Multivariate_Assignment2

Aniket Guha Roy-19200164

4/17/2020

```
if(!require(poLCA))
{
install.packages("poLCA")
library(poLCA)
}
```

```
## Loading required package: poLCA
```

```
## Warning: package 'poLCA' was built under R version 3.6.3
```

```
## Loading required package: scatterplot3d
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 3.6.2
```

```
if(!require(MASS))
{
install.packages("MASS")
library(MASS)
}
if(!require(vegan))
{
install.packages("vegan")
library(vegan)
}
```

```
## Loading required package: vegan
```

```
## Warning: package 'vegan' was built under R version 3.6.3
```

```
## Loading required package: permute
```

```
## Warning: package 'permute' was built under R version 3.6.3
```

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

```
## This is vegan 2.5-6
```

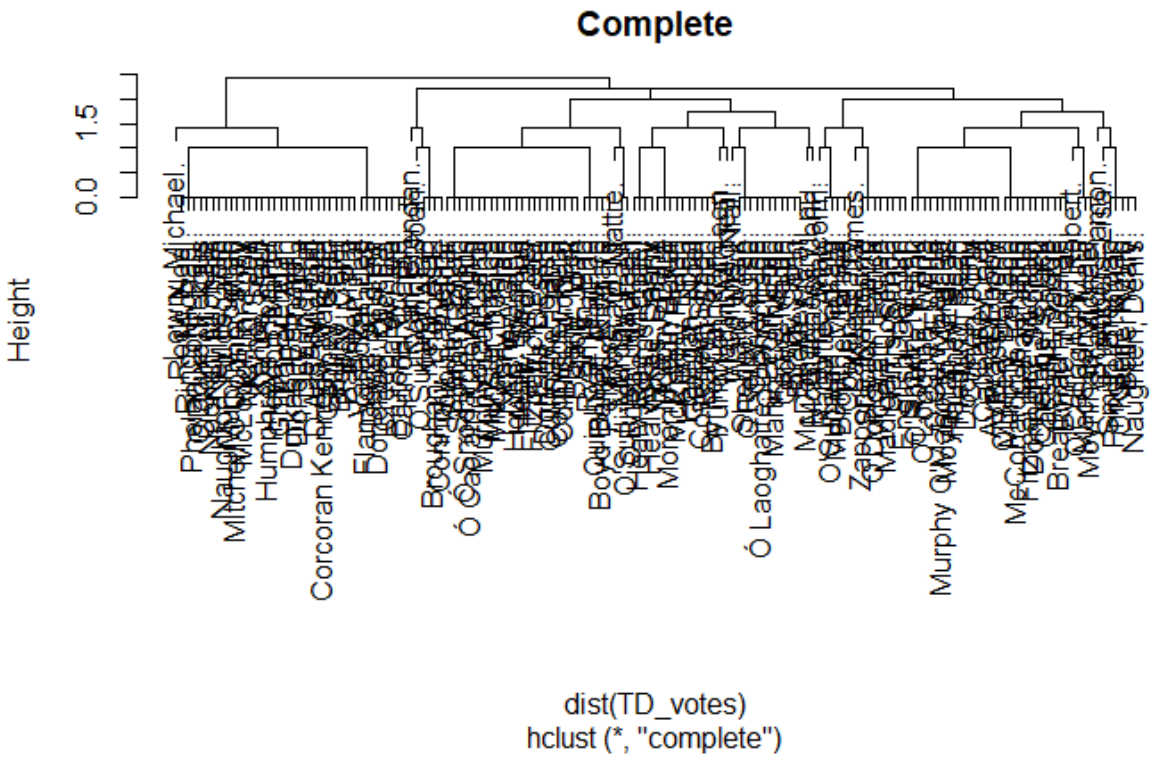
```
if(!require(e1071))
{
install.packages("e1071")
library(e1071)
}
```

```
## Loading required package: e1071

## Warning: package 'e1071' was built under R version 3.6.2

if(!require(mclust))
{
install.packages("mclust")
library(mclust)
}

## Loading required package: mclust

## Warning: package 'mclust' was built under R version 3.6.3

## Package 'mclust' version 5.4.5
## Type 'citation("mclust")' for citing this R package in publications.
```

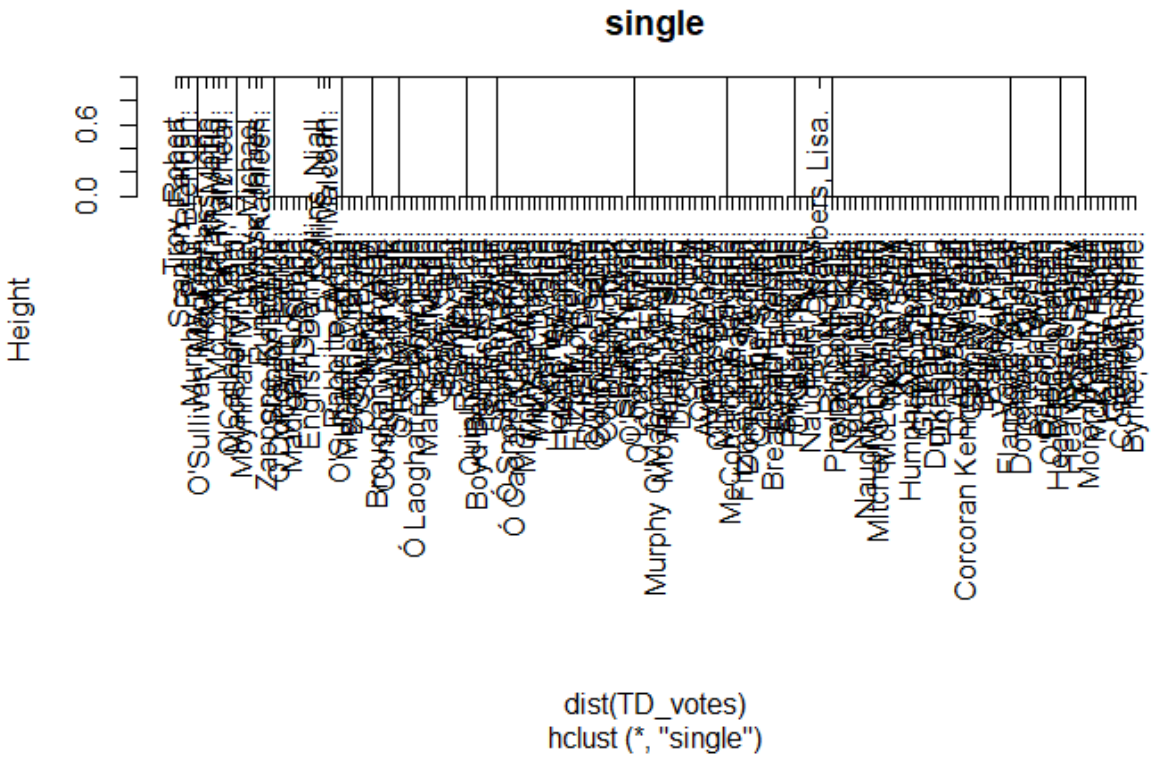
##Question 1

Before we apply hierarchical clustering, the response of votes recorded as “Yes” has been converted to 0 and “No” has been converted to 1. #Question 1.a)

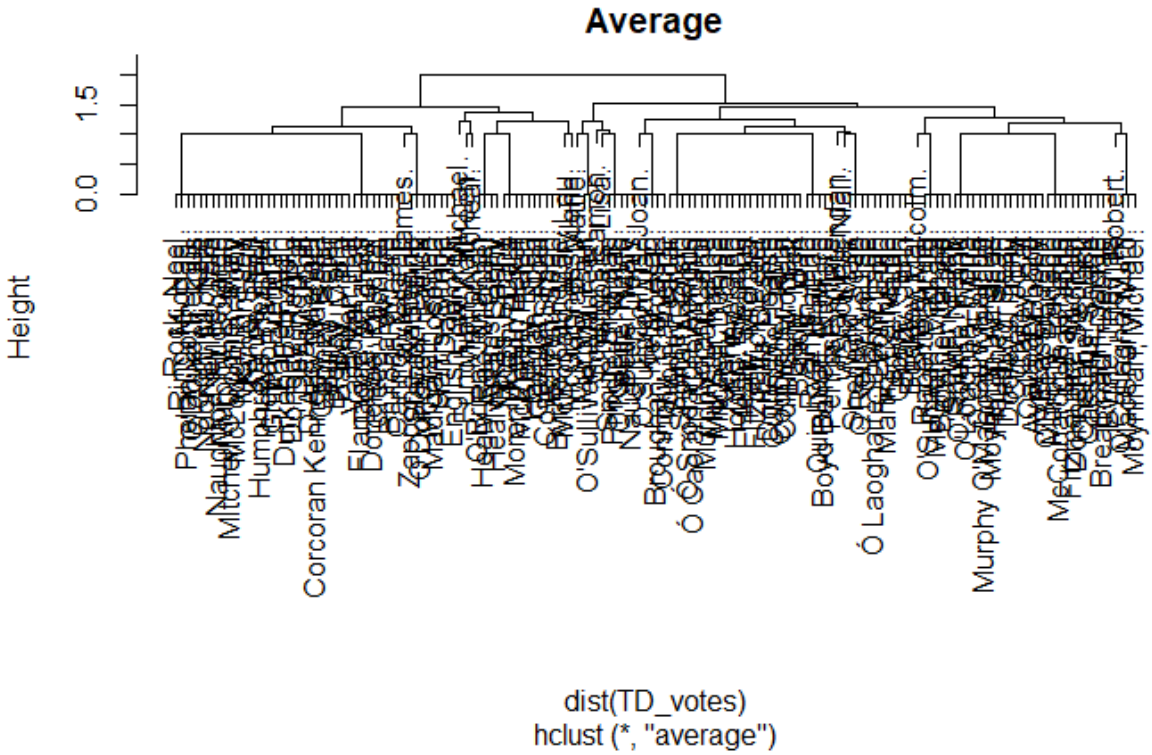
```
##Hclustering
TD.complete<-hclust(dist(TD_votes),method = "complete")
plot(TD.complete, main = "Complete")
```




```
TD.single<-hclust(dist(TD_votes),method = "single")
plot(TD.single, main = "single")
```




```
TD.average<-hclust(dist(TD_votes),method = "average")
plot(TD.average,main = "Average")
```



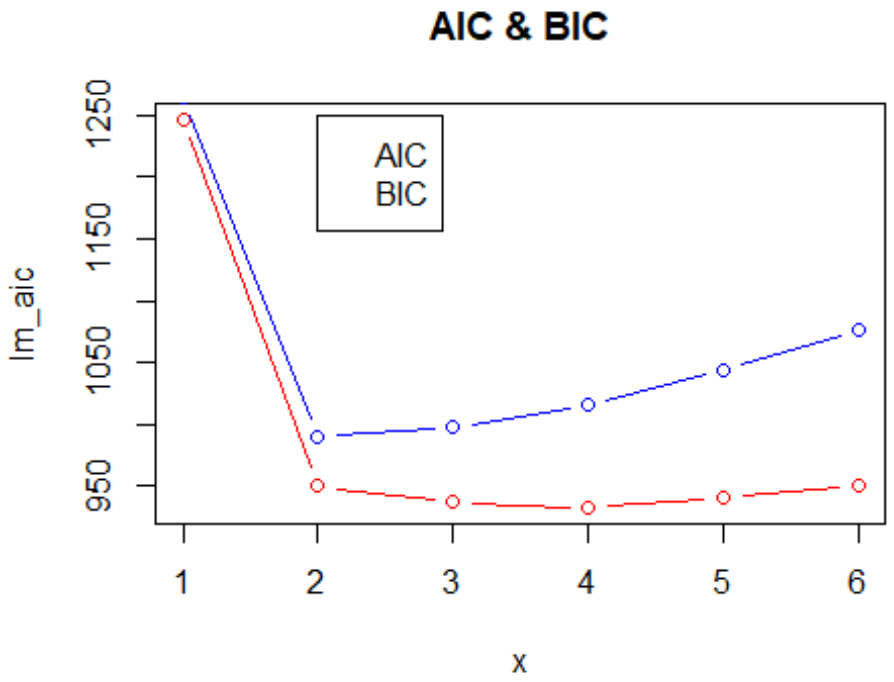

```
TD_average = cutree(TD.average,k=2)
```

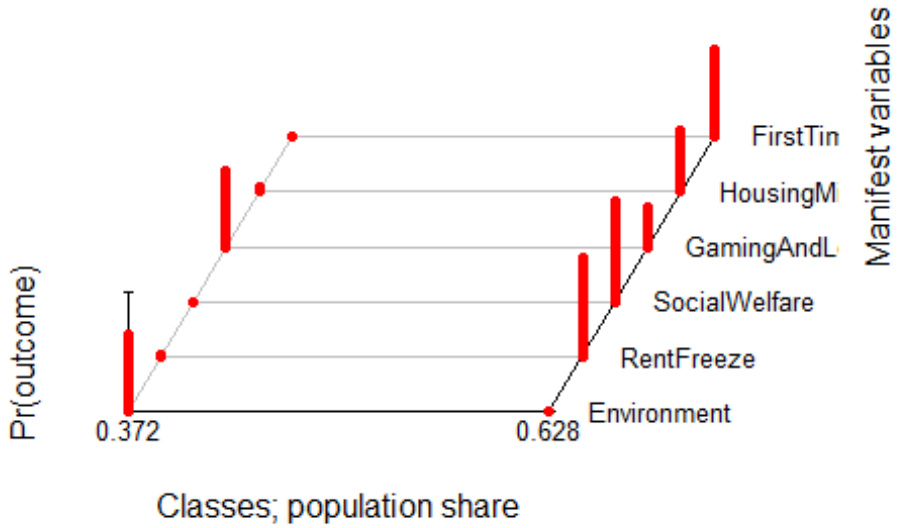
From the dendograms of the three different types of hierarchical clustering methods, we get to know that the average method seems to be the best as it classifies into two distinct clusters. In the single method, we can observe chaining while in complete method, there are numerous cluters with no clear distinctions.

#Question 1b)

```
## Conditional item response (column) probabilities,
##  by outcome variable, for each class (row)
##
## $Environment
##            Pr(1)  Pr(2)
## class 1:  0.3452 0.6548
## class 2:  1.0000 0.0000
##
## $RentFreeze
##            Pr(1)  Pr(2)
## class 1:  0.9739 0.0261
## class 2:  0.1683 0.8317
##
## $SocialWelfare
##           Pr(1) Pr(2)
## class 1:  1.000 0.000
## class 2:  0.163 0.837
##
## $GamingAndLotteries
##            Pr(1)  Pr(2)
## class 1:  0.3572 0.6428
## class 2:  0.6764 0.3236
##
## $HousingMinister
##            Pr(1)  Pr(2)
## class 1:  0.9669 0.0331
## class 2:  0.4786 0.5214
##
## $FirstTimeBuyers
##            Pr(1)  Pr(2)
## class 1:  1.0000 0.0000
## class 2:  0.2651 0.7349
##
## Estimated class population shares
##  0.372 0.628
##
## Predicted class memberships (by modal posterior prob.)
##  0.359 0.641
##
## =========================================================
## Fit for 2 latent classes:
## =========================================================
## number of observations: 156
```

```
## number of estimated parameters: 13
## residual degrees of freedom: 50
## maximum log-likelihood: -462.0166
##
## AIC(2): 950.0332
## BIC(2): 989.6813
## G^2(2): 58.95796 (Likelihood ratio/deviance statistic)
## X^2(2): 67.84483 (Chi-square goodness of fit)
##
```





The puropose of Latent Class Analysis is to classify features into latent groups/classes. LCA_best_model stores the model with the lowest BIC after sequentially testing every model from 1 to 6 classes with 5000 repetitions.We find that the BIC score for class 2 is the lowest and hence

is stored as the LCA_best_model. It is also evident from the AIC and BIC graph where we notice that the class 2 has the lowest AIC and BIC values.

#Question 1c)

```
tab = table(TD_average,lca_2$predclass)
tab
```

```
##
## TD_average  1  2
##          1 56  9
##          2  0 91
```

```
classAgreement(tab)
```

```
## $diag
## [1] 0.9423077
##
## $kappa
## [1] 0.8789238
##
## $rand
## [1] 0.8905707
##
## $crand
## [1] 0.7807944
```

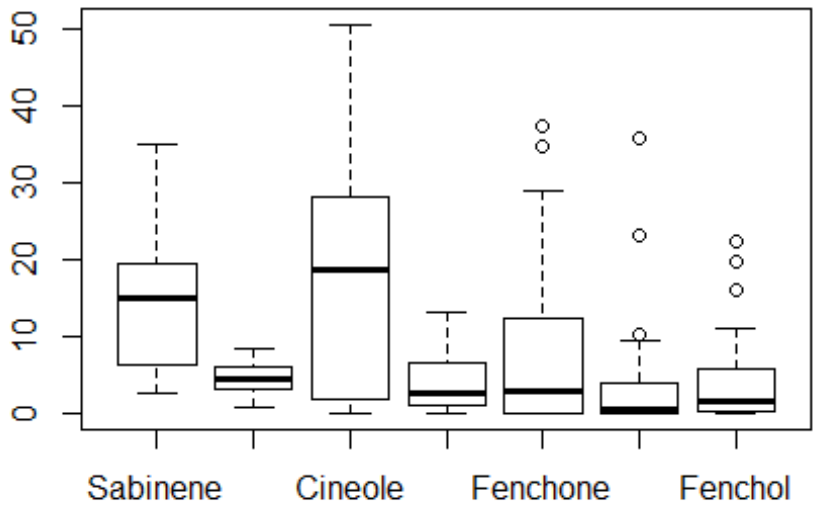
For models having two clusters, Rand index of 0.89 is quite high, indicating a high similarity between the hierarchical model and latent class analysis model. The Adjusted Rand index which takes randomness into account is also quite high at 0.78 confirming that both the models are quite similar. Question 1d)

##Question 2

```
plant = read.csv(file.choose(),header = TRUE)
terpenes = plant[,1:7]
summary(plant)
```

```
##     Sabinene          Pinene         Cineole         Terpinene
##  Min.   : 2.670   Min.   :0.740   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 6.263   1st Qu.:3.192   1st Qu.: 3.055   1st Qu.: 1.005
##  Median :14.965   Median :4.335   Median :18.810   Median : 2.650
##  Mean   :14.004   Mean   :4.611   Mean   :17.998   Mean   : 4.281
##  3rd Qu.:19.005   3rd Qu.:5.960   3rd Qu.:28.192   3rd Qu.: 6.435
##  Max.   :35.000   Max.   :8.430   Max.   :50.620   Max.   :13.050
##     Fenchone       Terpinolene        Fenchol           Location
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   East-high: 6
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.220   East-low : 5
##  Median : 2.840   Median : 0.545   Median : 1.450   North    : 9
##  Mean   : 8.464   Mean   : 3.807   Mean   : 4.485   South    :10
```

```
##  3rd Qu.:11.745   3rd Qu.: 3.908   3rd Qu.: 5.730
##  Max.   :37.430   Max.   :35.790   Max.   :22.470

boxplot(terpenes)
```

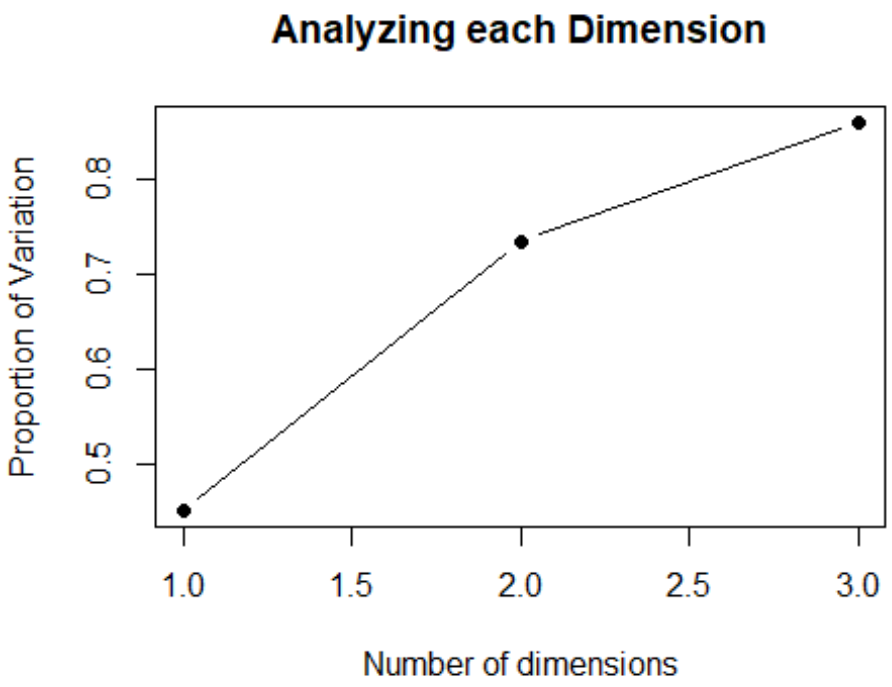



```
head(terpenes)

##   Sabinene Pinene Cineole Terpinene Fenchone Terpinolene Fenchol
## 1     6.22   4.11   22.31      5.42     4.72           0   16.12
## 2    14.46   5.93   28.17      2.34     4.74           0    5.34
## 3     5.13   3.82   21.87      8.33     4.48           0   16.18
## 4     3.09   0.74    0.78     13.05     1.20           0   22.47
## 5    23.98   4.38   11.49      2.69    12.36           0   11.02
## 6    19.61   3.17    6.61      3.47    12.27           0   19.67
```

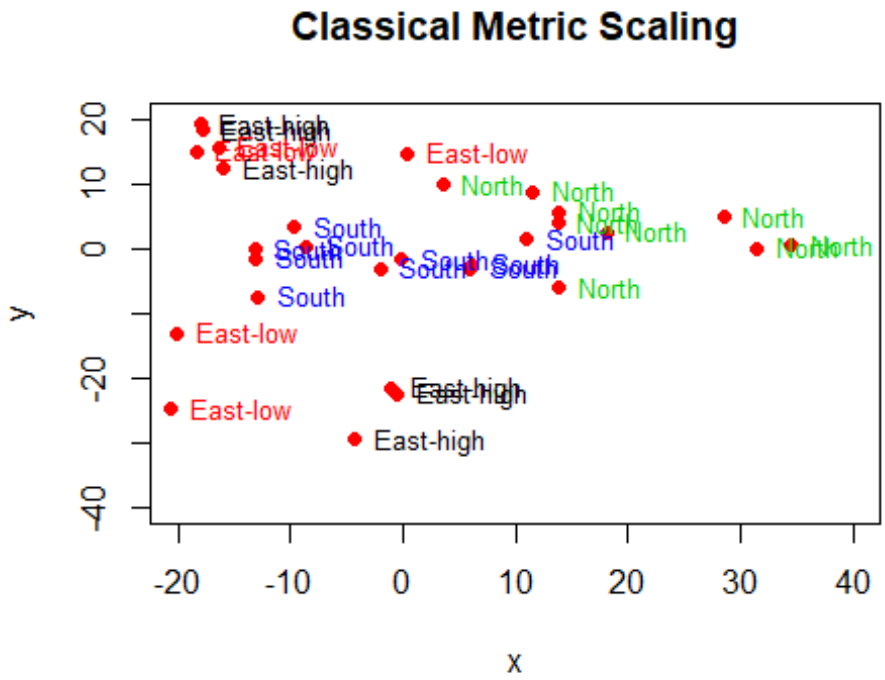
#Question 2 a) Classical Metric Scaling

```
cms = cmdscale(dist(terpenes),k=3,eig=TRUE)
cms

## $points
##              [,1]         [,2]        [,3]
##  [1,]   6.1674514  -2.51067441  -0.7399387
##  [2,]  11.0126932   1.49734771  -3.0777080
##  [3,]   5.9781805  -3.09292912  -0.3714343
##  [4,] -13.1729765  -0.17917709  -0.6027884
##  [5,]  -8.6051084   0.39199628 -10.4998927
##  [6,] -13.0158006  -1.56999138  -9.8392007
```

```
##  [7,] -12.8475582  -7.51294584  -8.2459563
##  [8,]  -2.0023185  -3.09193139  -4.4361310
##  [9,]  -9.6046097   3.36175622  -4.5402849
## [10,]  -0.1909021  -1.72693699  -4.7763259
## [11,]  -4.2383088 -29.43970984   5.1318018
## [12,]  -0.4876885 -22.51389441   4.8428270
## [13,]  -1.0394347 -21.60189993   5.9921620
## [14,] -15.9703333  12.34119238   4.4066451
## [15,] -18.0238554  19.25230551  -9.2660142
## [16,] -17.7935872  18.33362820 -11.5983340
## [17,] -20.5958399 -24.87258891   0.9451763
## [18,] -20.0971677 -13.12065301  -5.6811443
## [19,] -18.3609491  15.08484070  30.4069476
## [20,] -16.3963837  15.69894049  15.0411396
## [21,]   0.3499589  14.78995948  -1.2640114
## [22,]  13.8160385   5.72421580  -2.2865637
## [23,]   3.5110685   9.87666813   3.0108348
## [24,]  11.5143263   8.86162659  -5.4314002
## [25,]  13.8649697  -6.00970567   8.2572314
## [26,]  13.7592148   3.99299453   1.4553962
## [27,]  18.0957082   2.58794487   5.3803122
## [28,]  31.4158529   0.04199676   0.5903844
## [29,]  34.4659746   0.51503730   1.0016486
## [30,]  28.4913849   4.89058703  -3.8053780
##
## $eig
##  [1]  7.254790e+03  4.575679e+03  2.018634e+03  1.715059e+03  3.383429e+02
##  [6]  2.077295e+02  6.598575e+00  1.124236e-12  5.843278e-13  5.021714e-13
## [11]  2.070325e-13  2.021610e-13  1.994141e-13  1.658658e-13  1.611267e-13
## [16]  1.523798e-13  1.056488e-13  7.038776e-14  6.607250e-14  4.135444e-14
## [21]  6.603705e-15 -1.245242e-14 -5.646047e-14 -9.058943e-14 -1.038655e-13
## [26] -1.343232e-13 -1.898157e-13 -2.282284e-13 -2.720312e-13 -4.476032e-13
##
## $x
## NULL
##
## $ac
## [1] 0
##
## $GOF
## [1] 0.8592943 0.8592943

q1 = sum(abs(cms$eig[1]))/sum(abs(cms$eig))
q2 = sum(abs(cms$eig[1:2]))/sum(abs(cms$eig))
q3 = sum(abs(cms$eig[1:3]))/sum(abs(cms$eig))

plot(c(q1,q2,q3),col = 1,pch =19,type = "b",main="Analyzing each Dimension",
     xlab="Number of dimensions",ylab="Proportion of Variation")
```




```
x = cms$points[,1]
y = cms$points[,2]
{plot(x,y,col =2, pch = 19, xlim = c(-20,40), ylim = c(-40,20), main =
"Classical Metric Scaling")
text(x,y,plant$Location,pos =4, col = as.numeric(plant$Location), cex = 0.8)
}
```



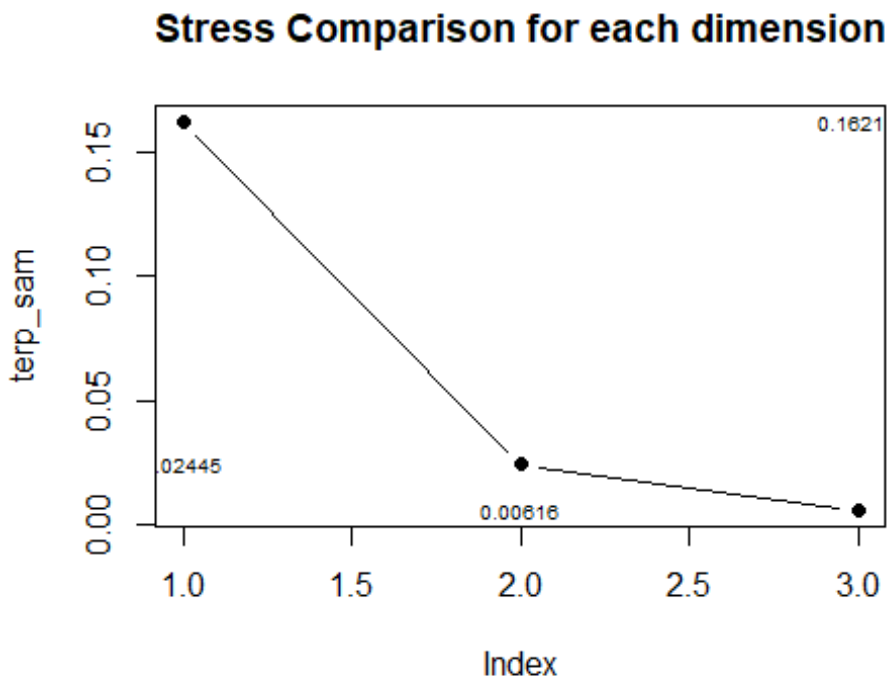
We can see that the proportion of variation explained by 2 dimensions is much more when compared to one dimension.The steep slope indicates the significant increase in variation compared to 3rd dimension as well. Hence, we select class 2 for our purpose.

#Question 2 b) Sammon's metric least squares scaling

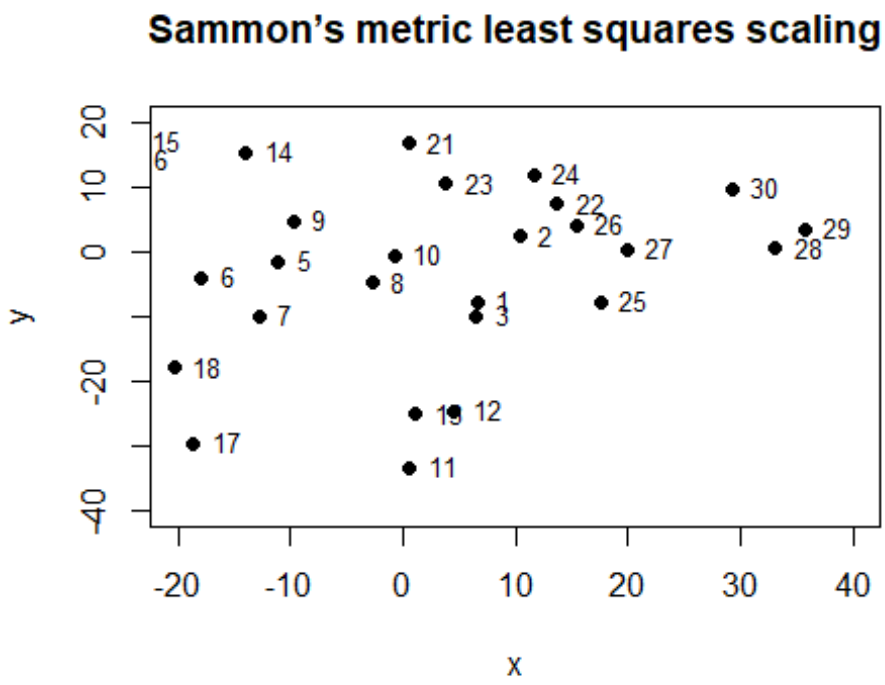
```
terp_sam = rep(0,3)
for(i in seq(3)){
terp_sam[i] = sammon(dist(terpenes),k=i)$stress
}
```

```
## Initial stress        : 0.26140
## stress after  10 iters: 0.16214, magic = 0.500
## stress after  20 iters: 0.16212, magic = 0.500
## Initial stress        : 0.06787
## stress after  10 iters: 0.02647, magic = 0.500
## stress after  20 iters: 0.02612, magic = 0.500
## stress after  30 iters: 0.02546, magic = 0.500
## stress after  40 iters: 0.02464, magic = 0.500
## stress after  50 iters: 0.02446, magic = 0.500
## stress after  60 iters: 0.02445, magic = 0.500
## Initial stress        : 0.02787
## stress after  10 iters: 0.01018, magic = 0.500
## stress after  20 iters: 0.00891, magic = 0.500
## stress after  30 iters: 0.00752, magic = 0.500
## stress after  40 iters: 0.00726, magic = 0.500
## stress after  50 iters: 0.00641, magic = 0.500
```

```
## stress after  60 iters: 0.00621, magic = 0.500
## stress after  70 iters: 0.00616, magic = 0.500

{plot(terp_sam, type = "b",pch = 19, col = 1, main = "Stress Comparison for
each dimension")
text(c(0:4),terp_sam,round(terp_sam,5), cex = 0.6)
}
```




```
terp_sam2 = sammon(dist(terpenes),k=2)

## Initial stress        : 0.06787
## stress after  10 iters: 0.02647, magic = 0.500
## stress after  20 iters: 0.02612, magic = 0.500
## stress after  30 iters: 0.02546, magic = 0.500
## stress after  40 iters: 0.02464, magic = 0.500
## stress after  50 iters: 0.02446, magic = 0.500
## stress after  60 iters: 0.02445, magic = 0.500

x = terp_sam2$points[,1]
y = terp_sam2$points[,2]
{plot(x,y,col =1, pch = 19, xlim = c(-20,40), ylim = c(-40,20), main =
"Sammon's metric least squares scaling")
text(x,y,row.names(plant),pos =4,cex = 0.8)
}
```



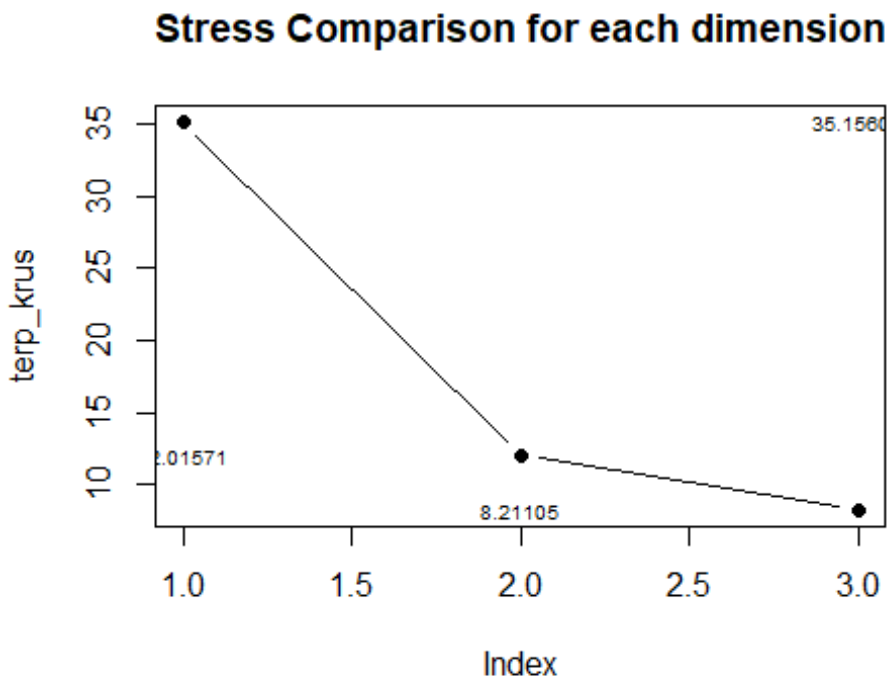
The stress levels decrease substantially from class 1 to class 2 whereas it doesn't decrease that much from class 2 to class 3. Hence we fit the Sammon scaling for 2 classes and the observations points plotted are similar to Classical Metric Scaling.

Kruskal's non-metric scaling

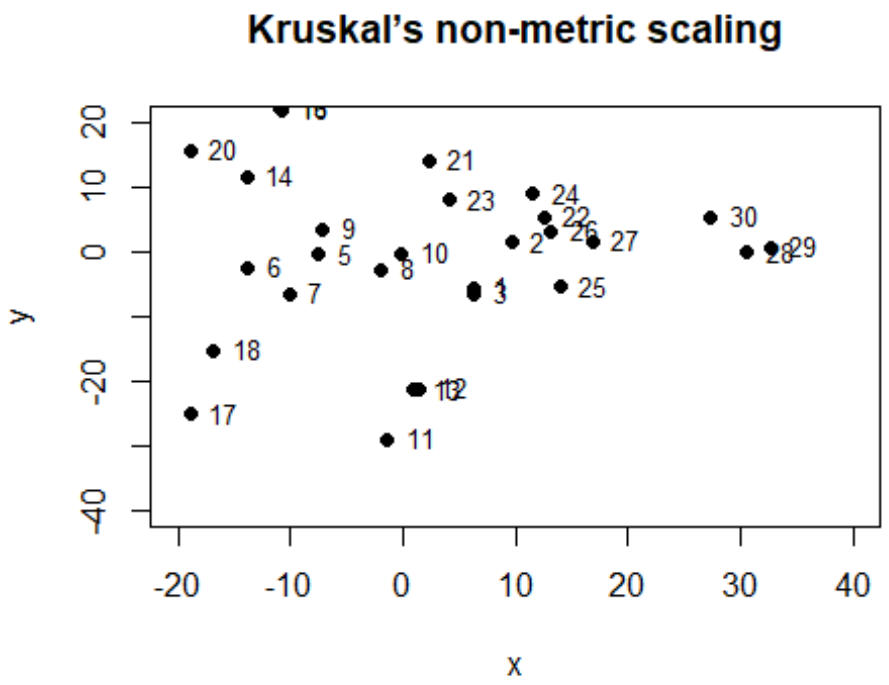
```
terp_krus = rep(NA,3)
for(i in seq(3)){
terp_krus[i] = isoMDS(dist(terpenes),k=i)$stress
}
```

```
## initial  value 40.484914
## final  value 35.156076
## converged
## initial  value 19.099008
## iter   5 value 18.422428
## iter  10 value 17.550163
## iter  15 value 12.874160
## iter  20 value 12.185171
## iter  25 value 12.021814
## iter  25 value 12.016702
## iter  25 value 12.015710
## final  value 12.015710
## converged
## initial  value 11.588897
## iter   5 value 8.245610
```

```
## final  value 8.211046
## converged

{plot(terp_krus, type = "b",pch = 19, col = 1, main = "Stress Comparison for
each dimension")
text(c(0:4),terp_krus,round(terp_krus,5), cex = 0.6)
}
```



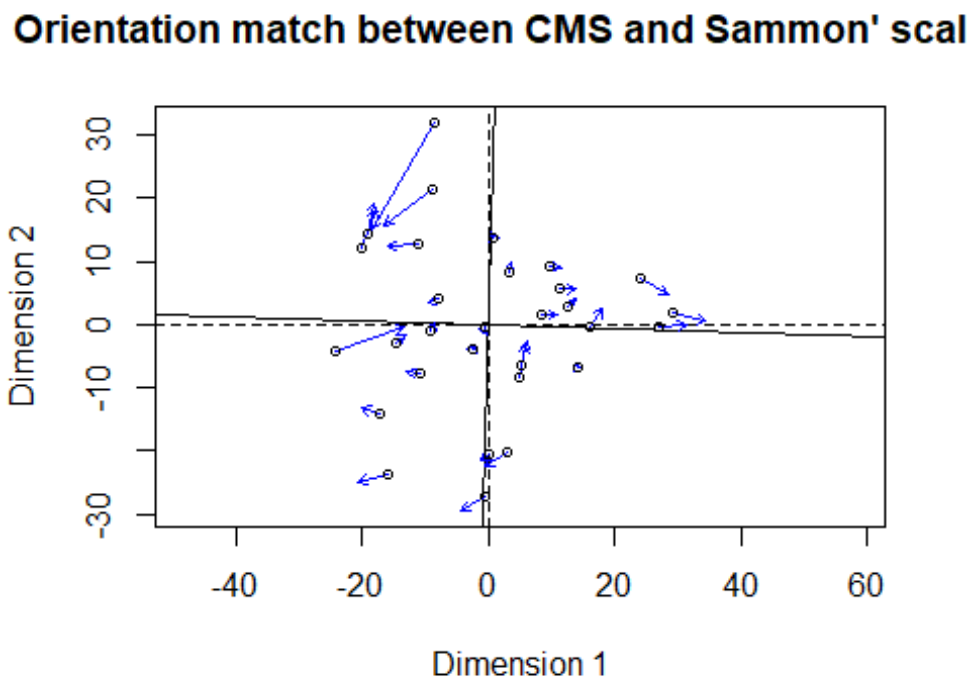

```
terp_krus2 = isoMDS(dist(terpenes),k=2)

## initial  value 19.099008
## iter   5 value 18.422428
## iter  10 value 17.550163
## iter  15 value 12.874160
## iter  20 value 12.185171
## iter  25 value 12.021814
## iter  25 value 12.016702
## iter  25 value 12.015710
## final  value 12.015710
## converged

x = terp_krus2$points[,1]
y = terp_krus2$points[,2]
{plot(x,y,col =1, pch = 19, xlim = c(-20,40), ylim = c(-40,20), main =
"Kruskal's non-metric scaling")
text(x,y,row.names(plant),pos =4,cex = 0.8)
}
```



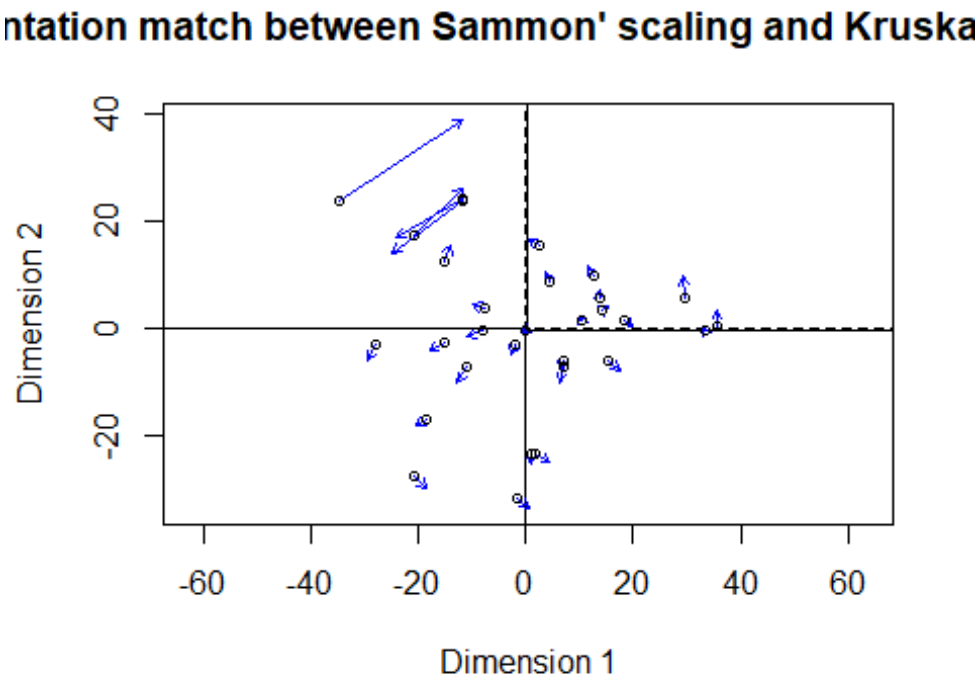
The stress levels decrease more in case of class 1 to class 2 than class 2 to class 3.Hence, we fit Kruskal with 2 classes again and notice that we get similar observation points as Classical Metric Scaling.

#Question 2c) Procrustes analysis

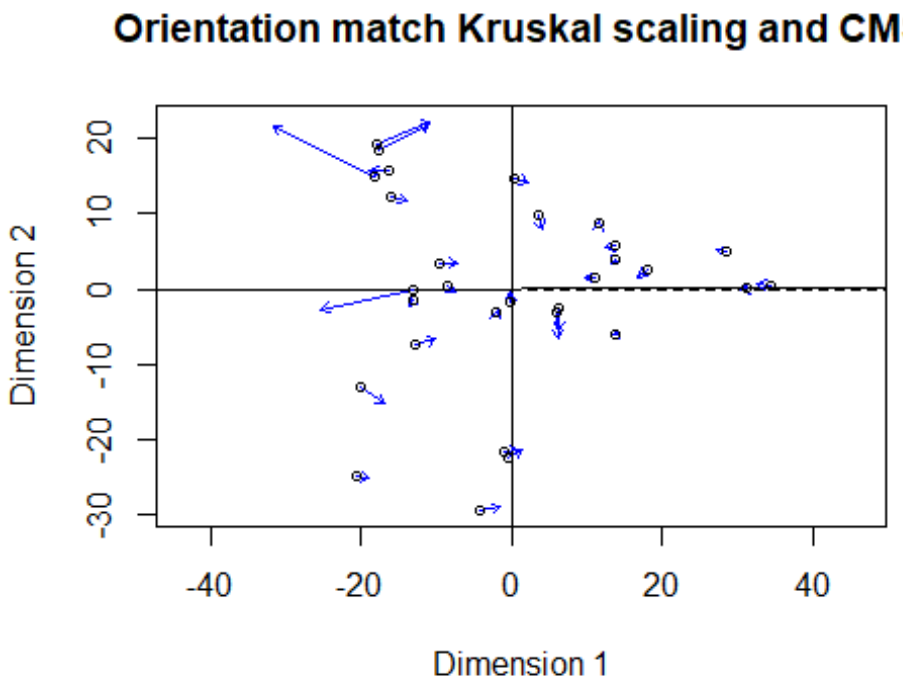
```
proc_cms_sam = procrustes(cms$points[,1:2], terp_sam2$points)
proc_sam_krus = procrustes(terp_sam2$points, terp_krus2$points)
proc_krus_cms = procrustes(terp_krus2$points, cms$points[,1:2])
plot(proc_cms_sam,main = "Orientation match between CMS and Sammon' scaling")
```



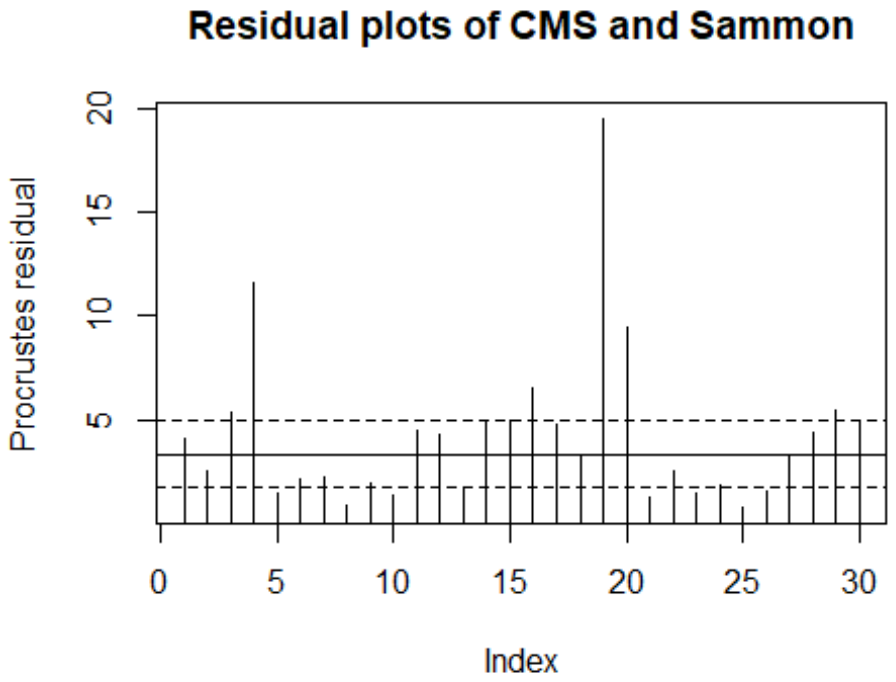

```r
plot(proc_sam_krus,main = "Orientation match between Sammon' scaling and
Kruskal scaling")
```



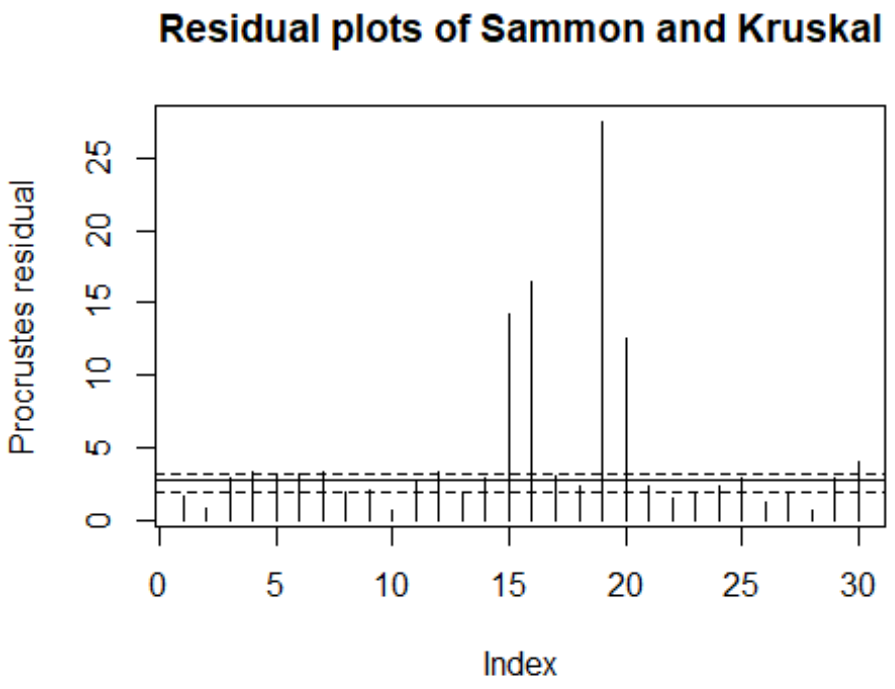
```r
plot(proc_krus_cms,main = "Orientation match Kruskal scaling and CMS")
```



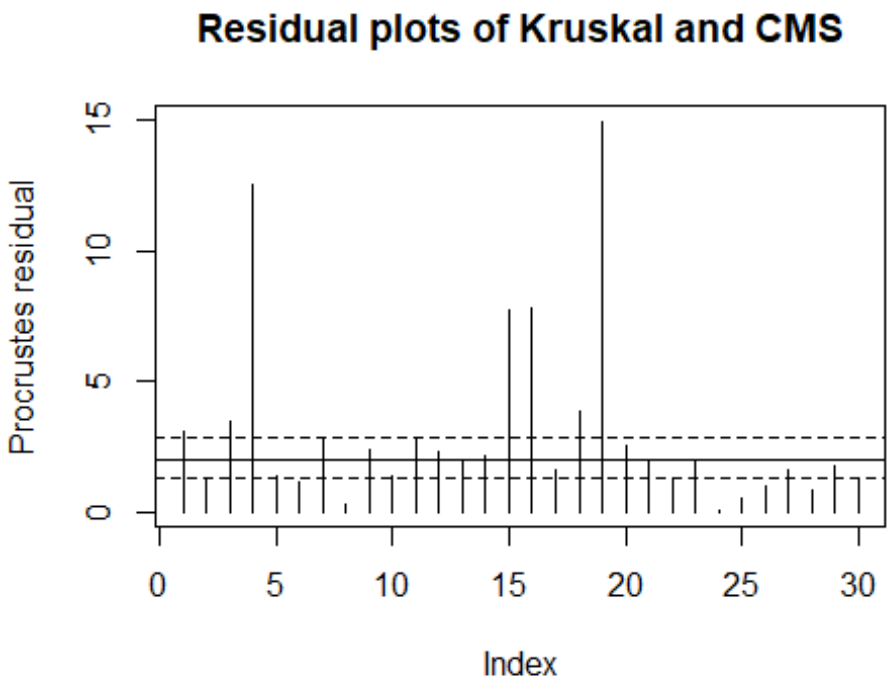

```r
plot(proc_cms_sam, kind=2,main = "Residual plots of CMS and Sammon")
```



```r
plot(proc_sam_krus, kind=2,main = "Residual plots of Sammon and Kruskal")
```




```r
plot(proc_krus_cms, kind=2,main = "Residual plots of Kruskal and CMS")
```



```
proc_cms_sam

##
## Call:
## procrustes(X = cms$points[, 1:2], Y = terp_sam2$points)
##
## Procrustes sum of squares:
## 943.1

proc_sam_krus

##
## Call:
## procrustes(X = terp_sam2$points, Y = terp_krus2$points)
##
## Procrustes sum of squares:
##  1546

proc_krus_cms

##
## Call:
## procrustes(X = terp_krus2$points, Y = cms$points[, 1:2])
##
## Procrustes sum of squares:
## 608.4
```

Kind 1 gives a visual indication of the degree of match between the two ordinations. Symbols or labels show the position of the samples in the first ordination, and arrows point to their positions in the target ordination. The plot also shows the rotation between the two ordinations necessary to make them match as closely as possible.

Kind 2 plots show the residuals for each sample. This allows identification of samples with the worst fit. The horizontal lines, from bottom to top, are the 25% (dashed), 50% (solid), and 75% (dashed) quantiles of the residuals.

Considering both the kind graphs, we find that procrustes analysis between CMS and kruskal scaling is the best as the residuals are quite low and oreientation of both the graphs is almost similar with very few observations apart from each other. #Question 2d)

```
model= Mclust(terpenes,G=1:9)
summary(model)

## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 6 components:
##
##  log-likelihood  n  df       BIC       ICL
##        -273.297 30 185 -1175.816 -1175.816
```

```
##
## Clustering table:
##  1  2  3  4  5  6
## 10  8  4  3  2  3
```

```
model$BIC
```

```
## Bayesian Information Criterion (BIC):
##         EII       VII       EEI       VEI       EVI       VVI       EEE
## 1 -1534.671 -1534.671 -1430.731 -1430.731 -1430.731 -1430.731 -1363.418
## 2 -1516.394 -1492.713 -1416.033 -1395.465 -1398.577 -1387.067 -1378.315
## 3 -1490.860 -1463.129 -1404.612 -1385.553 -1392.777 -1365.842 -1381.151
## 4 -1479.260 -1460.120 -1377.685 -1374.947        NA        NA -1364.810
## 5 -1475.484 -1435.108 -1391.356 -1355.610        NA        NA -1350.639
## 6 -1458.831 -1400.271 -1387.737 -1349.284        NA        NA -1339.974
## 7 -1452.880 -1429.906 -1365.618 -1338.798        NA        NA -1329.136
## 8 -1434.802 -1433.173 -1357.873 -1333.186        NA        NA -1333.114
## 9 -1444.199 -1415.257 -1370.629 -1334.397        NA        NA -1351.426
##         EVE       VEE       VVE       EEV       VEV       EVV       VVV
## 1 -1363.418 -1363.418 -1363.418 -1363.418 -1363.418 -1363.418 -1363.418
## 2 -1348.799 -1373.693 -1349.190 -1406.801 -1391.472 -1400.549 -1377.730
## 3        NA        NA        NA -1422.666 -1400.862        NA        NA
## 4        NA        NA        NA -1286.380 -1400.790        NA        NA
## 5        NA        NA        NA -1273.187 -1320.322        NA        NA
## 6        NA        NA        NA -1302.723 -1175.816        NA        NA
## 7        NA        NA        NA        NA        NA        NA        NA
## 8        NA        NA        NA -1482.555        NA        NA        NA
## 9        NA        NA        NA -1576.475        NA        NA        NA
##
## Top 3 models based on the BIC criterion:
##     VEV,6     EEV,5     EEV,4
## -1175.816 -1273.187 -1286.380
```

Mclust package uses Bayesian Information Criterion (BIC) to find the number of clusters (model selection). BIC uses the likelihood and a penalty term to guard against overfitting. After the data is fit into the model, we plot the model based on clustering results Mclust uses an identifier for each possible parametrization of the covariance matrix that has three letters: E for “equal”, V for “variable” and I for “coordinate axes”. The first identifier refers to volume, the second to shape and the third to orientation

We observe that the the minimum BIC is generated for the model VEV with 6 clusters.