lab10

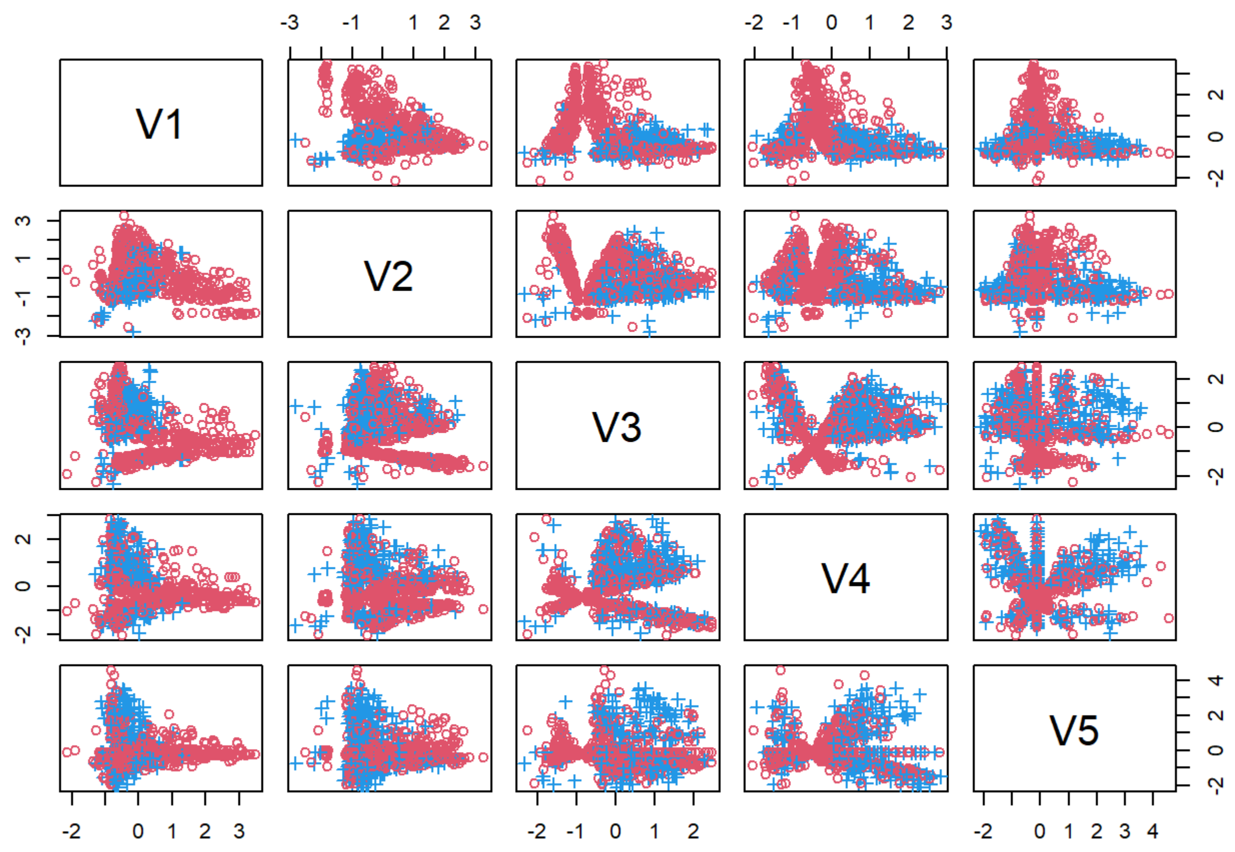
Yixuan Li, UPI:yil845

17 October 2021

The Data: The source of the data we use in this lab is from Open ML, Phoneme data set.

Import data

```
phoneme = read.csv("phoneme.csv", stringsAsFactors=TRUE)
with(phoneme, pairs(phoneme[,-6], col=c(2,4)[Class], pch=c(1,3)[Class]))
```



```
library(mclust)
```

```
## Warning: package 'mclust' was built under R version 4.0.5
```

```
## Package 'mclust' version 5.4.7
## Type 'citation("mclust")' for citing this R package in publications.
```

```
knitr::opts_chunk$set(echo = TRUE)
```

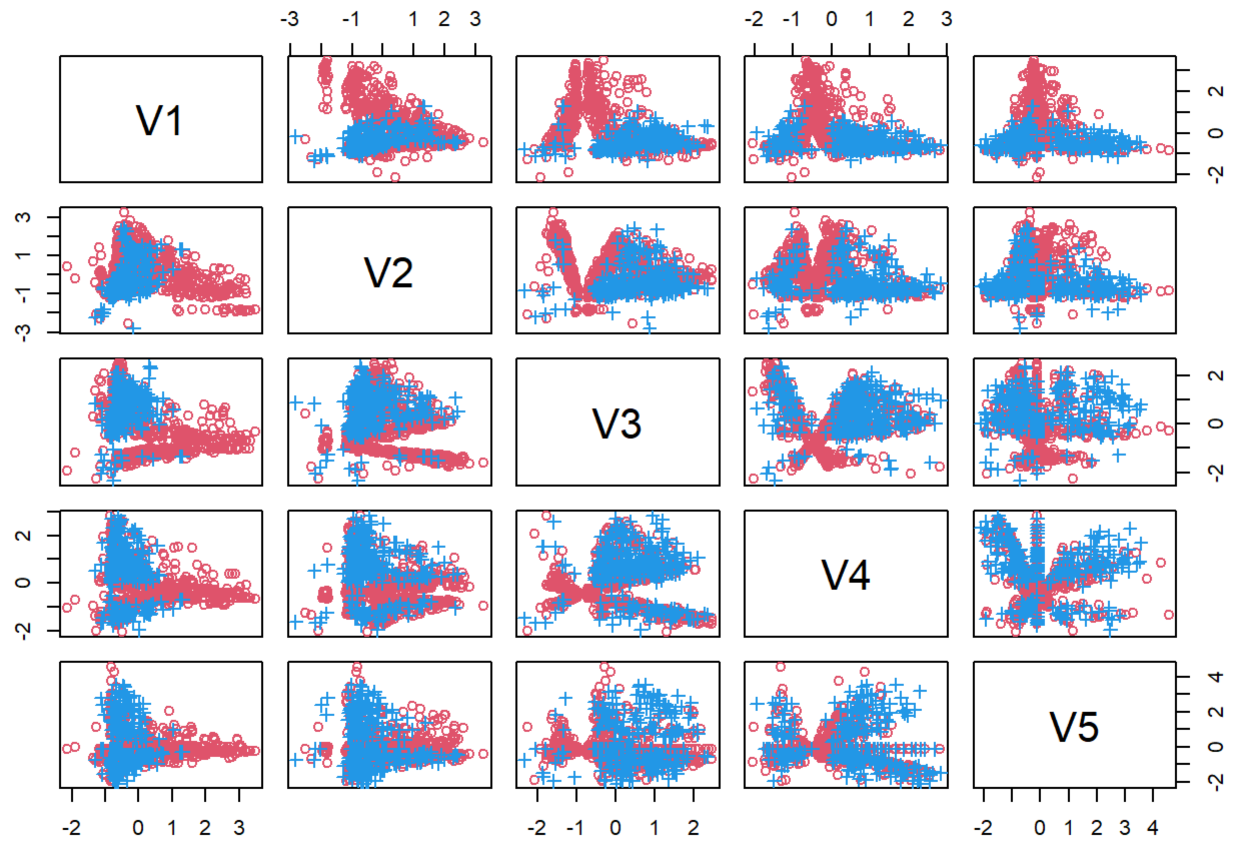
Visualisation

1, The data was sorted according to class label, the majority is in the front, thus is plotted firstly by pairs(), the minority class is plotted secondly, therefore on top.

```
table(as.numeric(factor(phoneme$Class)))                          ## majority is class 1 -Nasal
```

```
##
##   1   2
## 709 291
```

```
phoneme<- phoneme[order(as.numeric(factor(phoneme$Class))),]     ## sort data with class number o
f column Class
with(phoneme, pairs(phoneme[,-6], col=c(2,4)[Class], pch=c(1,3)[Class]))
```



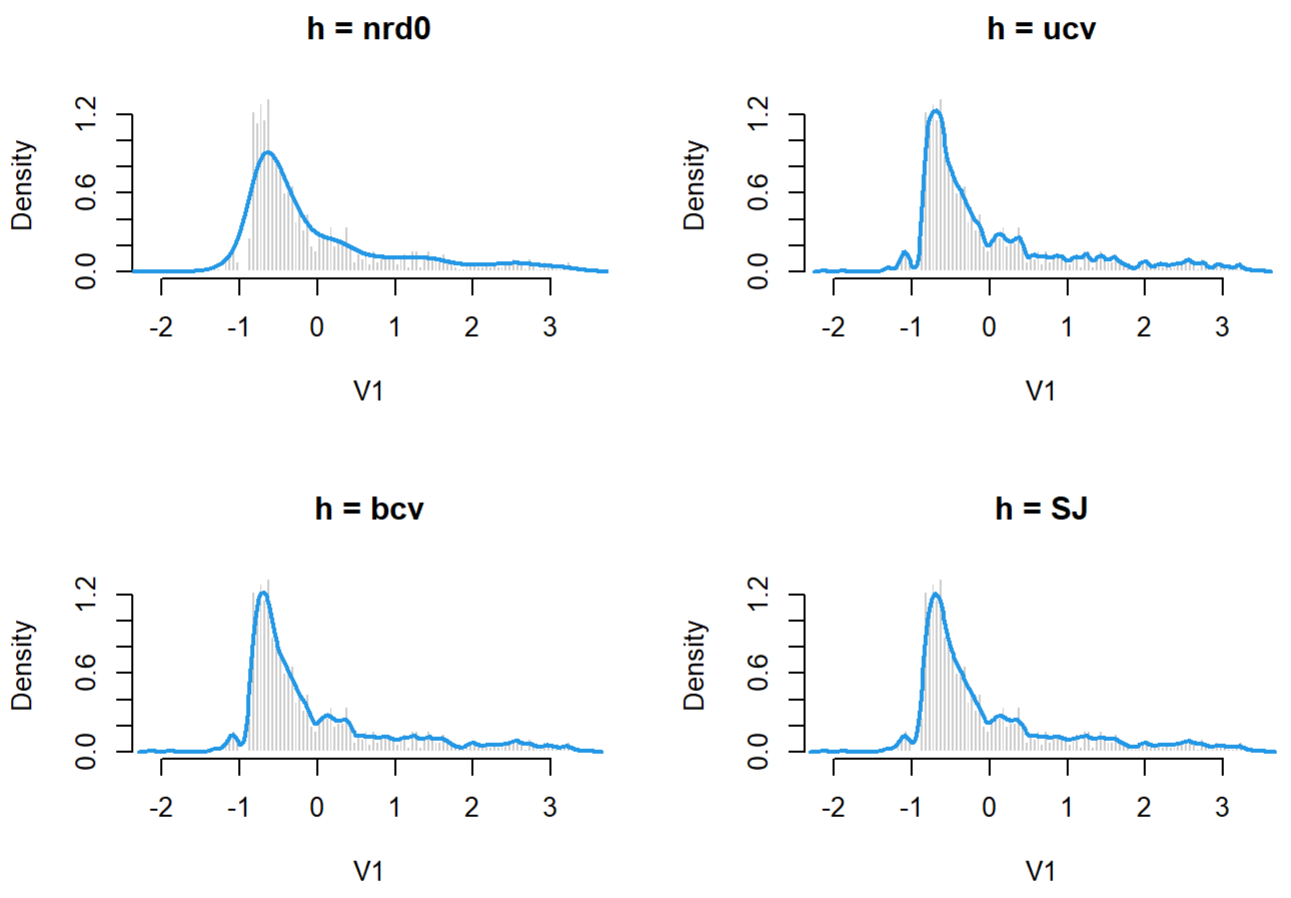
Univariate Density Estimation

2, Plot each of the kernel density estimate, by superimposing it on a histogram (with breaks=100), for V1, with the bandwidth values chosen by methods nrd0, ucv, bcv and SJ, respectively.

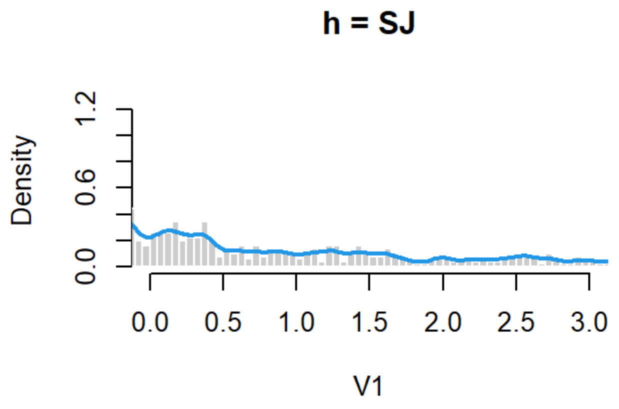
The nrd0 is underfitting and over-smoothing. Ucv, bcv and SJ methods are all somewhat overtting, esp at high range.

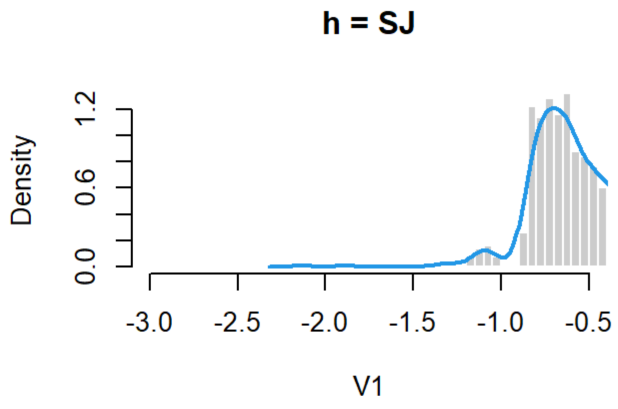
from teacher -Overfitting and underfitting can occur in different regions. Let’s take SJ as an example (which is usually considered as a good default bandwidth selector). It is underfitted/oversmoothed around V1 =−1, but at the same time is overfitted/undersmoothed for V1 ≥1, as shown in the next two plots. The reason is that different regions require different bandwidth values for correct smoothing, but KDE uses a common one everywhere.

```
par(mfrow=c(2,2))
for(bw in c("nrd0", "ucv", "bcv", "SJ")) {
   hist(phoneme$V1, freq=FALSE, breaks=100,
        col="gray80", border="white",
        main=paste0("h = ", bw), xlab="V1")
   lines(density(phoneme$V1, bw=bw), col=4, lwd=2)
}
```



```
## lower limit
hist(phoneme$V1, freq=FALSE, breaks=100,
        col="gray80", border="white",
        main=paste0("h = ", "SJ"),xlim=c(0,3), xlab="V1")
   lines(density(phoneme$V1, bw="SJ"), col=4, lwd=2)
##upper limit
hist(phoneme$V1, freq=FALSE, breaks=100,
        col="gray80", border="white",
        main=paste0("h = ", "SJ"),xlim=c(-3,-0.5), xlab="V1")
   lines(density(phoneme$V1, bw="SJ"), col=4, lwd=2)
```

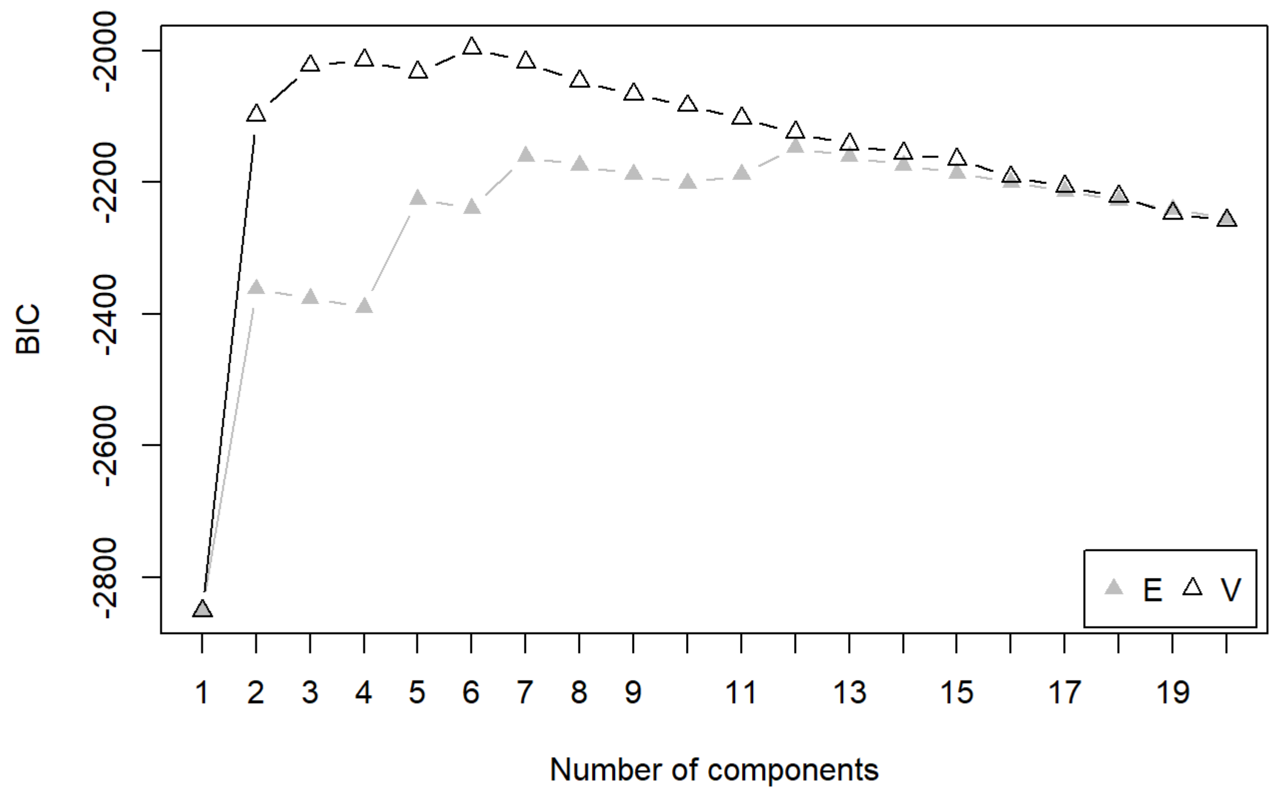




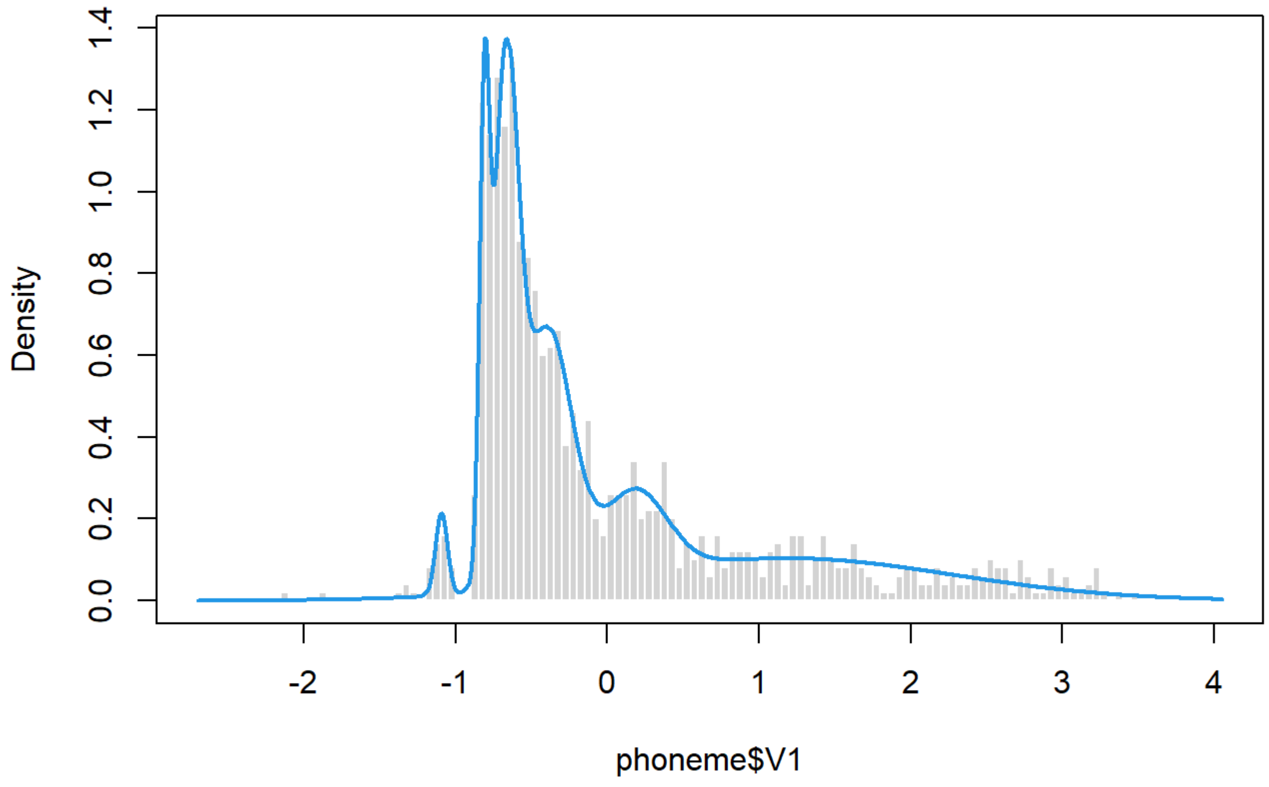
3, Find BIC selected the best normal mixture fit to V1 in both equal and varying variance subfamilies, with the number of components ranging from 1 to 20.Plot the fitted density, along with a histogram of the data.

BIC values of varying variance is bigger than that of equal variance density estimates,indicating varying variance fitting is better. The plot looks well-fitted, better than the results of KDEs.(teacher)It has much less underfitting or overfitting and is a better fit than any of the above KDEs, as can also be seen in the plots below. This is because a mixture with varying variances is more adaptive to the data everywhere.

```
r<-densityMclust(phoneme$V1,G=1:20)
#r$parameters
plot(r, what="BIC")
```



```
### the fitted density with 6 components - maximum BIC for model Varying variance
r6 = densityMclust(phoneme$V1, G=6, modelNames="V")  # 6 components, varying variance
plot(r6, phoneme$V1, "density", breaks=100, lwd=2, col=4)
```



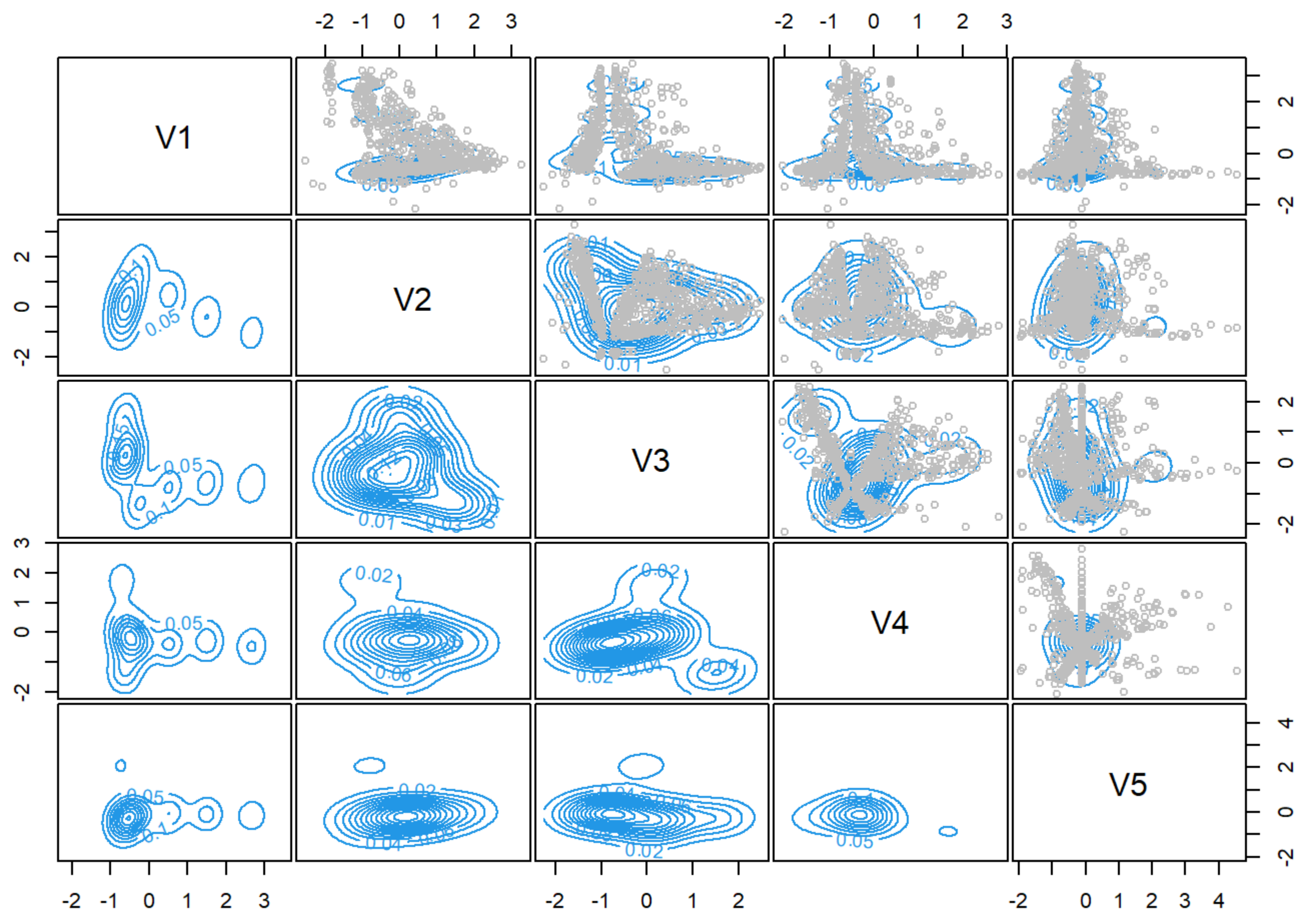
Multivariate density estimation

4, For each of the two classes, find a density estimate in the equal variance (EEE)subfamily of multivariate normal mixtures, with the number of components ranging from 1 to 9. Pairwise plot of estimated density with data are plotted.

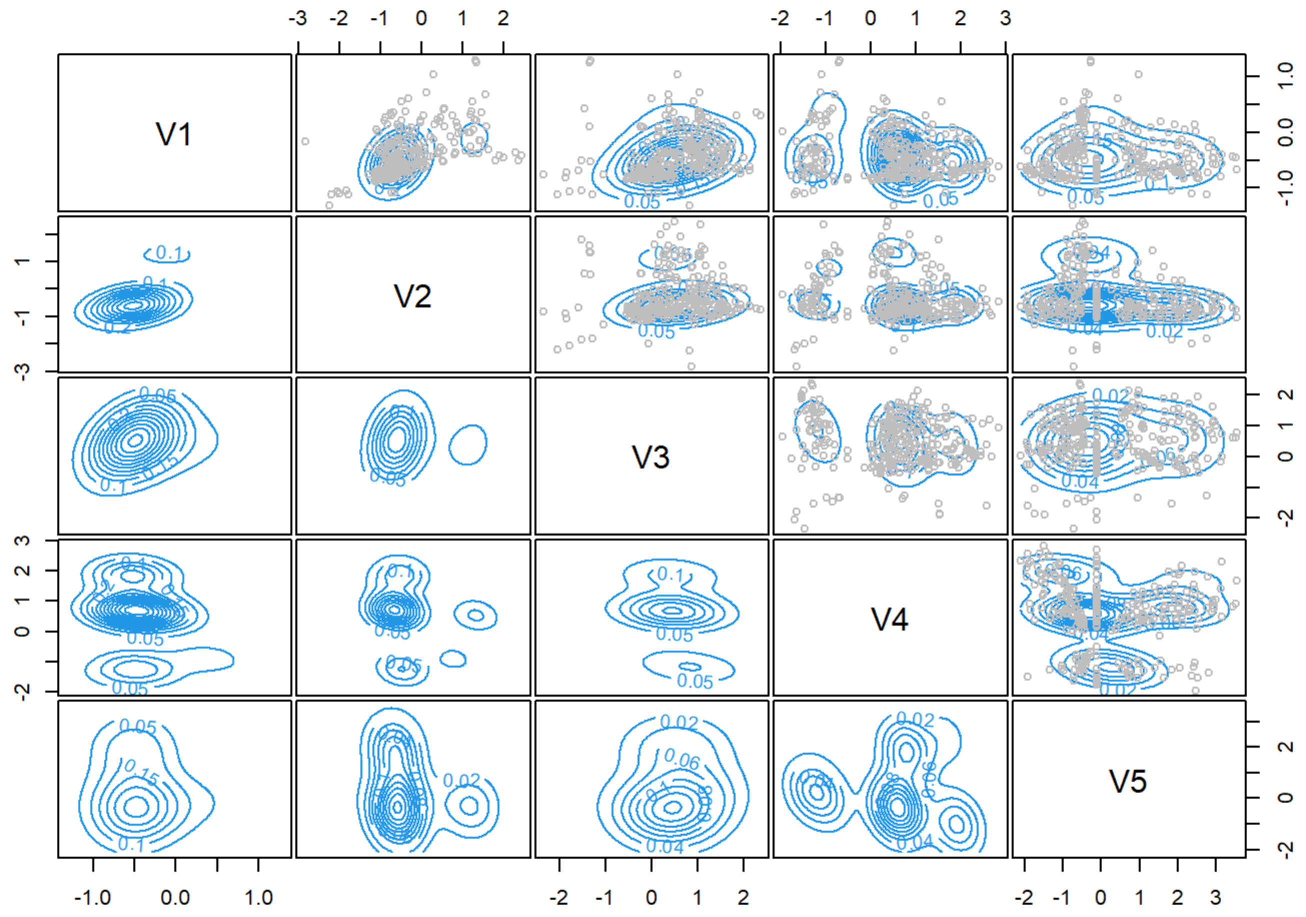
```
## subset data into oral and nasal
oral<- phoneme[phoneme$Class=="Oral",]
nasal<- phoneme[phoneme$Class=="Nasal",]

## estimated EEE density for oral and nasal
r.eoral<-densityMclust(oral[,-6],G=1:9 ,modelNames="EEE")
r.enasal<-densityMclust(nasal[,-6],G=1:9 ,modelNames="EEE")

## pairwise plot of estimated density of Nasal & Oral with method "EEE"
plot(r.enasal,nasal[,-6],what="density",col=4, points.col="grey")
```

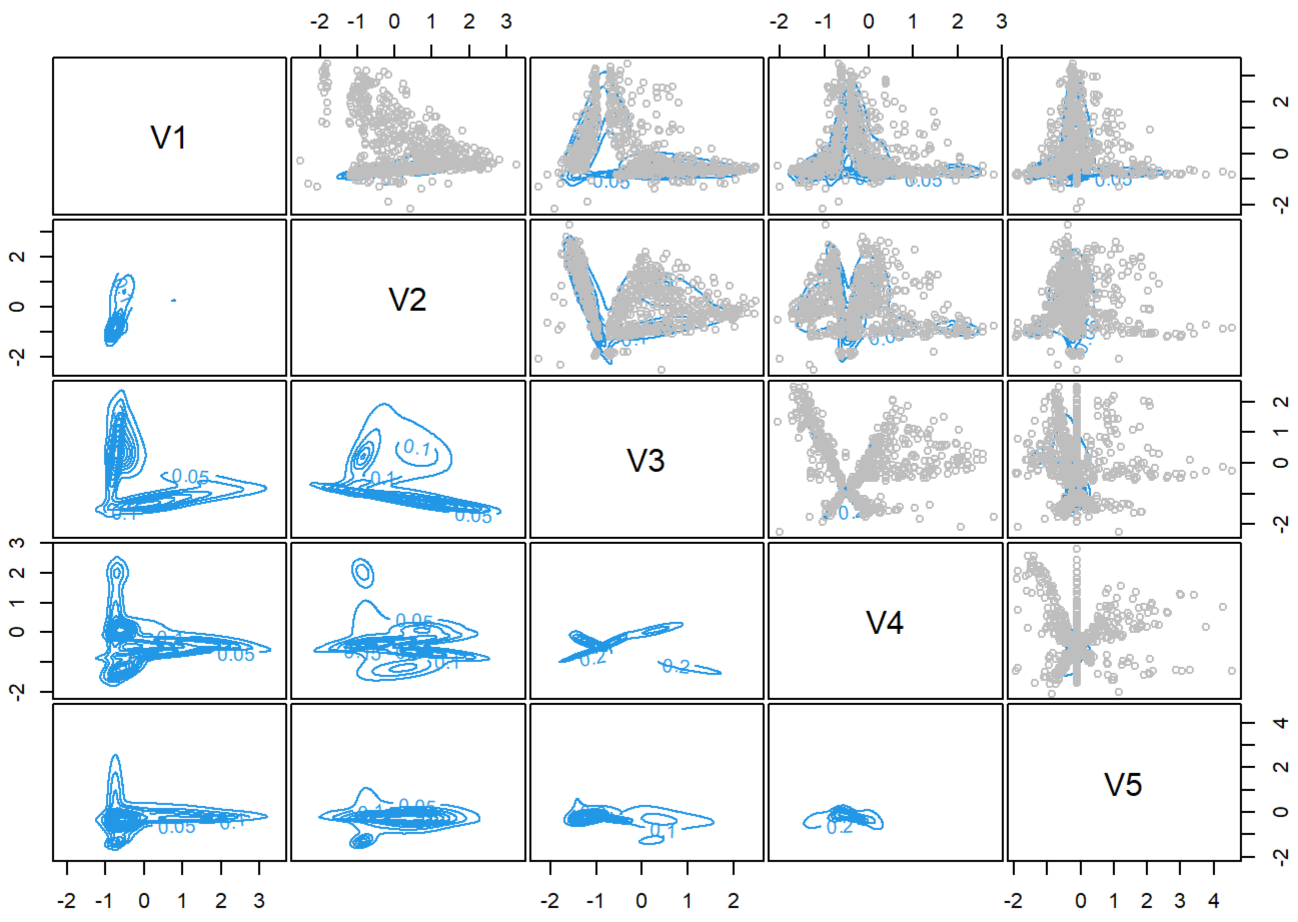


```
plot(r.eoral,oral[,-6],what="density",col=4, points.col="grey")
```

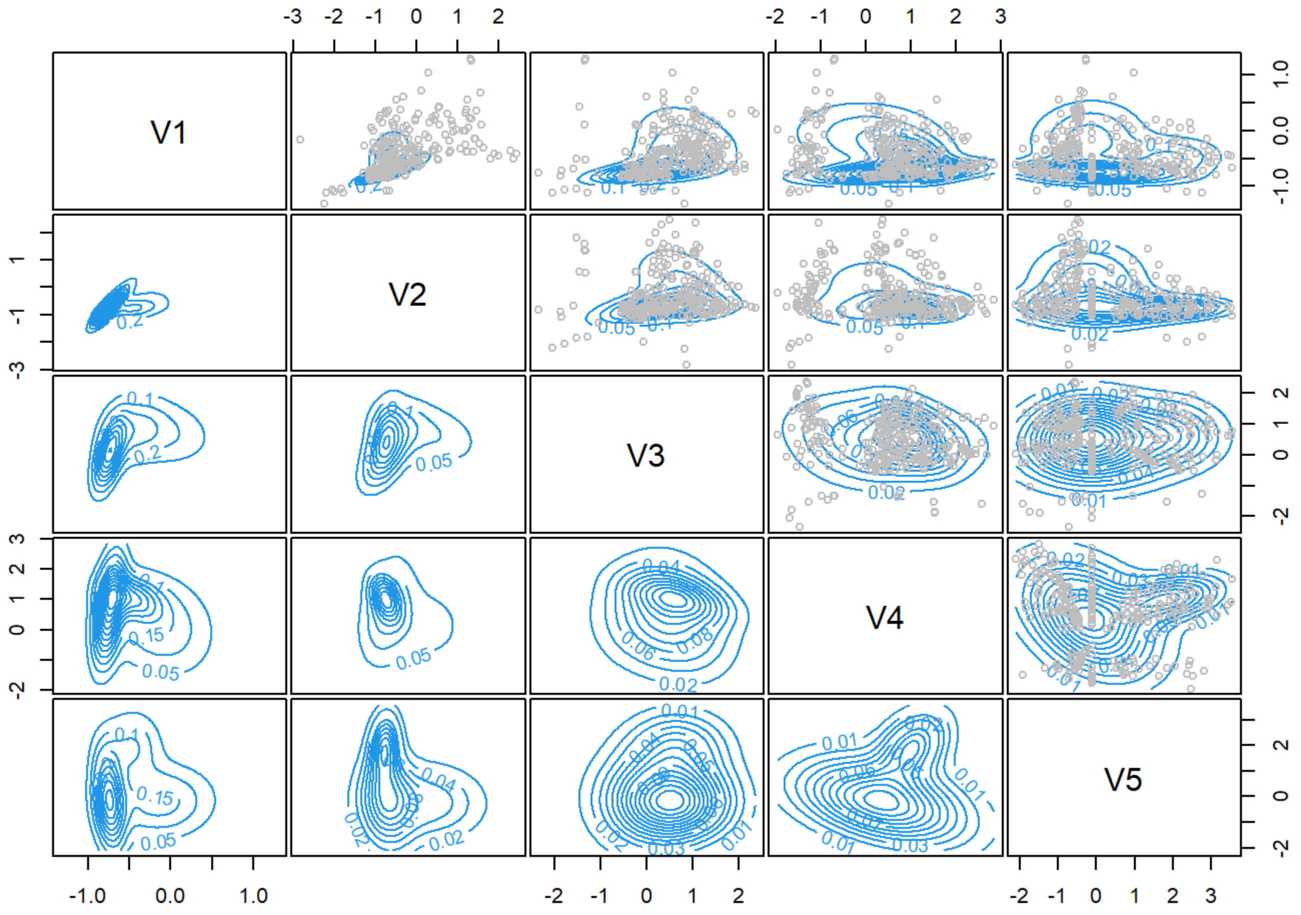


5, Repeat Task 4, but for the varying variance subfamily of normal mixtures.

```
## estimated VVV density for oral and nasal
r.voral<-densityMclust(oral[,-6],G=1:9 ,modelNames="VVV")
r.vnasal<-densityMclust(nasal[,-6],G=1:9 ,modelNames="VVV")
## pairwise plot of density of Nasal & Oral with method "VVV"
plot(r.vnasal,nasal[,-6],what="density",col=4, points.col="grey")
```



```
plot(r.voral,oral[,-6],what="density",col=4, points.col="grey")
```



6, By applying the fundamental rule of classification, we multiply the prior probability of each class with predicted posterior probability of each class, and compared them to get the number of classification. we compare this classification with the column “Class” in phoneme and calculate the misclassification rate for both model “EEE” & “VVV”.

the misclassification rate for method “EEE” is 18%; 16.5 for method “VVV”.

```
prior.p<-prop.table(table(phoneme$Class))            ## prior probability of oral & Nasal in pho
neme
## EEE model
p.eoral<-predict(r.eoral,phoneme[,1:5])*prior.p[2]   ## posterior probability of Oral (EEE)
p.enasal<-predict(r.enasal,phoneme[,1:5])*prior.p[1] ## posterior probability of Nasal (EEE)
t.e<- ifelse(p.eoral>p.enasal,2,1)

## misclassification rate of model EEE
mean(t.e!=as.numeric(factor(phoneme[,6])))
```

```
## [1] 0.18
## VVV model
p.voral<-predict(r.voral,phoneme[,1:5])*prior.p[2]   ## posterior probability of Oral (VVV)
p.vnasal<-predict(r.vnasal,phoneme[,1:5])*prior.p[1] ## posterior probability of Nasal (VVV)
t.v<- ifelse(p.voral>p.vnasal,2,1)

## misclassification rate of model VVV
mean(t.v!=as.numeric(factor(phoneme[,6])))
```

```
## [1] 0.165
```

all data estimation with both VVV & EEE

```
### test with whole data phoneme for both model "EEE" and "VVV"
r.all<-densityMclust(phoneme[,-6],G=1:9 ,modelNames=c("VVV","EEE"))
#(r.all$parameters), model with 9 components has highest BIC
p.all<-ifelse(predict(r.all,phoneme[,-6])>0.5,2,1)
#misclassification rate of whole dataset phoneme ~34%, larger error
mean(p.all!=as.numeric(factor(phoneme[,6])))
```

```
## [1] 0.337
```

K-means

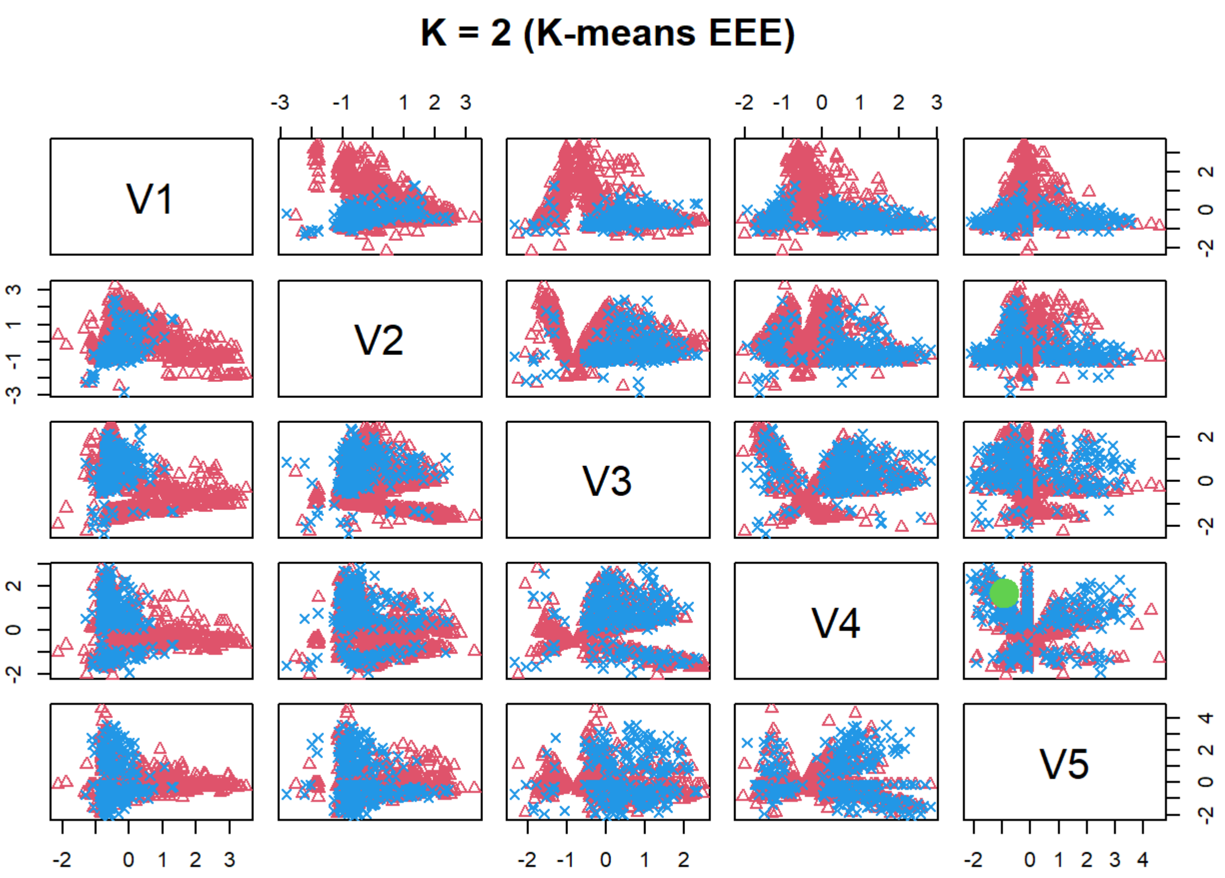
7, With the K-means method, find the clustering results with two clusters. Show the results in a pairwise plot of the data, using different colors and point types for observations of different clusters.

The majority of class (Nasal) was plotted firstly as red triangles, then the class (Oral) in blue crosses. The center of clusters is in either black (Nasal) or green (oral). Hardly any centers (black/green) can be seen, indicating poor clustering.

```
## with two clusters
set.seed(432)
(r0 = kmeans(phoneme[,-6], centers=2)) # K = 2
```

```
## K-means clustering with 2 clusters of sizes 608, 392
##
## Cluster means:
##           V1         V2         V3         V4          V5
## 1 -0.5356951 -0.2092974  0.5824347  0.2396484  0.07774704
## 2  0.8308753  0.3246247 -0.9033689 -0.3716992 -0.12057806
##
## Clustering vector:
##    1    2    5    7    8    9   10   12   13   14   15   17   18   19   20   22
##    2    2    2    1    2    2    2    2    1    1    2    1    2    2    2    1
##   23   25   26   29   31   32   33   34   35   36   37   38   39   41   43   44
##    2    1    1    2    1    2    2    2    1    2    2    1    1    1    1    2
##   45   47   48   49   50   51   52   53   54   55   57   58   59   60   61   62
##    2    1    2    2    1    2    1    1    2    2    1    1    1    1    2    2
##   63   64   66   68   70   72   73   74   76   77   78   79   80   81   82   83
##    2    2    1    2    2    2    1    2    2    2    2    1    2    1    2    2
##   84   88   89   90   91   93   94  101  102  103  104  105  107  108  109  110
##    1    2    1    1    2    2    1    1    2    2    2    2    2    2    1    2
##  111  113  114  115  116  117  118  119  120  122  124  125  126  127  130  131
##    2    1    2    2    1    1    1    1    1    2    1    1    1    1    1    1
##  132  134  135  138  143  144  145  146  148  150  152  153  154  155  156  158
##    1    1    1    2    1    2    2    1    1    2    2    1    2    1    2    2
##  159  161  162  164  165  166  167  168  169  170  172  173  174  177  178  179
##    2    2    2    2    2    1    1    2    2    2    2    2    2    1    1    2
##  180  181  183  185  186  187  188  189  190  191  193  194  198  199  201  203
##    2    2    2    2    2    1    2    2    1    2    2    2    2    1    2    2
##  205  206  208  210  212  213  215  216  217  218  220  221  222  223  224  225
##    2    1    1    1    2    2    2    2    1    1    2    2    2    1    2    1
##  230  232  233  234  235  237  238  239  241  242  243  246  248  251  252  254
##    1    1    2    1    1    1    1    2    1    1    2    2    2    1    1    1
##  257  259  261  262  263  264  266  268  269  270  271  274  276  277  278  279
##    1    1    1    1    1    2    1    1    1    2    1    1    1    2    1    1
##  281  282  283  287  288  289  290  291  292  293  294  295  296  298  299  301
##    2    2    1    1    2    2    1    1    1    1    2    2    1    2    2    2
##  302  303  304  305  309  310  312  313  314  315  317  319  320  321  322  323
##    2    1    2    2    1    1    2    2    2    2    2    2    1    1    2    2
##  324  325  328  329  331  332  333  335  337  338  340  342  343  344  345  347
##    2    1    2    2    2    1    1    2    2    2    1    1    2    2    2    2
##  348  350  351  352  353  357  358  359  360  361  362  363  364  365  368  369
##    2    2    1    2    2    1    2    2    1    2    2    1    2    2    2    1
##  371  372  374  375  376  377  378  379  381  382  383  384  385  386  387  388
##    1    2    2    2    2    1    2    2    2    2    2    1    1    1    1    2
##  389  390  391  393  394  395  397  398  399  400  401  403  404  408  409  410
##    2    2    2    2    1    2    1    1    2    1    1    1    1    1    1    2
##  411  413  414  415  416  417  418  419  420  421  423  424  425  426  427  428
##    1    2    2    1    2    2    2    2    2    1    2    1    1    2    1    2
##  429  430  431  432  433  434  435  436  437  438  439  440  442  443  445  446
##    2    1    2    2    1    2    2    1    2    1    1    1    2    2    1    2
##  448  449  450  451  452  453  454  456  458  459  460  461  462  465  466  467
##    1    2    2    2    1    2    2    1    2    1    1    1    1    1    2    1
##  468  470  471  472  473  474  475  476  479  481  482  483  485  487  489  490
##    2    1    1    1    2    1    2    2    1    1    2    2    2    1    2    2
##  491  492  494  495  496  497  498  499  500  501  502  503  504  505  506  507
##    1    2    1    1    1    2    2    2    1    1    1    1    2    2    2    1
##  508  514  516  517  518  519  520  523  524  525  526  527  528  529  530  531
##    2    1    1    2    2    2    2    2    1    1    1    1    1    1    1    2
##  533  534  537  538  539  540  541  542  544  545  546  547  550  552  553  554
##    1    1    2    1    1    2    1    1    2    2    2    2    2    2    2    1
##  555  557  558  559  560  562  563  565  566  567  568  569  570  571  573  574
##    1    1    1    2    2    2    1    1    1    2    1    2    2    2    2    2
##  576  577  578  579  580  581  583  584  586  588  589  591  592  593  594  595
##    2    2    2    1    2    1    1    2    2    1    2    1    2    2    2    1
##  596  598  600  601  603  604  605  606  609  611  612  613  615  616  617  620
##    2    2    2    1    1    1    2    1    1    2    1    2    1    2    2    2
##  621  623  624  625  626  627  628  629  630  631  632  633  635  637  639  640
##    2    1    2    1    2    2    1    2    1    2    2    1    2    1    2    2
##  641  643  644  645  646  648  649  651  652  653  654  655  656  657  658  659
##    1    1    1    2    2    2    2    2    1    2    1    2    2    1    1    1
##  660  661  662  663  664  665  668  670  671  672  673  674  675  676  677  678
##    2    1    2    2    2    1    1    2    2    2    2    1    1    1    2    2
##  679  681  682  684  685  687  689  690  691  692  693  694  695  696  699  700
##    2    1    1    2    2    1    1    2    1    2    1    2    2    1    1    2
##  702  703  704  705  706  707  708  710  711  714  715  716  717  718  719  721
##    1    1    1    2    1    2    2    2    2    2    1    2    1    1    2    1
##  722  723  724  725  728  729  731  732  733  734  736  738  739  740  743  744
##    1    1    2    2    2    2    1    2    2    2    1    1    2    2    1    2
##  745  746  747  748  750  751  753  755  756  758  759  760  761  763  764  766
##    2    1    2    2    2    1    1    2    1    2    2    1    2    1    1    1
##  768  769  770  771  772  775  777  781  783  784  786  788  789  790  791  793
##    1    2    1    2    1    1    2    1    2    2    1    1    1    2    1    1
##  795  797  801  802  804  805  806  809  810  812  814  816  817  820  822  823
##    2    1    2    2    1    1    1    1    1    1    2    2    2    2    1    2
##  824  825  827  828  831  832  833  834  835  837  838  840  844  845  846  847
##    1    2    1    2    2    1    1    1    2    1    2    1    1    2    1    1
##  848  849  852  853  855  858  859  860  861  863  865  866  868  869  870  873
##    1    2    2    1    2    2    2    1    2    2    1    1    1    2    1    2
##  874  875  878  880  883  887  888  890  891  892  893  895  896  897  901  903
##    1    2    1    1    2    2    2    2    1    2    1    2    1    2    1    1
##  904  905  906  907  908  909  910  911  912  914  915  916  917  918  922  924
##    1    2    2    1    1    2    1    2    1    1    1    2    1    1    1    1
##  926  927  928  929  931  933  934  939  940  941  942  943  944  947  948  949
##    1    1    2    2    1    1    2    2    2    2    1    2    2    1    1    1
##  950  951  953  954  955  957  958  959  960  961  962  963  964  965  966  967
##    1    1    1    1    1    1    2    1    1    2    2    1    1    2    2    1
##  968  969  970  972  974  975  976  979  982  983  986  988  989  990  993  994
##    2    1    1    1    2    2    1    1    2    2    2    1    2    1    1    2
##  995  996  997  999 1000    3    4    6   11   16   21   24   27   28   30   40
##    1    1    2    1    2    1    1    1    1    1    1    1    1    1    1    1
##   42   46   56   65   67   69   71   75   85   86   87   92   95   96   97   98
##    1    2    1    1    1    1    1    1    1    1    1    1    1    1    1    1
##   99  100  106  112  121  123  128  129  133  136  137  139  140  141  142  147
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
##  149  151  157  160  163  171  175  176  182  184  192  195  196  197  200  202
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
##  204  207  209  211  214  219  226  227  228  229  231  236  240  244  245  247 
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1 
##  249  250  253  255  256  258  260  265  267  272  273  275  280  284  285  286 
##    1    1    1    1    1    2    1    1    1    1    1    1    1    1    2    1 
##  297  300  306  307  308  311  316  318  326  327  330  334  336  339  341  346 
##    1    2    1    1    2    1    1    1    2    1    1    1    1    1    1    1 
##  349  354  355  356  366  367  370  373  380  392  396  402  405  406  407  412 
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1 
##  422  441  444  447  455  457  463  464  469  477  478  480  484  486  488  493 
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1 
##  509  510  511  512  513  515  521  522  532  535  536  543  548  549  551  556 
##    1    2    1    1    1    1    1    1    1    1    2    1    1    1    1    1 
##  561  564  572  575  582  585  587  590  597  599  602  607  608  610  614  618 
##    1    1    1    1    1    1    1    2    1    1    1    1    1    1    1    1 
##  619  622  634  636  638  642  647  650  666  667  669  680  683  686  688  697 
##    2    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1 
##  698  701  709  712  713  720  726  727  730  735  737  741  742  749  752  754 
##    1    1    1    1    1    1    1    1    1    1    1    2    1    2    1    1 
##  757  762  765  767  773  774  776  778  779  780  782  785  787  792  794  796 
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1 
##  798  799  800  803  807  808  811  813  815  818  819  821  826  829  830  836 
##    1    2    1    1    1    1    1    1    1    1    1    1    1    1    1    1 
##  839  841  842  843  850  851  854  856  857  862  864  867  871  872  876  877 
##    1    1    2    2    1    1    1    1    1    1    1    1    1    1    1    1 
##  879  881  882  884  885  886  889  894  898  899  900  902  913  919  920  921 
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1 
##  923  925  930  932  935  936  937  938  945  946  952  956  971  973  977  978 
##    1    1    1    2    1    1    1    1    1    1    1    1    1    2    1    1 
##  980  981  984  985  987  991  992  998 
##    1    1    1    1    1    1    1    1 
## 
## Within cluster sum of squares by cluster:
## [1] 2519.854 1337.500
##  (between_SS / total_SS =  22.8 %)
## 
## Available components:
## 
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
#o = order.majority(r0$cluster)
#(count1 = attr(o, "count.majority"))
with(phoneme, pairs(phoneme[,-6], col=c(2,4)[Class],pch=c(2,4)[Class], main="K = 2 (K-means EE
E)"))
points(r0$centers, col=c(1,3), pch=19, cex=2)
```



8, Compare the results obtained with K means clustering (unsupervised clustering) with the availalbe class in phoneme dataset.

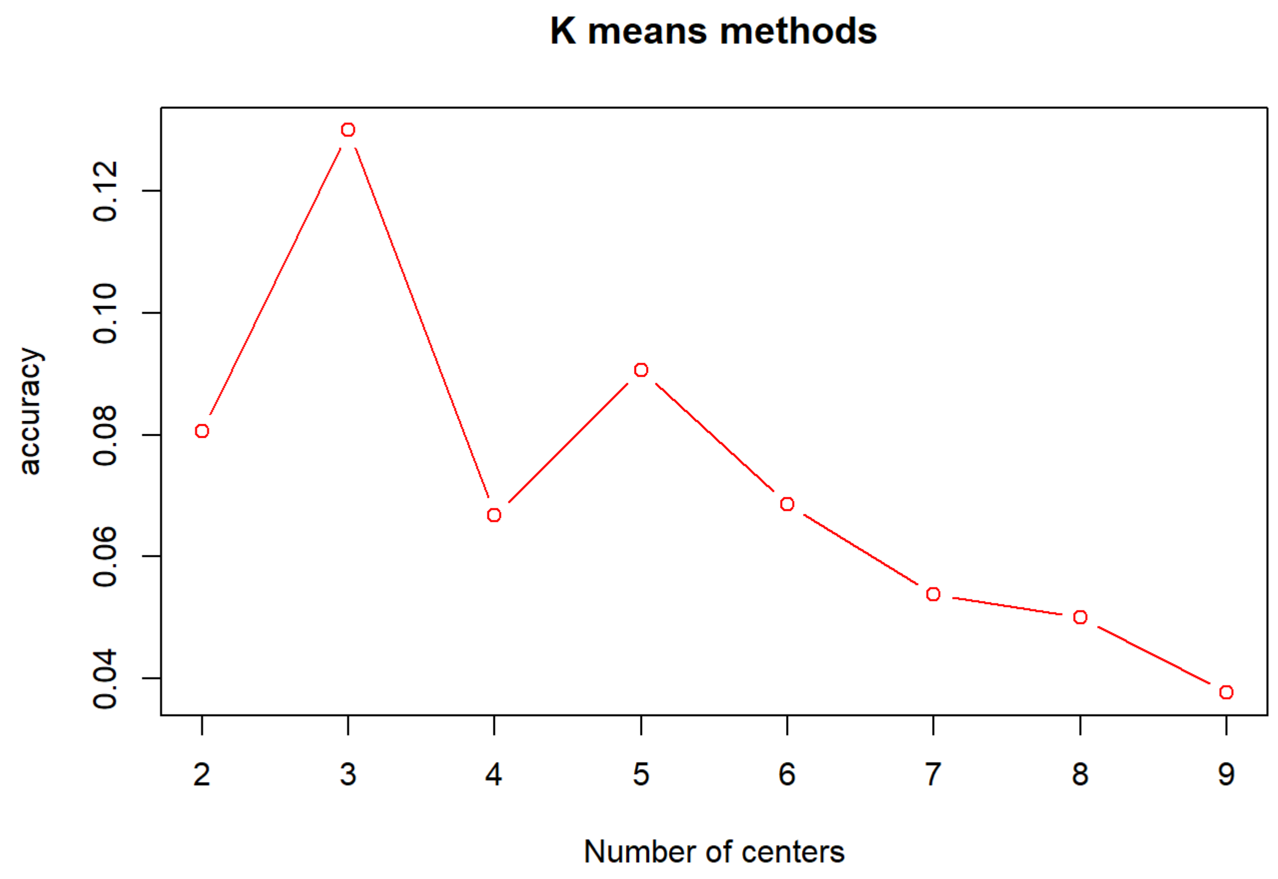
The classification accuracy is very low about 35% with 2 clustering centers, and reducing with the increasing of clustering centers.

```
accuracy = double(8)

for(k in 2:9) {
  set.seed(432)
  r = kmeans(phoneme[,-6], centers=k)
  accuracy[k-1]<- adjustedRandIndex(r$cluster,as.numeric(factor(phoneme$Class)))
}
## clustering accuracy
accuracy
```

```
## [1] 0.08058490 0.12999026 0.06678252 0.09069357 0.06855240 0.05386771 0.04999859
## [8] 0.03767955
```

```
plot(2:9,accuracy, type="b",col="red", xlab="Number of centers",ylab="accuracy", main="K means m
ethods")
```



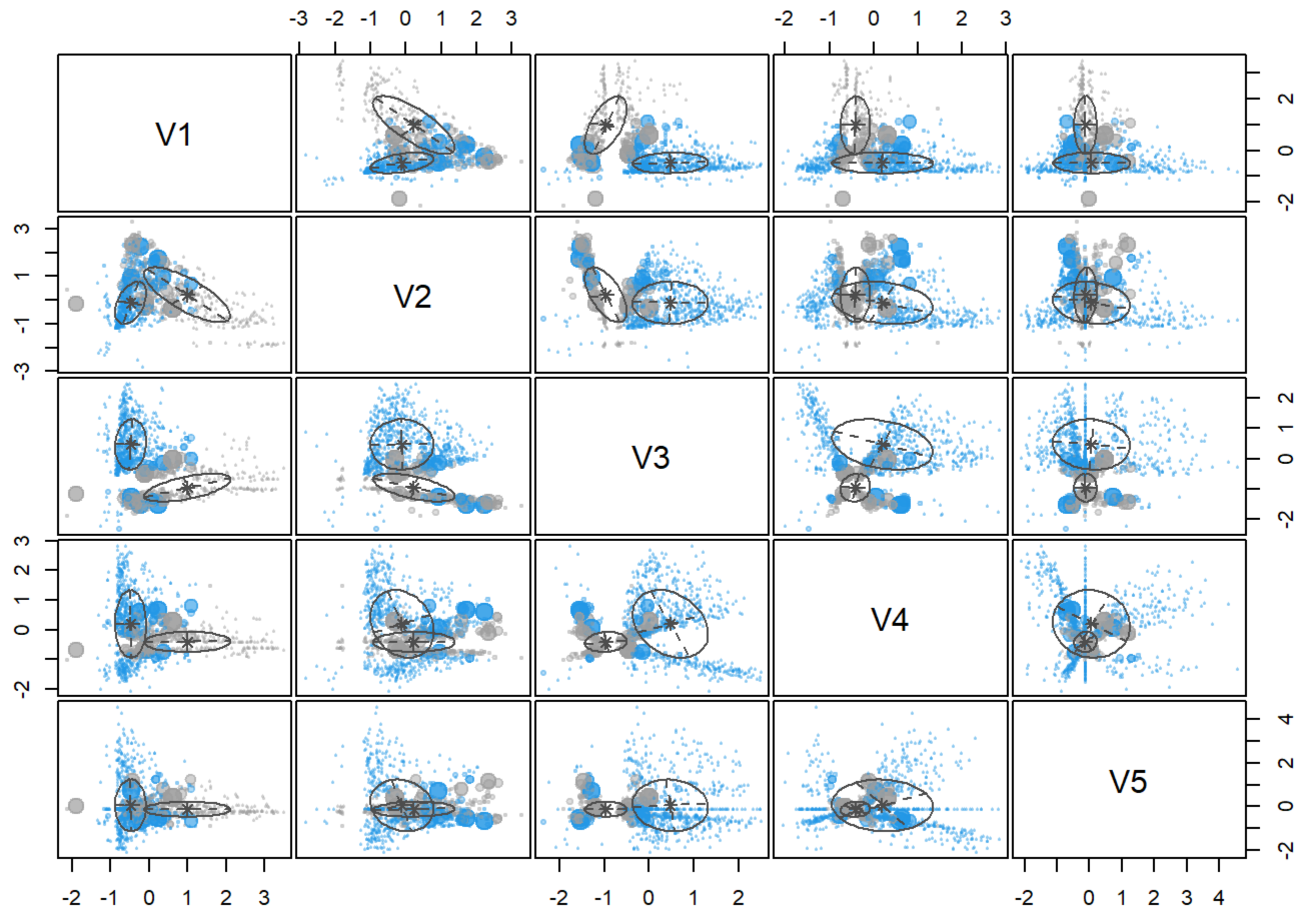
Mixture-based clustering

9, Use the varying variance subfamily of multivariate normal mixtures to find the clustering results with two clusters, and show the results in a pairwise plot,compute the adjusted Rand indices for K=2,…9 clusters.

The classification accuracy is almost 0, indicating no similarity of mixture based clustering comparing to original class.

```
set.seed(432)
r.mbc = Mclust(phoneme[,-6], G=2, modelNames="VVV")

plot(r.mbc,"uncertainty",col=c(8,4), main="K = 2 (k-means, VVV)")  ## uncertainty plot of varyin
g variance method with 2 centers
```

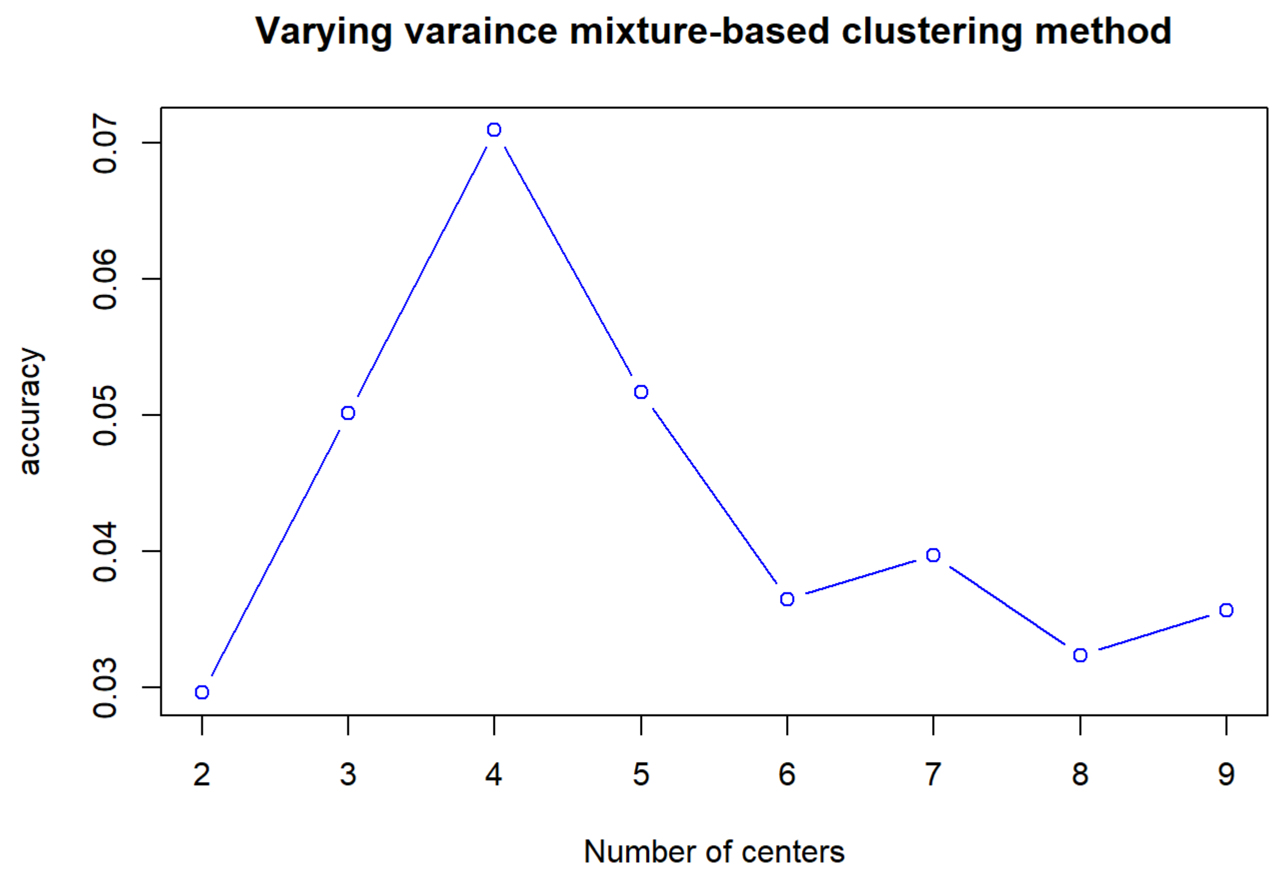


```
### compute clustering accuracy for k = 2~9
accuracy.v = double(8)
for(k in 2:9) {
  r.mbc = Mclust(phoneme[,-6], G=k, modelNames = "VVV")
  accuracy.v[k-1]<- adjustedRandIndex(r.mbc$classification, as.numeric(factor(phoneme$Class)))
}

## clustering accuracy
accuracy.v
```

```
## [1] 0.02963741 0.05014537 0.07091669 0.05169660 0.03648900 0.03973215 0.03236527
## [8] 0.03569147
```

```
plot(2:9,accuracy.v,type="b",col="blue", xlab="Number of centers",ylab="accuracy", main="Varying
varaince mixture-based clustering method")
```



Hierarchical Clustering

10,Produce a dendrogram of the hierarchical clustering results using the complete and the single linkage method, respectively.

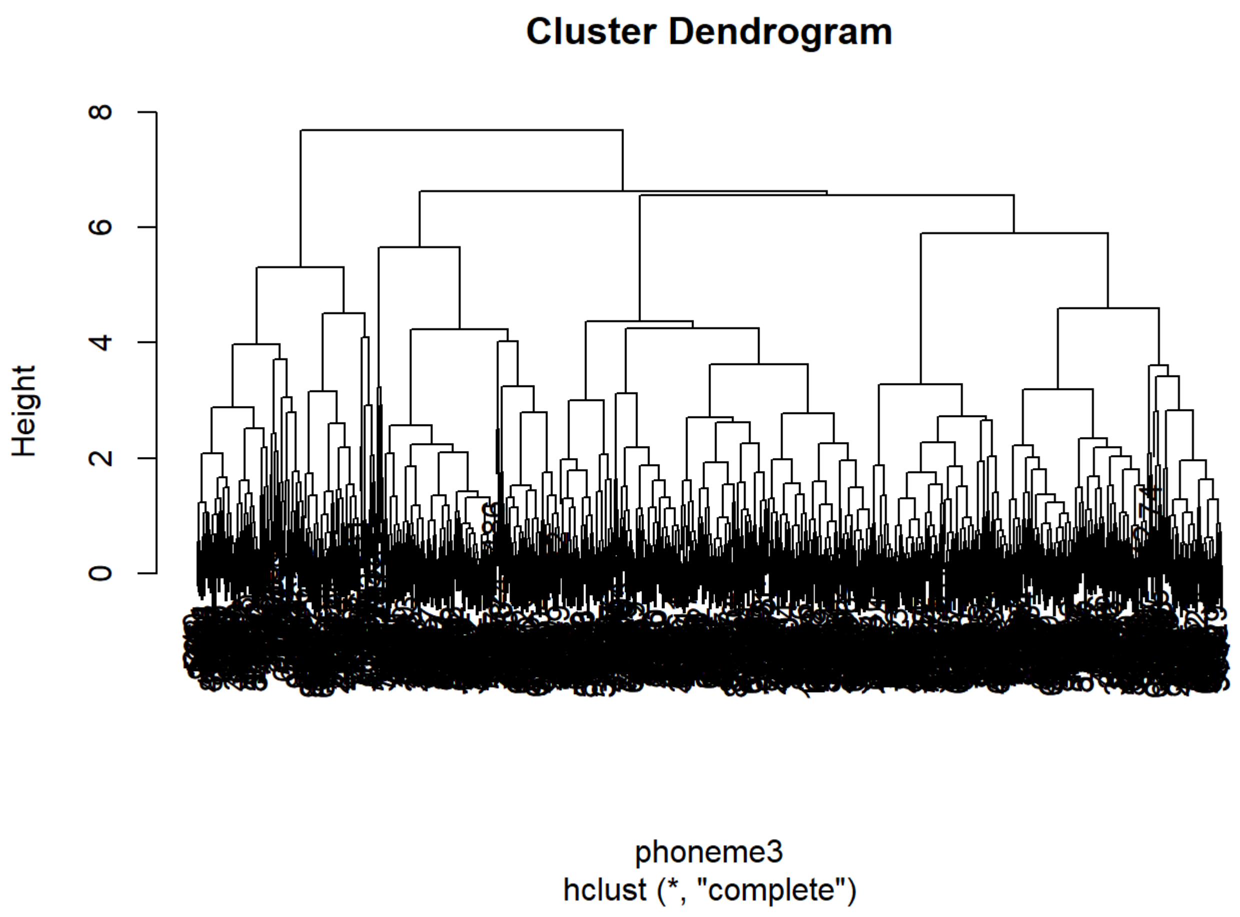
Explain why they look very much different? Complete linkage clustering maximizes the pairwise observations' dissimilarity, while single linkage clustering minimizes the dissimilarity. For this dataset, there is no clear boundary between different clusters, thus by comparing euclidean distance of pairwised observations is not sufficient for clustering.

(from teacher)The complete-linkage method tends to produce more compact clusters and relatively large clusters near the top of the clustering tree.With the single-linkage method, clusters are linked purely based on two closest observations and may exhibit an extended, chain shape. It thus tends to produce clusters with very few observations even near the top of the clustering tree.

```
## standardize phoneme - euclidean distance matrix
phoneme3<-dist(phoneme[,-6])
## complete linkage
r.cl = hclust(phoneme3)                # complete linkage by default
names(r.cl)
```

```
## [1] "merge"        "height"       "order"        "labels"       "method"
## [6] "call"         "dist.method"
```

```
plot(r.cl)
```

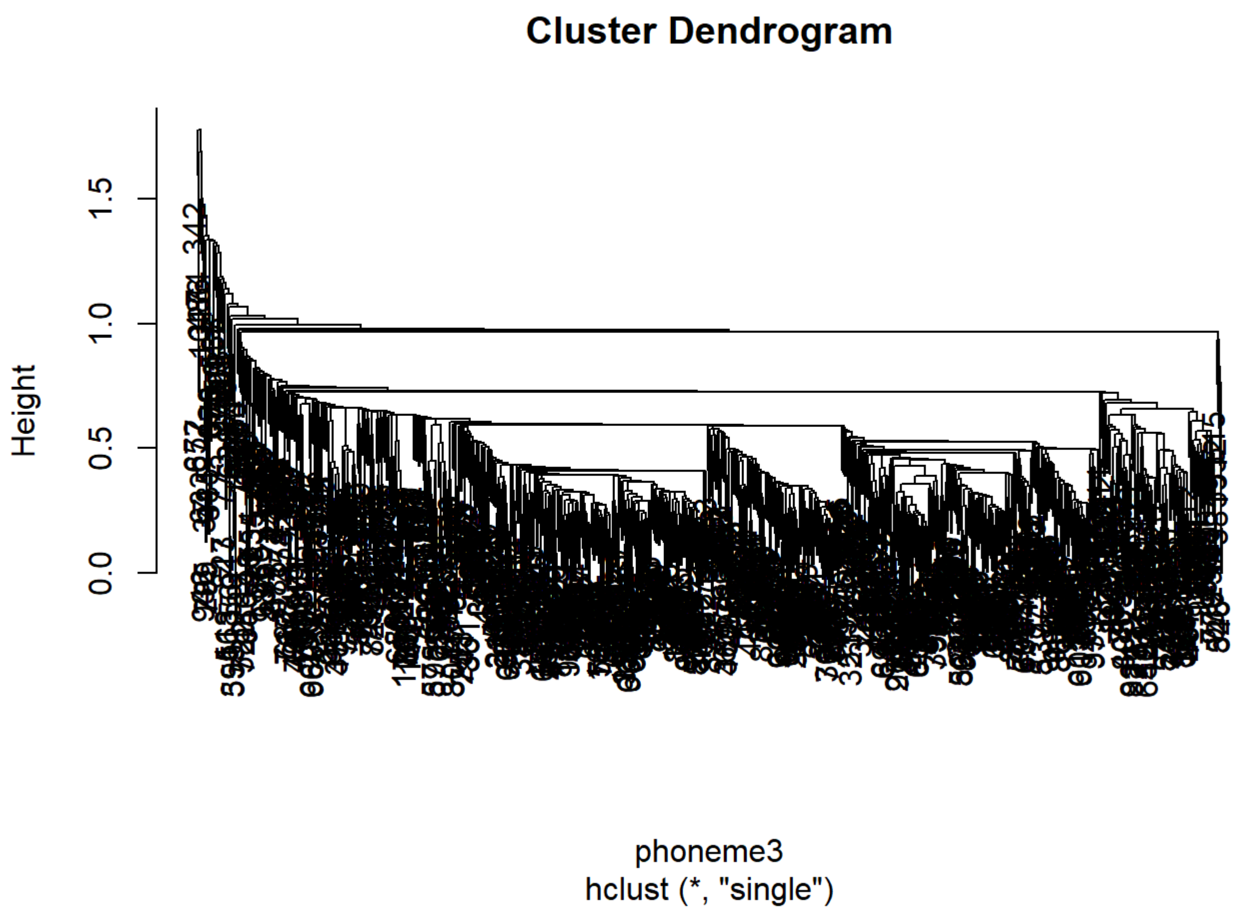


```
## single linkage
r.sl= hclust(phoneme3,method="single")

names(r.sl)
```

```
## [1] "merge"        "height"       "order"        "labels"       "method"
## [6] "call"         "dist.method"
```

```
plot(r.sl)
```



11, Compute the adjusted Rand indices for K=2,…,9 clusters produced by the complete and the single linkage method, respectively.

The result shows very low ARI values, close to 0, which means both tree clustering methods of complete and single linkage methods are not consistent, indicating data can not be seperable with pairwise euclidean distances dissimilarity.

```
### compute ari
ari = double(9)
for(k in 2:9) {
  r.cl = hclust(phoneme3)
  ari[k-1] = adjustedRandIndex(cutree(r.cl,k), cutree(r.sl,k))
}
ari
```

```
## [1] -0.0015711706  0.0033555333  0.0002705179  0.0006832815  0.0017865693
## [6]  0.0019591100  0.0051464297  0.0080570515  0.0000000000
```

```
## For the complete-linkage method, the 2-cluster partition is a little bit similar to that acco
rding to class labels, as ARI =0.152.

## For the single-linkage method, all ARI-values are virtually 0, indicating no similarity betwe
en any of its clustering partitions and the one according to class labels.
round(sapply(1:8, function(j) adjustedRandIndex(phoneme[,6], cutree(r.cl,j+1))), 3)
```

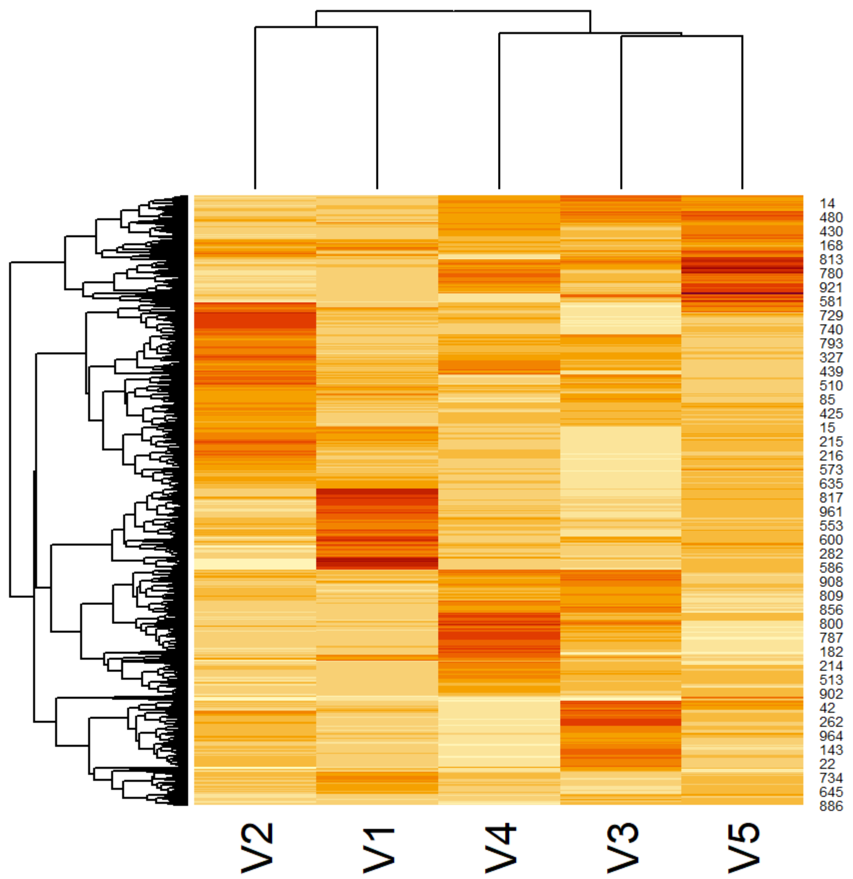
```
## [1] 0.152 0.087 0.048 0.069 0.070 0.075 0.081 0.082
```

```
round(sapply(1:8, function(j) adjustedRandIndex(phoneme[,6], cutree(r.sl,j+1))), 3)
```

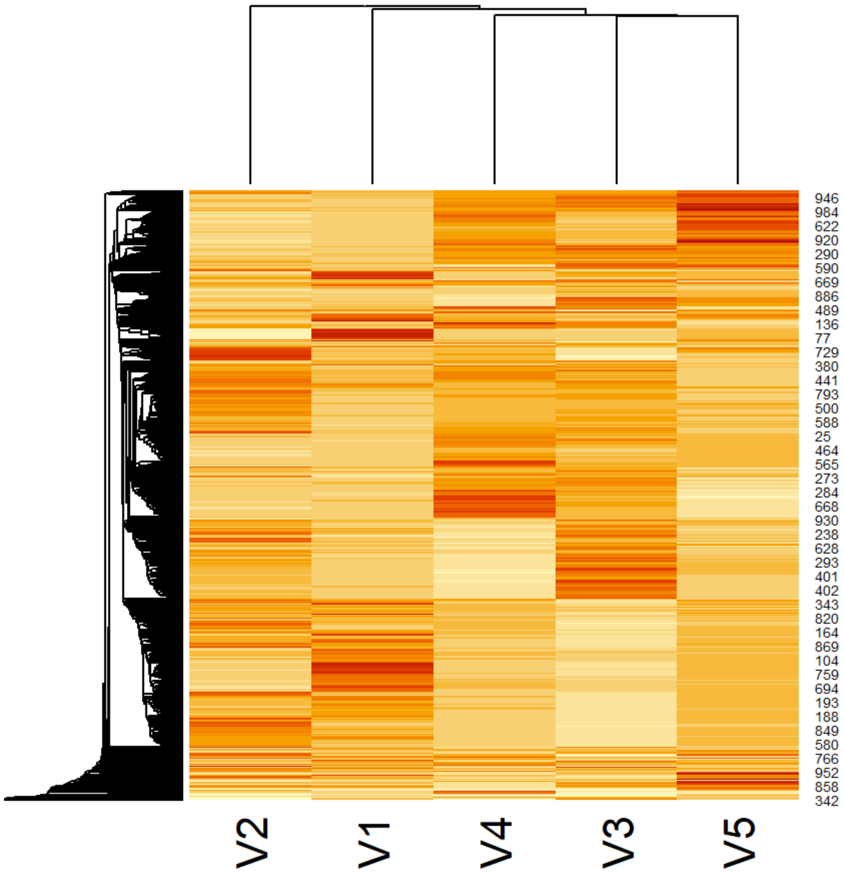
```
## [1] -0.001  0.001 -0.001  0.002  0.005  0.004  0.008  0.014
```

12, Produce the heatmaps for both the complete and single linkage methods.High values are in red, low values are in yellow.

```
## complete linkage
heatmap(as.matrix(phoneme[,-6]), scale="column", distfun=dist, hclustfun=hclust)
```



```
## single linkage
hclust.single<- function(x, ...) hclust(x, method="single", ...)
heatmap(as.matrix(phoneme[,-6]), scale="none", distfun=dist, hclustfun=hclust.single)
```



Summary

In this lab we get to know density estimation and clustering methods. For density estimated we compared results of equal variance and varying variance density estimation, the result shows that nrd0 is underfitted, ucv, bcv and SJ methods are better fitted. Further we use BIC selected normal mxiture estimate and choose 6 variables VVV model for density estimate, the result looks overfitting, especially at peak. Afterwards, we do the multivariant density estimate for both classes oral and nasal and for both “EEE”,“VVV” methods, the calculated misclassification rate for method “EEE” is 18%; for method “VVV” is 16.5%.

For clustering, k means method does not give good results, probably because of no clear boundary between the class, thus methods using Euclidean distances fail, including hierachical clustering. The heatmaps generated from single and complete linkage do not have consistency (ARI ~0), indicating poor clustering. While K-means clustering with two components gives a bit better results for phoneme data.