Practicum-3.2

Harsh

26/07/2020

```
#Importing Libraries
library(fpc)
```

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

```
library(arules)
```

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

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

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

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

```
#unloadNamespace("arules")
#update.packages("arules")
#library(arules)
```

---Problem 2---

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. The description of each column can be found in the data set explanation. This assignment must be completed within a separate R Markdown Notebook. Use read.transaction() from the arules package to read the data.
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?

- I used read.transactions function to import the plants transaction data
- Since this the data is transaction i.e sparse matrix we cannot observe the skewness or distribution of the data
- Using summary and inspect I did explanatory analysis

```r
#Importing Dataset
plant_data <- read.transactions("C:\\Users\\harsh\\Desktop\\Introduction to Machine learning and Data M

#Observing transactions
summary(plant_data)
```

```
## transactions as itemMatrix in sparse format with
##  34781 rows (elements/itemsets/transactions) and
##  34851 columns (items) and a density of 0.0002779312
##
## most frequent items:
##      ca      tx      or      az      fl (Other)
##   11676    8483    7028    6778    6621  296309
##
## element (itemset/transaction) length distribution:
## sizes
##     2     3     4     5     6     7     8     9    10    11    12    13    14
## 11566  4874  2954  2107  1366  1094   859   744   655   562   503   421   421
##    15    16    17    18    19    20    21    22    23    24    25    26    27
##   333   322   284   252   241   249   207   200   212   198   195   155   152
##    28    29    30    31    32    33    34    35    36    37    38    39    40
##   190   179   159   146   140   146   148   147   119   123   110   124   118
##    41    42    43    44    45    46    47    48    49    50    51    52    53
##   114    83    85   102    90    90    75    64    89    63    60    64    68
##    54    55    56    57    58    59    60    61    62    63    64    65    66
##    56    54    51    59    47    45    39    59    52    43    39    43    47
##    67    68    69    70
##    60    35    26     4
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   4.000   9.686  10.000  70.000
##
## includes extended item information - examples:
##             labels
## 1        ×achnella
## 2 ×achnella caduca
## 3      ×agroelymus
```

```r
inspect(plant_data[1:5])
```

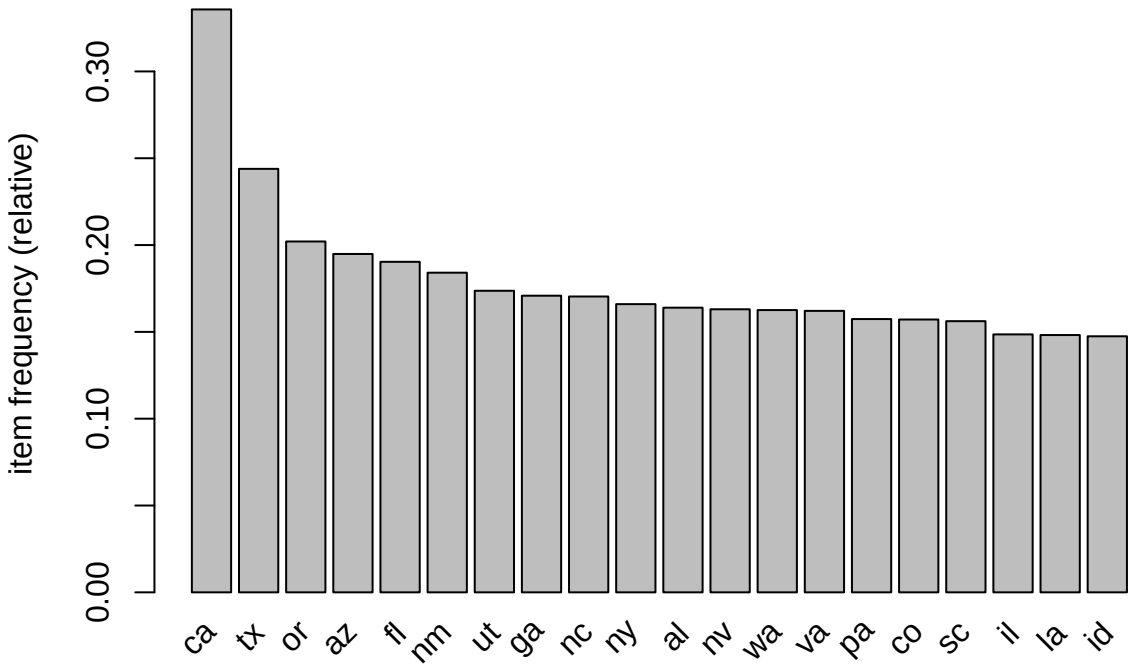
```
##     items
## [1] {abelia,
##      fl,
##      nc}
## [2] {abelia x grandiflora,
##      fl,
##      nc}
## [3] {abelmoschus,
##      ct,
##      dc,
##      fl,
##      hi,
```

```
##      il,
##      ky,
##      la,
##      md,
##      mi,
##      ms,
##      nc,
##      pr,
##      sc,
##      va,
##      vi}
## [4] {abelmoschus esculentus,
##      ct,
##      dc,
##      fl,
##      il,
##      ky,
##      la,
##      md,
##      mi,
##      ms,
##      nc,
##      pr,
##      sc,
##      va,
##      vi}
## [5] {abelmoschus moschatus,
##      hi,
##      pr}
```

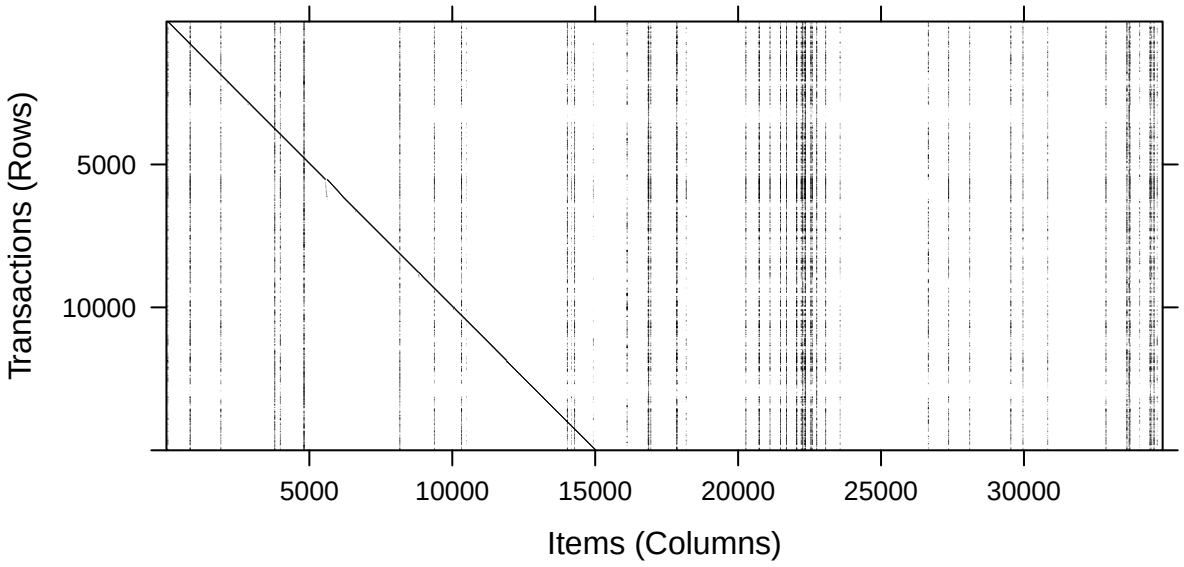
```
#Checking top 20 items using itemfrequency plot
itemFrequency(plant_data[,1:5])
```

```
##            ×achnella     ×achnella caduca          ×agroelymus
##         2.875133e-05         2.875133e-05         2.875133e-05
##  ×agroelymus adamsii ×agroelymus bowdenii
##         2.875133e-05         2.875133e-05
```

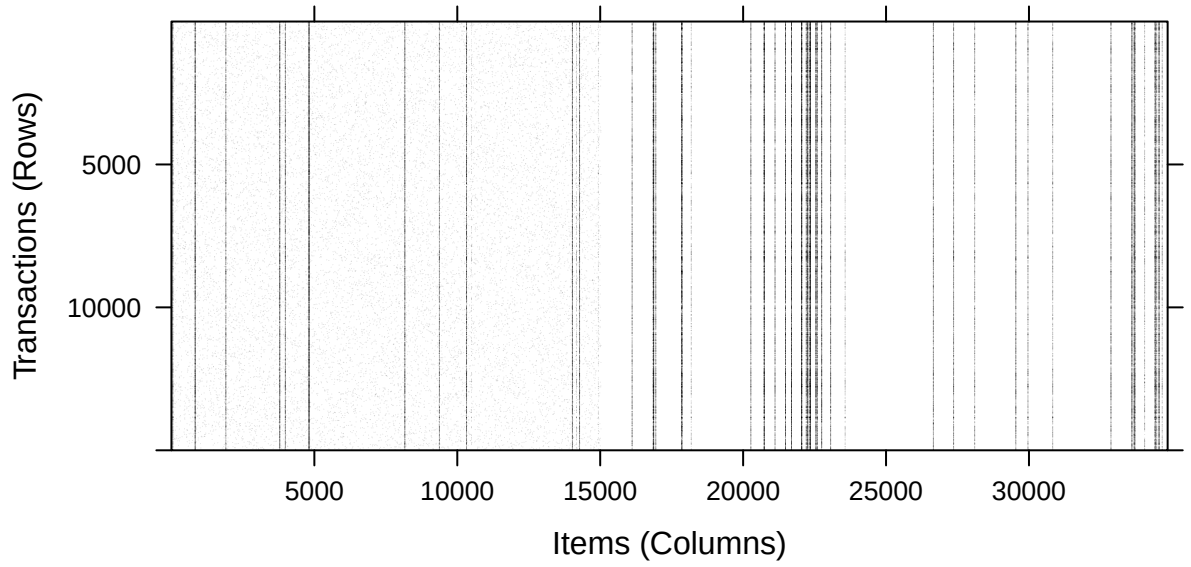
```
itemFrequencyPlot(plant_data,topN=20)
```




```
#Observing the image of the transaction data with and without sampling
image(plant_data[1:15000])
```




```
image(sample(plant_data[1:15000]))
```



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.

- Using apriori function to build association rules for plants dataset
- Improving the model by adjusting the support and confidence
- Using Inspect and summary, we can observe the output

```
#using apriori function to build rules
default_rules <- apriori(plant_data)
```

```
## 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 ...[34851 item(s), 34781 transaction(s)] done [0.14s].
## sorting and recoding items ... [49 item(s)] done [0.01s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 done [0.03s].
## writing ... [506 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(default_rules)
```

```
## set of 506 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4   5
##  89 307 100  10
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   3.000   3.000   3.061   3.000   5.000
##
## summary of quality measures:
##     support         confidence        coverage            lift
##  Min.   :0.1001   Min.   :0.8001   Min.   :0.1015   Min.   :2.470
##  1st Qu.:0.1015   1st Qu.:0.8582   1st Qu.:0.1110   1st Qu.:5.449
##  Median :0.1043   Median :0.8980   Median :0.1175   Median :5.793
##  Mean   :0.1071   Mean   :0.8969   Mean   :0.1200   Mean   :5.767
##  3rd Qu.:0.1100   3rd Qu.:0.9414   3rd Qu.:0.1247   3rd Qu.:6.092
##  Max.   :0.1426   Max.   :0.9893   Max.   :0.1708   Max.   :7.424
##      count
##  Min.   :3480
##  1st Qu.:3532
##  Median :3626
##  Mean   :3726
##  3rd Qu.:3825
##  Max.   :4959
##
## mining info:
##        data ntransactions support confidence
##  plant_data         34781     0.1        0.8
```

```
inspect(default_rules[1:5])
```

```
##     lhs     rhs  support   confidence coverage  lift     count
## [1] {nv} => {ca} 0.1351600 0.8291005  0.1630200 2.469762 4701
## [2] {wy} => {mt} 0.1098013 0.8108280  0.1354188 5.875294 3819
## [3] {wy} => {co} 0.1104339 0.8154989  0.1354188 5.190095 3841
## [4] {mt} => {id} 0.1123027 0.8137500  0.1380064 5.518237 3906
## [5] {id} => {or} 0.1207268 0.8186781  0.1474656 4.051571 4199
```

```
#Improving rules by changing default parameters
new_rules <- apriori(plant_data, parameter = list (support = 0.001, confidence = 0.75))
```

```
## Apriori
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.75    0.1    1 none FALSE            TRUE       5   0.001      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: 34
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[34851 item(s), 34781 transaction(s)] done [0.12s].
## sorting and recoding items ... [70 item(s)] done [0.01s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5

## Warning in apriori(plant_data, parameter = list(support = 0.001, confidence =
## 0.75)): Mining stopped (time limit reached). Only patterns up to a length of 5
## returned!

##  done [17.61s].
## writing ... [43995074 rule(s)] done [5.66s].
## creating S4 object  ... done [13.08s].
```

```
summary(new_rules)
```

```
## set of 43995074 rules
##
## rule length distribution (lhs + rhs):sizes
##        2        3        4        5
##      469    68831  2201628 41724146
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   5.000   5.000   4.947   5.000   5.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.001006   Min.   :0.7500   Min.   :0.001006   Min.   : 2.234
##  1st Qu.:0.007360   1st Qu.:0.8729   1st Qu.:0.007935   1st Qu.: 6.013
##  Median :0.015411   Median :0.9399   Median :0.016791   Median : 7.018
##  Mean   :0.017252   Mean   :0.9202   Mean   :0.018989   Mean   : 7.515
##  3rd Qu.:0.024381   3rd Qu.:0.9788   3rd Qu.:0.027055   3rd Qu.: 8.652
##  Max.   :0.158765   Max.   :1.0000   Max.   :0.202064   Max.   :67.350
##      count
##  Min.   :  35
##  1st Qu.: 256
##  Median : 536
##  Mean   : 600
##  3rd Qu.: 848
##  Max.   :5522
```

```
##
## mining info:
##        data ntransactions support confidence
##  plant_data         34781   0.001       0.75
```

```
inspect(new_rules[1:5])
```

```
##     lhs        rhs      support    confidence coverage   lift      count
## [1] {gl}    => {qc}     0.01236307 0.7664884  0.01612950  6.240457 430
## [2] {dengl} => {lb}     0.01204681 0.8747390  0.01377189 21.231192 419
## [3] {dengl} => {fraspm} 0.01377189 1.0000000  0.01377189 28.744628 479
## [4] {dengl} => {yt}     0.01049424 0.7620042  0.01377189 12.620603 365
## [5] {dengl} => {nt}     0.01081050 0.7849687  0.01377189 13.489128 376
```

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

- First I converted the transactions to separate items and transaction ID by importing it again
- Later I stored this data in a dataframe in matrix format
- Using kmeans function, I built clusters of the data by using k = 10 for creating 10 clusters
- Using size and centers we can observe the center and size of the cluster
- Using plotCluster function I visualize the clusters
- For this problem, I referred a few links I have added them in this chunk

```
#Importing the data as separate columns
plants <- read.transactions("C:\\Users\\harsh\\Desktop\\Introduction to Machine learning and Data Mining

#Observing the new data
summary(plants)
```

```
## transactions as itemMatrix in sparse format with
##  34781 rows (elements/itemsets/transactions) and
##  70 columns (items) and a density of 0.1240883
##
## most frequent items:
##      ca      tx      or      az      fl (Other)
##   11676    8483    7028    6778    6621  261528
##
## element (itemset/transaction) length distribution:
## sizes
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## 11566  4874  2954  2107  1366  1094   859   744   655   562   503   421   421
##    14    15    16    17    18    19    20    21    22    23    24    25    26
##   333   322   284   252   241   249   207   200   212   198   195   155   152
##    27    28    29    30    31    32    33    34    35    36    37    38    39
##   190   179   159   146   140   146   148   147   119   123   110   124   118
##    40    41    42    43    44    45    46    47    48    49    50    51    52
##   114    83    85   102    90    90    75    64    89    63    60    64    68
##    53    54    55    56    57    58    59    60    61    62    63    64    65
##    56    54    51    59    47    45    39    59    52    43    39    43    47
##    66    67    68    69
##    60    35    26     4
```

```
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   3.000   8.686   9.000  69.000
##
## includes extended item information - examples:
##   labels
## 1     ab
## 2     ak
## 3     al
##
## includes extended transaction information - examples:
##          transactionID
## 1               abelia
## 2 abelia x grandiflora
## 3          abelmoschus
```

```
#Converting the data to matrix format and as integer datatype
plants_matrix <- as.data.frame(as(plants,"matrix"))
plants_matrix[,1:70] <- lapply(plants_matrix[,1:70],as.integer)

#Verifying the data
str(plants_matrix)
```

```
## 'data.frame':    34781 obs. of  70 variables:
##  $ ab    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ ak    : int  0 0 0 0 0 1 0 1 0 0 ...
##  $ al    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ar    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ az    : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ bc    : int  0 0 0 0 0 1 0 1 0 0 ...
##  $ ca    : int  0 0 0 0 0 1 0 1 0 0 ...
##  $ co    : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ ct    : int  0 0 1 1 0 1 0 0 1 1 ...
##  $ dc    : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ de    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ dengl : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fl    : int  1 1 1 1 0 0 0 0 0 0 ...
##  $ fraspm: int  0 0 0 0 0 1 0 0 1 1 ...
##  $ ga    : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ gl    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hi    : int  0 0 1 0 1 0 0 0 0 0 ...
##  $ ia    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ id    : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ il    : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ in    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ ks    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ky    : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ la    : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ lb    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ ma    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ mb    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ md    : int  0 0 1 1 0 1 0 0 1 1 ...
##  $ me    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ mi    : int  0 0 1 1 0 1 0 0 1 1 ...
```

```
##  $ mn    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ mo    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ms    : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ mt    : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ nb    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ nc    : int  1 1 1 1 0 1 1 0 0 0 ...
##  $ nd    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ne    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nf    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ nh    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ nj    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nm    : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ ns    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ nt    : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ nu    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ nv    : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ ny    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ oh    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ ok    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ on    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ or    : int  0 0 0 0 0 1 0 1 0 0 ...
##  $ pa    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ pe    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ pr    : int  0 0 1 1 1 0 0 0 0 0 ...
##  $ qc    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ ri    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ sc    : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ sd    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ sk    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ tn    : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ tx    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ut    : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ va    : int  0 0 1 1 0 1 0 0 1 1 ...
##  $ vi    : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ vt    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ wa    : int  0 0 0 0 0 1 0 1 0 0 ...
##  $ wi    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ wv    : int  0 0 0 0 0 1 0 0 1 1 ...
##  $ wy    : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ yt    : int  0 0 0 0 0 1 0 0 0 0 ...
```

```
#Creating the clusters using kmeans function
clusters <- kmeans(plants_matrix,10)
clusters$size
```

```
##  [1]  2097  2454 12769  6474  1346  2009  1376  2582  1757  1917
```

```
clusters$centers
```

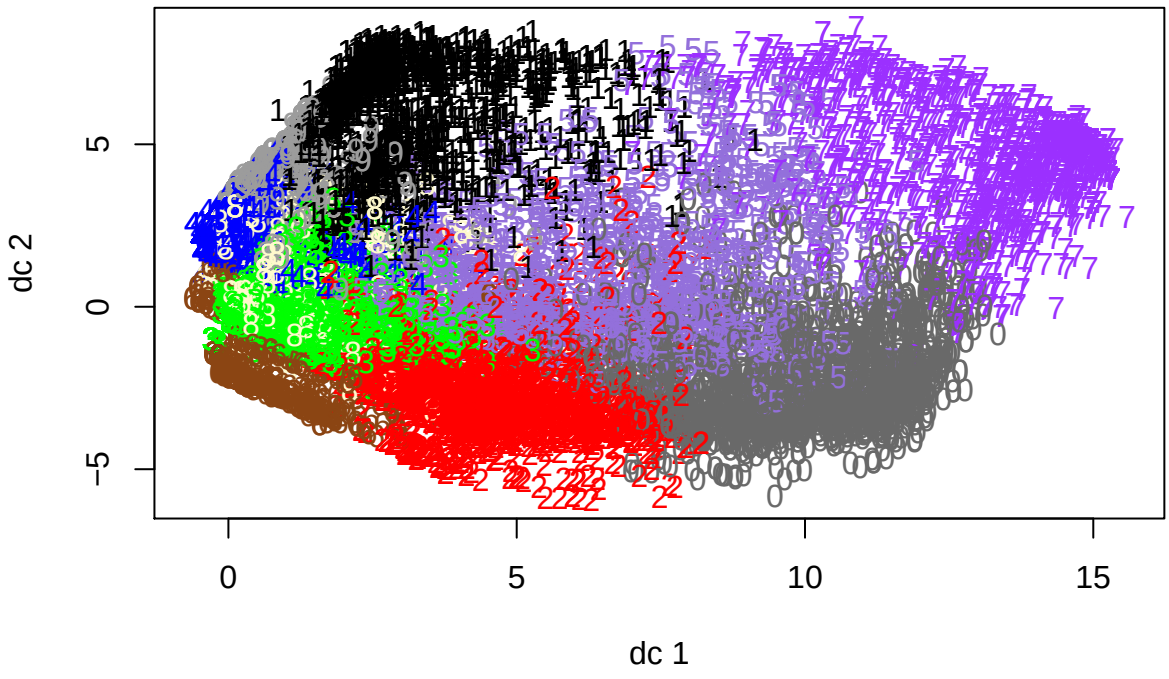
```
##             ab           ak          al          ar         az           bc
## 1  0.695278970 0.3681449690 0.009060563 0.042918455 0.44110634 0.7758702909
## 2  0.003259984 0.0020374898 0.901792991 0.473512632 0.10024450 0.0199674002
## 3  0.024590806 0.0689952228 0.033440363 0.022867883 0.05059128 0.0517659958
```

```
## 4  0.005560704 0.0194624652 0.004633920 0.002316960 0.06889095 0.0755329008
## 5  0.292719168 0.2555720654 0.101040119 0.053491828 0.03417533 0.4108469539
## 6  0.000000000 0.0009955202 0.057242409 0.004977601 0.04330513 0.0009955202
## 7  0.806686047 0.5813953488 0.702761628 0.743459302 0.79360465 0.9091569767
## 8  0.006196747 0.0007745933 0.013942680 0.077846631 0.66227730 0.0023237800
## 9  0.002276608 0.0056915196 0.005691520 0.016505407 0.73875925 0.0313033580
## 10 0.035472092 0.0140845070 0.912363067 0.895148670 0.14710485 0.0949400104
##            ca           co          ct          dc          de       dengl
## 1  0.65951359 0.7663328565 0.025274201 0.008583691 0.008106819 0.013352408
## 2  0.14384678 0.0346373268 0.077424613 0.109209454 0.188264059 0.000000000
## 3  0.00000000 0.0305427207 0.017699115 0.006265173 0.007674837 0.001801237
## 4  1.00000000 0.0038616002 0.005715168 0.001390176 0.001699104 0.000463392
## 5  0.20059435 0.1560178306 0.797919762 0.195393759 0.320208024 0.101783061
## 6  0.04629169 0.0004977601 0.002986560 0.006470881 0.000000000 0.000000000
## 7  0.89098837 0.9353197674 0.950581395 0.725290698 0.795784884 0.209302326
## 8  0.08520527 0.3508907823 0.001936483 0.001161890 0.001936483 0.000000000
## 9  0.72737621 0.3932840068 0.004553216 0.002276608 0.001138304 0.000000000
## 10 0.19770475 0.1371935316 0.774126239 0.742827334 0.787167449 0.000000000
##             fl      fraspm          ga          gl          hi          ia
## 1  0.006199332 0.021459227 0.011444921 0.077730091 0.01430615 0.140677158
## 2  0.788508557 0.000407498 0.902200489 0.000407498 0.11776691 0.042787286
## 3  0.093351085 0.004777195 0.043308012 0.020675072 0.16477406 0.011355627
## 4  0.012511585 0.001390176 0.004324992 0.001235712 0.02301514 0.002008032
## 5  0.038632987 0.312778603 0.155274889 0.037890045 0.02748886 0.329866270
## 6  0.608262817 0.000000000 0.062220010 0.000000000 0.24240916 0.000000000
## 7  0.577034884 0.421511628 0.726017442 0.049418605 0.34229651 0.943313953
## 8  0.024012393 0.000000000 0.006196747 0.000000000 0.00929512 0.033307514
## 9  0.013659647 0.000000000 0.007398976 0.000000000 0.02105862 0.001707456
## 10 0.649973918 0.048513302 0.918622848 0.003129890 0.09181012 0.658841941
##             id          il           in          ks          ky          la
## 1  0.8950882213 0.116833572 0.0557939914 0.172627563 0.021459227 0.023843586
## 2  0.0114099430 0.238386308 0.1483292584 0.142216789 0.346780766 0.740016300
## 3  0.0422116062 0.021379904 0.0144882136 0.009476075 0.016367766 0.030307777
## 4  0.0586963238 0.004324992 0.0009267841 0.001235712 0.002162496 0.008958913
## 5  0.2169390788 0.560921248 0.4680534918 0.077265973 0.284546805 0.040861813
## 6  0.0004977601 0.001991040 0.0000000000 0.001493280 0.002488800 0.100049776
## 7  0.9367732558 0.984011628 0.9527616279 0.821220930 0.852470930 0.670058140
## 8  0.0081332301 0.039116964 0.0151045701 0.181642138 0.008520527 0.061967467
## 9  0.3386454183 0.008537279 0.0028457598 0.042117245 0.001707456 0.017643711
## 10 0.0558163798 0.942618675 0.9300991132 0.651538863 0.965049557 0.768909755
##             lb          ma          mb          md          me          mi
## 1  0.1092036242 0.061516452 0.323319027 0.027181688 0.082021936 0.1468764902
## 2  0.0008149959 0.107986960 0.003259984 0.345558272 0.027302363 0.0383048085
## 3  0.0184039471 0.031404182 0.018560576 0.022789568 0.020910016 0.0240426032
## 4  0.0015446401 0.018690145 0.003398208 0.007568736 0.004170528 0.0046339203
## 5  0.3112927192 0.869985141 0.445765230 0.531203566 0.852897474 0.8068350669
## 6  0.0000000000 0.006968641 0.000000000 0.017421603 0.000000000 0.0009955202
## 7  0.3626453488 0.962209302 0.848110465 0.918604651 0.926598837 0.9811046512
## 8  0.0000000000 0.009682417 0.006196747 0.014717273 0.005809450 0.0123934934
## 9  0.0000000000 0.015367103 0.000569152 0.013659647 0.005122368 0.0062606716
## 10 0.0203442879 0.775169536 0.153364632 0.932185707 0.515910276 0.7897756912
##             mn          mo         ms          mt          nb          nc
## 1  0.227467811 0.092513114 0.01144492 0.9713876967 0.0629470672 0.013352408
## 2  0.015077425 0.298288509 0.77791361 0.0126324368 0.0052974735 0.781581092
```

```
## 3  0.014174955 0.018638891 0.01417495 0.0335969927 0.0116688856 0.034145195
## 4  0.001390176 0.004016064 0.00308928 0.0154464010 0.0018535681 0.006641952
## 5  0.602526003 0.208766716 0.03789004 0.2637444279 0.6901931649 0.377414562
## 6  0.000000000 0.002488800 0.03235441 0.0009955202 0.0000000000 0.014932802
## 7  0.963662791 0.880087209 0.64970930 0.9636627907 0.8241279070 0.842296512
## 8  0.020914020 0.072037180 0.02052672 0.0395042603 0.0003872967 0.005422153
## 9  0.001707456 0.018212863 0.00569152 0.1548093341 0.0005691520 0.003414912
## 10 0.537819510 0.904016693 0.83881064 0.0761606677 0.2529994784 0.930620762
##             nd          ne          nf          nh          nj          nm
## 1  0.3290414878 0.290414878 0.105388650 0.049594659 0.025274201 0.522174535
## 2  0.0073349633 0.035452323 0.002852486 0.025264874 0.216381418 0.100651997
## 3  0.0056386561 0.007126635 0.020283499 0.011590571 0.019735296 0.011042368
## 4  0.0009267841 0.000617856 0.002316960 0.001390176 0.009113377 0.001699104
## 5  0.2043090639 0.115156018 0.514115899 0.779346211 0.684249629 0.061664190
## 6  0.0009955202 0.000000000 0.000000000 0.000000000 0.007964161 0.019910403
## 7  0.8604651163 0.871366279 0.615552326 0.902616279 0.947674419 0.872093023
## 8  0.0251742835 0.099535244 0.000000000 0.000000000 0.004647560 0.929899303
## 9  0.0039840637 0.025611838 0.000000000 0.000000000 0.004553216 0.459305635
## 10 0.1893583725 0.435054773 0.076682316 0.532603026 0.869066249 0.197183099
##             ns          nt          nu          nv          ny          oh
## 1  0.056270863 0.2665712923 0.113018598 0.67858846 0.113972341 0.051502146
## 2  0.015077425 0.0008149959 0.000407498 0.02770986 0.165036675 0.134881826
## 3  0.012138774 0.0364946354 0.023337771 0.01777743 0.046283969 0.020440128
## 4  0.003398208 0.0023169601 0.001390176 0.16203275 0.014519617 0.003707136
## 5  0.685735513 0.2243684993 0.138930163 0.03863299 0.955423477 0.621842496
## 6  0.000000000 0.0000000000 0.000000000 0.00248880 0.006470881 0.006470881
## 7  0.809593023 0.4854651163 0.177325581 0.74345930 0.992005814 0.974563953
## 8  0.000000000 0.0003872967 0.000000000 0.03253292 0.006971340 0.009295120
## 9  0.001138304 0.0011383039 0.000000000 0.93454752 0.023904382 0.005122368
## 10 0.246739697 0.0046948357 0.001564945 0.05059990 0.897235263 0.951486698
##             ok          on          or          pa          pe          pr
## 1  0.135431569 0.253695756 0.798760134 0.048640916 0.019551741 0.001430615
## 2  0.326405868 0.032599837 0.060717196 0.219233904 0.001629992 0.182966585
## 3  0.021928107 0.041663404 0.059284204 0.033440363 0.003837419 0.135484376
## 4  0.002316960 0.006796416 0.295180723 0.014674081 0.001390176 0.004633920
## 5  0.028231798 0.868499257 0.299405646 0.795690936 0.451708767 0.014858841
## 6  0.006968641 0.001493280 0.003484321 0.011448482 0.000000000 0.988053758
## 7  0.736191860 0.980377907 0.932412791 0.983284884 0.655523256 0.199854651
## 8  0.244771495 0.002323780 0.007358637 0.006196747 0.000000000 0.013555383
## 9  0.075128059 0.006829824 0.317017644 0.013659647 0.001138304 0.011383039
## 10 0.752217006 0.699530516 0.139280125 0.952008346 0.117892540 0.122065728
##             qc          ri          sc          sd          sk          tn
## 1  0.2060085837 0.009537434 0.020028612 0.407725322 0.433953267 0.017644254
## 2  0.0097799511 0.048899756 0.834148329 0.017522412 0.002852486 0.491850041
## 3  0.0448743050 0.006030229 0.026548673 0.006030229 0.012686976 0.018638891
## 4  0.0052517763 0.000308928 0.006641952 0.000772320 0.002008032 0.001699104
## 5  0.8142644874 0.549777117 0.152303120 0.169390788 0.343982169 0.277117385
## 6  0.0000000000 0.000000000 0.026879044 0.000000000 0.000000000 0.003484321
## 7  0.9389534884 0.859011628 0.710755814 0.899709302 0.825581395 0.834302326
## 8  0.0003872967 0.001161890 0.012780790 0.068164214 0.008520527 0.010457010
## 9  0.0011383039 0.002276608 0.009106431 0.020489471 0.002845760 0.001138304
## 10 0.4267083985 0.598330725 0.873761085 0.274908712 0.066770996 0.965049557
##             tx          ut          va          vi          vt          wa
## 1  0.18454936 0.751549833 0.023843586 0.0009537434 0.0548402480 0.820219361
```

```
## 2  0.62184189 0.046454768 0.578239609 0.1022819886 0.0126324368 0.032599837
## 3  0.12444201 0.046049025 0.023885974 0.0048555094 0.0132351789 0.054037121
## 4  0.01513747 0.005715168 0.004324992 0.0006178560 0.0033982082 0.136546185
## 5  0.03789004 0.121099554 0.514858841 0.0052005944 0.7971768202 0.355126300
## 6  0.15181682 0.001991040 0.007964161 0.7849676456 0.0000000000 0.001991040
## 7  0.74491279 0.861918605 0.916424419 0.1010174419 0.9156976744 0.951308140
## 8  0.68048025 0.187838885 0.004260263 0.0061967467 0.0003872967 0.002711077
## 9  0.22936824 0.963005122 0.003984064 0.0073989755 0.0022766079 0.137734775
## 10 0.70005216 0.102243088 0.964006260 0.0594679186 0.5414710485 0.125195618
##              wi           wv          wy           yt
## 1  0.141154030 0.0157367668 0.880305198 0.3171196948
## 2  0.027302363 0.1572942135 0.010187449 0.0004074980
## 3  0.020283499 0.0118255149 0.030777665 0.0402537395
## 4  0.002625888 0.0013901761 0.005560704 0.0035526722
## 5  0.720653789 0.4316493314 0.176077266 0.1827637444
## 6  0.001493280 0.0024888004 0.000000000 0.0000000000
## 7  0.983284884 0.8706395349 0.906250000 0.4622093023
## 8  0.017041053 0.0003872967 0.127807901 0.0003872967
## 9  0.002845760 0.0022766079 0.253272624 0.0011383039
## 10 0.681794471 0.8836724048 0.078768910 0.0062597809
```

```
#Visualizing the clusters
plotcluster(plants_matrix,clusters$cluster)
```



Reference: https://stats.stackexchange.com/questions/31083/how-to-produce-a-pretty-plot-of-the-results-of-k-means-cluster-analysis https://stackoverflow.com/questions/41972270/how-to-convert-object-of-transaction-to-dataframe-in-r