Question-2 Practicum 3 DA5030

Shivam Verma

Problem 2

Part 1. Download the data set Plant Disease Data Set. Note that the data file does not contain header names; you may wish to add those.

```
library(arules)
```

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

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
# Loading the data
plant <- read.transactions("plants.data")
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

```
# printing summary
summary(plant)
```

```
## transactions as itemMatrix in sparse format with
##  34781 rows (elements/itemsets/transactions) and
##  35463 columns (items) and a density of 6.717562e-05
##
## most frequent items:
##       var.       ssp.      carex astragalus  eriogonum    (Other)
##       5359       3069        694        615        436      72684
##
## element (itemset/transaction) length distribution:
## sizes
##     1     2     3     4     5     6
##  3383 23250     1  7865   181   101
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   1.000   2.000   2.000   2.382   2.000   6.000
##
## includes extended item information - examples:
##                       labels
## 1 <d7>abbeae,mi,mn,wi,on,sk
## 2          <d7>abbottiae,hi
## 3         <d7>abitibiana,qc
```

Part 2. Explore the data set as you see fit and that allows you to get a sense of the data and get comfortable with it. Is there distributional skew in any of the features? Is there a need to apply a transform?

Exploring the dataset using functions like inspect, itemFrequency, frequencyPlot, and image to find out skewness of the data. The data mostly consisted of plant species and states. The data is scewed but, we dont have to normalize the data for further functionality.

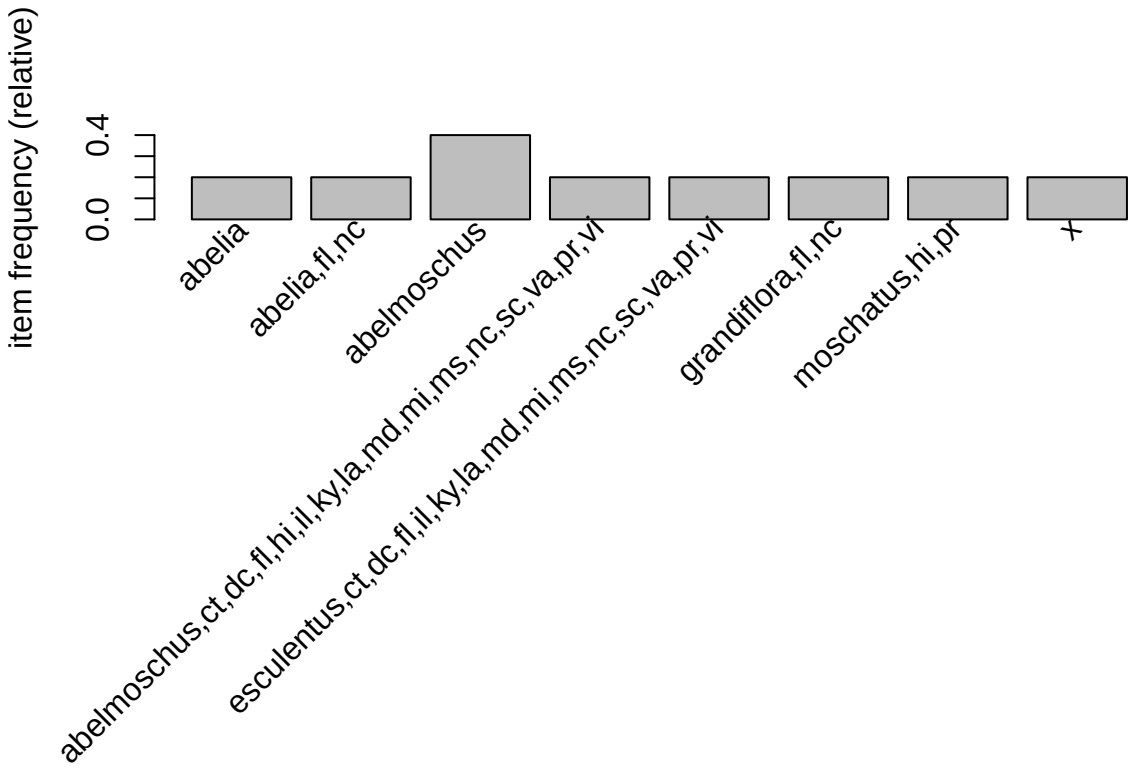
```
# viewing first five plants using inspect()
inspect(plant[1:5])
```

```
##     items
## [1] {abelia,fl,nc}
## [2] {abelia,
##      grandiflora,fl,nc,
##      x}
## [3] {abelmoschus,ct,dc,fl,hi,il,ky,la,md,mi,ms,nc,sc,va,pr,vi}
## [4] {abelmoschus,
##      esculentus,ct,dc,fl,il,ky,la,md,mi,ms,nc,sc,va,pr,vi}
## [5] {abelmoschus,
##      moschatus,hi,pr}
```

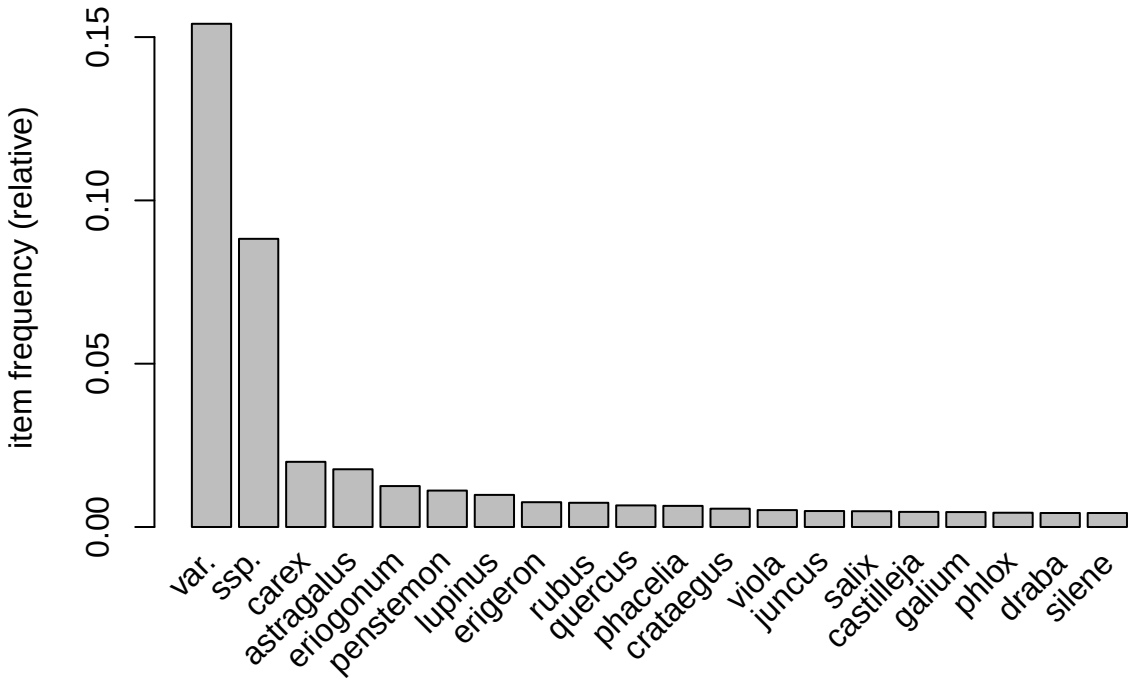
```
# see the proportion of data that contain the species
itemFrequency(plant[, 1:3])
```

```
## <d7>abbeae,mi,mn,wi,on,sk          <d7>abbottiae,hi         <d7>abitibiana,qc
##              2.875133e-05              2.875133e-05              2.875133e-05
```

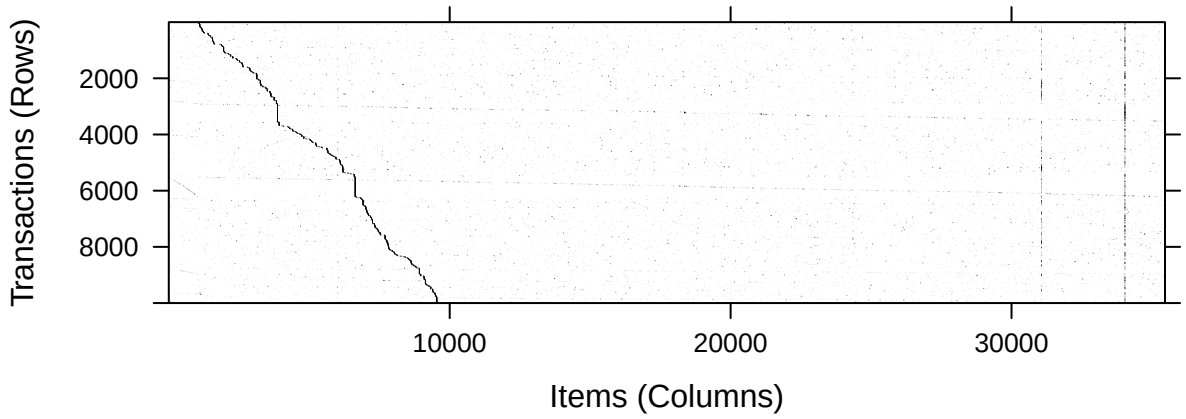
```
# getting visualization with atleast 10% support
itemFrequencyPlot(plant[1:5], support = 0.1)
```



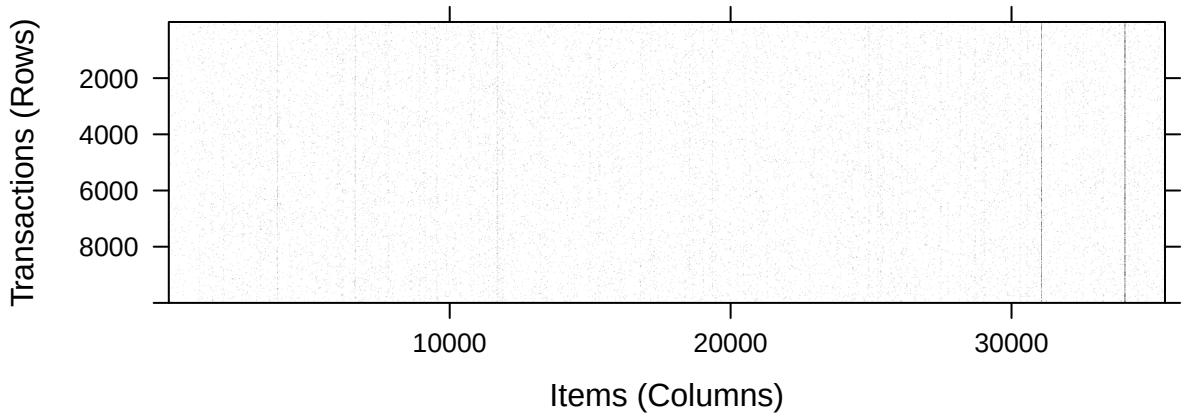

```
# Visualizing top 20 plants
itemFrequencyPlot(plant, topN = 20)
```




```
# the sparse matrix for the first five transactions
image(plant[1:10000])
```




```
# sparse matrix for the random selection of 10000 plants
image(sample(plant, 10000))
```



Part 3. Use association rules to segment the data similar to what was done in Hämäläinen, W., & Nykänen, M. (2008, December). Efficient discovery of statistically significant association rules. In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on (pp. 203-212). IEEE.

Setting the association rules for segmentation of the data using apriori function from arules package. Setting up support equal to 0.0001 and confidence level equal to 0.50. A confidence threshold of 0.50, which means that in order to be included in the results, the rule has to be correct at least 50 percent of the time. A support = 0.0001, in order to generate a rule, an item must have appeared in at least 0.0001 * 34781 = 3.4781 transactions.

```
# using default setting of apriori results in non-significant rules
apriori(plant)
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
```

```
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 3478
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[35463 item(s), 34781 transaction(s)] done [0.05s].
## sorting and recoding items ... [1 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].

## set of 0 rules
```

```
# setting customized parameters using trial and error method to generate rules.
plantrules <- apriori(plant, parameter = list(support = 0.0001, confidence = 0.50))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5   1e-04      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 3
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[35463 item(s), 34781 transaction(s)] done [0.05s].
## sorting and recoding items ... [2327 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 done [0.02s].
## writing ... [2657 rule(s)] done [0.01s].
## creating S4 object  ... done [0.01s].
```

```
# summarizing the generated rules
summary(plantrules)
```

```
## set of 2657 rules
##
## rule length distribution (lhs + rhs):sizes
##    2    3    4    5
## 1160 1242  216   39
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   3.000   2.674   3.000   5.000
##
## summary of quality measures:
```

```
##     support            confidence        coverage             lift
##  Min.   :0.0001150   Min.   :0.5000   Min.   :0.0001150   Min.   :   3.245
##  1st Qu.:0.0001150   1st Qu.:0.7500   1st Qu.:0.0001150   1st Qu.:   6.490
##  Median :0.0001438   Median :1.0000   Median :0.0001725   Median :  11.333
##  Mean   :0.0001755   Mean   :0.8876   Mean   :0.0002140   Mean   : 572.191
##  3rd Qu.:0.0001725   3rd Qu.:1.0000   3rd Qu.:0.0002300   3rd Qu.: 457.645
##  Max.   :0.0023864   Max.   :1.0000   Max.   :0.0043702   Max.   :8695.250
##      count
##  Min.   : 4.000
##  1st Qu.: 4.000
##  Median : 5.000
##  Mean   : 6.104
##  3rd Qu.: 6.000
##  Max.   :83.000
##
## mining info:
##   data ntransactions support confidence
##  plant         34781   1e-04        0.5
```

```
# Inspecting the rules
inspect(plantrules[1:10])
```

```
##      lhs                rhs          support      confidence coverage
## [1]  {collina}       => {calystegia} 0.0001150053 1          0.0001150053
## [2]  {collina}       => {ssp.}       0.0001150053 1          0.0001150053
## [3]  {brunnescens}   => {carex}      0.0001150053 1          0.0001150053
## [4]  {brunnescens}   => {ssp.}       0.0001150053 1          0.0001150053
## [5]  {texensis}      => {var.}       0.0001150053 1          0.0001150053
## [6]  {deustus}       => {penstemon}  0.0001150053 1          0.0001150053
## [7]  {deustus}       => {var.}       0.0001150053 1          0.0001150053
## [8]  {serpyllifolia} => {ssp.}       0.0001150053 1          0.0001150053
## [9]  {aridus}        => {lupinus}    0.0001150053 1          0.0001150053
## [10] {aridus}        => {ssp.}       0.0001150053 1          0.0001150053
##      lift       count
## [1]  632.381818 4
## [2]   11.333007 4
## [3]   50.116715 4
## [4]   11.333007 4
## [5]    6.490203 4
## [6]   89.641753 4
## [7]    6.490203 4
## [8]   11.333007 4
## [9]  101.698830 4
## [10]  11.333007 4
```

Part 4. Are there clusters in the data? Can plants be segmented into groups? Build a k-means clustering model to investigate.

To identify clusters in the data k-means algorithm can be applied, but for that the data format is not suitable. Data from transaction format of S4 vector needs to be convert into a matrix of binary variables with species on row and states on coloumn to represent 1 being present species in that state and 0 means not present. T convert it first I extracted plants with states each row unlist as vector, and then write it into the new csv file with seperated by comma ",". Then read that file, with 100 pre columns to store states. Then removed the empty coloumns from the data.

Next, applied reshape2 library, to melt the data on the basis of species to get just 2 rows with state and species. then used dcast and acast function to create a 35000 species X 70 row matrix with 1, or 0 variable.

After getting the matrix, I applied kmeans function with cluster value as 10.

```
# unlist and converting it into vecotr
plant_vector <- as.vector(unlist(plant@itemInfo))

# writing it into file
write(x = plant_vector, "plant_data.csv", sep=",")

# reading file as dataframe
plant_data <- read.csv("plant_data.csv", header=FALSE, fill = TRUE, col.names = paste0("V",1:100))

# removing empty coloumns
plant_data <- plant_data[,which(!is.na(plant_data[1,]))]

library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 3.6.2
```

```
# melting data on based of V1 coloumn which is our species
plant_data_modified <- melt(plant_data,id=c("V1"))
```

```
## Warning: attributes are not identical across measure variables; they will be
## dropped
```

```
# removing unwanted row created by melting
plant_data_modified <- plant_data_modified[,c(1,3)]

# creating data frame with 1, 0 binary variables represent if species available or not
plant_data_modified1 <- dcast(plant_data_modified,V1~value)
```

```
## Aggregation function missing: defaulting to length
```

```
# creating matrix with same features
plant_data_modified <- acast(plant_data_modified,V1~value)
```

```
## Aggregation function missing: defaulting to length
```

```
plant_data_modified <-plant_data_modified[,-1]
```

Applying K-means clustering algorithm setting k as 10. Large gap between cluster sizes is slightly concerning, it may be case some species are perdominantly present in some some states or it may be a random fluke caused by the initial k-means cluster centers.

```
set.seed(123)
# Creating clusters
plant_clusters <- kmeans(plant_data_modified, 10)
# Getting cluster sizes
plant_clusters$size
```

```
##  [1]   99 7015  980 1565 3901  299 1083   37   92    7
```

```
# Getting cluster centers
plant_clusters$centers
```

```
##              ab          ak         al          ar          az          bc
## 1   2.717171717 1.989898990  5.5656566  4.83838384  4.26262626  3.595959596
## 2   0.049037776 0.084248040  0.0168211  0.02794013  0.21240200  0.098075552
## 3   0.651020408 0.459183673  0.1438776  0.12448980  1.17040816  0.896938776
## 4   0.047923323 0.042172524  0.9987220  0.81597444  0.28626198  0.149520767
## 5   0.002563445 0.005383235  0.1871315  0.06588054  0.05331966  0.007690336
## 6   1.023411371 0.742474916  2.7859532  2.31103679  1.62207358  1.387959866
## 7   0.910433980 0.722068329  0.8060942  0.75438596  0.87165282  1.070175439
## 8   5.864864865 4.567567568 10.6216216  8.48648649  7.43243243  8.162162162
## 9   2.858695652 1.967391304  1.1630435  0.96739130  5.81521739  4.021739130
## 10 12.714285714 9.571428571 19.1428571 19.28571429 11.42857143 16.285714286
##             ca          co          ct          dc          de        dengl
## 1   5.73737374  4.01010101  4.161616162  3.070707071  3.65656566 0.454545455
## 2   0.45388453  0.10563079  0.045901639  0.009978617  0.01168924 0.003991447
## 3   1.59591837  1.18061224  0.061224490  0.025510204  0.03979592 0.013265306
## 4   0.45239617  0.18530351  0.624920128  0.538658147  0.64856230 0.007667732
## 5   0.07177647  0.01332992  0.008459369  0.006921302  0.01230454 0.000000000
## 6   2.35451505  1.50167224  2.214046823  1.668896321  1.89632107 0.123745819
## 7   1.15974146  1.00646353  1.078485688  0.701754386  0.79963066 0.222530009
## 8  10.56756757  7.83783784  9.027027027  6.351351351  7.89189189 1.081081081
## 9   7.76086957  5.68478261  0.750000000  0.347826087  0.41304348 0.163043478
## 10 18.00000000 14.57142857 23.142857143 17.000000000 19.14285714 2.285714286
##             fl       fraspm          ga          gl         hi          ia
## 1   5.52525253 1.0606060606  5.72727273 0.393939394 1.55555556  3.606060606
## 2   0.01696365 0.0106913756  0.01625089 0.020527441 0.02879544  0.022950820
## 3   0.21326531 0.0285714286  0.15408163 0.088775510 0.14489796  0.167346939
## 4   0.86389776 0.0664536741  1.03067093 0.008306709 0.22939297  0.460702875
## 5   0.45680595 0.0002563445  0.19943604 0.001025378 0.36093309  0.003332479
## 6   2.81270903 0.4481605351  2.96655518 0.130434783 0.87625418  1.752508361
## 7   0.72483841 0.4727608495  0.81809788 0.097876270 0.42566944  0.922437673
## 8  10.02702703 2.8378378378 11.21621622 0.729729730 2.56756757  7.891891892
## 9   1.34782609 0.3369565217  1.19565217 0.586956522 0.58695652  0.891304348
## 10 14.85714286 7.0000000000 21.14285714 0.857142857 4.00000000 20.857142857
##             id          il          in          ks          ky          la
## 1   3.838383838  5.21212121  4.454545455  3.93939394  4.74747475  5.04040404
## 2   0.100213828  0.04005702  0.026942267  0.03235923  0.02138275  0.02523165
## 3   1.186734694  0.17142857  0.098979592  0.27857143  0.07857143  0.15510204
## 4   0.115654952  0.83897764  0.748881789  0.49712460  0.85814696  0.82939297
## 5   0.004357857  0.01820046  0.007177647  0.01563702  0.03332479  0.17328890
## 6   1.317725753  2.47826087  2.204013378  1.79264214  2.31772575  2.51505017
## 7   0.987072946  1.09233610  0.999076639  0.77469991  0.87257618  0.72206833
## 8   7.081081081  9.97297297  8.756756757  7.27027027  8.56756757  9.35135135
## 9   5.282608696  1.09782609  0.782608696  1.60869565  0.75000000  1.20652174
## 10 15.714285714 25.28571429 22.428571429 17.00000000 21.57142857 17.14285714
##             lb          ma          mb          md          me          mi
## 1  1.1212121212  4.54545455  2.818181818  5.26262626  3.616161616  4.45454545
## 2  0.0262295082  0.06942267  0.035923022  0.03592302  0.051318603  0.06058446
## 3  0.1173469388  0.12551020  0.296938776  0.08571429  0.128571429  0.17040816
## 4  0.0415335463  0.67539936  0.127156550  0.88242812  0.428115016  0.60830671
```

```
## 5  0.0007690336  0.01409895  0.003332479  0.03511920  0.003845168  0.00281979
## 6  0.4013377926  2.36454849  1.063545151  2.57190635  1.806020067  2.18394649
## 7  0.4958448753  1.16066482  0.936288089  1.00738689  1.079409049  1.12834718
## 8  2.7297297297  9.64864865  6.243243243 10.37837838  8.567567568  9.75675676
## 9  0.7391304348  1.03260870  1.695652174  0.86956522  0.956521739  1.11956522
## 10 5.2857142857 23.28571429 15.428571429 24.85714286 21.142857143 24.14285714
##             mn          mo           ms           mt           nb          nc
## 1   3.909090909  4.85858586  4.989898990  3.818181818  2.585858586  5.83838384
## 2   0.034069850  0.03449751  0.007982894  0.072986458  0.034354954  0.02779758
## 3   0.234693878  0.15612245  0.124489796  1.093877551  0.088775510  0.12551020
## 4   0.409584665  0.78658147  0.836421725  0.120127796  0.219808307  1.03642173
## 5   0.001794412  0.02614714  0.142271213  0.003332479  0.001025378  0.14560369
## 6   1.732441472  2.22073579  2.498327759  1.347826087  1.210702341  2.86622074
## 7   1.069252078  0.89658356  0.689750693  1.030470914  0.933518006  0.94921514
## 8   8.135135135  8.72972973  9.189189189  7.702702703  6.081081081 11.59459459
## 9   1.250000000  0.95652174  0.967391304  4.630434783  0.695652174  1.15217391
## 10 20.857142857 22.14285714 18.000000000 15.285714286 16.857142857 23.57142857
##             nd          ne           nf           nh          nj          nm
## 1   2.565656566  3.222222222  1.989898990  3.404040404  4.55555556  4.45454545
## 2   0.016821098  0.022808268  0.036635780  0.034354954  0.04248040  0.16635780
## 3   0.324489796  0.362244898  0.113265306  0.076530612  0.07244898  1.12551020
## 4   0.141214058  0.291373802  0.100319489  0.425559105  0.78146965  0.31182109
## 5   0.001794412  0.002563445  0.002050756  0.002050756  0.01820046  0.04075878
## 6   1.010033445  1.397993311  0.725752508  1.752508361  2.44147157  1.69565217
## 7   0.816251154  0.801477378  0.747922438  1.013850416  1.07202216  0.92613112
## 8   5.243243243  6.729729730  4.459459459  7.864864865  9.89189189  8.16216216
## 9   1.402173913  1.630434783  0.793478261  0.847826087  0.83695652  5.57608696
## 10 14.857142857 15.714285714 10.428571429 20.714285714 23.85714286 12.57142857
##             ns          nt           nu          nv          ny          oh
## 1  2.626263e+00 1.53535354 0.575757576  3.57575758  5.40404040  4.54545455
## 2  3.421240e-02 0.04447612 0.025801853  0.16436208  0.09194583  0.03905916
## 3  7.755102e-02 0.27959184 0.132653061  1.26428571  0.17653061  0.10306122
## 4  2.217252e-01 0.01597444 0.005750799  0.14057508  0.80383387  0.78594249
## 5  5.126891e-04 0.00281979 0.001794412  0.01076647  0.02025122  0.01179185
## 6  1.237458e+00 0.53177258 0.177257525  1.14046823  2.68896321  2.36454849
## 7  9.215143e-01 0.60480148 0.296398892  0.81717452  1.22068329  1.04986150
## 8  6.432432e+00 3.21621622 1.513513514  6.24324324 11.35135135  9.67567568
## 9  5.978261e-01 1.41304348 0.804347826  6.04347826  1.25000000  0.85869565
## 10 1.657143e+01 7.28571429 2.857142857 11.00000000 26.14285714 24.42857143
##             ok          on          or          pa          pe          pr
## 1   4.74747475  4.636363636  4.61616162  5.12121212  1.71717172 2.53535354
## 2   0.04761226  0.081397006  0.19757662  0.06699929  0.01482537 0.01454027
## 3   0.29591837  0.263265306  1.30510204  0.11632653  0.02959184 0.20612245
## 4   0.67667732  0.558466454  0.25495208  0.86070288  0.11246006 0.34121406
## 5   0.04537298  0.004614201  0.01743143  0.02563445  0.00000000 0.57293002
## 6   2.17056856  2.244147157  1.82274247  2.65551839  0.78595318 1.22073579
## 7   0.74145891  1.174515235  1.08494922  1.13573407  0.68882733 0.34441367
## 8   7.97297297 10.297297297  9.16216216 10.83783784  4.70270270 3.27027027
## 9   1.85869565  1.619565217  6.11956522  1.03260870  0.32608696 0.84782609
## 10 18.00000000 24.714285714 18.00000000 25.85714286 11.85714286 4.14285714
##             qc          ri          sc          sd          sk          tn
## 1   3.747474747  3.131313131  5.40404040  3.010101010  2.373737374  5.09090909
## 2   0.082394868  0.019957234  0.01511048  0.021952958  0.032501782  0.01924448
## 3   0.230612245  0.027551020  0.13265306  0.437755102  0.408163265  0.08673469
```

```
## 4   0.350159744  0.449201278  0.95335463  0.187220447  0.081789137  0.93610224
## 5   0.001538067  0.003332479  0.15816457  0.001538067  0.001794412  0.05716483
## 6   1.759197324  1.729096990  2.74247492  1.240802676  0.946488294  2.44816054
## 7   1.110803324  0.897506925  0.78947368  0.868882733  0.882733149  0.86703601
## 8   8.513513514  7.081081081 10.51351351  6.027027027  5.486486486  9.21621622
## 9   1.413043478  0.554347826  1.05434783  1.858695652  1.836956522  0.89130435
## 10 21.857142857 19.714285714 20.28571429 15.857142857 12.428571429 23.00000000
##            tx          ut          va          vi           vt           wa
## 1   6.7777778  3.96969697  5.56565657 1.262626263  3.424242424  3.989898990
## 2   0.1995723  0.15837491  0.03121882 0.004989309  0.041910192  0.122594440
## 3   0.6622449  1.32448980  0.11530612 0.081632653  0.078571429  1.121428571
## 4   0.8626198  0.18785942  1.02236422 0.184025559  0.416613419  0.204472843
## 5   0.1922584  0.01614971  0.07408357 0.262753140  0.001025378  0.009997437
## 6   3.1137124  1.37458194  2.83612040 0.608695652  1.795986622  1.628762542
## 7   0.9104340  0.94090489  1.01846722 0.174515235  1.036934441  1.065558633
## 8  11.3513514  7.48648649 10.83783784 1.594594595  7.864864865  8.486486486
## 9   4.0326087  6.32608696  1.00000000 0.391304348  0.847826087  4.945652174
## 10 19.8571429 13.28571429 25.00000000 2.428571429 21.142857143 17.857142857
##              wi          wv           wy          yt
## 1   4.141414141  4.01010101  3.696969697 1.555555556
## 2   0.048325018  0.01895937  0.075124733 0.048610121
## 3   0.174489796  0.04183673  1.096938776 0.344897959
## 4   0.523961661  0.69584665  0.111821086 0.017252396
## 5   0.004357857  0.01384260  0.004614201 0.003332479
## 6   1.993311037  2.12040134  1.260869565 0.451505017
## 7   1.088642659  0.89658356  0.963988920 0.578947368
## 8   8.702702703  8.64864865  7.378378378 3.513513514
## 9   1.065217391  0.73913043  4.771739130 1.565217391
## 10 23.000000000 23.42857143 14.857142857 6.285714286
```

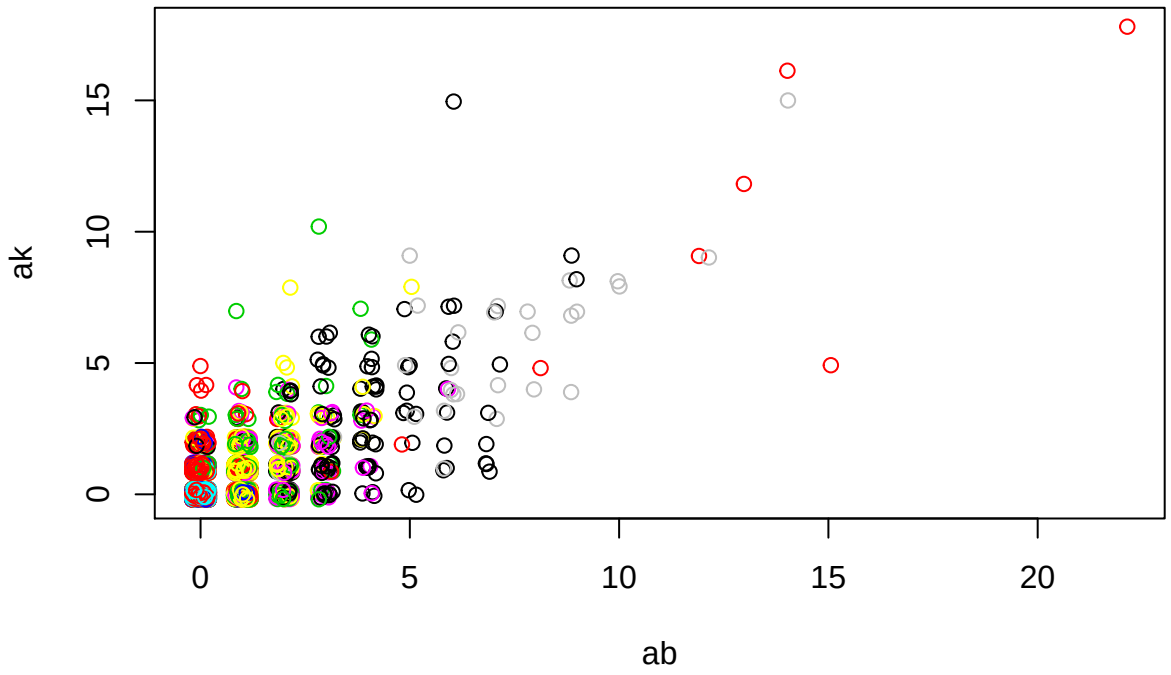
Part 5. Visualize the clusters.

Visualizing the clusters using 2-D scatter plots, 3-D scatter plot, plotting clusters in 2-D axis.

```
# generating 2-D scatter plot
print("2D graph with Jitters")
```

```
## [1] "2D graph with Jitters"
```

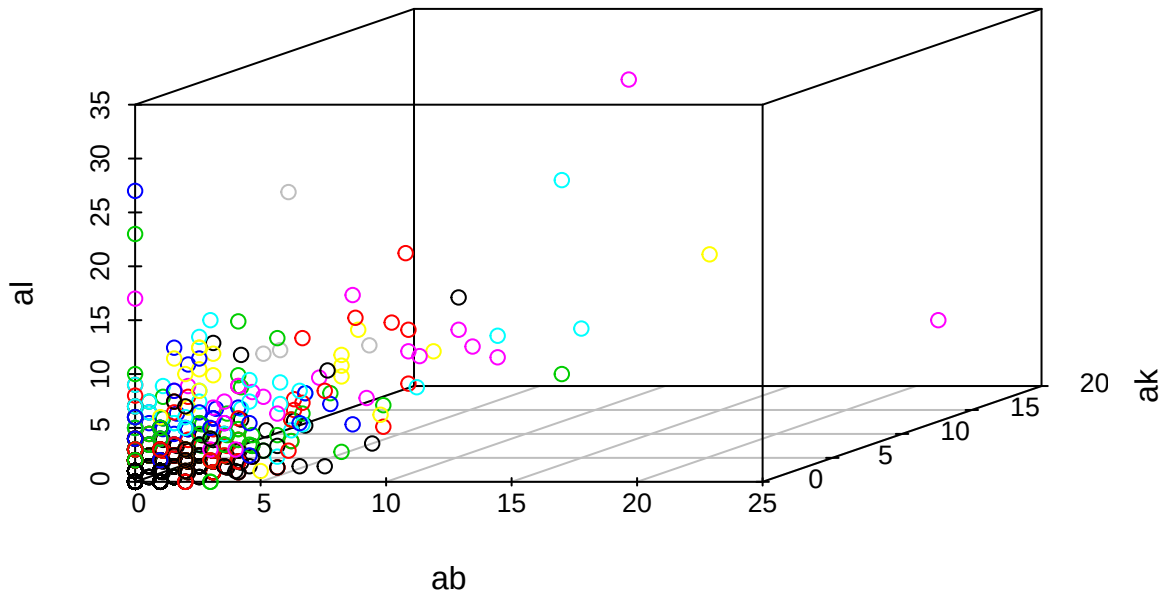
```
plot(jitter(plant_data_modified), col = plant_clusters$cluster)
points(plant_clusters$modes, col = 1:5, pch = 8)
```




```
# plotting 3D scatterplot
print("3D Scatter plot")
```

```
## [1] "3D Scatter plot"
```

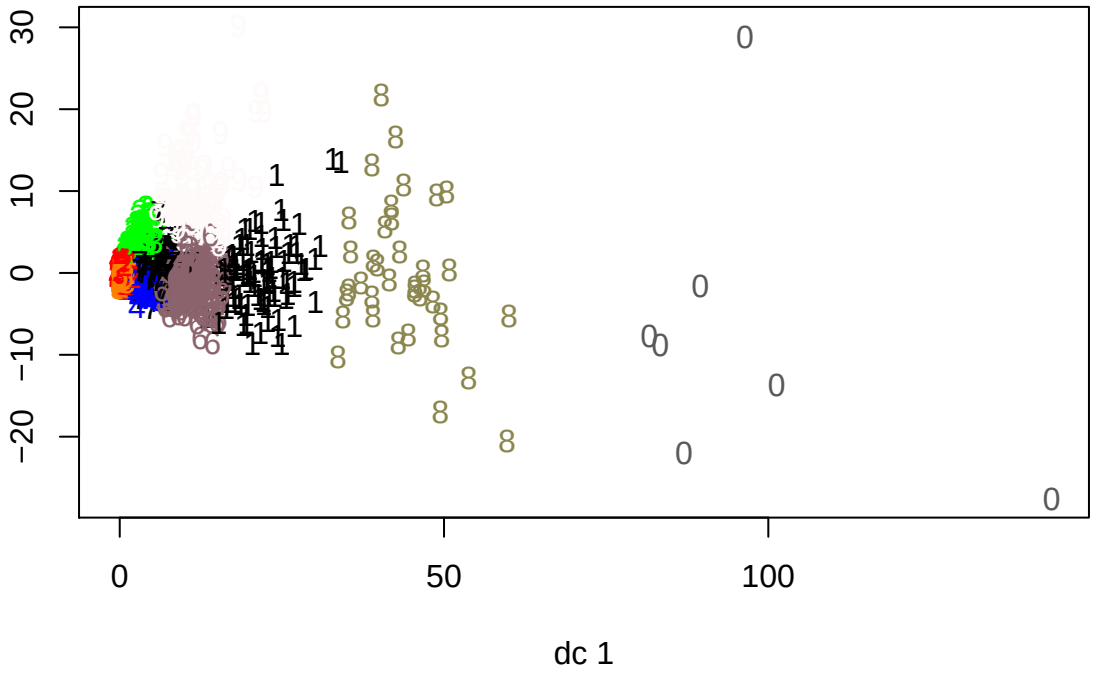
```
scatterplot3d::scatterplot3d(plant_data_modified, color = plant_clusters$cluster)
```




```
library(cluster)
library(fpc)
```

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

```
# Generating cluster plots
plotcluster(plant_data_modified, plant_clusters$cluster)
```




```
clusplot(plant_data_modified, plant_clusters$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```