STAT3500 Assignment 4

Chee Kitt Win

11/1/2021

```
library(lattice)
uni_data = read.csv("C:\\Users\\Owner\\Desktop\\UQ Year 3 Sem 2 Courses\\STAT3500\\Assignment 4\\Data-Ass4a(4).csv")
```

a) iii)

Starting points tried

Looking at the density plot, we see most of the data clustered around x = 1. We also observe 2 points near x = -2 which seem more like outliers rather than a separate cluster. Initializing the parameters as mu1 = 0, mu2 = 2, common variance = 1, pro1 = 0.5, pro2 = 0.5 seems reasonable here. The estimated parameters at convergence are given below.

Stopping criterion

According to the specification of Mclust, the algorithm stops iterating if any of the following are met:

1. The improvement of the value of the Q function between iterations is less than 0.00001 (Successful convergence)

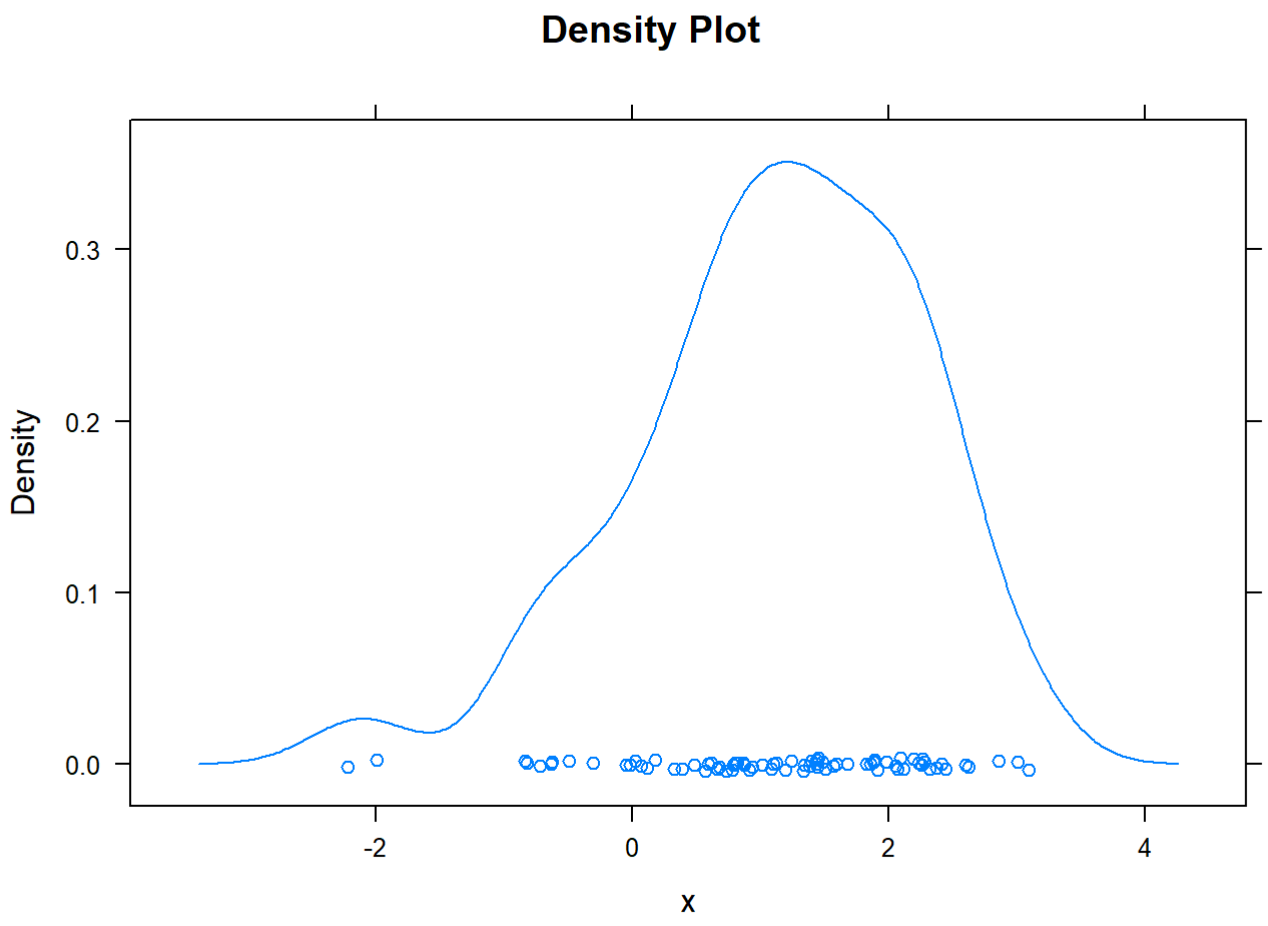
2. At any point, the updated covariance has a value of less than the relative machine precision .Machine$double.eps, which is approximately 2x10^− 16 on IEEEcompliant machines.

```
library(mclust)
```

```
## Warning: package 'mclust' was built under R version 4.1.1
```

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

```
densityplot(uni_data[,2], main = "Density Plot" , xlab = "x")
```



```
params = list(pro = c(0.5,0.5),mean = c(0,2), variance = list(modelName = "E", d = 1, G = 2,
 sigmasq = 1))


set.seed(5)
EM = em(uni_data[,2],modelName = "E", parameters = params)
EM$modelName
```

```
## [1] "E"
```

```
EM$parameters
```

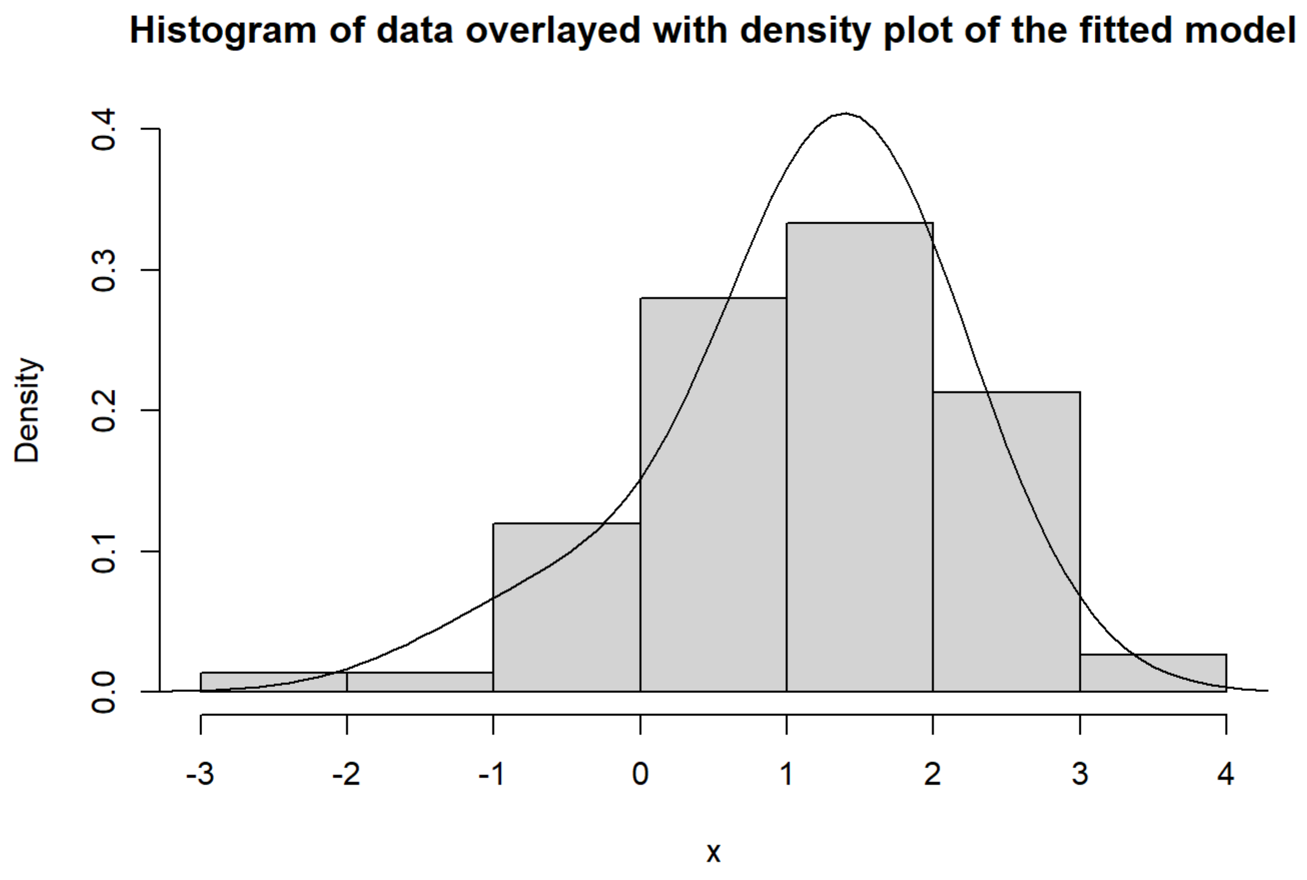
```
## $pro
## [1] 0.1438315 0.8561685
##
## $mean
##          1          2
## -0.5928387  1.4114699
##
## $variance
## $variance$modelName
## [1] "E"
##
## $variance$d
## [1] 1
##
## $variance$G
## [1] 2
##
## $variance$sigmasq
## [1] 0.7013157
##
##
## $Vinv
## NULL
```

a) iv)

Below is a histogram of the data overlayed with the pdf of the fitted model. I chose the number of bins, N, based on the formula provided in the question.

```
# N is the number of bins (based on the formula in the question)
N = log(75)/log(2) + 1
# Plot the histogram of the data
histogram = hist(uni_data[,2], breaks = floor(N), xlab = "x", freq = FALSE, ylim = c(0,0.4),
                 main ="Histogram of data overlayed with density plot of the fitted model")

# Plot the pdf of the fitted GMM
x <- seq(-5, 5, by = .1)
y <- EM$parameters$pro[1]*dnorm(x, mean = EM$parameters$mean[1], sd = sqrt(EM$parameters$vari
ance$sigmasq)) +
    EM$parameters$pro[2]*dnorm(x, mean = EM$parameters$mean[2], sd = sqrt(EM$parameters$varia
nce$sigmasq))
lines(x,y, xlab = "", ylab = "")
```



a) v)

We retain the null hypothesis since the p value is 0.7758. The fit is good.

```
library(AdaptGauss)
```

```
## Warning: package 'AdaptGauss' was built under R version 4.1.1
```

```
cdf = CDFMixtures(Kernels = c(-3,-2,-1,0,1,2,3,4), Means = c(EM$parameters$mean[1],EM$paramet
ers$mean[2]), SDs = c(sqrt(EM$parameters$variance$sigmasq),sqrt(EM$parameters$variance$sigmas
q)), Weights = c(EM$parameters$pro[1],EM$parameters$pro[2]))
list = c()
for (i in 1:8){
  list = c(list,(cdf$CDFGaussMixture[i] - cdf$CDFGaussMixture[i-1]))
}
list[7] = list[7] + 1 - sum(list)
chisq.test(x = histogram$counts, p = list)
```

```
## Warning in chisq.test(x = histogram$counts, p = list): Chi-squared approximation
## may be incorrect
```

```
##
##  Chi-squared test for given probabilities
##
## data:  histogram$counts
## X-squared = 3.2585, df = 6, p-value = 0.7758
```

a) vi)

From this section onwards, instead of manually initializing starting values for the parameters as done in part a) iii), I use the Mclust() function from the mclust package to implement the EM algorithm for Gaussian mixture models. This is because many of the functions in this package are useful for subsequent questions, but require an Mclust object to be taken as an argument.

As required, variance is no longer set to be equal (modelNames = “v”), and the estimated parameters of the fitted model are given below.

```
set.seed(5)
EM_unequal_variance = Mclust(uni_data[,2], G = 2, modelNames = "V")
EM_unequal_variance$parameters
```

```
## $pro
## [1] 0.4032817 0.5967183
##
## $mean
##         1         2
## 0.3901591 1.6185916
##
## $variance
## $variance$modelName
## [1] "V"
##
## $variance$d
## [1] 1
##
## $variance$G
## [1] 2
##
## $variance$sigmasq
## [1] 1.2840916 0.5279212
##
## $variance$scale
## [1] 1.2840916 0.5279212
```

a) vii)

Here, the function MclustBootstrap() is implementing a non parametric bootstrap (indicated by the argument type = “bs”) with 1000 replications. The standard errors of the parameters obtained are given below.

```
set.seed(5)
non_par_boot = MclustBootstrap(EM_unequal_variance, type = "bs", nboot = 1000)
summary(non_par_boot, what = "se")
```

```
## ----------------------------------------------------------
## Resampling standard errors
## ----------------------------------------------------------
## Model                      = V
## Num. of mixture components = 2
## Replications               = 1000
## Type                       = nonparametric bootstrap
##
## Mixing probabilities:
##         1         2
## 0.2125635 0.2125635
##
## Means:
##         1         2
## 0.7965001 0.3182876
##
## Variances:
##         1         2
## 0.4720941 0.2195587
```

a) viii)

Using the fitted model in part a)vi), we obtain standard errors for the estimated parameters via a parametric bootstrap (type = “pb”) with 1000 replications. The standard errors of each parameter are shown below.

```
set.seed(5)
par_boot = MclustBootstrap(EM_unequal_variance, type = "pb", nboot = 1000)
summary(par_boot, what = "se")
```

```
## ----------------------------------------------------------
## Resampling standard errors
## ----------------------------------------------------------
## Model                      = V
## Num. of mixture components = 2
## Replications               = 1000
## Type                       = parametric bootstrap
##
## Mixing probabilities:
##         1         2
## 0.1730114 0.1730114
##
## Means:
##         1         2
## 0.6830476 0.2187151
##
## Variances:
##         1         2
## 0.6171004 0.2094875
```

b) i)

Aside from fitting the mixture models, I also did some pairplots as a sanity check. We can see that for g = 1,2 and 3, in each individual plot, the ellipses from each component have the same shape and size which agrees with the fact that we have fitted a GMM with common covariance matrix (modelNames = “EEE”). The parameters of the fitted models are also provided below.

```
multi_data = read.csv("C:\\Users\\Owner\\Desktop\\UQ Year 3 Sem 2 Courses\\STAT3500\\Assignme
nt 4\\Data-Ass4b(3).csv", header = FALSE)
set.seed(5)
multi_EM1 = Mclust(multi_data, G = 1, modelNames = "EEE")
multi_EM1$parameters
```

```
## $pro
## [1] 1
##
## $mean
##     [,1]
## V1 6.262
## V2 2.872
## V3 4.906
## V4 1.676
##
## $variance
## $variance$modelName
## [1] "XXX"
##
## $variance$d
## [1] 4
##
## $variance$G
## [1] 1
##
## $variance$Sigma
##          V1       V2       V3       V4
## V1 0.434956 0.120936 0.448828 0.165488
## V2 0.120936 0.109616 0.141368 0.079228
## V3 0.448828 0.141368 0.674764 0.285844
## V4 0.165488 0.079228 0.285844 0.178624
##
## $variance$cholSigma
##           [,1]       [,2]        [,3]       [,4]
## [1,] 0.6595119  0.1833720  0.68054568  0.2509249
## [2,] 0.0000000 -0.2756641 -0.06012752 -0.1204923
## [3,] 0.0000000  0.0000000  0.45607704  0.2364364
## [4,] 0.0000000  0.0000000  0.00000000  0.2126972
##
## $variance$cholsigma
##           [,1]       [,2]        [,3]       [,4]
## [1,] 0.6595119  0.1833720  0.68054568  0.2509249
## [2,] 0.0000000 -0.2756641 -0.06012752 -0.1204923
## [3,] 0.0000000  0.0000000  0.45607704  0.2364364
## [4,] 0.0000000  0.0000000  0.00000000  0.2126972
##
## $variance$sigma
## , , 1
##
##          V1       V2       V3       V4
## V1 0.434956 0.120936 0.448828 0.165488
## V2 0.120936 0.109616 0.141368 0.079228
## V3 0.448828 0.141368 0.674764 0.285844
## V4 0.165488 0.079228 0.285844 0.178624
```

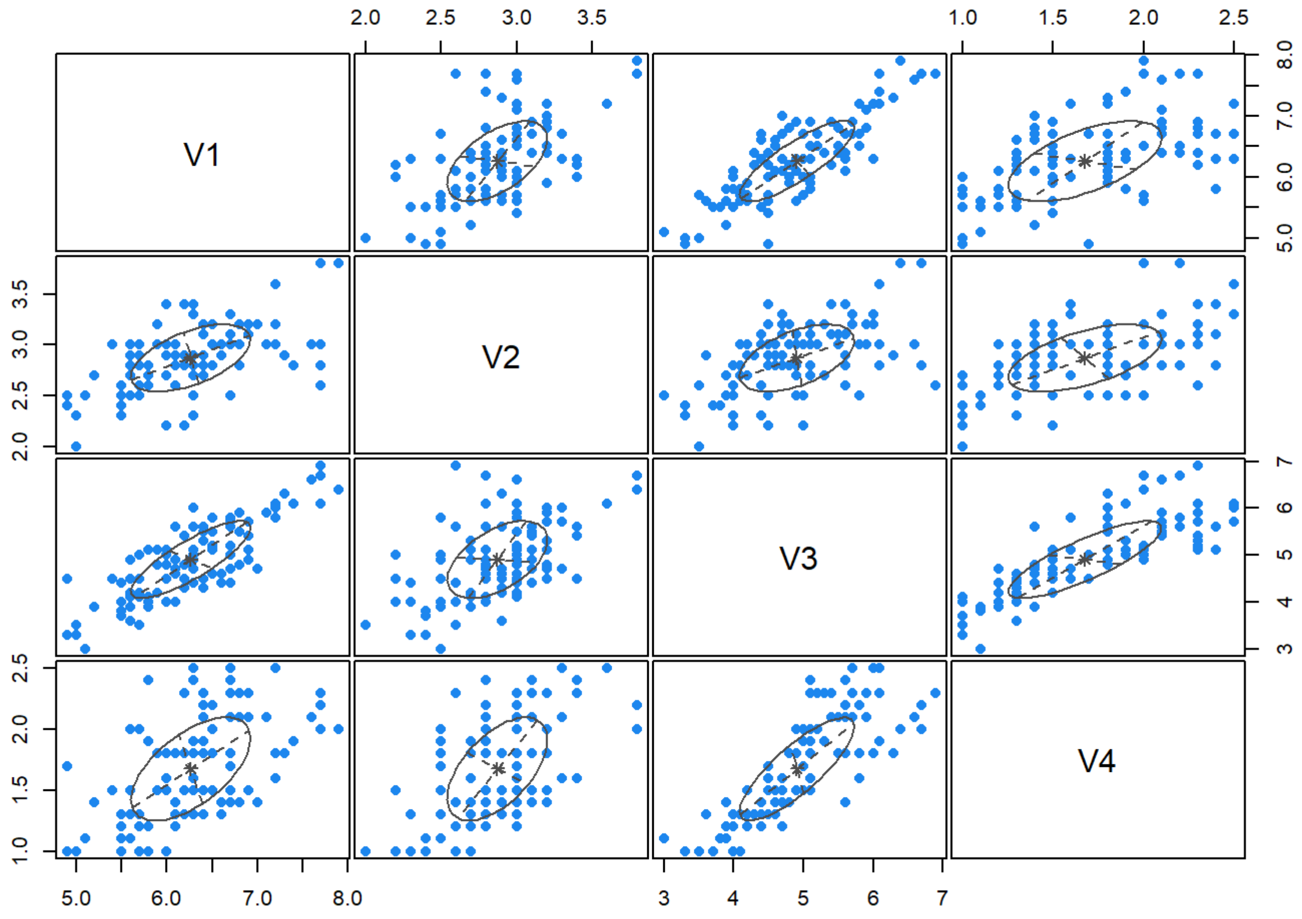
```
set.seed(5)
multi_EM2 = Mclust(multi_data, G = 2, modelNames = "EEE")
multi_EM2$parameters
```

```
## $pro
## [1] 0.4875028 0.5124972
##
## $mean
##        [,1]     [,2]
## V1 5.944076 6.564419
## V2 2.779609 2.959885
## V3 4.256671 5.523662
## V4 1.322206 2.012539
##
## $variance
## $variance$modelName
## [1] "EEE"
##
## $variance$d
## [1] 4
##
## $variance$G
## [1] 2
##
## $variance$sigma
## , , 1
##
##            V1         V2         V3         V4
## V1 0.33880988 0.09299513 0.25245862 0.05849407
## V2 0.09299513 0.10149615 0.08430141 0.04813466
## V3 0.25245862 0.08430141 0.27369805 0.06731898
## V4 0.05849407 0.04813466 0.06731898 0.05955834
##
## , , 2
##
##            V1         V2         V3         V4
## V1 0.33880988 0.09299513 0.25245862 0.05849407
## V2 0.09299513 0.10149615 0.08430141 0.04813466
## V3 0.25245862 0.08430141 0.27369805 0.06731898
## V4 0.05849407 0.04813466 0.06731898 0.05955834
##
##
## $variance$Sigma
##            V1         V2         V3         V4
## V1 0.33880988 0.09299513 0.25245862 0.05849407
## V2 0.09299513 0.10149615 0.08430141 0.04813466
## V3 0.25245862 0.08430141 0.27369805 0.06731898
## V4 0.05849407 0.04813466 0.06731898 0.05955834
##
## $variance$cholSigma
##           V1         V2          V3          V4
## V1 0.5820738  0.1597652  0.43372271  0.10049254
## V2 0.0000000 -0.2756288 -0.05444864 -0.11638644
## V3 0.0000000  0.0000000  0.28743347  0.06052184
## V4 0.0000000  0.0000000  0.00000000 -0.17958533
##
##
## $Vinv
## NULL
```

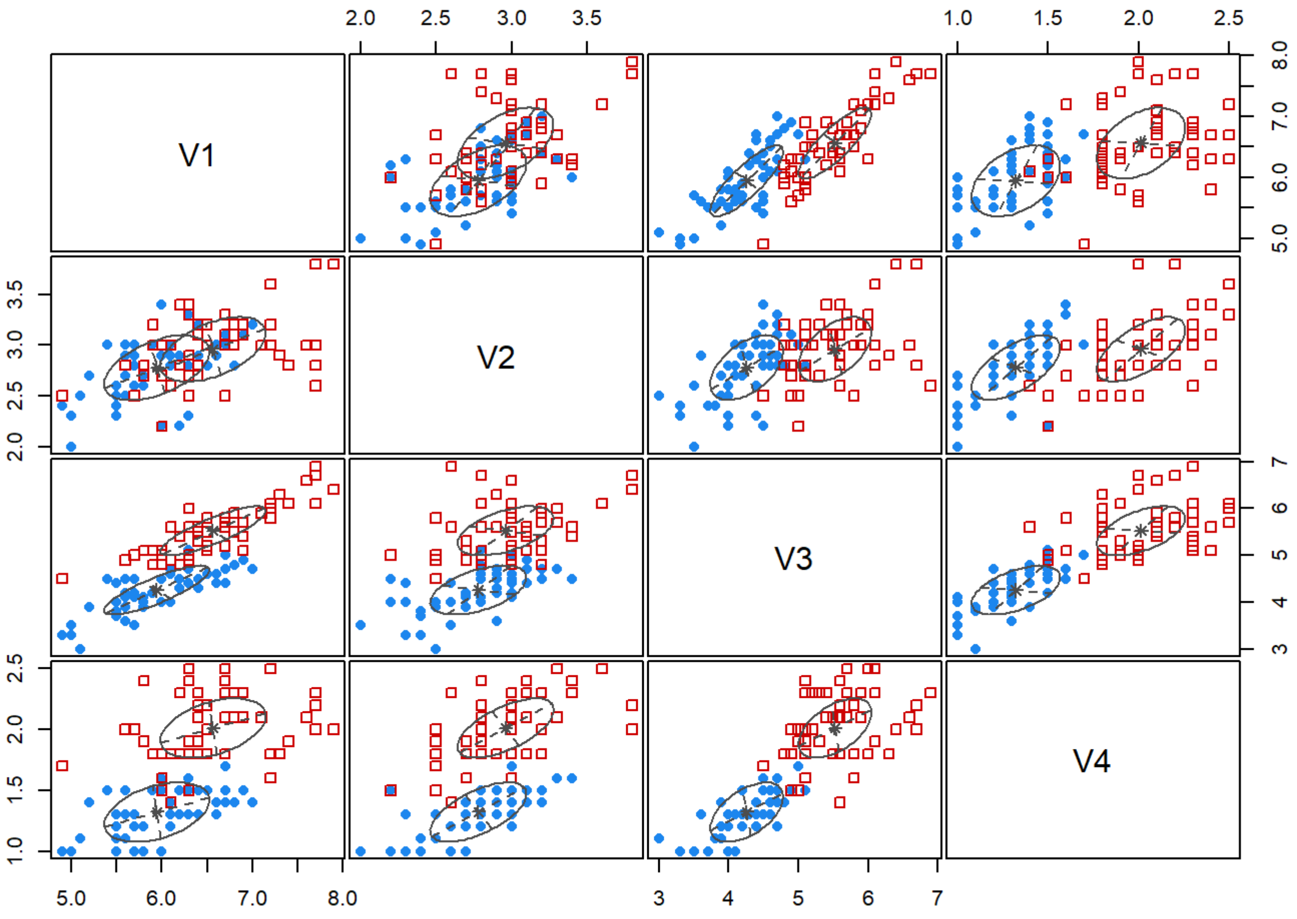
```
set.seed(5)
multi_EM3 = Mclust(multi_data, G = 3, modelNames = "EEE")
multi_EM3$parameters
```

```
## $pro
## [1] 0.3757430 0.5080942 0.1161628
##
## $mean
##        [,1]     [,2]     [,3]
## V1 6.023919 6.566193 5.701565
## V2 2.852898 2.965707 2.523912
## V3 4.363963 5.530623 3.927197
## V4 1.363914 2.016941 1.194213
##
## $variance
## $variance$modelName
## [1] "EEE"
##
## $variance$d
## [1] 4
##
## $variance$G
## [1] 3
##
## $variance$sigma
## , , 1
##
##            V1         V2         V3         V4
## V1 0.33015710 0.08208280 0.24007632 0.05350919
## V2 0.08208280 0.09094240 0.06816016 0.04127411
## V3 0.24007632 0.06816016 0.25484382 0.05929961
## V4 0.05350919 0.04127411 0.05929961 0.05600277
##
## , , 2
##
##            V1         V2         V3         V4
## V1 0.33015710 0.08208280 0.24007632 0.05350919
## V2 0.08208280 0.09094240 0.06816016 0.04127411
## V3 0.24007632 0.06816016 0.25484382 0.05929961
## V4 0.05350919 0.04127411 0.05929961 0.05600277
##
## , , 3
##
##            V1         V2         V3         V4
## V1 0.33015710 0.08208280 0.24007632 0.05350919
## V2 0.08208280 0.09094240 0.06816016 0.04127411
## V3 0.24007632 0.06816016 0.25484382 0.05929961
## V4 0.05350919 0.04127411 0.05929961 0.05600277
##
##
## $variance$Sigma
##            V1         V2         V3         V4
## V1 0.33015710 0.08208280 0.24007632 0.05350919
## V2 0.08208280 0.09094240 0.06816016 0.04127411
## V3 0.24007632 0.06816016 0.25484382 0.05929961
## V4 0.05350919 0.04127411 0.05929961 0.05600277
##
## $variance$cholSigma
##          V1         V2          V3          V4
## V1 0.574593  0.1428538  0.41781979  0.09312539
## V2 0.000000 -0.2655846 -0.03190323 -0.10531780
## V3 0.000000  0.0000000  0.28151843  0.06049338
## V4 0.000000  0.0000000  0.00000000  0.18049693
```

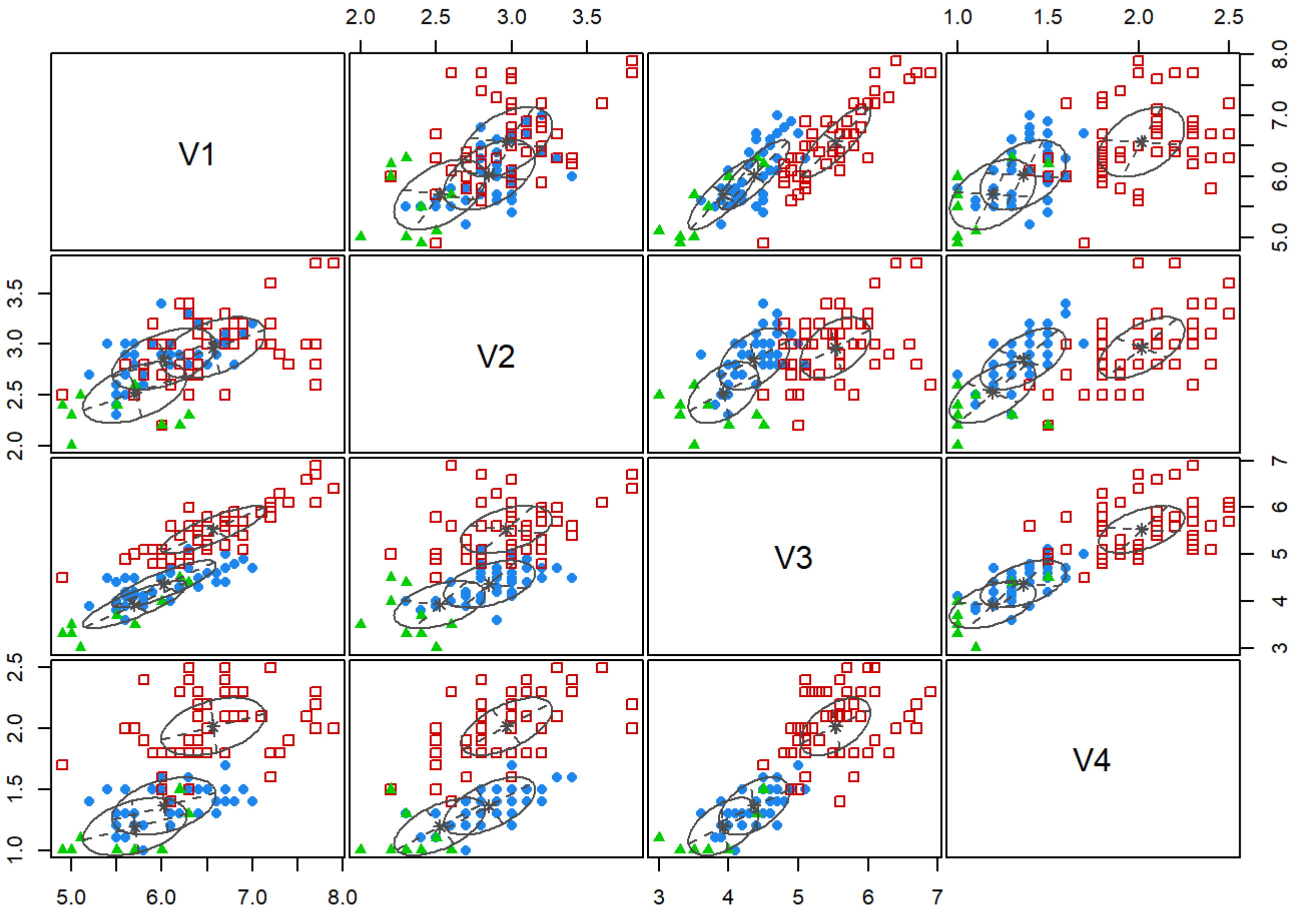
```
plot(multi_EM1, what = "classification")
```



```
plot(multi_EM2, what = "classification")
```



```
plot(multi_EM3, what = "classification")
```



b) ii)

We know that since the null hypothesis is g = 1, the likelihood ratio test statistic does not depend on any unknown parameters. It follows that the bootstrap is effectively sampling from the true null distribution of the LRTS. Therefore, to ensure that the test carried out by mclustBootstrapLRT() is a test of exact size 0.05, we simply set the significance level to 0.05, and ensure that the number of bootstrap replications, B, satisfies k/(B+1) = 0.05 for some integer k. Here, I have chosen k = 999. The p-value obtained is 0.001 < 0.05, strongly suggesting that we should reject the null hypothesis.

```
set.seed(5)
exact_test = mclustBootstrapLRT(multi_data,modelName = "EEE",level = 0.05, maxG = 1, nboot =
999)
exact_test
```

```
## -------------------------------------------------------------
## Bootstrap sequential LRT for the number of mixture components
## -------------------------------------------------------------
## Model        = EEE
## Replications = 999
##              LRTS bootstrap p-value
## 1 vs 2   28.95669             0.001
```

b) iii)

From the results of the likelihood ratio test below, with 99 bootstrap replications, we obtain a p value of 0.99 > 0.05, strongly suggesting that we retain the null hypothesis, i.e g = 2.

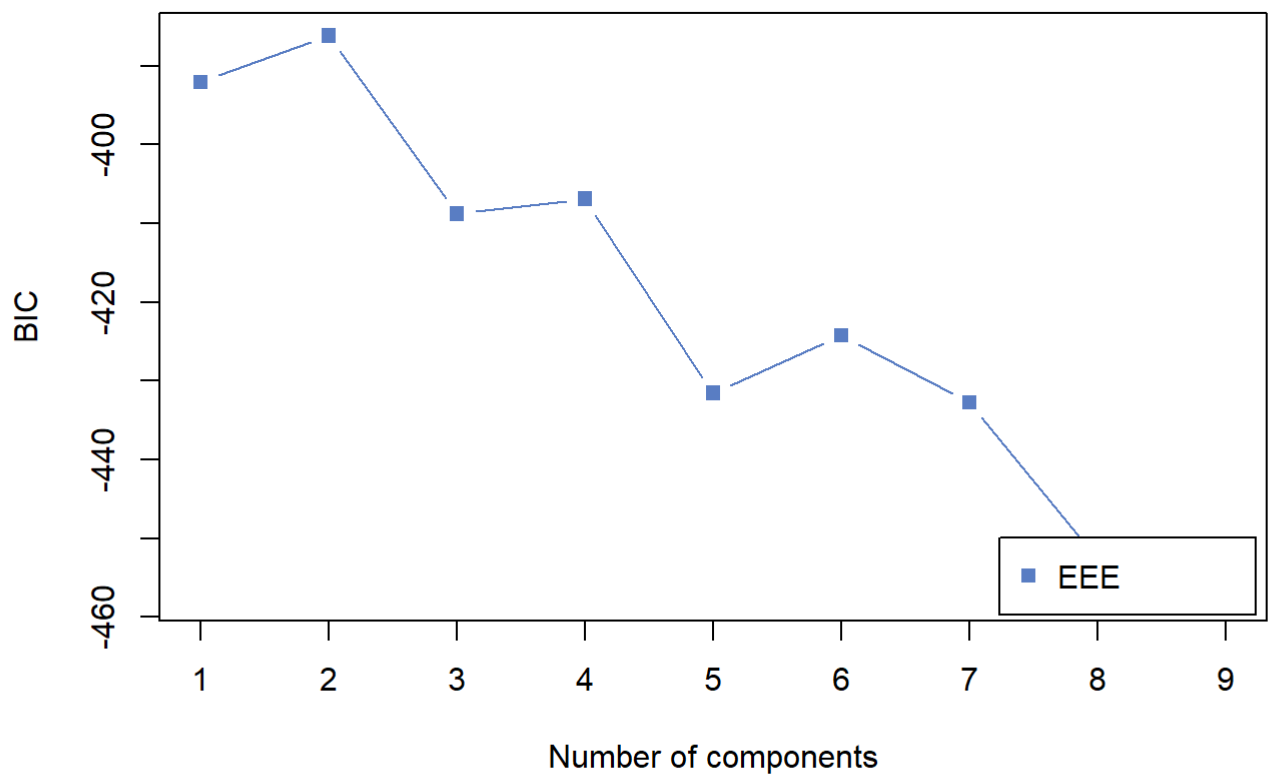
```
set.seed(5)
LRT = mclustBootstrapLRT(multi_data, modelName = "EEE", level = 0.05, nboot = 99, maxG = 2)
LRT
```

```
## -------------------------------------------------------------
## Bootstrap sequential LRT for the number of mixture components
## -------------------------------------------------------------
## Model        = EEE
## Replications = 99
##                LRTS bootstrap p-value
## 1 vs 2   28.9566882              0.01
## 2 vs 3    0.3562638              0.99
```

b) iv)

From the graph below, we see that the BIC suggests that we choose g=2 as the number of components since it attains its largest value at g=2. This agrees with the results in the previous sections.

```
set.seed(5)
BIC = mclustBIC(multi_data, modelNames = "EEE")
plot(BIC)
```


BIC
Number of components
EEE