Module 7: Classification and Clustering

Weixing Song

July 24, 2018

First, we use the following R-codes to install all necessary packages.

```
list.of.packages=c("ggplot2","rrcov","klaR","latticeExtra","rgl","tree","cluster","mclust")
if(length(which(!list.of.packages %in% installed.packages()))){
  install.packages(list.of.packages[!list.of.packages %in% installed.packages()])}
```

Note that the some R codes in this handout are adapted from https://rstudio-pubs-static.s3.amazonaws.com/35817_2552e05f1d4e4db8ba87b334101a43da.html

1. Classification with Two MVN Populations — Common Σ and Equal Costs

Example 7.1 Detection of Hemophilia A Carriers

R-Codes

```
  library(rrcov)
```

```
## Loading required package: robustbase
```

```
## Scalable Robust Estimators with High Breakdown Point (version 1.4-4)
```

```
  data(hemophilia)

  Pi1=hemophilia[1:30,1:2]
  Pi2=hemophilia[31:75,1:2]
  xbar1=colMeans(Pi1)
  xbar2=colMeans(Pi2)
  S1=cov(Pi1)
  S2=cov(Pi2)
  Spinv=solve(((30-1)*S1+(45-1)*S2)/(30+45-2))

  x0=c(-0.210,-0.044)
  D=t(xbar1-xbar2)%*%Spinv%*%x0-0.5*t(xbar1-xbar2)%*%Spinv%*%(xbar1+xbar2)

  # equal prior and equal cost case
  c(D,0)
```

```
## [1] 0.2559494 0.0000000
```

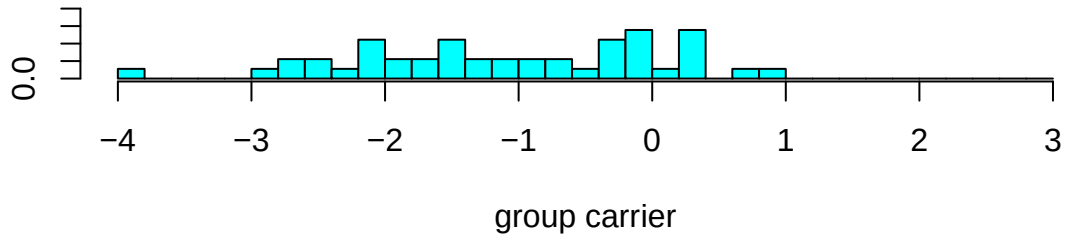
```
  # the unequal prior case
  p1=.75
  p2=.25
  c(D,log(p2/p1))
```

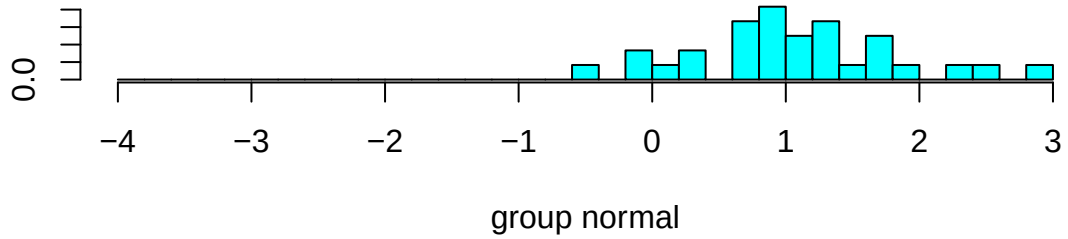
```
## [1]  0.2559494 -1.0986123
```

```
  # Using lda package

library(MASS)
mylda=lda(gr~., data=hemophilia,prior=c(0.5,0.5))
predict(mylda, newdata=data.frame(AHFactivity=x0[1],AHFantigen=x0[2]))$class
```

```
## [1] normal
## Levels: carrier normal
```

```
mylda.predict=predict(mylda)
ldahist(data=mylda.predict$x,g=hemophilia$gr)
```

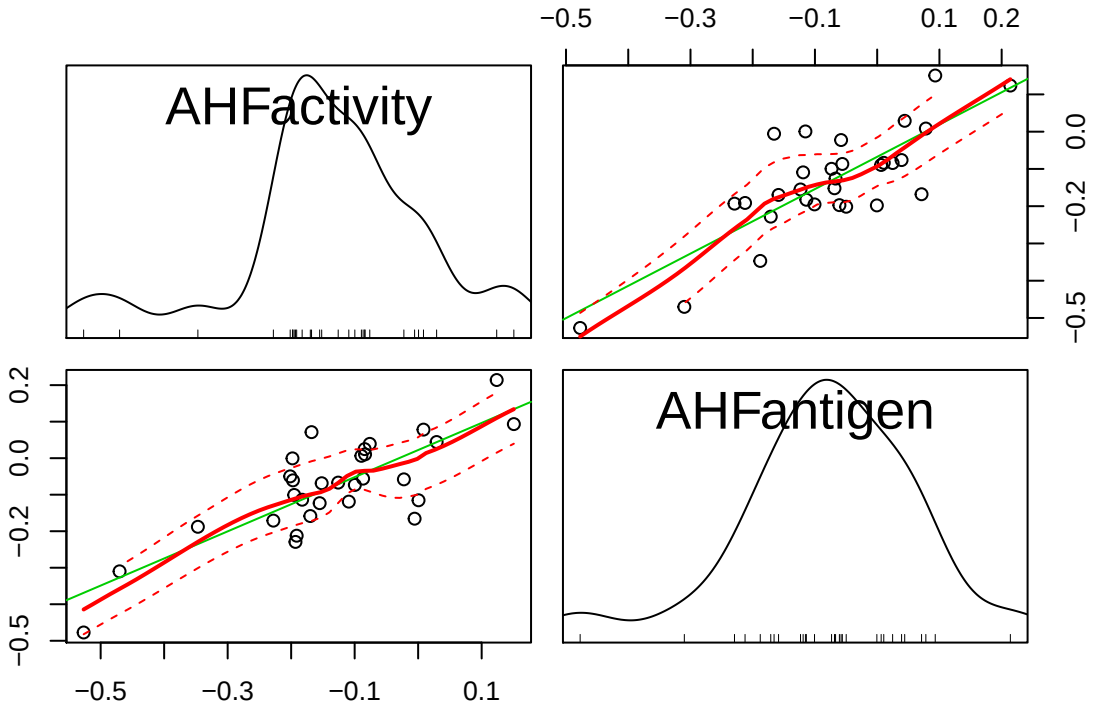




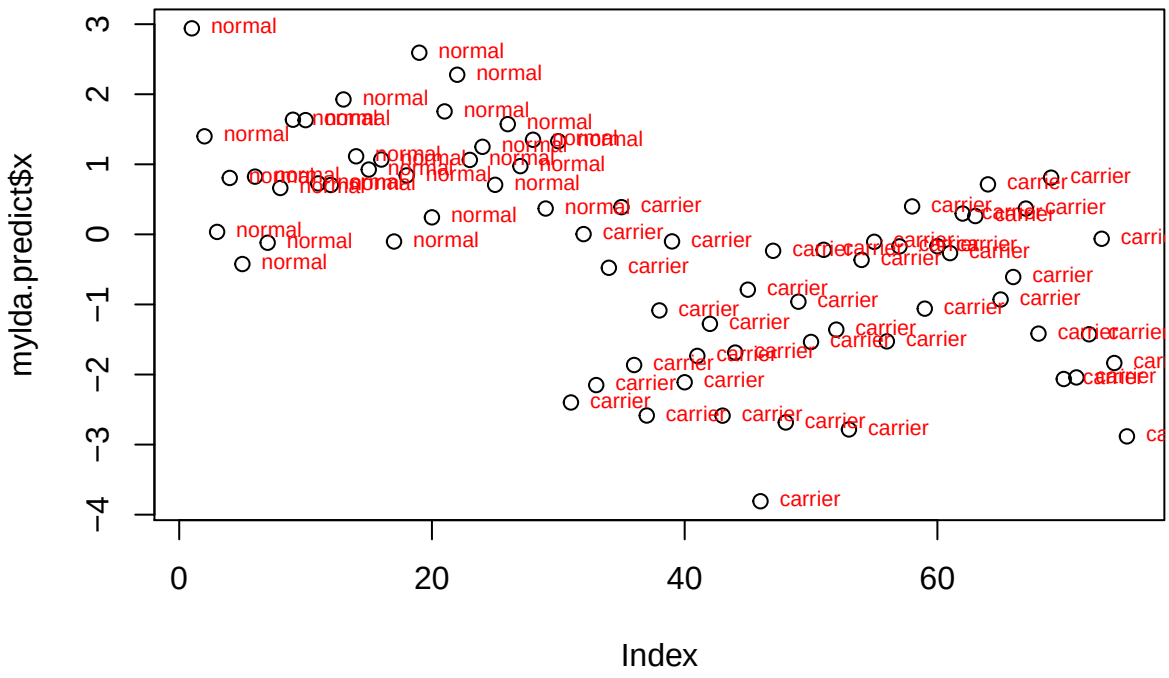

```
# Confusion Matrix

table(mylda.predict$class,hemophilia$gr)
```

```
##
##           carrier normal
##   carrier      37      3
##   normal        8     27
```

```
library(car) # to use scatterplotMatrix
scatterplotMatrix(Pi1)
```




```
#Scatterplots of the Discriminant Functions
#We can obtain a scatterplot of the best two discriminant functions, with the data
# points labelled by cultivar, by typing:

plot(mylda.predict$x) # make a scatterplot
text(mylda.predict$x,hemophilia$gr,cex=0.7,pos=4,col="red") # add labels
```



2 Classification with Two MVN Populations — Uncommon Σ's.

Example 7.2 (Classifying Alaskan and Canadian Salmon) The salmon fishery is a valuable resource for both the United States and Canada. Because it is a limited resource, it must be managed efficiently. Moreover, since more than one country is involved, problems must be solved equitably. that is, Alaskan commercial fishermen cannot catch too many Canadian salmon and vice versa.

These fish have a remarkable life cycle. They are born in freshwater streams and after a year or two swim into the ocean. After a couple of years in salt water, they return to their place of birth to spawn and die. At the time they are about to return as mature life, they are harvested while still in the ocean. To help regulate from Alaskan or Canadian waters. The fish carry some information about their birthplace in the growth rings on their scales. Typically, the rings associated with freshwater growth are smaller for the Alaskan-born than for the Canadian-born salmon.

The dataset salmon contains $n_1 = 50$ Alaskan born and $n_2 = 50$ Canadian born salmons along with variables X_1 =diameter of rings of the first-year freshwater growth, and X_2 =diamter of rings for the first year marine growth. In addition, females are coded as 1 and males are coded as 2.

Construct a classification rule for Alaskan and Canadian salmon.

```
myfile="S:/Workshop/Data/T11-2.DAT";
salmon=read.table(myfile)
type=salmon$V1;
gender=salmon$V2;
x1=salmon$V3;
x2=salmon$V4;
```

```
  myqda=qda(type~x1+x2,prior=c(0.5,0.5))

  mypredict=predict(myqda)

  # Confusion Matrix

  table(mypredict$class,type)
```

```
##    type
##      1  2
##   1 45  2
##   2  5 48
```

2. Classification with More Than Two MVN Populations — Common Σ and Equal Costs

Example 7.3 (Classifying a potential business-school graduate student) The adimission officer of a business school has used an "index" of undergraduate grade point average (GPA) and graduate management aptitude test (GMAT) scores to help decide which applicants should be admitted to the school's graduate programs. The data are listed in Table 11.6, with three variables x_1 =GPA, $x_2 = GMAT$, and the values for groups of recent applicants who have been categorized as π_1 : admitted; π_2 : do not admit; and π_3 : boderline.

R-Codes

```
   myfile="S:/Workshop/Data/T11-6.DAT";
   bsgs=read.table(myfile);
   x1=bsgs$V1;
   x2=bsgs$V2;
   gr=bsgs$V3;

   library(MASS)
   mylda=lda(gr~x1+x2, prior=c(1,1,1)/3)
   predict(mylda, newdata=data.frame(x1=3.21,x2=497))$class
```

```
## [1] 3
## Levels: 1 2 3
```

```
   mylda=lda(gr~x1+x2, prior=c(1,1,1)/3,CV=T) # Lachenbruch's Procedure.
 #  ldahist(g=gr,data=mylda$class,type="both")

   # Confusion Matrix

   table(mylda$class,gr)
```

```
##    gr
##      1  2  3
##   1 26  0  1
##   2  0 26  1
##   3  5  2 24
```

```r
# Posterior Probability

round(mylda.predict$posterior,digits=4)
```

```
##    carrier normal
## 1   0.0019 0.9981
## 2   0.0477 0.9523
## 3   0.4812 0.5188
## 4   0.1515 0.8485
## 5   0.7120 0.2880
## 6   0.1464 0.8536
## 7   0.5635 0.4365
## 8   0.1953 0.8047
## 9   0.0293 0.9707
## 10  0.0297 0.9703
## 11  0.1743 0.8257
## 12  0.1812 0.8188
## 13  0.0160 0.9840
## 14  0.0845 0.9155
## 15  0.1215 0.8785
## 16  0.0924 0.9076
## 17  0.5542 0.4458
## 18  0.1419 0.8581
## 19  0.0039 0.9961
## 20  0.3730 0.6270
## 21  0.0229 0.9771
## 22  0.0076 0.9924
## 23  0.0934 0.9066
## 24  0.0646 0.9354
## 25  0.1805 0.8195
## 26  0.0334 0.9666
## 27  0.1108 0.8892
## 28  0.0527 0.9473
## 29  0.3123 0.6877
## 30  0.0550 0.9450
## 31  0.9941 0.0059
## 32  0.4979 0.5021
## 33  0.9901 0.0099
## 34  0.7348 0.2652
## 35  0.3033 0.6967
## 36  0.9818 0.0182
## 37  0.9960 0.0040
## 38  0.9106 0.0894
## 39  0.5534 0.4466
## 40  0.9892 0.0108
## 41  0.9761 0.0239
## 42  0.9389 0.0611
## 43  0.9961 0.0039
## 44  0.9736 0.0264
## 45  0.8440 0.1560
## 46  0.9997 0.0003
## 47  0.6227 0.3773
## 48  0.9968 0.0032
## 49  0.8864 0.1136
```

```
## 50  0.9638 0.0362
## 51  0.6156 0.3844
## 52  0.9482 0.0518
## 53  0.9974 0.0026
## 54  0.6855 0.3145
## 55  0.5565 0.4435
## 56  0.9632 0.0368
## 57  0.5906 0.4094
## 58  0.2987 0.7013
## 59  0.9061 0.0939
## 60  0.5895 0.4105
## 61  0.6392 0.3608
## 62  0.3461 0.6539
## 63  0.3637 0.6363
## 64  0.1785 0.8215
## 65  0.8795 0.1205
## 66  0.7860 0.2140
## 67  0.3126 0.6874
## 68  0.9539 0.0461
## 69  0.1505 0.8495
## 70  0.9880 0.0120
## 71  0.9875 0.0125
## 72  0.9548 0.0452
## 73  0.5334 0.4666
## 74  0.9807 0.0193
## 75  0.9979 0.0021
```

Example 7.4 (Fisher's discriminants for the crude-oil data)

Gerrild and Lantz collected curde-oil samples from standstone in the Elk Hills, California, petroleum reserve. These crude oils can be assigned to one of the three stratigraphic populations

$$\pi_1: \text{ Wilhelm sandstone}, \quad \pi_2: \text{ Sub-Mulinia sandstone}, \quad \pi_3: \text{ Upper sandstone}$$

on the basis of thier chemistry. For illustration purposes, we consider only the five variables

$$\begin{aligned} X_1 &= \text{vanadium in percent ash} \\ X_2 &= \sqrt{\text{iron in percent ash}} \\ X_3 &= \sqrt{\text{berylium in percent ash}} \\ X_4 &= 1/\text{saturated hydrocarbons in percent ash} \\ X_5 &= \text{aromatic hydrocarbons in percent ash} \end{aligned}$$

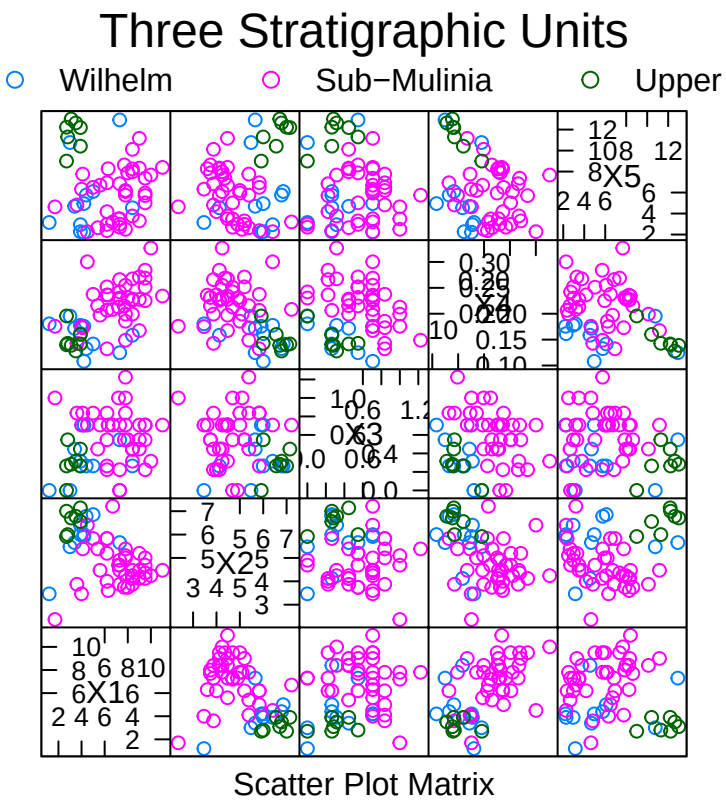
The following dat set gives the values of the five original variables for 56 cases whose populaton assignment was certain.

```
myfile="S:/Workshop/Data/T11-7.DAT";
crude=read.table(myfile);
X1=crude$V1;
X2=sqrt(crude$V2);
X3=sqrt(crude$V3);
X4=1/crude$V4;
X5=crude$V5;
gr=crude$V6;

require(latticeExtra); # for trellis display
```

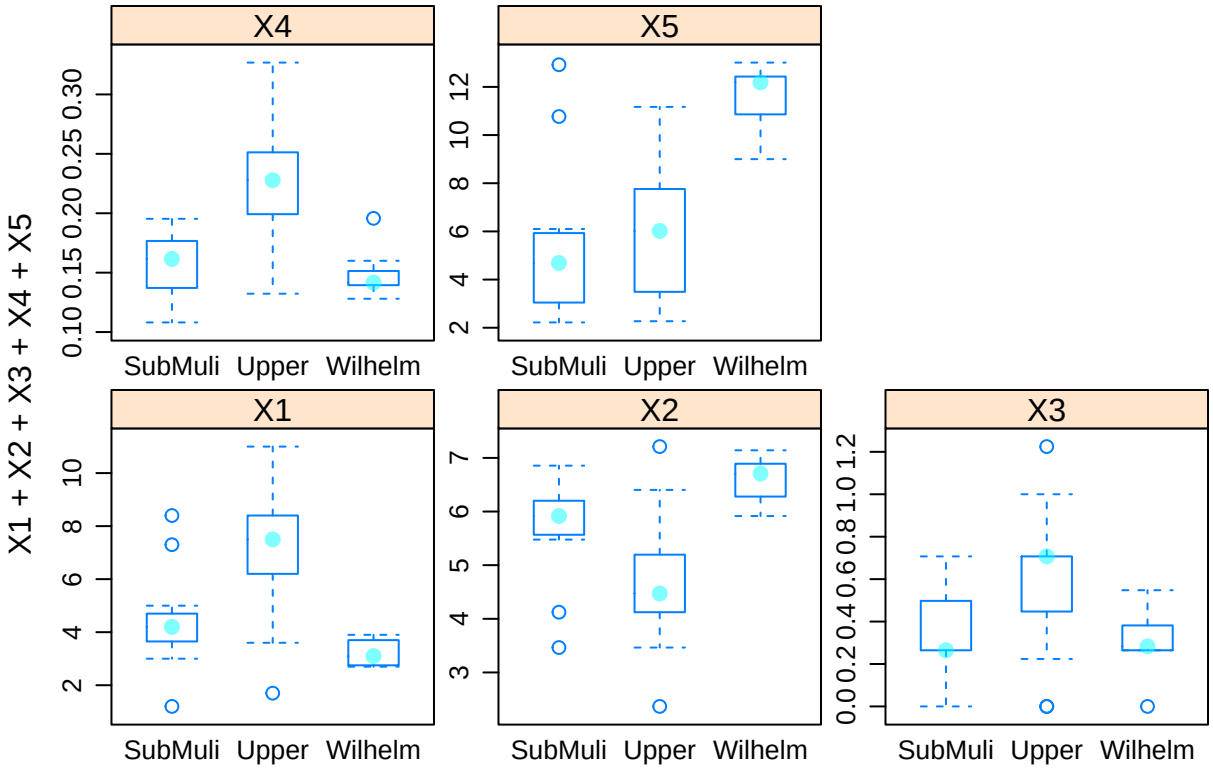
```
## Loading required package: latticeExtra
```

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

```
   # Get trellis graphics parameters form the theme "superpose.symbol"
   # to be used with panel function panel.superpose
   super.sym=trellis.par.get("superpose.symbol");

   # Scatter plot matrix of X1-X5 variables by groups
   splom(~data.frame(X1,X2,X3,X4,X5),
         groups=gr,panel=panel.superpose,
         key=list(title="Three Stratigraphic Units",
         columns=3,
         points=list(pch=super.sym$pch[1:3],col=super.sym$col[1:3]),
         text=list(c("Wilhelm","Sub-Mulinia","Upper"))));
```




```
   # box-whisker plot of X1-X5 by groups
   bwplot(X1+X2+X3+X4+X5~gr,allow.multiple=T,outer=T,pch=16,col="cyan",alpha=0.5,scale="free")
```




```
# Linear discriminant analysis (from MASS library)

library(MASS)
crudeLDA=lda(gr~X1+X2+X3+X4+X5);

# Confusion Matrix: rows=actual, columns=classfied into

table(Actual=(actual=gr),Classified=(classified=predict(crudeLDA)$class));
```

```
##           Classified
## Actual     SubMuli Upper Wilhelm
##   SubMuli        8     2       1
##   Upper          1    37       0
##   Wilhelm        0     0       7
```

```
# Apparent Error Rate (error rate based on train data, underestimating!!!)

mean(miss<-(actual!=classified))  # an interesting exmple of = is not same as <-
```

```
## [1] 0.07142857
```

```
# Posterior probabilities of misclassified objects

p=as.data.frame(round(predict(crudeLDA)$posterior[miss,],4));

# Actual membership

A=as.character(actual[miss]);
```

```r
# Classified membership
C=as.character(classified[miss]);

# Summary of misclassified objects
cbind(p,actual=A,classified=C);
```

```
##    SubMuli  Upper Wilhelm  actual classified
## 11  0.4271 0.5729  0.0000 SubMuli      Upper
## 13  0.2921 0.7079  0.0000 SubMuli      Upper
## 18  0.0117 0.0000  0.9883 SubMuli    Wilhelm
## 51  0.6070 0.3930  0.0000   Upper    SubMuli
```

```r
# Cross-validation confusion matrix based on holdout/Jackknife/Lachenbruch's procedure
table(actual,crudeLDAcv=update(crudeLDA,CV=T)$class);
```

```
##          crudeLDAcv
## actual    SubMuli Upper Wilhelm
##   SubMuli       7     2       2
##   Upper         3    35       0
##   Wilhelm       0     0       7
```

```r
# Expected error rate
crudeLDAcv=update(crudeLDA,CV=T)$class
mean(actual!=crudeLDAcv);
```

```
## [1] 0.125
```

```r
# Confusion matrix when unequal priors assumed
table(actual, predict(update(crudeLDA,prior=c(4,3,3)/10))$class);
```

```
##
## actual    SubMuli Upper Wilhelm
##   SubMuli      10     0       1
##   Upper         3    35       0
##   Wilhelm       0     0       7
```

```r
## Quadratic discriminant analysis
crudeQDA=qda(gr~X1+X2+X3+X4);

# Confusion matrix: row=actual, columns=classified
table(actual=gr,classified=predict(crudeQDA)$class);
```

```
##          classified
## actual    SubMuli Upper Wilhelm
##   SubMuli       9     1       1
##   Upper         1    37       0
##   Wilhelm       0     0       7
```

```r
# Apparent Error Rate
mean(miss<-actual!=classified);
```

```
## [1] 0.07142857
```

```r
# Posterior Probabilities of misclassified objects
p=as.data.frame(round(predict(crudeQDA)$posterior[miss,],4));

# Actural Membership
A=as.character(actual[miss])
```

```
  # Classified Membership
  C=as.character(classified[miss])

  # Summary of missclassified objects
  cbind(p,actal=A,classified=C)
```

```
##    SubMuli  Upper Wilhelm   actal classified
## 11  0.6455 0.3545  0.0000 SubMuli      Upper
## 13  0.2782 0.7216  0.0001 SubMuli      Upper
## 18  0.5844 0.0014  0.4142 SubMuli    Wilhelm
## 51  0.1067 0.8933  0.0000   Upper    SubMuli
```

```
  # Cross validation confusion matrix based on holdout/Jackknife/Lachenbruch's procedure
  crudeQDAcv=table(actual=gr,classified=update(crudeQDA,CV=T)$class)

  # Expected AER
  mean(gr!=classified)
```

```
## [1] 0.07142857
```

Example 7.5 (Calculating Fisher's Sample Discriminants for Three Populations) Consider the observations on $p = 2$ variables from $g = 3$ populations given below. Assuming that the populations have a common covariance matrix Σ.

```
myData=data.frame(X1=c(-2,0,-1,0,2,1,1,0,-1),X2=c(5,3,1,6,4,2,-2,0,-4),
                  gr=rep(c("pop1","pop2","pop3"),c(3,3,3)));

# Variance and Covariance matrices of the three groups.
# subset: logical expression indicating elements or rows to keep, missing values are taken as false
# select: expression indicating columns to select from a data frame.
S1=cov(subset(myData,subset=gr=="pop1",select=-gr))
S2=cov(subset(myData,subset=gr=="pop2",select=-gr))
S3=cov(subset(myData,subset=gr=="pop3",select=-gr))

# Degrees of freedom for the 3 groups
dof=table(myData$gr)-1;

# Within Groups

W=dof[1]*S1+dof[2]*S2+dof[3]*S3

# Between Groups

m1=colMeans(subset(myData,subset=gr=="pop1",select=-gr))
m2=colMeans(subset(myData,subset=gr=="pop2",select=-gr))
m3=colMeans(subset(myData,subset=gr=="pop3",select=-gr))
mu=colMeans(myData[,1:2])
B=dof[1]*cov(rbind(m1,m2,m3))

# Eigenvectors of W^(-1)B

myeigen=eigen(solve(W)%*%B)
e1=myeigen$vectors[,1]
e2=myeigen$vectors[,2]
```

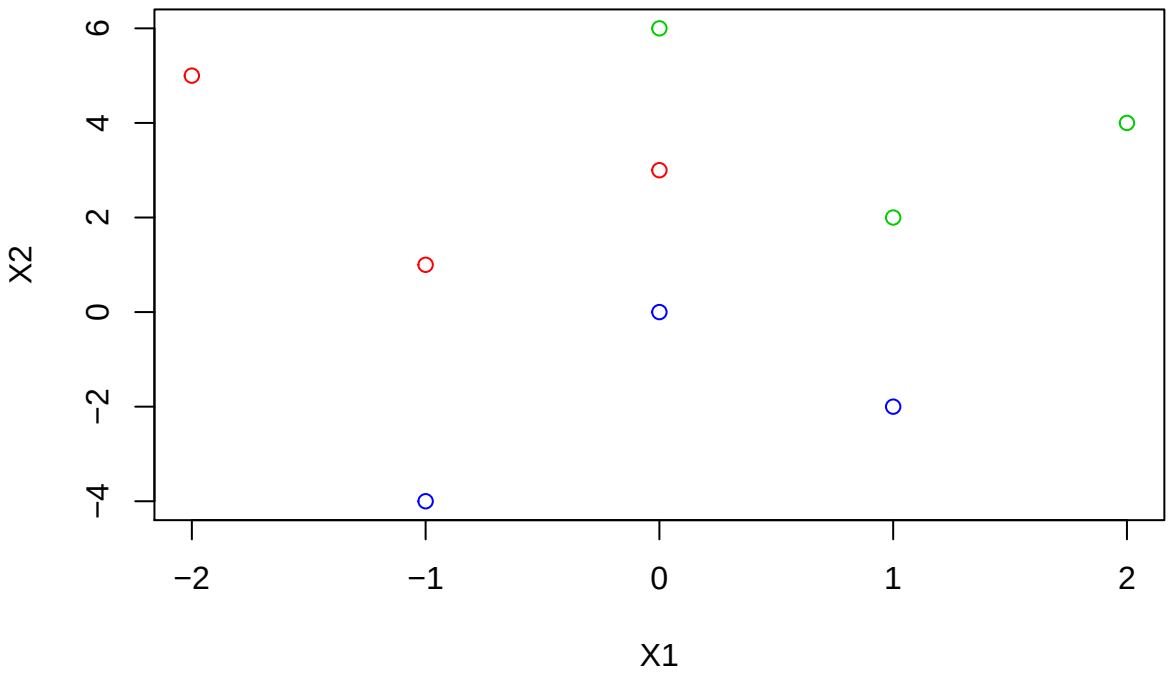
```r
# Normalized eigenvectors, Fisher's Discriminant
S=W/sum(dof);
a1=-e1/c(sqrt(t(e1)%*%S%*%e1))
a2=-e2/c(sqrt(t(e2)%*%S%*%e2))

# Classifying a new observation with Fisher's Discriminant

x0=c(1,3)
coeA=rbind(a1,a2);
yhat=coeA%*%x0;

by1=coeA%*%m1;
by2=coeA%*%m2;
by3=coeA%*%m3;

D1=sum((yhat-by1)^2)
D2=sum((yhat-by2)^2)
D3=sum((yhat-by3)^2)

# minimum distance method

# Squared Mahalanobis distances of all objects to the 3 means
d1=mahalanobis(myData[,-3],m1,S)
d2=mahalanobis(myData[,-3],m2,S)
d3=mahalanobis(myData[,-3],m3,S)

# Place thses distances into a data frame
d=data.frame(pop1=d1,pop2=d2,pop3=d3)

# Check ties
any(apply(d,1,function(x){length(unique(x))<3}))
```

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

```r
# Add classified result
d=cbind(d,classified=names(d)[apply(d,1,which.min)])

# Add actual identities
d=cbind(d,actual=myData$gr)

# Report missclassified objects
subset(d,subset=classified!=actual)
```

```
## [1] pop1       pop2       pop3       classified actual
## <0 rows> (or 0-length row.names)
```

```r
# 3-D rotatable plot may give insight as what variables to use in discriminant and classification
# analysis. Require rgl package

library(rgl)
plot(myData[1:2],col=c("red","green3","blue")[as.numeric(myData$gr)])
```



3. Classification via Logistic Regression

Example 7.6 We re-analyze the salmon data.

```
myfile="S:/Workshop/Data/T11-2.DAT";
salmon=read.table(myfile)
type=salmon$V1-1;
gender=salmon$V2;
x1=salmon$V3;
x2=salmon$V4;

# Logistic Regression
myreg=glm(type~gender+x1+x2,family=binomial(link='logit'))

# Model fit: Analyze the table of deviance

anova(myreg,test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: type
##
## Terms added sequentially (first to last)
```

```
##
##
##        Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                     99    138.629
## gender  1    0.000        98    138.629        1
## x1      1   80.793        97     57.837  < 2e-16 ***
## x2      1   19.163        96     38.674  1.2e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
  pred=ifelse(myreg$fitted>0.5,1,0)

  # Confusion Matrix

  table(type,pred)
```

```
##     pred
## type  0  1
##    0 46  4
##    1  3 47
```

```
  # Apparent Error Rate

  mean(type!=pred)
```

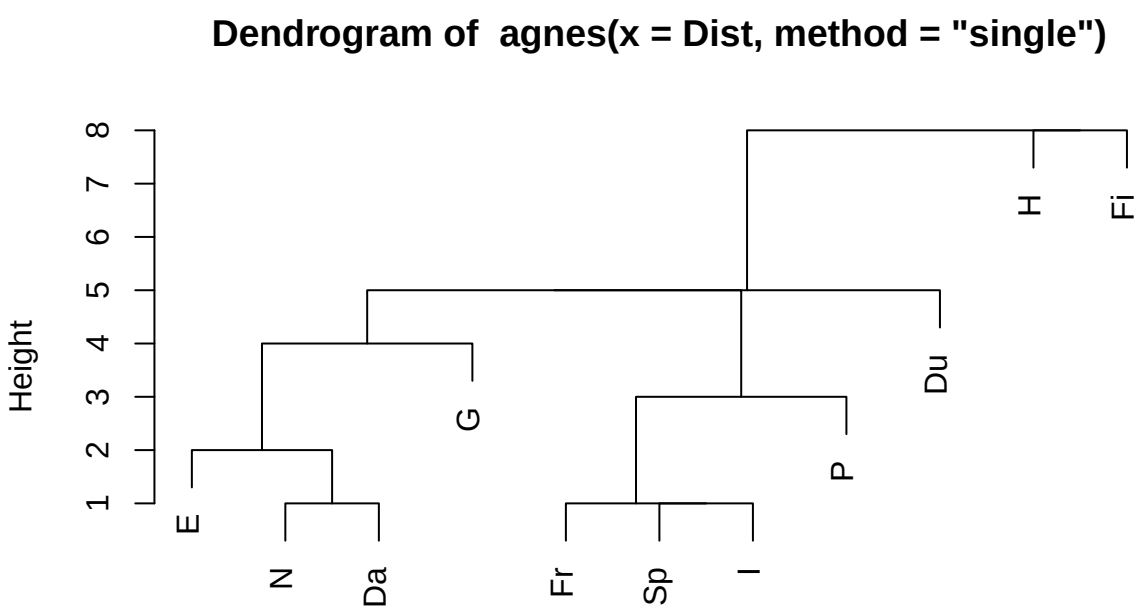
```
## [1] 0.07
```

4. Agglomerative Hierarchical Clustering: Linkage

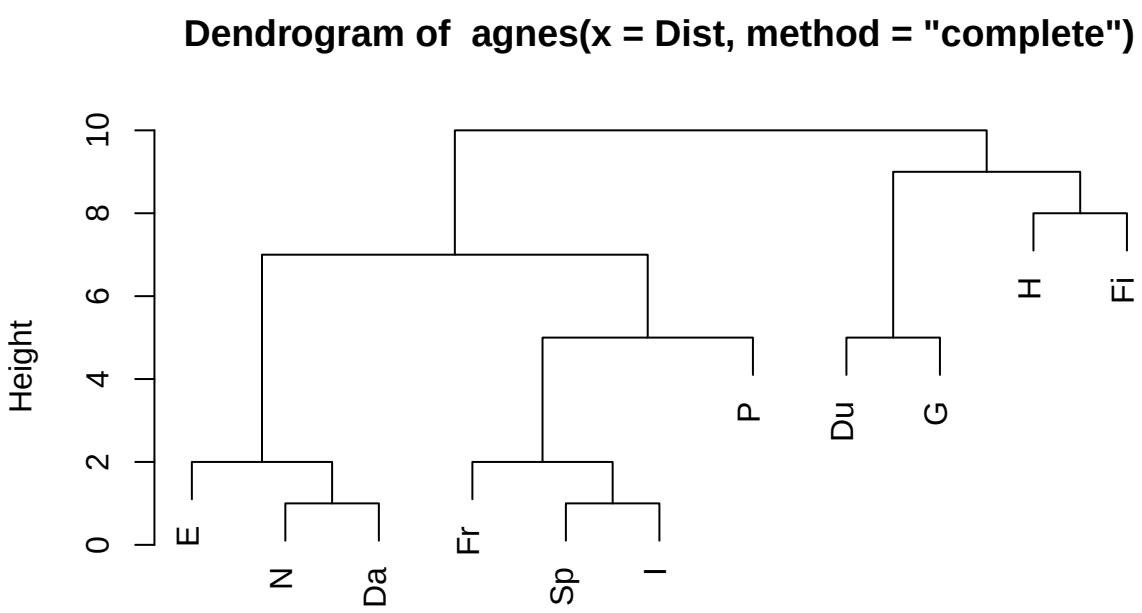
Example 7.7 Single linkage clustering of 11 languages

R-Codes

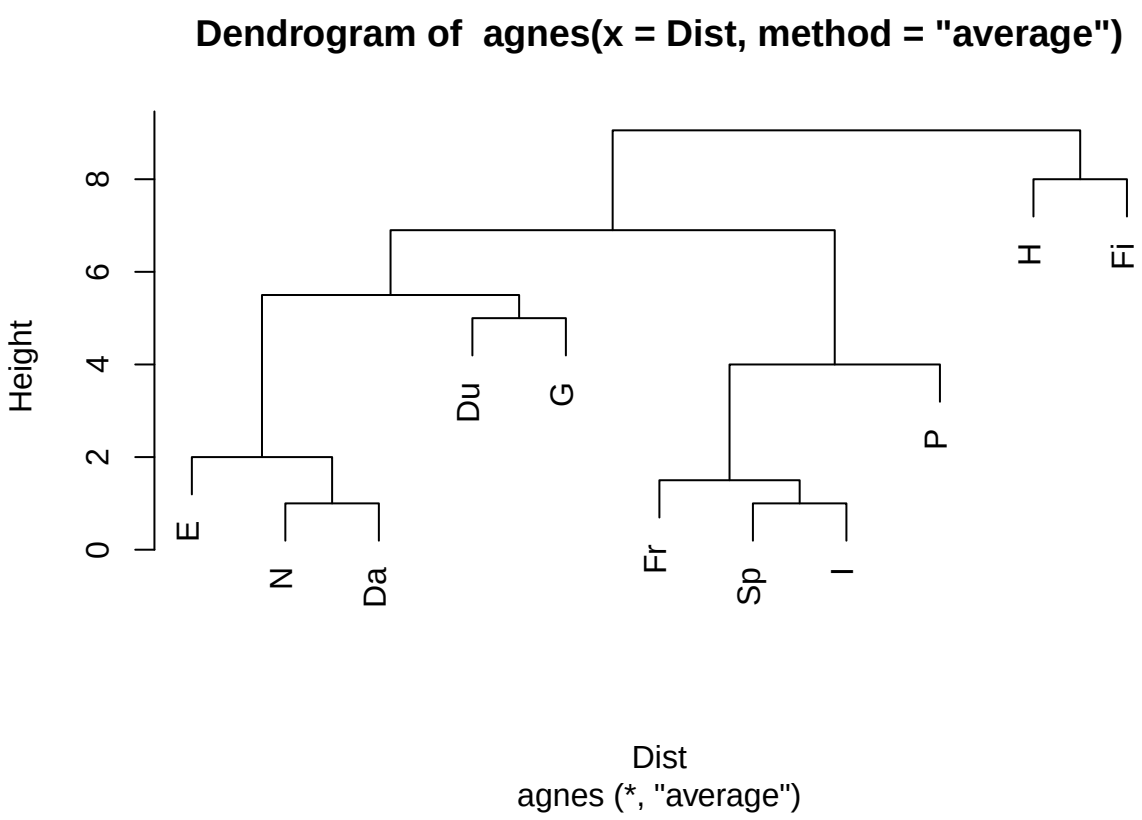
```
  library("cluster")
  #enter similarities and convert to distances using dist
  lang=read.table("C:/Users/weixi/Dropbox/Teaching/Stat730/References/RforExamples/R for Examples/E12-4
  m10=matrix(10,nrow=11, ncol=11)
  Dist=as.dist(m10-lang)
  clangS=agnes(Dist,method="single")  # Agglomerative Nesting
                                      # method options: average, single, complete, ward.
  pltree(clangS,labels=c("E","N","Da","Du","G","Fr","Sp","I","P","H","Fi") )
```




```
clangC=agnes(Dist,method="complete")  # Agglomerative Nesting
                                      # method options: average, single, complete, ward.
pltree(clangC,labels=c("E","N","Da","Du","G","Fr","Sp","I","P","H","Fi") )
```

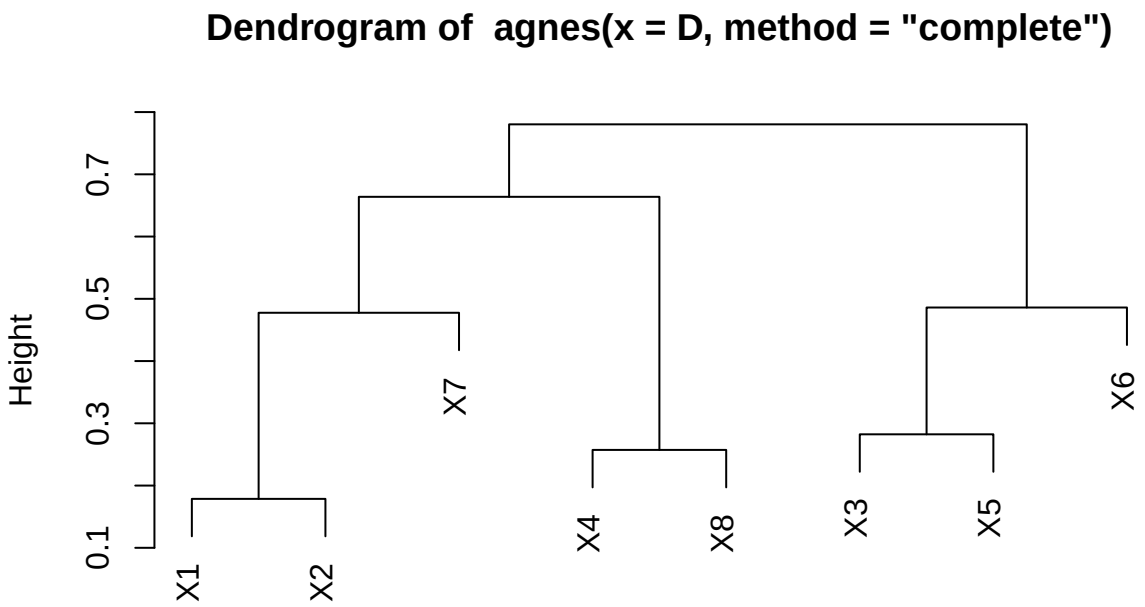



```
clangA=agnes(Dist,method="average")  # Agglomerative Nesting
                                     # method options: average, single, complete, ward.
pltree(clangA,labels=c("E","N","Da","Du","G","Fr","Sp","I","P","H","Fi") )
```

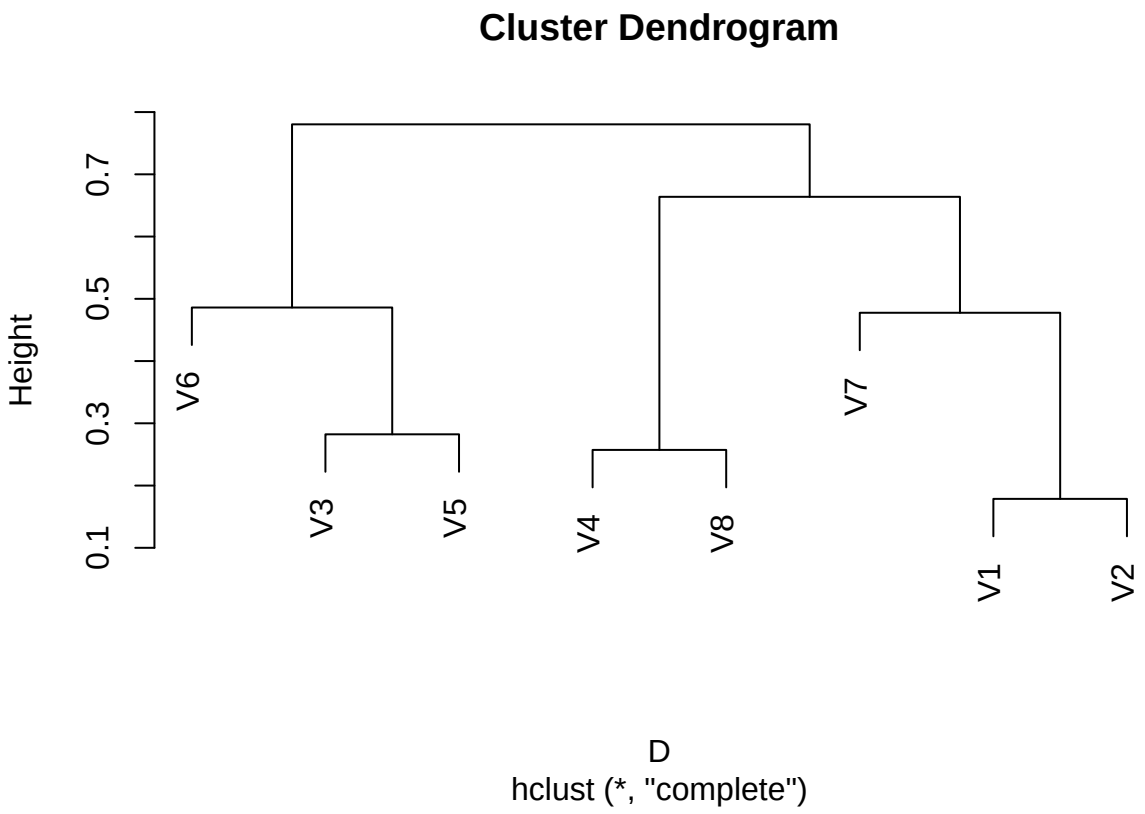


Example 7.8 Clustering variables using complete linkage.

```
library("cluster")
utility=read.table("S:/Workshop/Data/T12-4.DAT")
Rcor= cor(utility[,1:8])
D=as.dist((1-Rcor)/2)

#similarities need to used---not distances
## Use correlations between variables "as distance"
clangC1=agnes(D,method="complete")  # Agglomerative Nesting
                                    # method options: average, single, complete, ward.
pltree(clangC1,labels=paste("X",1:8, sep=""))
```




```r
# Using hclust function

D=as.dist((1-Rcor)/2)
hC= hclust(D,method="complete")
plot(hC)
```

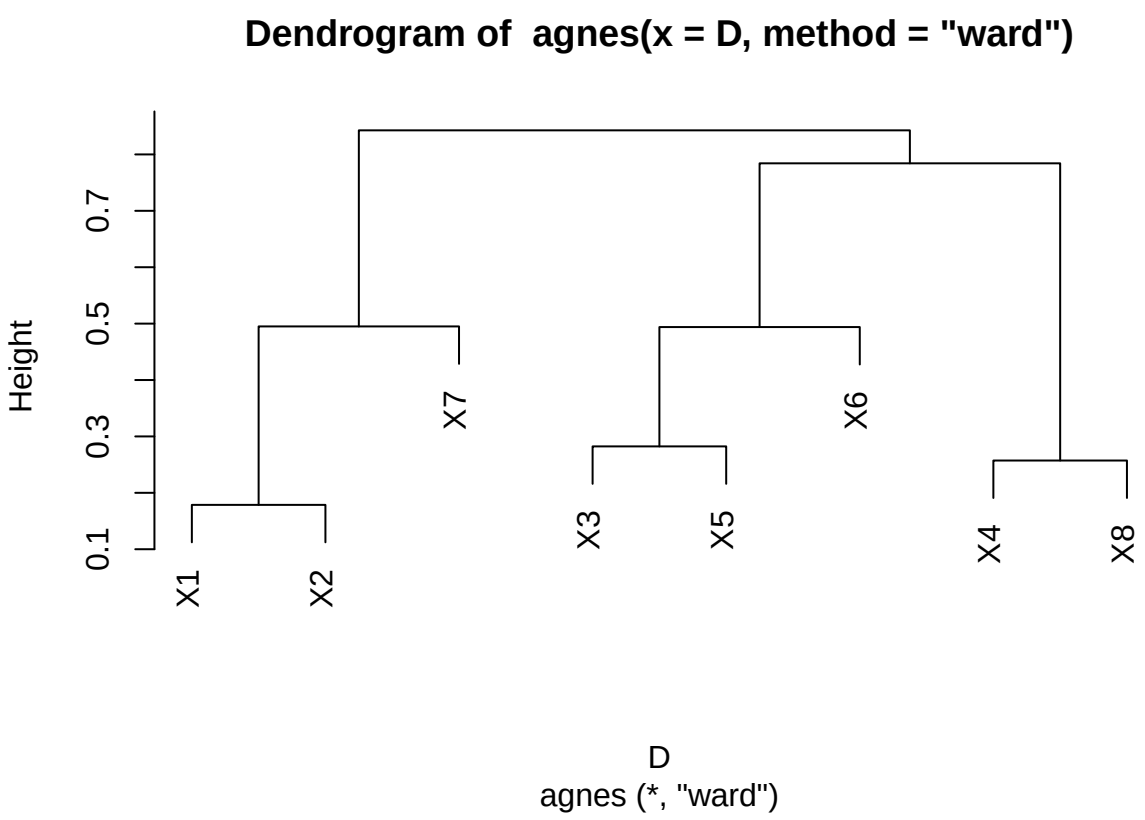


5. Agglomerative Hierarchical Clustering: Ward's Procedure

Example 7.9 Clustering pure malt scotch whiskies.

R-Codes

```
library("cluster")
utility=read.table("S:/Workshop/Data/T12-4.DAT")
Rcor= cor(utility[,1:8])
D=as.dist((1-Rcor)/2)

#similarities need to used---not distances
## Use correlations between variables "as distance"
clangC1=agnes(D,method="ward")  # Agglomerative Nesting
                                  # method options: average, single, complete, ward.
pltree(clangC1,labels=paste("X",1:8, sep=""))
```



6. Non-Hierarchical Clustering: K-means method

Example 7.10 Clustering variables using k-means method

```
library("cluster")
utility=read.table("S:/Workshop/Data/T12-4.DAT")
dat=utility[,1:8];
fit=kmeans(dat,4); # 4 cluster solution
aggregate(dat, by=list(fit$cluster),FUN=mean) # get cluster means
```

```
##   Group.1       V1       V2       V3       V4       V5        V6        V7
## 1       1 1.066667 10.42222 174.3333 59.70000 3.466667  6663.556 19.333333
## 2       2 1.190000  9.55000 197.0000 54.55000 1.600000  4194.000 20.450000
## 3       3 1.042500  9.57500 193.5000 54.62500 4.225000 15005.250  0.000000
## 4       4 1.194286 12.14286 137.5714 55.51429 2.857143  9675.429  7.014286
##          V8
## 1 1.2818889
## 2 1.7995000
## 3 0.5832500
## 4 0.9701429
```

```
myutility=data.frame(utility,fit$cluster)
```

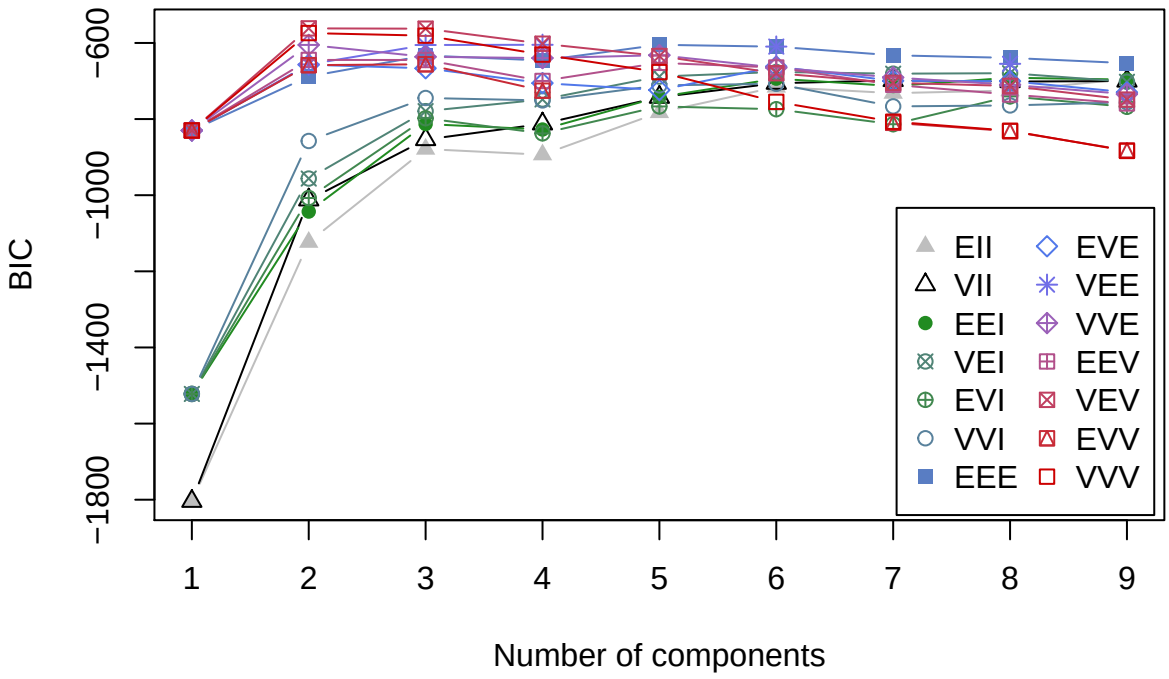
7. Mixture Modeling

Example 7.11 Consider the Iris data. Fit a $p = 4$ dimensional normal mixture model restricting the covariance matrices to satisfy $\Sigma_k = \eta_k I, k = 1, 2, 3$.

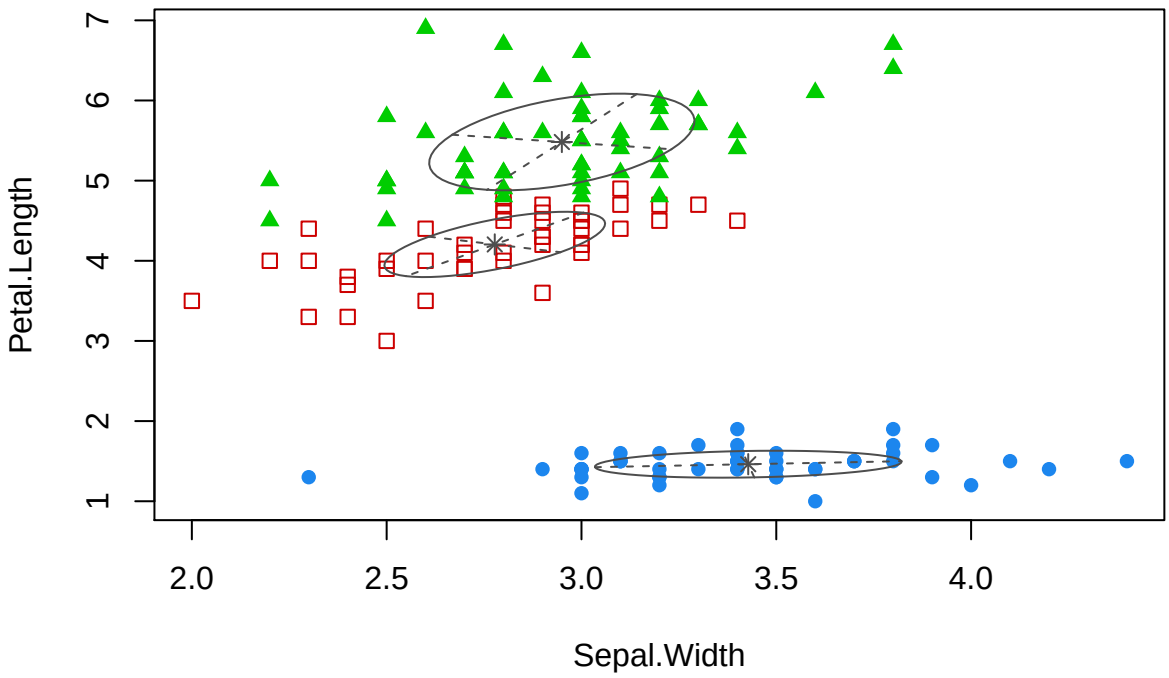
```
attach(iris)
library(mclust)
```

```
## Package 'mclust' version 5.4
## Type 'citation("mclust")' for citing this R package in publications.
```

```
# Clustering using Mclust function
irisModel=Mclust(iris[,-5])
# BIC plots for different types of covariance matrices structures.
plot(irisModel,iris[,-5],what="BIC")
```




```
# Clustering by specifying 3 clusters
irisModel=Mclust(iris[,-5],G=3)
# plot the culsters
coordProj(iris[,-5],dimens=c(2,3),what="classification",classification=irisModel$classification,
          parameters=irisModel$parameters)
```




```
# probability of membership
irisModel$z
```

```
##                [,1]         [,2]         [,3]
##   [1,]  1.000000e+00 4.916819e-40 3.345948e-29
##   [2,]  1.000000e+00 5.846760e-29 1.599452e-23
##   [3,]  1.000000e+00 2.486168e-33 2.724243e-25
##   [4,]  1.000000e+00 2.050996e-28 2.180764e-22
##   [5,]  1.000000e+00 8.283066e-42 8.710458e-30
##   [6,]  1.000000e+00 2.118466e-41 1.791499e-29
##   [7,]  1.000000e+00 2.272415e-33 1.488306e-24
##   [8,]  1.000000e+00 5.627283e-36 6.810311e-27
##   [9,]  1.000000e+00 2.191516e-25 1.168399e-20
##  [10,]  1.000000e+00 4.280141e-31 3.858191e-24
##  [11,]  1.000000e+00 2.228290e-44 1.310754e-31
##  [12,]  1.000000e+00 1.484405e-33 5.116424e-25
##  [13,]  1.000000e+00 3.370422e-30 8.670710e-24
##  [14,]  1.000000e+00 1.466098e-31 1.653532e-23
##  [15,]  1.000000e+00 2.147653e-58 5.434571e-39
##  [16,]  1.000000e+00 1.801107e-59 7.697554e-39
##  [17,]  1.000000e+00 6.150723e-48 5.272890e-33
##  [18,]  1.000000e+00 4.703045e-38 3.742860e-28
##  [19,]  1.000000e+00 1.825528e-42 1.183789e-30
##  [20,]  1.000000e+00 3.247181e-43 2.409369e-30
##  [21,]  1.000000e+00 5.018328e-34 6.502112e-26
##  [22,]  1.000000e+00 1.156985e-38 7.313433e-28
##  [23,]  1.000000e+00 2.353800e-44 1.549600e-29
```

```
##  [24,]  1.000000e+00 4.079464e-25 1.983593e-20
##  [25,]  1.000000e+00 2.284469e-28 2.986694e-21
##  [26,]  1.000000e+00 1.921336e-26 6.682767e-22
##  [27,]  1.000000e+00 2.046362e-30 1.406767e-23
##  [28,]  1.000000e+00 6.094074e-39 1.100330e-28
##  [29,]  1.000000e+00 3.481680e-38 1.641146e-28
##  [30,]  1.000000e+00 3.118168e-29 9.903246e-23
##  [31,]  1.000000e+00 9.405036e-28 3.111619e-22
##  [32,]  1.000000e+00 1.602197e-33 3.939744e-26
##  [33,]  1.000000e+00 2.013315e-55 4.486532e-36
##  [34,]  1.000000e+00 2.092088e-59 7.086472e-39
##  [35,]  1.000000e+00 1.543479e-29 1.364608e-23
##  [36,]  1.000000e+00 1.007083e-35 3.697238e-27
##  [37,]  1.000000e+00 7.608891e-43 2.806438e-31
##  [38,]  1.000000e+00 3.328870e-43 4.377978e-30
##  [39,]  1.000000e+00 3.032036e-28 4.037820e-22
##  [40,]  1.000000e+00 2.299563e-36 2.845053e-27
##  [41,]  1.000000e+00 6.169023e-39 2.213949e-28
##  [42,]  1.000000e+00 4.335053e-16 2.833331e-16
##  [43,]  1.000000e+00 8.596140e-32 1.149420e-23
##  [44,]  1.000000e+00 1.887810e-27 1.360444e-20
##  [45,]  1.000000e+00 9.633551e-34 2.280518e-24
##  [46,]  1.000000e+00 6.480651e-27 3.325717e-22
##  [47,]  1.000000e+00 1.032431e-43 1.863733e-30
##  [48,]  1.000000e+00 1.339992e-31 4.420889e-24
##  [49,]  1.000000e+00 5.633662e-44 2.981771e-31
##  [50,]  1.000000e+00 2.638259e-35 8.924242e-27
##  [51,]  4.164351e-93 9.974365e-01 2.563472e-03
##  [52,]  5.911972e-86 9.949442e-01 5.055808e-03
##  [53,] 5.285769e-106 9.761907e-01 2.380931e-02
##  [54,]  1.235585e-65 9.444404e-01 5.555959e-02
##  [55,]  8.460223e-94 9.607909e-01 3.920909e-02
##  [56,]  1.243576e-82 9.726051e-01 2.739491e-02
##  [57,]  3.099355e-97 9.718710e-01 2.812895e-02
##  [58,]  2.772674e-35 9.991939e-01 8.060884e-04
##  [59,]  7.478128e-88 9.954658e-01 4.534204e-03
##  [60,]  1.496232e-63 9.591811e-01 4.081894e-02
##  [61,]  5.612898e-43 9.934148e-01 6.585219e-03
##  [62,]  2.945767e-75 9.875515e-01 1.244853e-02
##  [63,]  4.064047e-61 9.968534e-01 3.146570e-03
##  [64,]  9.175764e-94 9.572139e-01 4.278608e-02
##  [65,]  2.337741e-48 9.986804e-01 1.319553e-03
##  [66,]  3.956053e-80 9.989061e-01 1.093889e-03
##  [67,]  9.311070e-88 9.213443e-01 7.865569e-02
##  [68,]  4.282264e-60 9.985955e-01 1.404483e-03
##  [69,]  4.402578e-93 6.793960e-02 9.320604e-01
##  [70,]  3.108474e-56 9.987522e-01 1.247756e-03
##  [71,] 4.551440e-111 1.244538e-01 8.755462e-01
##  [72,]  3.015684e-63 9.993093e-01 6.906517e-04
##  [73,] 2.021128e-109 5.434374e-02 9.456563e-01
##  [74,]  3.184465e-89 9.605724e-01 3.942760e-02
##  [75,]  1.422141e-74 9.989884e-01 1.011609e-03
##  [76,]  6.178720e-81 9.984106e-01 1.589391e-03
##  [77,] 4.517743e-101 9.595442e-01 4.045584e-02
```

```
##  [78,] 1.631797e-117 4.068849e-01 5.931151e-01
##  [79,]  4.629859e-88 9.555906e-01 4.440938e-02
##  [80,]  7.538495e-40 9.998911e-01 1.089491e-04
##  [81,]  2.680928e-53 9.985677e-01 1.432310e-03
##  [82,]  9.830603e-48 9.994317e-01 5.683231e-04
##  [83,]  1.978709e-57 9.993721e-01 6.278961e-04
##  [84,] 6.708053e-121 7.177998e-03 9.928220e-01
##  [85,]  3.584021e-88 8.613277e-01 1.386723e-01
##  [86,]  5.501060e-88 9.741119e-01 2.588807e-02
##  [87,]  3.563033e-96 9.904169e-01 9.583076e-03
##  [88,]  4.571674e-83 9.132369e-01 8.676306e-02
##  [89,]  2.256417e-65 9.976954e-01 2.304607e-03
##  [90,]  9.473163e-65 9.856135e-01 1.438654e-02
##  [91,]  7.490910e-77 9.508626e-01 4.913736e-02
##  [92,]  1.158125e-88 9.872020e-01 1.279805e-02
##  [93,]  8.311457e-62 9.986387e-01 1.361269e-03
##  [94,]  8.839759e-36 9.993348e-01 6.651923e-04
##  [95,]  3.806304e-71 9.900557e-01 9.944262e-03
##  [96,]  2.778790e-66 9.984856e-01 1.514417e-03
##  [97,]  7.195258e-70 9.967909e-01 3.209139e-03
##  [98,]  3.613432e-74 9.984704e-01 1.529567e-03
##  [99,]  8.960657e-29 9.993054e-01 6.946056e-04
## [100,]  1.017088e-66 9.970471e-01 2.952912e-03
## [101,] 1.433666e-213 3.365991e-13 1.000000e+00
## [102,] 3.798224e-134 4.985483e-06 9.999950e-01
## [103,] 7.112294e-186 9.792479e-08 9.999999e-01
## [104,] 1.753986e-154 4.229595e-05 9.999577e-01
## [105,] 9.455979e-185 1.847738e-09 1.000000e+00
## [106,] 8.075517e-234 1.833146e-11 1.000000e+00
## [107,] 5.848745e-101 9.503826e-05 9.999050e-01
## [108,] 3.888990e-201 2.532701e-08 1.000000e+00
## [109,] 7.933266e-171 1.128283e-08 1.000000e+00
## [110,] 5.947311e-215 5.980370e-10 1.000000e+00
## [111,] 3.455211e-134 1.911209e-03 9.980888e-01
## [112,] 5.417513e-144 9.189437e-06 9.999908e-01
## [113,] 3.462539e-162 2.039138e-06 9.999980e-01
## [114,] 8.149080e-136 1.307782e-08 1.000000e+00
## [115,] 2.661731e-163 1.066229e-13 1.000000e+00
## [116,] 6.727807e-162 2.557241e-08 1.000000e+00
## [117,] 6.109996e-148 9.281696e-04 9.990718e-01
## [118,] 7.488963e-236 5.690720e-07 9.999994e-01
## [119,] 5.538026e-270 4.898595e-20 1.000000e+00
## [120,] 1.713701e-116 8.304031e-05 9.999170e-01
## [121,] 1.022015e-182 1.633728e-08 1.000000e+00
## [122,] 8.911794e-130 1.546635e-06 9.999985e-01
## [123,] 4.270567e-240 3.503086e-13 1.000000e+00
## [124,] 4.459420e-119 2.704888e-03 9.972951e-01
## [125,] 9.326805e-171 1.626337e-05 9.999837e-01
## [126,] 3.012402e-177 1.891484e-04 9.998109e-01
## [127,] 1.667730e-113 1.153151e-02 9.884685e-01
## [128,] 8.827274e-117 3.169089e-02 9.683091e-01
## [129,] 1.752611e-169 7.275275e-09 1.000000e+00
## [130,] 1.793903e-160 1.153537e-03 9.988465e-01
## [131,] 3.374587e-193 3.604260e-08 1.000000e+00
```

```
## [132,] 2.886010e-206 3.806074e-04 9.996194e-01
## [133,] 6.264799e-175 2.587063e-10 1.000000e+00
## [134,] 8.519286e-117 1.608950e-01 8.391050e-01
## [135,] 2.580359e-143 2.016170e-05 9.999798e-01
## [136,] 2.689585e-210 1.821532e-10 1.000000e+00
## [137,] 2.617508e-183 1.213626e-09 1.000000e+00
## [138,] 5.527209e-147 2.416771e-03 9.975832e-01
## [139,] 3.071356e-112 3.810761e-02 9.618924e-01
## [140,] 8.434848e-156 2.326481e-05 9.999767e-01
## [141,] 2.703950e-184 7.298620e-11 1.000000e+00
## [142,] 1.150487e-151 1.211823e-07 9.999999e-01
## [143,] 3.798224e-134 4.985483e-06 9.999950e-01
## [144,] 1.805088e-194 2.242421e-09 1.000000e+00
## [145,] 6.535933e-195 1.309451e-11 1.000000e+00
## [146,] 1.071971e-157 1.010247e-08 1.000000e+00
## [147,] 3.478192e-130 5.551018e-06 9.999944e-01
## [148,] 5.198716e-141 1.264816e-04 9.998735e-01
## [149,] 1.833272e-166 9.268658e-08 9.999999e-01
## [150,] 4.715422e-127 4.486829e-03 9.955132e-01
```