Assignment 1 - Modern Statistics

Henrik Olaussen

2023-08-25

a)

(i) The EM algorithm is an iterative method used to calculate the maximum likelihood estimates (MLE). Often, the algorithm is used for incomplete-data problems. We denote $\mathbf{w}$ as the observed data $(w_1, ..., w_n)$, $\mathbf{x}$ as the complete data and $\mathbf{z}$ as the additional (or missing) data such that $\mathbf{x} = (\mathbf{w}^T, \mathbf{z}^T)^T$. Moreover, we want to estimate the unknown parameters contained in the vector Ψ by maximum likelihood. The EM algorithm makes use of the complete-data log likelihood function:

$$\log L_c(\Psi) = \log g_c(\mathbf{x}; \Psi)$$

Where g is the pdf of the observed random sample $\mathbf{x}_1, ..\mathbf{x}_n$ from the random vector $\mathbf{X}$, corresponding to the complete data vector.

(ii) However, $L_c(\Psi)$ is unobservable. Hence we use the conditional expectation given $\mathbf{w}$ instead, such that we have a function of $\mathbf{w}$. This leads to the E-step:

$$Q(\Psi; \Psi^{(k)}) = E_{\Psi^{(k)}}[\log L_c(\Psi) \mid \mathbf{w}]$$

Furthermore, after computing the E-step for the current parameter updates (k+1), the EM algorithm maximizes this update. We want to find the values of $\Psi^{(k+1)}$ that optimizes the likelihood function:

$$\Psi^{(k+1)} = \text{argmax} Q(\Psi; \Psi^{(k)})$$

(iii) Now, we want to fit the EM-algorithm to the given two-component normal mixture model with common variances to our n = 75 observations. Firstly, we load dataset 1:

```r
setwd("/Users/henrikolaussen/Desktop/Modern-stat/Assignments/Assignment1")

data1 <- read.csv('Data_A1a_v2.csv', header = TRUE)
```

Secondly, the Mclust-package is used to fit the model with the EM-algorithm:

```r
mod1 <- Mclust(data = data1$x, G = 2, modelNames = 'E') #modelNames = 'E' means equal variance.

mod1.mu <- as.data.frame(mod1$parameters$mean)$`mod1$parameters$mean`

mod1.proportions <- mod1$parameters$pro

mod1.variance <- mod1$parameters$variance$sigmasq
```

The parameter values are:

```
mod1.mu
```

```
## [1] -0.5954022  1.4106478
```

```
mod1.proportions
```

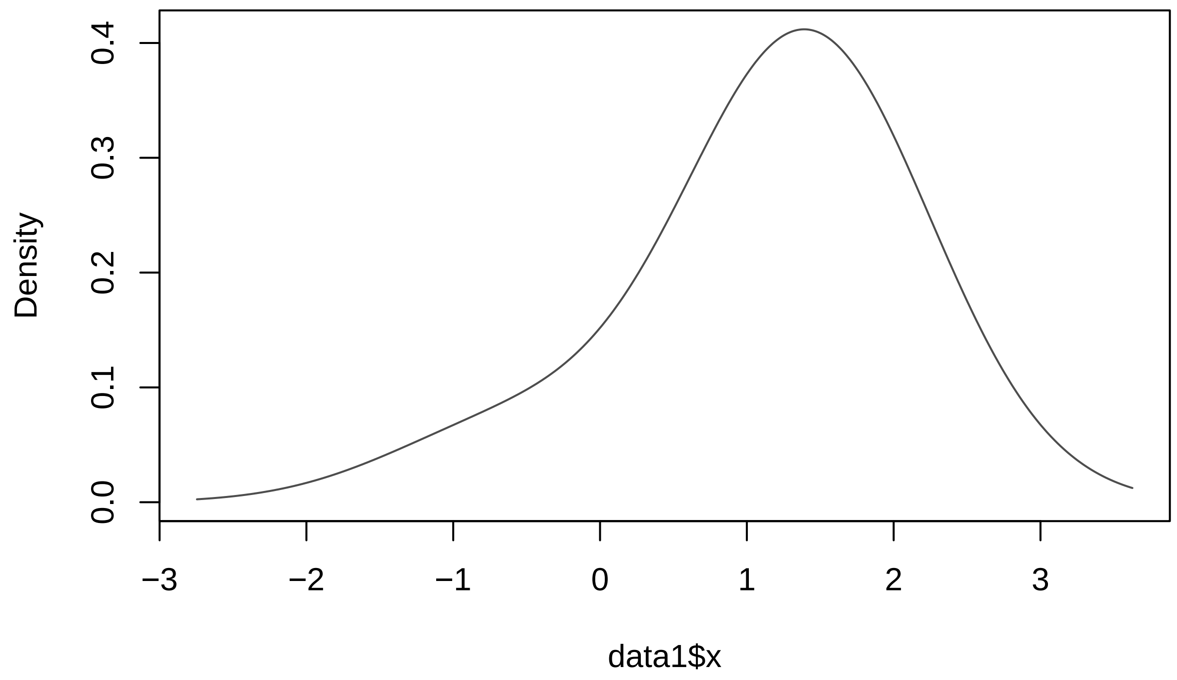
```
## [1] 0.1432968 0.8567032
```

```
mod1.variance
```

```
## [1] 0.7019895
```

Below is a plotting of the density of the normal mixture model.

```
plot(mod1, what = "density")
```



Moreover, the mclust-function initializes the EM-algorithm by hierarchical model-based agglomerative clustering, which can be found in the hc()-function. By default for the multivariate case, a hierarchical agglomerative clustering tree is computed through the hc()-function, whereas the result is used as initialization of the EM-algorithm. In the univariate case, quantiles are used as a default initialization. On the other hand, the hc() function can be called as well, with modelName equal 'V' or 'E', depending on the variance of the model. The function call of hc() would then be as below:

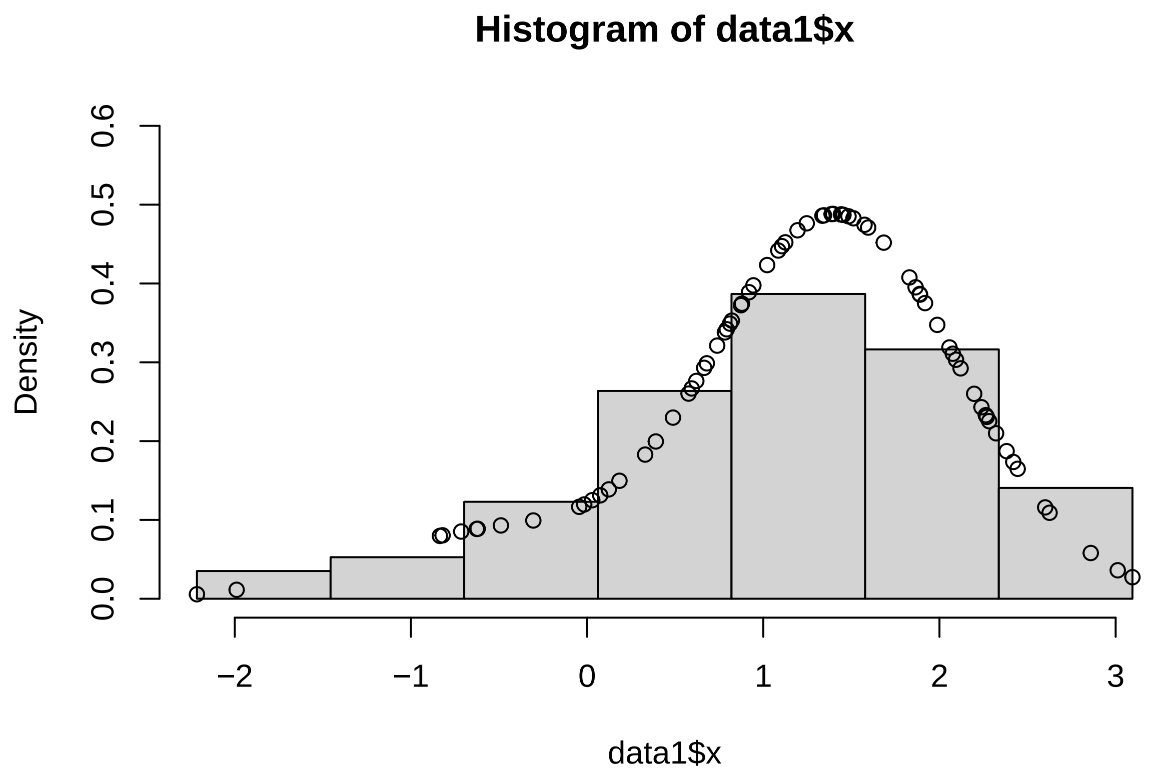
```r
hc_init <- hc(data1$x, modelName = 'E')
hc_init
```

```
## Call:
## hc(data = data1$x, modelName = "E")
##
## Model-Based Agglomerative Hierarchical Clustering
## Model name        = E
## Use               = VARS
## Number of objects = 75
```

For the stopping criterion, the default value is used since the stopping criterion was not specified in the fitting of the model above. From the emControl, the relative convergence tolerance for the log-likelihood is, by default, 1.e-5.

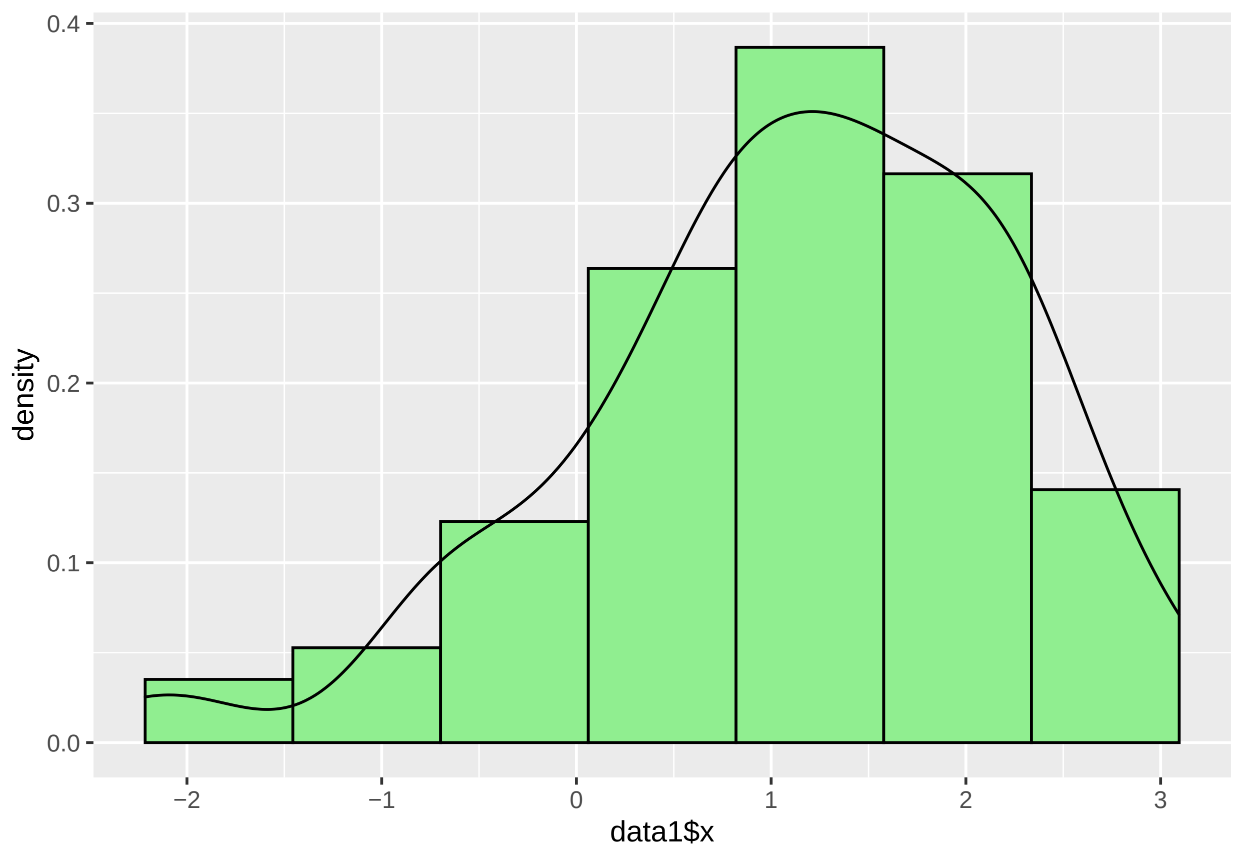
(iv) Furthermore, we want to divide our data into bins and then plot the resulting histogram in the same plot as the density. The number of bins is determined through the following formula: $n = 2^{N-1} \implies N = \log(n)/\log(2) + 1 \approx 7$ bins.

```r
N_bins = 7 #number of bins in histogram
max_data = max(data1$x)
min_data = min(data1$x)

breaks = seq(from = min_data, to = max_data, length.out = N_bins+1)

dens = mod1.proportions[1] * dnorm(data1$x, mean = mod1.mu[1], sd = mod1.variance) + mod1.proportions[2

hist(data1$x, breaks = breaks, prob = TRUE, ylim = c(0, 0.6))
points(data1$x, dens)
```



One can also plot the histogram and density through ggplot. However, using ggplot, the density is different from the model fitted above:

```
df = data.frame(x = data1$x, dens = dens)
ggplot(df) + geom_histogram(breaks = breaks, aes(x = data1$x, y = ..density..), color = "black", fill =
```



(v) Furthermore, one can carry out a chi-squared goodness-of-fit test. Doing this, we test the adequacy of the fit of the fitted model. In a Chi-squared goodness-of-fit test, we test: H_0: There is no significant difference between the observed value and the expected distribution. H_1: There is a significant difference.

The test statistic is:

$$Q = \sum_{j=1}^{N} (o_j - e_j)^2/e_j \sim \chi^2_{N-1-d} \quad \text{under } H_0 : W \sim F_\Psi$$

where $d = \dim(\Psi)$ and F_Ψ is the corresponding distribution of our two-component normal mixture model with density $f(w;\Psi)$. o_j denotes the number of observations in the j'th interval, and e_j denotes the estimated expected frequencies for the N intervals. $(a_j, b_j), j = 1, .., n$ are the boundaries of the j'th interval. We have:

$$e_j \approx n(F(b_j;\hat{\Psi}) - F(a_j;\hat{\Psi})) \quad j = 1, .., N$$

Where:

$$F(t;\hat{\Psi}) = \int_{-\infty}^{t} f(w;\hat{\Psi})dt$$

The test is carried out in the following code-chunk:

```
N = 7
n = 75
d = 4 #number of parameters to be estimated, dim(Psi)
```

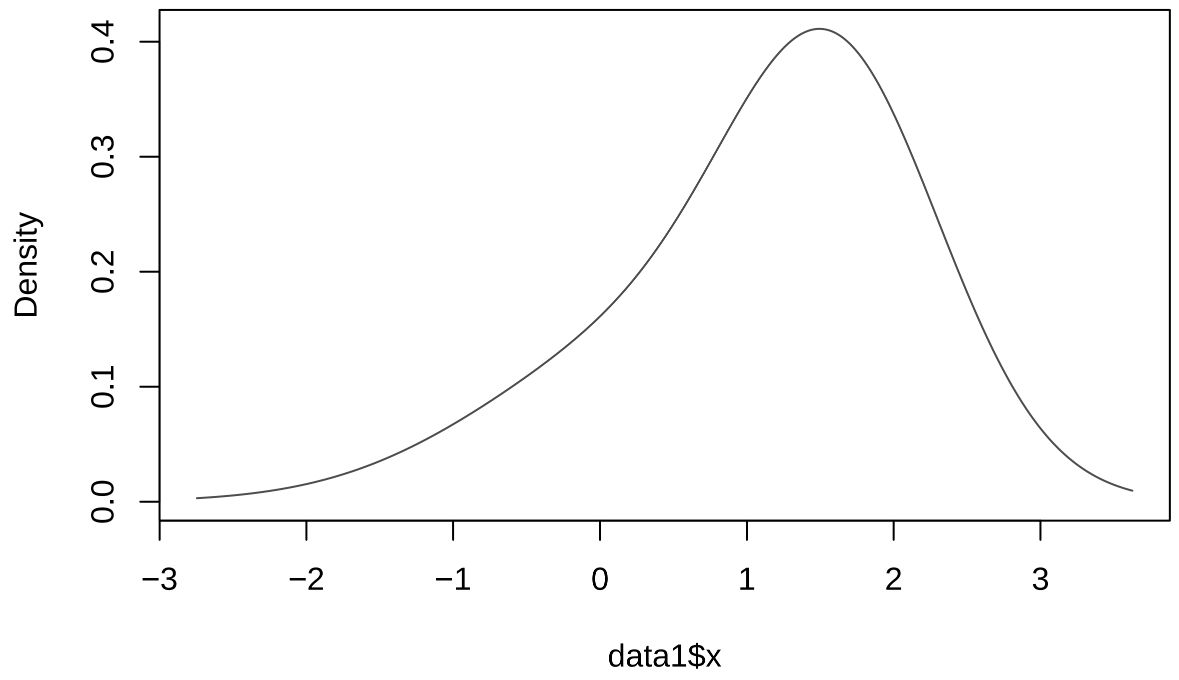
```r
e = list()
o = list()

dens_func <- function(x) {
  mod1.proportions[1] * dnorm(x, mean = mod1.mu[1], sd = mod1.variance) + mod1.proportions[2] * dnorm(x
}

Q = 0
for (i in 1:N) {
  o[[i]] <- length(data1$x[data1$x >= breaks[i] & data1$x <= breaks[i+1]])
  e[[i]] <- n * (integrate(dens_func, -Inf, breaks[i+1])$value - integrate(dens_func, -Inf, breaks[i])$
  Q = Q + (o[[i]]-e[[i]])^2 / e[[i]]
}

p_val <- 1 - pchisq(Q, df = N-1-d)

print(p_val)
```

```
## [1] 0.1785405
```

The p-value is given above. By using the significance level 0.05, the conclusion of the test is that there are no evidence to discard H_0. This means that the observed data is consistent with the fitted two-component normal mixture model.

(vi) Now the problem is expanded to the case where the component variances are arbitrary, meaning they are no longer equal. The new model is fitted and plotted below:

```r
mod2 <- Mclust(data = data1$x, G = 2, modelNames = "V") #V means unequal variances.
plot(mod2, what = "density")
```



The estimated parameter values from the fitted model:

```
mod2$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
```

(vi) Non-parametric bootstrap is applied below to decide the standard errors of the estimated values in 'mod2':

```
np_se = MclustBootstrap(mod2, nboot=100, type="bs")
summary(np_se)
```

```
## ----------------------------------------------------------
## Resampling standard errors
## ----------------------------------------------------------
## Model                        = V
## Num. of mixture components = 2
## Replications                 = 100
## Type                         = nonparametric bootstrap
##
## Mixing probabilities:
##         1         2
## 0.2126267 0.2126267
##
## Means:
##         1         2
## 0.6970875 0.3069287
##
## Variances:
##         1         2
## 0.4487149 0.2219669
```

(vii) Moreover, we can also apply the parametric bootstrap to decide the standard errors:

```
p_se = MclustBootstrap(mod2, nboot = 2, type = 'pb')

summary(p_se, what="se")
```

```
## ----------------------------------------------------------
## Resampling standard errors
## ----------------------------------------------------------
## Model                        = V
## Num. of mixture components = 2
## Replications                 = 2
## Type                         = parametric bootstrap
##
## Mixing probabilities:
##          1          2
## 0.09159739 0.09159739
##
## Means:
##          1          2
## 0.58037159 0.01352464
##
## Variances:
##          1          2
## 0.43192664 0.03037542
```

By comparing the two bootstrapping approaches above, we see that the bound of the parametric bootstrap is tighter than the bound of the non-parametric bootstrap.

b)

(i) Now, we want to fit and plot the normal mixture models with g = 1, g = 2 and g = 3 for a different dataset. The new dataset is loaded in the code-chunk below:

```
data2 <- read.csv('Data_A1b.csv', header = TRUE)
```

Then we fit the three different models:

```
mixture_g1 = Mclust(data = data2[2:5], G = 1, modelNames = "EEE")
mixture_g2 = Mclust(data = data2[2:5], G = 2, modelNames = "EEE")
mixture_g3 = Mclust(data = data2[2:5], G = 3, modelNames = "EEE")
```

The parameters from the three models are:

```
mixture_g1$parameters
```

Model with g = 1:

```
## $pro
## [1] 1
##
## $mean
##     [,1]
## V1 6.210
## V2 2.848
## V3 4.788
## V4 1.639
##
## $variance
## $variance$modelName
## [1] "XXX"
##
## $variance$d
## [1] 4
##
## $variance$G
## [1] 1
##
## $variance$Sigma
##         V1       V2       V3       V4
## V1 0.40670 0.119120 0.419120 0.165710
## V2 0.11912 0.105896 0.147876 0.086128
## V3 0.41912 0.147876 0.615456 0.269468
## V4 0.16571 0.086128 0.269468 0.171379
##
## $variance$cholSigma
##           [,1]       [,2]        [,3]       [,4]
## [1,] 0.6377304  0.1867874  0.65720567  0.2598434
## [2,] 0.0000000 -0.2664704 -0.09426285 -0.1410758
## [3,] 0.0000000  0.0000000  0.41791294  0.2043470
## [4,] 0.0000000  0.0000000  0.00000000  0.2054272
##
## $variance$cholsigma
##           [,1]       [,2]        [,3]       [,4]
## [1,] 0.6377304  0.1867874  0.65720567  0.2598434
## [2,] 0.0000000 -0.2664704 -0.09426285 -0.1410758
## [3,] 0.0000000  0.0000000  0.41791294  0.2043470
## [4,] 0.0000000  0.0000000  0.00000000  0.2054272
##
## $variance$sigma
## , , 1
##
##         V1       V2       V3       V4
## V1 0.40670 0.119120 0.419120 0.165710
## V2 0.11912 0.105896 0.147876 0.086128
## V3 0.41912 0.147876 0.615456 0.269468
## V4 0.16571 0.086128 0.269468 0.171379
```

```
mixture_g2$parameters
```

Model with g = 2:

```
## $pro
## [1] 0.5084774 0.4915226
##
## $mean
##        [,1]     [,2]
## V1 5.881324 6.550014
## V2 2.762712 2.936230
## V3 4.189313 5.407339
## V4 1.311575 1.977719
##
## $variance
## $variance$modelName
## [1] "EEE"
##
## $variance$d
## [1] 4
##
## $variance$G
## [1] 2
##
## $variance$sigma
## , , 1
##
##            V1         V2         V3         V4
## V1 0.29494557 0.09012081 0.21555807 0.05438108
## V2 0.09012081 0.09837099 0.09505366 0.05723923
## V3 0.21555807 0.09505366 0.24466571 0.06668115
## V4 0.05438108 0.05723923 0.06668115 0.06047398
```

```
##
## , , 2
##
##            V1         V2         V3         V4
## V1 0.29494557 0.09012081 0.21555807 0.05438108
## V2 0.09012081 0.09837099 0.09505366 0.05723923
## V3 0.21555807 0.09505366 0.24466571 0.06668115
## V4 0.05438108 0.05723923 0.06668115 0.06047398
##
##
## $variance$Sigma
##            V1         V2         V3         V4
## V1 0.29494557 0.09012081 0.21555807 0.05438108
## V2 0.09012081 0.09837099 0.09505366 0.05723923
## V3 0.21555807 0.09505366 0.24466571 0.06668115
## V4 0.05438108 0.05723923 0.06668115 0.06047398
##
## $variance$cholSigma
##           V1         V2         V3          V4
## V1 0.5430889  0.1659412  0.3969112  0.10013293
## V2 0.0000000 -0.2661476 -0.1096751 -0.15263358
## V3 0.0000000  0.0000000  0.2740412  0.03721037
## V4 0.0000000  0.0000000  0.0000000 -0.16051715
```

```
mixture_g3$parameters
```

Model with g = 3:

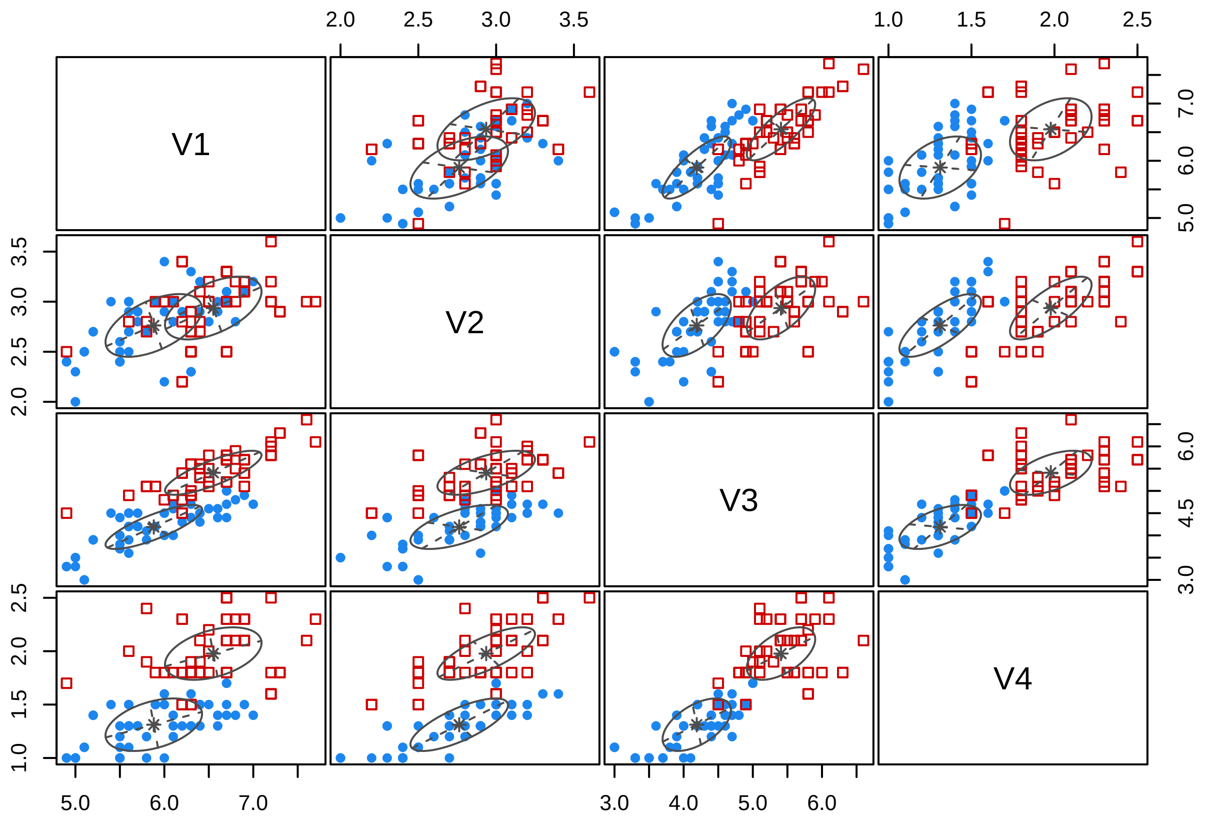
```
## $pro
## [1] 0.3702236 0.4886247 0.1411517
##
## $mean
##        [,1]     [,2]     [,3]
## V1 6.068854 6.549804 5.403910
## V2 2.887492 2.938459 2.431273
## V3 4.381564 5.410547 3.698961
## V4 1.382929 1.980760 1.127573
##
## $variance
## $variance$modelName
## [1] "EEE"
##
## $variance$d
## [1] 4
##
## $variance$G
## [1] 3
##
## $variance$sigma
## , , 1
##
```

```
##            V1         V2         V3         V4
## V1 0.25118689 0.05874864 0.17060400 0.03739361
## V2 0.05874864 0.07680768 0.06224248 0.04468304
## V3 0.17060400 0.06224248 0.19751788 0.04835935
## V4 0.03739361 0.04468304 0.04835935 0.05311198
##
## , , 2
##
##            V1         V2         V3         V4
## V1 0.25118689 0.05874864 0.17060400 0.03739361
## V2 0.05874864 0.07680768 0.06224248 0.04468304
## V3 0.17060400 0.06224248 0.19751788 0.04835935
## V4 0.03739361 0.04468304 0.04835935 0.05311198
##
## , , 3
##
##            V1         V2         V3         V4
## V1 0.25118689 0.05874864 0.17060400 0.03739361
## V2 0.05874864 0.07680768 0.06224248 0.04468304
## V3 0.17060400 0.06224248 0.19751788 0.04835935
## V4 0.03739361 0.04468304 0.04835935 0.05311198
##
##
## $variance$Sigma
##            V1         V2         V3         V4
## V1 0.25118689 0.05874864 0.17060400 0.03739361
## V2 0.05874864 0.07680768 0.06224248 0.04468304
## V3 0.17060400 0.06224248 0.19751788 0.04835935
## V4 0.03739361 0.04468304 0.04835935 0.05311198
##
## $variance$cholSigma
##            V1         V2          V3          V4
## V1 -0.5011855 -0.1172194 -0.34040092 -0.07461032
## V2  0.0000000 -0.2511320 -0.08896079 -0.14310110
## V3  0.0000000  0.0000000 -0.27153466 -0.03768043
## V4  0.0000000  0.0000000  0.00000000  0.16014851
```

Furthermore, we can plot the result of g = 2 and g = 3.

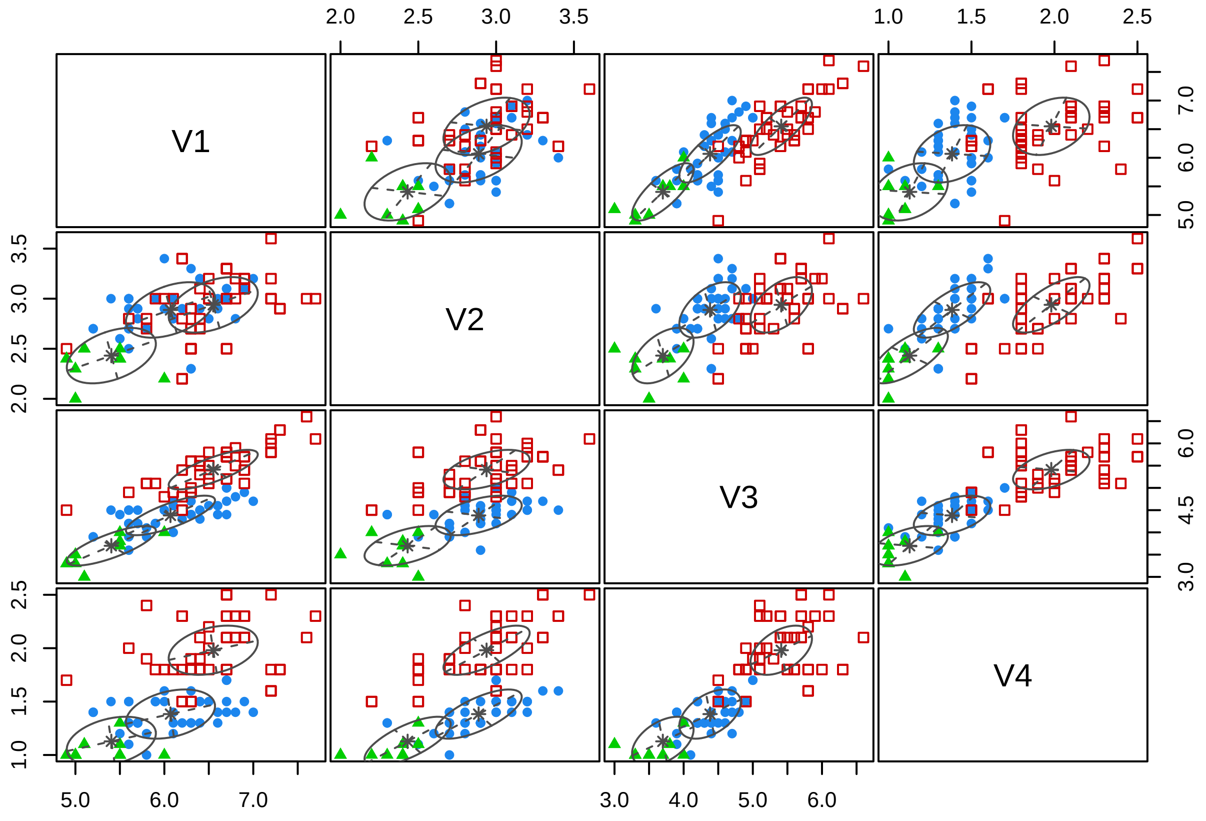
g = 2 plot:

```
plot(mixture_g2, what = "classification")
```



g = 3 plot:

```
plot(mixture_g3, what = "classification")
```



(ii) Next, the test H0: g = 1 vs H1: g = 2, is performed by bootstrapping the LRTS. This is done below:

```
LRT_test = mclustBootstrapLRT(data = data2[2:5], modelName = 'EEE')

LRT_test
```

```
## -------------------------------------------------------------
## Bootstrap sequential LRT for the number of mixture components
## -------------------------------------------------------------
## Model        = EEE
## Replications = 999
##                LRTS bootstrap p-value
## 1 vs 2    38.288410             0.001
## 2 vs 3     3.073707             0.860
```

We have that the p-value is below the significance level 0.05. Hence, there are evidence that we can discard H0, meaning that we should go with g = 2.

(iii) Now, we bootstrap with B=99 bootstrap replications and perform the test $H_0 : g = 2$ vs $H_1 : g = 3$:

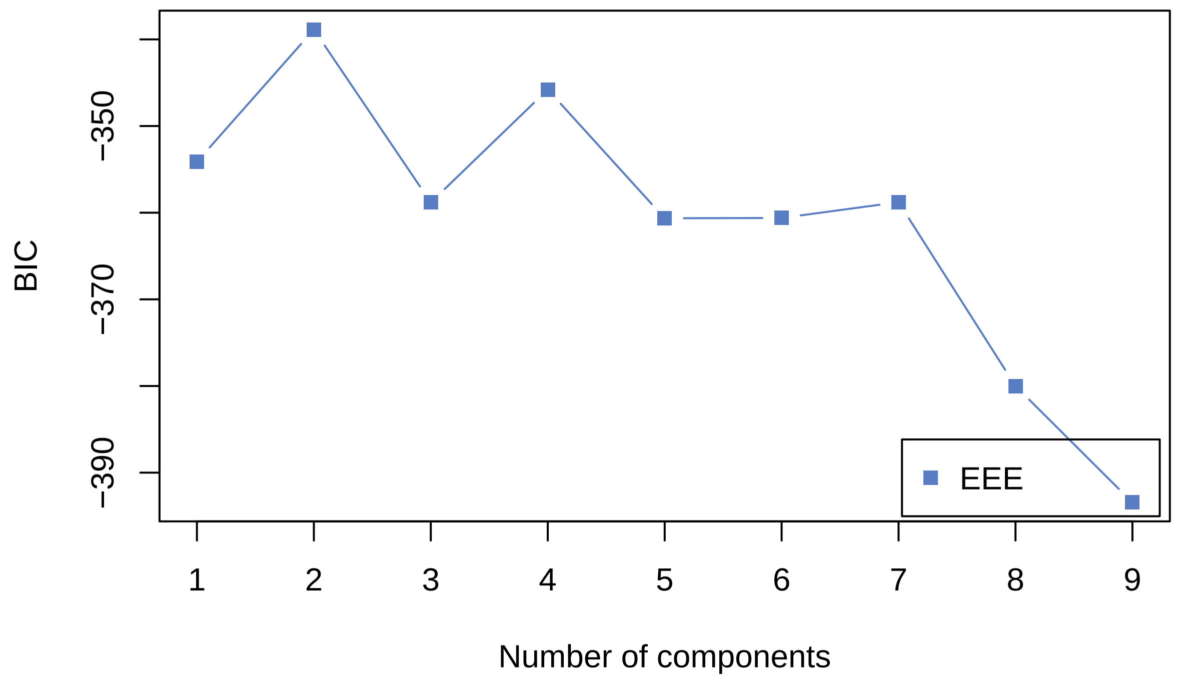
```
LRT_test2 = mclustBootstrapLRT(data = data2[2:5], modelName = 'EEE', nboot = 99)

LRT_test2
```

```
## -------------------------------------------------------------
## Bootstrap sequential LRT for the number of mixture components
## -------------------------------------------------------------
## Model        = EEE
## Replications = 99
##                LRTS bootstrap p-value
## 1 vs 2   38.288410              0.01
## 2 vs 3    3.073707              0.82
```

We have that the p-value is above 0.05. Hence there are no evidence that H_0 should be discarded. Hence, with this test, we should go with g = 2, like we did in (ii).

(iv) Lastly, the BIC is used to choose between the models with g=2 and g=3 component normal mixtures.

```
BIC_test <- mclustBIC(data = data2[2:5], modelNames = 'EEE')
plot(BIC_test)
```



Since we use the mClust package, we choose the model with the maximum BIC value. Normally, one would minimize the BIC. However, mClust uses a different formula, such that we look to maximize the BIC. Furthermore, we choose the modelNames = 'EEE', since we want common covariance matrices. According to the test above, as we want to choose the largest BIC, we see that g = 2 components are favorable to g = 3 components.