          0.938     0.419  0.375 0.396     0.0909         0.0341              0.0814            0.661
Class: 7       0.500       0.946          0.480          0.950     0.480  0.500 0.490     0.0909         0.0455              0.0947            0.723
Class: 8       0.750       0.963          0.667          0.975     0.667  0.750 0.706     0.0909         0.0682              0.1023            0.856
Class: 9       0.417       0.973          0.606          0.943     0.606  0.417 0.494     0.0909         0.0379              0.0625            0.695
Class: 10      0.750       0.950          0.600          0.974     0.600  0.750 0.667     0.0909         0.0682              0.1136            0.850
Class: 11      0.792       0.965          0.691          0.979     0.691  0.792 0.738     0.0909         0.0720              0.1042            0.878
> p2=confusionMatrix(pred.test.pc,as.factor(labelts$y))$byClass;print(p2,digit = 3)
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision Recall    F1 Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1       0.690       0.938          0.527          0.968     0.527  0.690 0.598     0.0909         0.0628              0.1190            0.814
Class: 2       0.262       0.945          0.324          0.928     0.324  0.262 0.289     0.0909         0.0238              0.0736            0.604
Class: 3       0.333       0.929          0.318          0.933     0.318  0.333 0.326     0.0909         0.0303              0.0952            0.631
Class: 4       0.786       0.883          0.402          0.976     0.402  0.786 0.532     0.0909         0.0714              0.1775            0.835
Class: 5       0.286       0.919          0.261          0.928     0.261  0.286 0.273     0.0909         0.0260              0.0996            0.602
Class: 6       0.262       0.952          0.355          0.928     0.355  0.262 0.301     0.0909         0.0238              0.0671            0.607
Class: 7       0.143       0.955          0.240          0.918     0.240  0.143 0.179     0.0909         0.0130              0.0541            0.549
Class: 8       0.667       0.976          0.737          0.967     0.737  0.667 0.700     0.0909         0.0606              0.0823            0.821
Class: 9       0.310       0.924          0.289          0.930     0.289  0.310 0.299     0.0909         0.0281              0.0974            0.617
Class: 10      0.381       0.967          0.533          0.940     0.533  0.381 0.444     0.0909         0.0346              0.0649            0.674
Class: 11      0.381       0.962          0.500          0.940     0.500  0.381 0.432     0.0909         0.0346              0.0693            0.671
> p3=confusionMatrix(pred.tr.pc,as.factor(labeltr$y))$byClass;print(p3,digit = 3)
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision Recall    F1 Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1       0.833       0.998          0.976          0.984     0.976  0.833 0.899     0.0909         0.0758              0.0777            0.916
Class: 2       0.875       0.975          0.778          0.987     0.778  0.875 0.824     0.0909         0.0795              0.1023            0.925
Class: 3       0.896       0.983          0.843          0.990     0.843  0.896 0.869     0.0909         0.0814              0.0966            0.940
Class: 4       0.896       0.992          0.915          0.990     0.915  0.896 0.905     0.0909         0.0814              0.0890            0.944
Class: 5       0.917       0.981          0.830          0.992     0.830  0.917 0.871     0.0909         0.0833              0.1004            0.949
Class: 6       0.688       0.983          0.805          0.969     0.805  0.688 0.742     0.0909         0.0625              0.0777            0.835
Class: 7       0.938       0.990          0.900          0.994     0.900  0.938 0.918     0.0909         0.0852              0.0947            0.964
Class: 8       1.000       0.992          0.923          1.000     0.923  1.000 0.960     0.0909         0.0909              0.0985            0.996
Class: 9       0.771       0.988          0.860          0.977     0.860  0.771 0.813     0.0909         0.0701              0.0814            0.879
Class: 10      0.938       0.998          0.978          0.994     0.978  0.938 0.957     0.0909         0.0852              0.0871            0.968
Class: 11      0.896       0.985          0.860          0.990     0.860  0.896 0.878     0.0909         0.0814              0.0947            0.941
> p4=confusionMatrix(pred.ts.pc,as.factor(labelts$y))$byClass;print(p4,digit = 3)
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision Recall    F1 Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1       0.714       0.945          0.566          0.971     0.566  0.714 0.632     0.0909         0.0649              0.1147            0.830
Class: 2       0.690       0.933          0.509          0.968     0.509  0.690 0.586     0.0909         0.0628              0.1234            0.812
Class: 3       0.524       0.981          0.733          0.954     0.733  0.524 0.611     0.0909         0.0476              0.0649            0.752
Class: 4       0.571       0.983          0.774          0.958     0.774  0.571 0.658     0.0909         0.0519              0.0671            0.777
Class: 5       0.262       0.981          0.579          0.930     0.579  0.262 0.361     0.0909         0.0238              0.0411            0.621
Class: 6       0.762       0.881          0.390          0.974     0.390  0.762 0.516     0.0909         0.0693              0.1775            0.821
Class: 7       0.786       0.902          0.446          0.977     0.446  0.786 0.569     0.0909         0.0714              0.1602            0.844
Class: 8       0.476       0.967          0.588          0.949     0.588  0.476 0.526     0.0909         0.0433              0.0736            0.721
Class: 9       0.381       0.960          0.485          0.939     0.485  0.381 0.427     0.0909         0.0346              0.0714            0.670
Class: 10      0.357       0.990          0.789          0.939     0.789  0.357 0.492     0.0909         0.0325              0.0411            0.674
Class: 11      0.571       0.986          0.800          0.958     0.800  0.571 0.667     0.0909         0.0519              0.0649            0.779
```

```
> #Originaldata
> o1=confusionMatrix(pred.train,as.factor(labeltr$y))$byClass;print(o1,digit = 3)
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision Recall    F1 Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1       0.667       0.979          0.762          0.967     0.762  0.667 0.711     0.0909         0.0606              0.0795             0.823
Class: 2       0.583       0.971          0.667          0.959     0.667  0.583 0.622     0.0909         0.0530              0.0795             0.777
Class: 3       0.875       0.973          0.764          0.987     0.764  0.875 0.816     0.0909         0.0795              0.1042             0.924
Class: 4       0.750       0.988          0.857          0.975     0.857  0.750 0.800     0.0909         0.0682              0.0795             0.869
Class: 5       0.688       0.960          0.635          0.968     0.635  0.688 0.660     0.0909         0.0625              0.0985             0.824
Class: 6       0.479       0.969          0.605          0.949     0.605  0.479 0.535     0.0909         0.0436              0.0720             0.724
Class: 7       0.688       0.958          0.623          0.968     0.623  0.688 0.653     0.0909         0.0625              0.1004             0.823
Class: 8       0.708       0.977          0.756          0.971     0.756  0.708 0.731     0.0909         0.0644              0.0852             0.843
Class: 9       0.604       0.969          0.659          0.961     0.659  0.604 0.630     0.0909         0.0549              0.0833             0.786
Class: 10      0.688       0.952          0.589          0.968     0.589  0.688 0.635     0.0909         0.0625              0.1061             0.820
Class: 11      0.792       0.956          0.644          0.979     0.644  0.792 0.710     0.0909         0.0720              0.1117             0.874
> o2=confusionMatrix(pred.test,as.factor(labelts$y))$byClass;print(o2,digit = 3)
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision Recall    F1 Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1       0.667       0.926          0.475          0.965     0.475  0.667 0.554     0.0909         0.0606              0.1277             0.796
Class: 2       0.381       0.940          0.390          0.938     0.390  0.381 0.386     0.0909         0.0346              0.0887             0.661
Class: 3       0.381       0.957          0.471          0.939     0.471  0.381 0.421     0.0909         0.0346              0.0736             0.669
Class: 4       0.786       0.964          0.688          0.978     0.688  0.786 0.733     0.0909         0.0714              0.1039             0.875
Class: 5       0.167       0.957          0.280          0.920     0.280  0.167 0.209     0.0909         0.0152              0.0541             0.562
Class: 6       0.452       0.867          0.253          0.941     0.253  0.452 0.325     0.0909         0.0411              0.1623             0.660
Class: 7       0.262       0.969          0.458          0.929     0.458  0.262 0.333     0.0909         0.0238              0.0519             0.615
Class: 8       0.548       0.976          0.697          0.956     0.697  0.548 0.613     0.0909         0.0498              0.0714             0.762
Class: 9       0.357       0.938          0.366          0.936     0.366  0.357 0.361     0.0909         0.0325              0.0887             0.648
Class: 10      0.310       0.945          0.361          0.932     0.361  0.310 0.333     0.0909         0.0281              0.0779             0.627
Class: 11      0.571       0.948          0.522          0.957     0.522  0.571 0.545     0.0909         0.0519              0.0996             0.760
> o3=confusionMatrix(pred.tr,as.factor(labeltr$y))$byClass;print(o3,digit = 3)
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision Recall    F1 Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1       1.000       0.998          0.980          1.000     0.980  1.000 0.990     0.0909         0.0909              0.0928             0.999
Class: 2       1.000       0.998          0.980          1.000     0.980  1.000 0.990     0.0909         0.0909              0.0928             0.999
Class: 3       0.979       1.000          1.000          0.998     1.000  0.979 0.989     0.0909         0.0890              0.0890             0.990
Class: 4       1.000       1.000          1.000          1.000     1.000  1.000 1.000     0.0909         0.0909              0.0909             1.000
Class: 5       0.979       1.000          1.000          0.998     1.000  0.979 0.989     0.0909         0.0890              0.0890             0.990
Class: 6       0.938       0.998          0.978          0.994     0.978  0.938 0.957     0.0909         0.0852              0.0871             0.968
Class: 7       1.000       1.000          1.000          1.000     1.000  1.000 1.000     0.0909         0.0909              0.0909             1.000
Class: 8       1.000       1.000          1.000          1.000     1.000  1.000 1.000     0.0909         0.0909              0.0909             1.000
Class: 9       1.000       0.998          0.980          1.000     0.980  1.000 0.990     0.0909         0.0909              0.0928             0.999
Class: 10      0.979       1.000          1.000          0.998     1.000  0.979 0.989     0.0909         0.0890              0.0890             0.990
Class: 11      1.000       0.996          0.960          1.000     0.960  1.000 0.980     0.0909         0.0909              0.0947             0.998
> o4=confusionMatrix(pred.ts,as.factor(labelts$y))$byClass;print(o4,digit = 3)
         Sensitivity Specificity Pos Pred Value Neg Pred Value Precision Recall    F1 Prevalence Detection Rate Detection Prevalence Balanced Accuracy
Class: 1       0.881       0.931          0.561          0.987     0.561  0.881 0.685     0.0909         0.0801              0.1429             0.906
Class: 2       0.524       0.943          0.478          0.952     0.478  0.524 0.500     0.0909         0.0476              0.0996             0.733
Class: 3       0.286       0.990          0.750          0.933     0.750  0.286 0.414     0.0909         0.0260              0.0346             0.638
Class: 4       0.286       0.981          0.600          0.932     0.600  0.286 0.387     0.0909         0.0260              0.0433             0.633
Class: 5       0.381       0.962          0.500          0.940     0.500  0.381 0.432     0.0909         0.0346              0.0693             0.671
Class: 6       0.524       0.933          0.440          0.951     0.440  0.524 0.478     0.0909         0.0476              0.1082             0.729
Class: 7       0.524       0.860          0.272          0.948     0.272  0.524 0.358     0.0909         0.0476              0.1753             0.692
Class: 8       0.143       0.998          0.857          0.921     0.857  0.143 0.245     0.0909         0.0130              0.0152             0.570
Class: 9       0.905       0.850          0.376          0.989     0.376  0.905 0.531     0.0909         0.0823              0.2186             0.877
Class: 10      0.262       0.998          0.917          0.931     0.917  0.262 0.407     0.0909         0.0238              0.0260             0.630
Class: 11      0.476       0.974          0.645          0.949     0.645  0.476 0.548     0.0909         0.0433              0.0671             0.725
```

Based on the confusion matrix result, we compare each measure among classes in each dataset. The conclusion is that class 5 and class 6 are difficult to distinguish from others.

```
> '%!in%' <- Negate('%in%')
> newtr=as.data.frame(labeltr[labeltr$y %!in% c(5,6),])
> newts=as.data.frame(labelts[labelts$y %!in% c(5,6),])
> ##LDA
> newvl.lda <- lda(y ~., newtr)
> newpred.train <- predict(newvl.lda,newtr)$class
> newpred.test <- predict(newvl.lda,newts)$class
> mean(newpred.train != newtr$y)
[1] 0.2523148
> mean(newpred.test != newts$y)
[1] 0.478836
>
> ##QDA
> newvl.qda <- qda(y ~., newtr)
> newpred.tr <- predict(newvl.qda,newtr)$class
> newpred.ts <- predict(newvl.qda,newts)$class
> mean(newpred.tr != newtr$y)
[1] 0.00462963
> mean(newpred.ts != newts$y)
[1] 0.4603175
```

The following table shows the comparison with the results from Q4:

		Original data (all classes)	Classes {5,6} removed
Training	LDA	0.3162879	0.2523148
	QDA	0.0113636	0.0046296
Testing	LDA	0.5562771	0.4788360
	QDA	0.5281385	0.4603175

From this table, we can see that after removed classes {5,6}, all methods' error rate is smaller than before.

Q6 Clustering Analysis

To simplify the analysis, we select observations from classes {1,3,6,10} and conduct clustering analysis on these observations. And in order to get more comparison among different methods, we conduct hierarchical clustering methods (single, complete, average), K-means methods and model-based clustering method for both training and testing data sets.

A. Training

```
> library(stats)
> labeltr.sub=data.frame(labeltr[labeltr$y %in% c(1,3,6,10),])
> lel=c(1,2,3,4);labeltr.sub$y=factor(labeltr.sub$y,labels=lel)
> rownames(labeltr.sub) <- seq(length=nrow(labeltr.sub))
> sub=labeltr.sub[,2:11]

> #Hierarchical method
> dist.sub=dist(sub)
> ##single
> singl.sub = hclust(dist.sub, method = 'single')
> singl.clusterCut=cutree(singl.sub, 4)
> mean(singl.clusterCut!=labeltr.sub$y)
[1] 0.78125
> ##complete
> comp.sub = hclust(dist.sub, method = 'complete')
> comp.clusterCut=cutree(comp.sub, 4)
> mean(comp.clusterCut!=labeltr.sub$y)
[1] 0.6875
> ##average
> avg.sub = hclust(dist.sub, method = 'aver')
> avg.clusterCut=cutree(avg.sub, 4)
> mean(avg.clusterCut!=labeltr.sub$y)
[1] 0.875

> #K Means
> cl=kmeans(sub, 4)
> table(cl$cluster,labeltr.sub$y)

     1  2  3  4
  1  6  8  4 17
  2 12 24 43  6
  3 30 16  1  0
  4  0  0  0 25
> mean(cl$cluster != labeltr.sub$y)
[1] 0.7083333

> #model-based
> library("mclust")
> mc <- Mclust(sub,4)
fitting ...
  |======================================================| 100%
```

```
> table(mc$classification,labeltr.sub$y)

     1  2  3  4
  1 24 12 30 48
  2 24  0  0  0
  3  0 12 18  0
  4  0 24  0  0
> mean(mc$classification != labeltr.sub$y)
[1] 0.78125
```

B. Testing

```
> labelts.sub=data.frame(labelts[labelts$y %in% c(1,3,6,10),])
> lel=c(1,2,3,4);labelts.sub$y=factor(labelts.sub$y,labels=lel)
> rownames(labelts.sub) <- seq(length=nrow(labelts.sub))
> ts.sub=labelts.sub[,2:11]

> #Hierarchical method
> dist.ts.sub=dist(ts.sub)
> ##single
> ts.singl.sub = hclust(dist.ts.sub, method = 'single')
> ts.singl.clusterCut=cutree(ts.singl.sub, 4)
> mean(ts.singl.clusterCut!=labelts.sub$y)
[1] 0.5
> ##complete
> ts.comp.sub = hclust(dist.ts.sub, method = 'complete')
> ts.comp.clusterCut=cutree(ts.comp.sub, 4)
> mean(ts.comp.clusterCut!=labelts.sub$y)
[1] 0.6071429
> ##average
> ts.avg.sub = hclust(dist.ts.sub, method = 'aver')
> ts.avg.clusterCut=cutree(ts.avg.sub, 4)
> mean(ts.avg.clusterCut!=labelts.sub$y)
[1] 0.5

> #K Means
> ts.cl=kmeans(ts.sub, 4)
> table(ts.cl$cluster,labelts.sub$y)

     1  2  3  4
  1 16  0  0  6
  2  0 42 42  0
  3 26  0  0  3
  4  0  0  0 33
> mean(ts.cl$cluster != labelts.sub$y)
[1] 0.4583333
> #model-based
> ts.mc <- Mclust(ts.sub,4)
fitting ...
  |============================================================| 100%
> table(ts.mc$classification,labelts.sub$y)

     1  2  3  4
  1 24  0  0  6
  2  0 12 36 18
  3 18 18  6 18
  4  0 12  0  0
> mean(ts.mc$classification != labelts.sub$y)
[1] 0.75
```

The following table shows the performance results for each clustering method:

	Hierarchical			K Means	model-based
	single	complete	average		
Training	0.781	0.688	0.875	0.708	0.781
Testing	0.500	0.607	0.500	0.458	0.750

From this table, we can see that:

- There is no significant difference among different methods. The approaches regardless of performance, each approach has its own benefits.
- In K Means clustering, since we start with random choice of clusters, the results produced by running the algorithm multiple times are different. While results are reproducible in Hierarchical and model-based clustering method.
- Comparing with results before, LDA/QDA performs better.

Question 7 Conclusion

- For high-dimension data, to reduce the risk of overfitting and reduce computational complexity, PCA is a good method. Since it's a well-established mathematical technique for reducing the dimensionality of data, while keeping as much variation as possible.
- QDA has better performance in our case. And since the error rate difference between LDA and QDA are small, the LDA method is a convenient substitution.
- Removing these hard-distinguish classes can help improve the classification performance.
- For unsupervised clustering methods, each approach has its own benefits, it may work great in other cases. In our case, since our data is labeled training data, supervised methods like LDA/QDA performs better.