ashishsa_hw3_p2

We first read the US Arrests Data.

```
library(ISLR)
```

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

```
library(philentropy)
```

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

```
library(ggdendro)
```

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

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(dendextend)
```

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

```
##
## ---------------------
## Welcome to dendextend version 1.13.4
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:ggdendro':
##
##     theme_dendro
```

```
## The following object is masked from 'package:stats':
```

```
##
##     cutree
```

```
library(ggplot2)
library(dendextend)
library(kohonen)
```

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

```
data("USArrests")
df1 <- USArrests
head(df1)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

We scale the data first to avoid large values to skew the analysis.

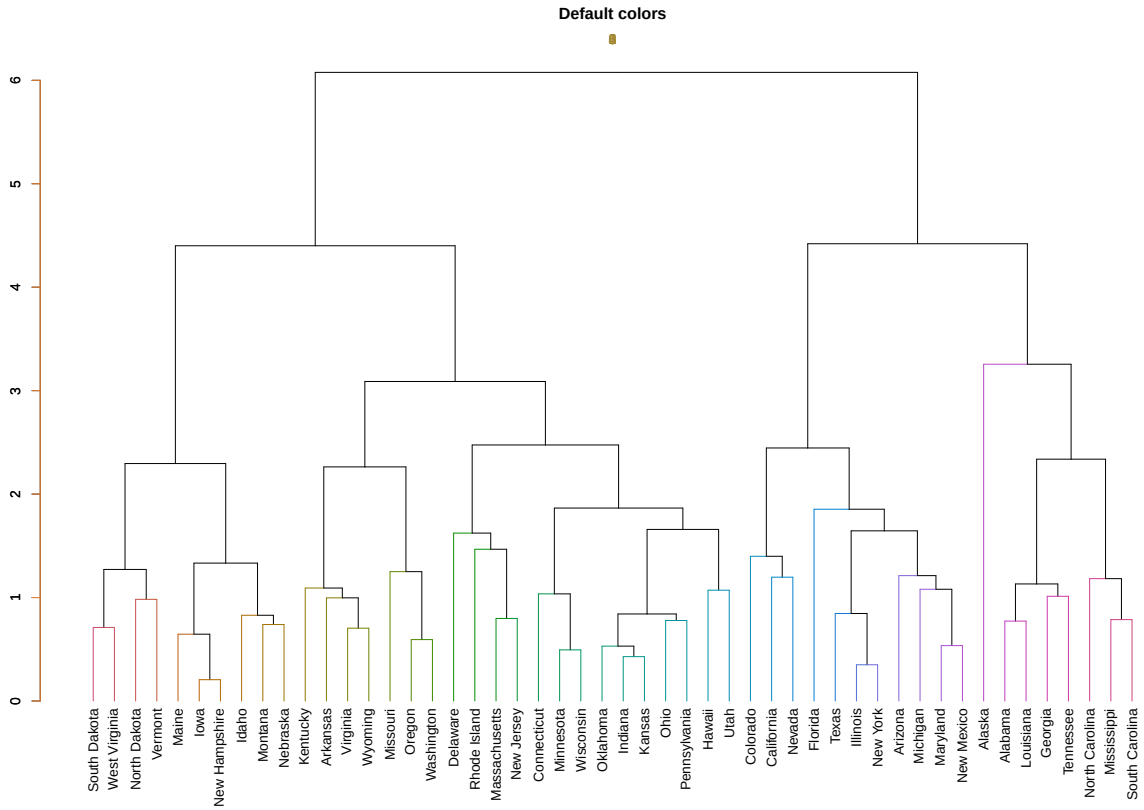
```
m <- apply(df1,2,mean)
s <- apply(df1,2,sd)
z <- scale(df1,m,s)
```

We perform Heirarchical Clustering using Complete Linkages and Eucledian Distance as a measure of distance.

```
dist1 <- dist(z,method="euclidean")
hclust1 <- hclust(dist1,method='complete')
```

We create a Dendrogram from the Heirarchical Cluster.

```
dend1 <- as.dendrogram(hclust1)
dend1 %>% set("branches_k_color") %>%
  plot(main = "Default colors") %>%
  axis(side = 2,col = "#F38630",labels = TRUE) %>%
  mtext(col = "#A38630")
```



As we can observe from the plot the data is partitioned into 4 groups at the same level.

We first try to cluster the Dendrogram into 3 groups and use k=3 to cluster the data into 3 groups.

```
cutree2 <- cutree(dend1,k=3)
cutree2
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              2              3              2
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              3              2              1
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              3              2              3              3
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              3              1              3              2
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              3              2              3              1              3
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              3              3              2              3              3
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              2              2              1              3              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              3              3              3              3              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              3              1              2              3              3
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              3              3              3              3              3
```

We now try to cluster the Dendrogram into 3 groups by cutting at an appropriate height. We observe that it

clusters into 2 groups at a height of 4.5 and it clusters into 4 groups at a height of 4.4.

```
cutree1 <- cutree(dend1,h=4.4)
cutree1
```

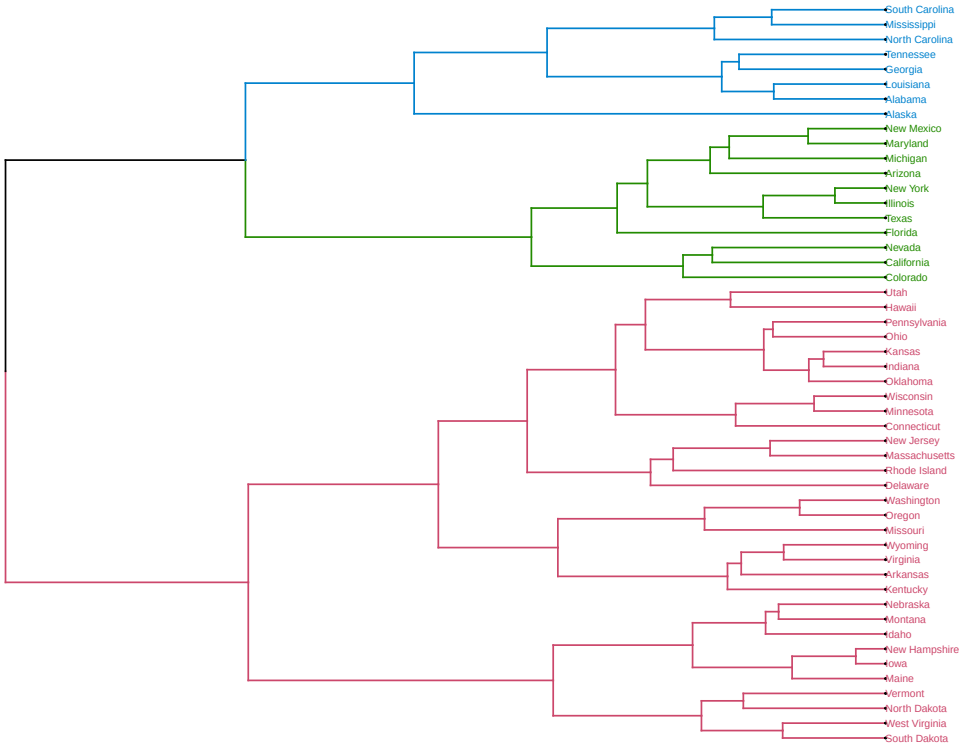
```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              2              3              2
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              3              2              1
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              4              2              3              4
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              3              1              4              2
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              3              2              3              1              3
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              4              4              2              4              3
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              2              2              1              4              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              3              3              3              3              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              4              1              2              3              4
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              3              3              4              3              3
```

By manually comparing the values we can infer that the Dendrogram cannot actually be divided into 3 clusters by a clear break a the 2 main groups cut into 4 groups at the same height.

The function k=3 actually divides the cluster into 3 groups by dividing the cluster in a way that 2 groups cluster as 1 group(That is groups 3,4 clusters into group3) and 1 group clusters into 2(That is groups 1,2).

This can be visually confirmed by the following Dendrogram where 2 entires sub group clusters into the Red Group, while 2 more groups are formed that is groups Blue and Green.

```
dend <- z %>% dist %>%
  hclust %>% as.dendrogram %>%
  set("branches_k_color", k = 3) %>% set("branches_lwd", 0.7) %>%
  set("labels_cex", 0.6) %>% set("labels_colors", k = 3) %>%
  set("leaves_pch", 19) %>% set("leaves_cex", 0.5)
ggd1 <- as.ggdend(dend)
ggplot(ggd1, horiz = TRUE)
```




```
training_data <- as.matrix(z)
```

We need to apply SOM to the data and we know that the size of SOM is given by 5*sqrt(N) Here N is the number of observation. So to break it down, If Columns are your feature and each row is one observation then Number of Nodes = 5 sqrt (# of Rows * # of columns).

We observe that there are 50 Rows and 4 Columns. So the Number Of Observations can be summed up as:

```
nr <- nrow(df1)
nc <- ncol(df1)
s_lower <- 0.25*sqrt((nr*nc))
s_lower
```

```
## [1] 3.535534
```

```
s_upper <- 4*sqrt((nr*nc))
s_upper
```

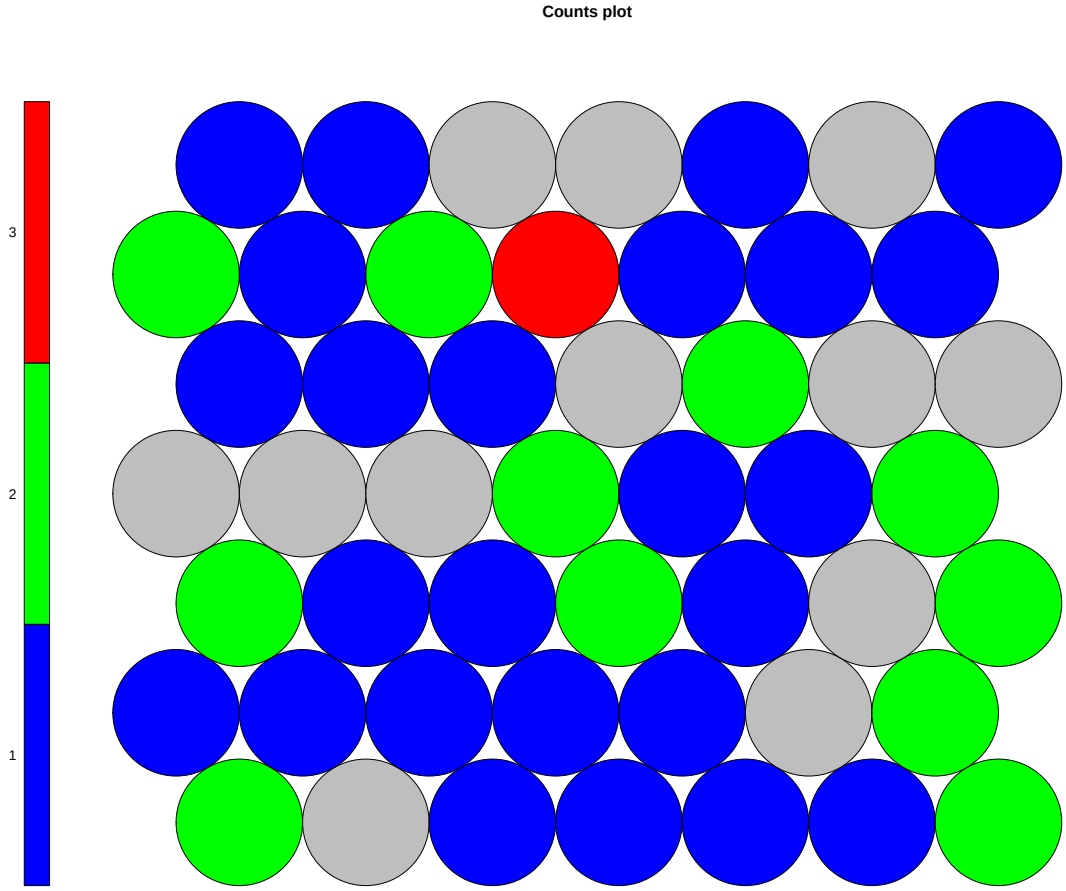
```
## [1] 56.56854
```

We observe that the nearest square less than 56 is obtaiuned by dim=7. Here the feature map derives the size of 49=>(Xdim=7*Ydim=7). So we create a SOM having x and y dimension as 7.

```
data_train_matrix <- training_data
som_grid <- somgrid(xdim = 7, ydim=7, topo="hexagonal")
som_model <- som(data_train_matrix,grid=som_grid,rlen=10000)
```

The counts lot gives us the count of the number of states map into the different units.

```
coolBlueHotRed <- function(n,alpha=1){rainbow(n,end=4/6,alpha=alpha)[n:1]}
plot(som_model,type="counts",palette.name=coolBlueHotRed)
```



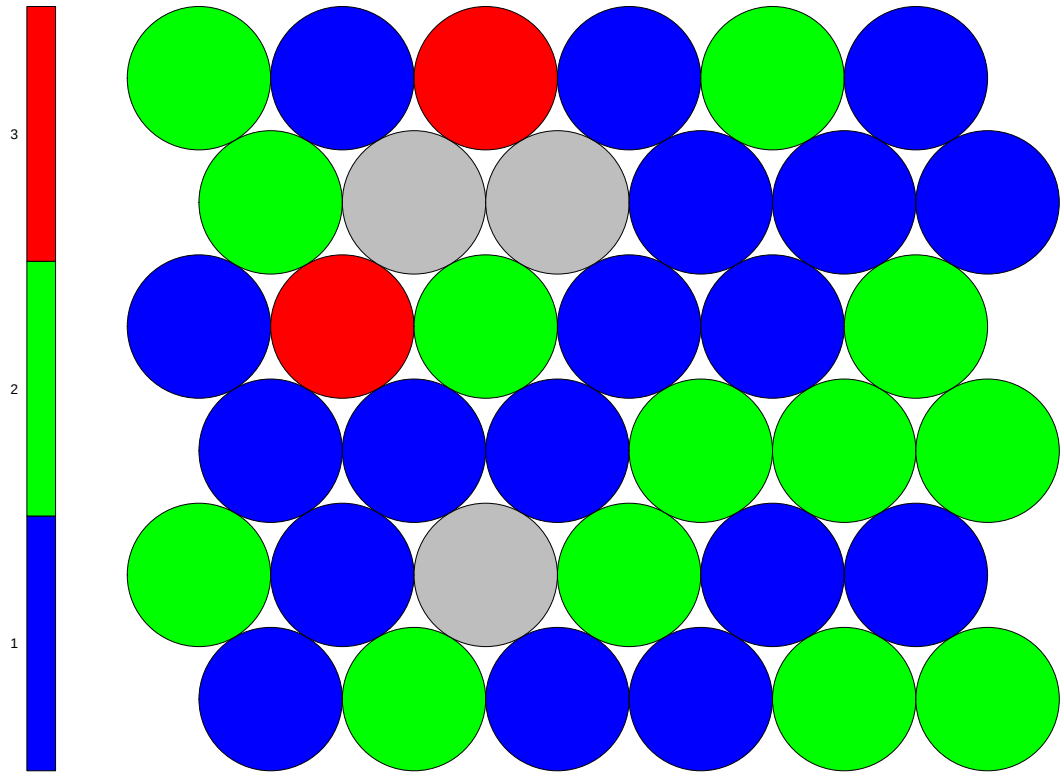
As we can observe here there are too many states that donot map into different units we reduce the grid size

to reduce the number of unnecessary nodes.

```
som_grid <- somgrid(xdim = 6, ydim=6, topo="hexagonal")
som_model <- som(data_train_matrix,grid=som_grid,rlen=10000)
```

The counts lot gives us the count of the number of states map into the different units.

```
plot(som_model,type="counts",palette.name=coolBlueHotRed)
```



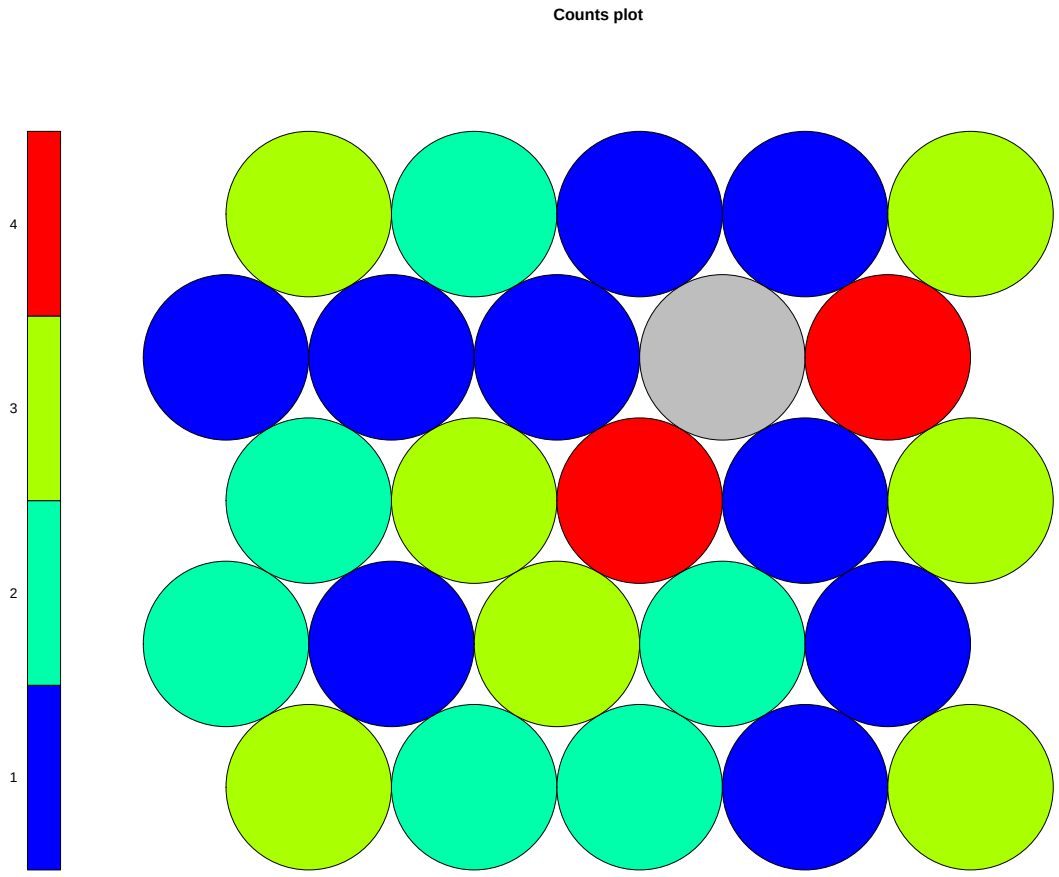
As we can observe there are still units that donot map to any state. We further reduce the size of the SOM

Map.

```
som_grid <- somgrid(xdim = 5, ydim=5, topo="hexagonal")
som_model <- som(data_train_matrix,grid=som_grid,rlen=10000)
```

The counts lot gives us the count of the number of states map into the different units.

```
plot(som_model,type="counts",palette.name=coolBlueHotRed)
```



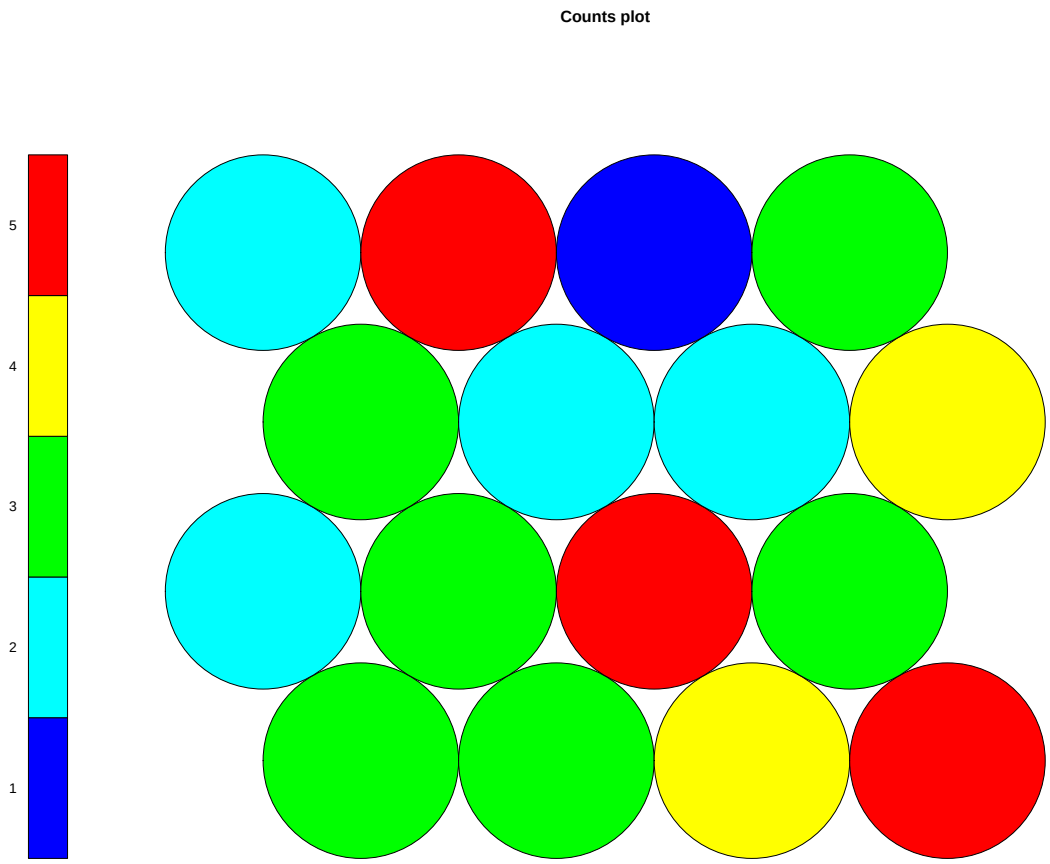
Although the number of states getting mapped to the plot increases there are still instances when there are

empty nodes in the map and to avoid it we further reduce the size of the SOM Map.

```
som_grid <- somgrid(xdim = 4, ydim=4, topo="hexagonal")
som_model <- som(data_train_matrix,grid=som_grid,rlen=10000)
```

The counts lot gives us the count of the number of states map into the different units.

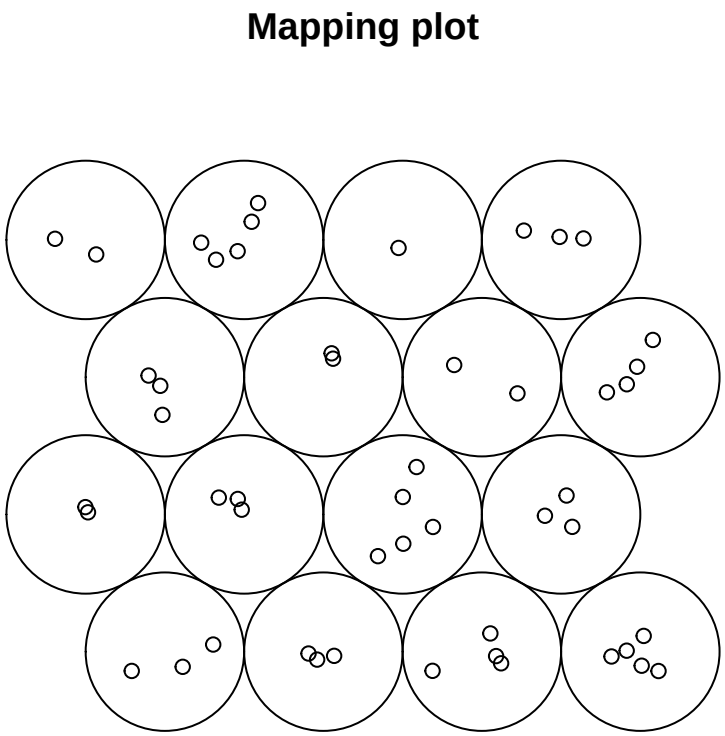
```
plot(som_model,type="counts",palette.name=coolBlueHotRed)
```



We now plot all this Information onto a Grid. This further reinforcees the information provided by the above

Heat Map.

```
plot(som_model,type="mapping")
```



This gives us the summary of the Self Organization Map.

```
summary(som_model)
```

```
## SOM of size 4x4 with a hexagonal topology and a bubble neighbourhood function.
## The number of data layers is 1.
## Distance measure(s) used: sumofsquares.
## Training data included: 50 objects.
## Mean distance to the closest unit in the map: 0.332.
```

The codes give us the list of 16 different prototypes and the code books for all of the different prototypes.

```
som_model$codes
```

```
## [[1]]
##         Murder    Assault    UrbanPop       Rape
## V1  -0.9474192 -0.3465109  1.17538893 -1.0094802
## V2  -0.3451161 -0.9667896  0.77111342 -0.2593686
## V3  -1.2399737 -1.3400079 -0.31166831 -1.0367450
## V4  -1.0773691 -1.2077822 -1.63134204 -1.2068564
## V5  -0.2574792  0.3224058  1.04647849 -0.4162659
## V6  -0.8752023 -0.3665368  0.56688835  0.5011652
## V7  -0.1774348 -0.3814432 -0.09416719 -0.2442084
## V8  -0.8076697 -0.7210928 -0.64653189 -0.5951015
## V9   0.8488411  0.7469500  1.20020817  0.4306367
```

```
## V10  0.1435059  0.2533496  0.60303503  1.3421172
## V11  0.3341305 -0.2478807 -1.00553417 -0.3532002
## V12  1.6218421  0.6039149 -0.35212984  0.2914396
## V13  0.6382549  1.1217085  1.42041932  2.3503064
## V14  0.8690294  1.4677226  0.60656108  1.1069105
## V15  0.5078625  1.1068225 -1.21176419  2.4842029
## V16  1.5850498  1.3838009 -1.36927344 -0.2567538
```

We now analyze the list of all the states and the prototypes that are closest to them.

```
som_model$unit.classif
```

```
##  [1] 12 15 14 11 13 10  1  5 14 12  2  8  9  7  3  7 11 12  4 14  1 14  3 16 10
## [26]  8  8 13  3  5 14  9 16  4  2  7  6  2  1 16  4 12  9  6  4  7  6  4  3  7
```

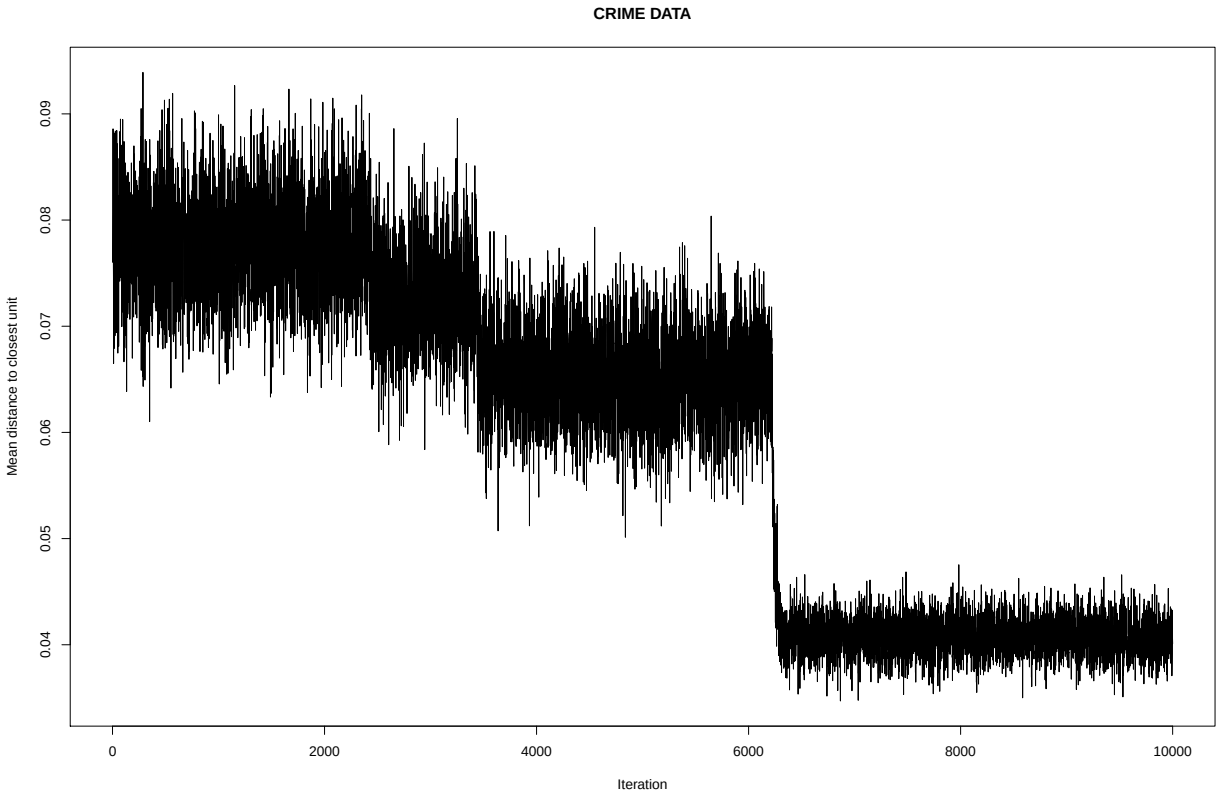
We now analyze the relative changes of the prototype code book vectors as we proceed through 200 different iteration.

```
head(som_model$changes)
```

```
##            [,1]
## [1,] 0.07602028
## [2,] 0.07880837
## [3,] 0.08104744
## [4,] 0.08284668
## [5,] 0.08858940
## [6,] 0.08293387
```

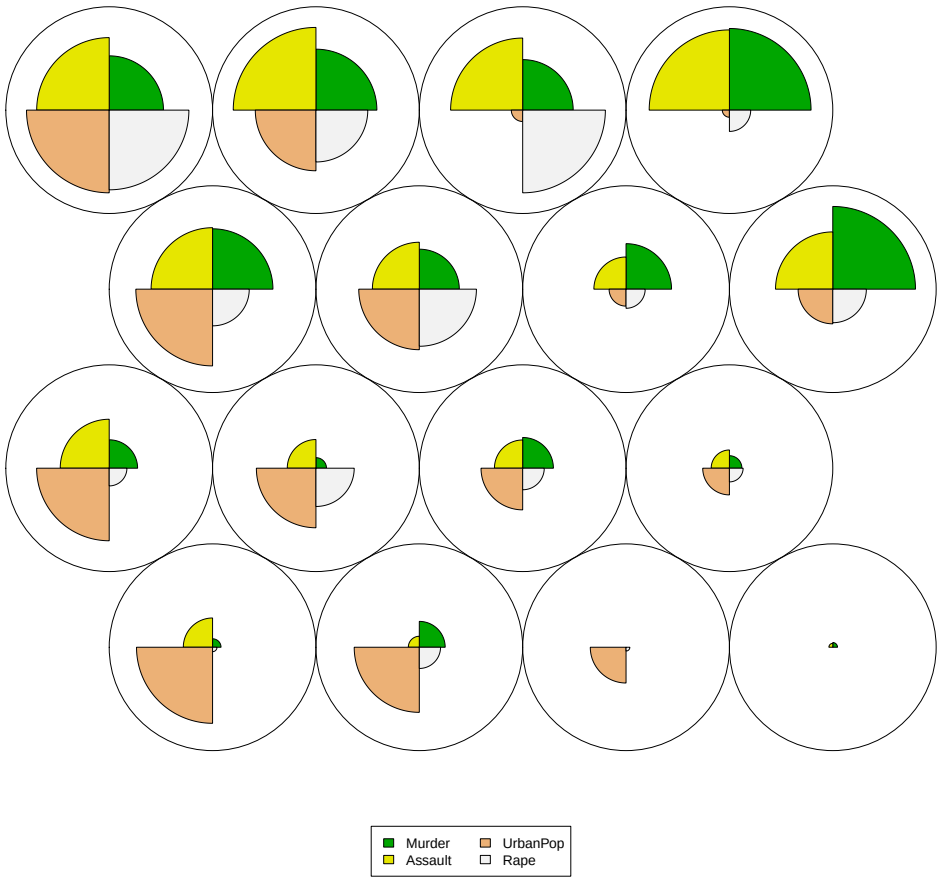
We observe that the changes start decreasing as we start increasing the number of iterations. A better understanding can be obtained from the Graph which represents the changes in the prototype. The things start converging as we move in the iteration.

```
plot(som_model,type="changes",main="CRIME DATA")
```



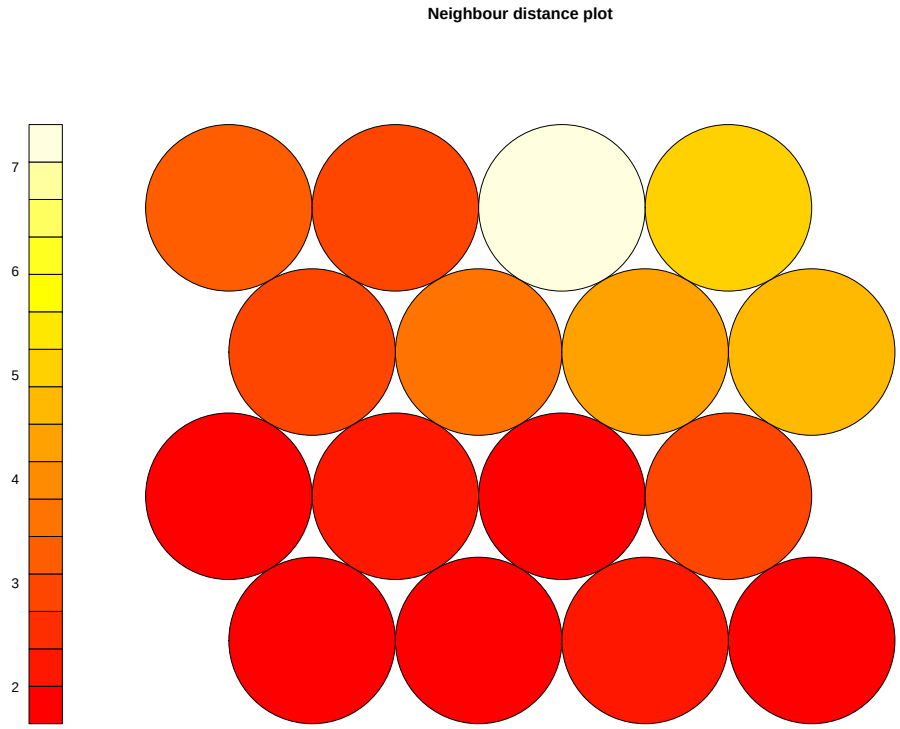
We now plot the Codes plot which gives us the distribution of various crimes across various nodes.

```
plot(som_model,type = "codes")
```



We now plot the distance to the Neighbours between each states that are plotted into a particular node.

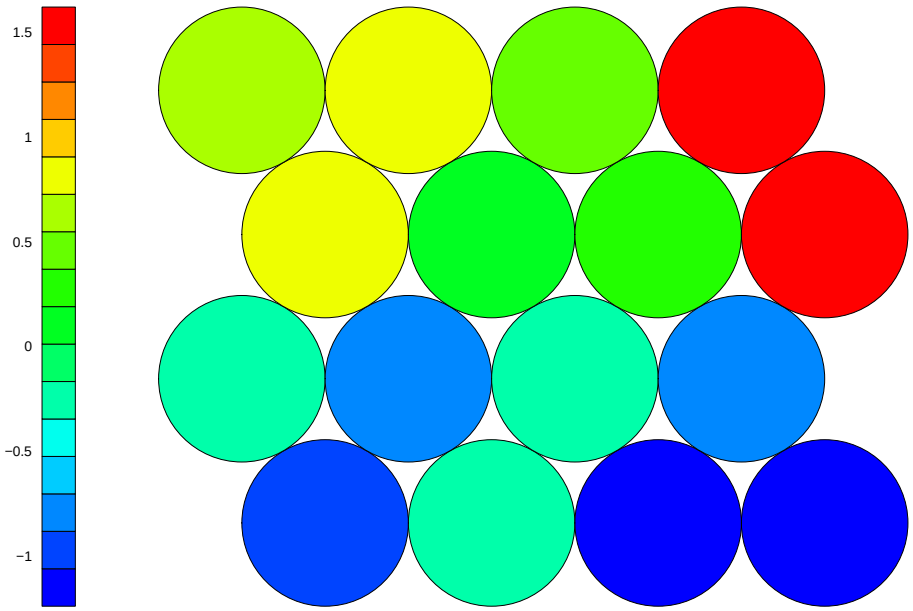
```
plot(som_model,type="dist.neighbours")
```



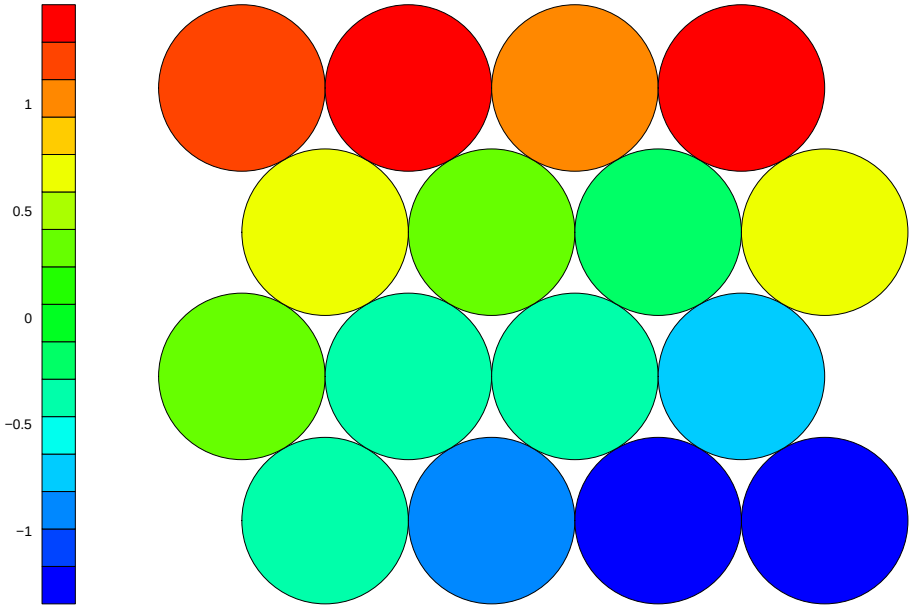
We now plot the componenet plane plots. Which Visualizes the original data over the SOM that we have created.

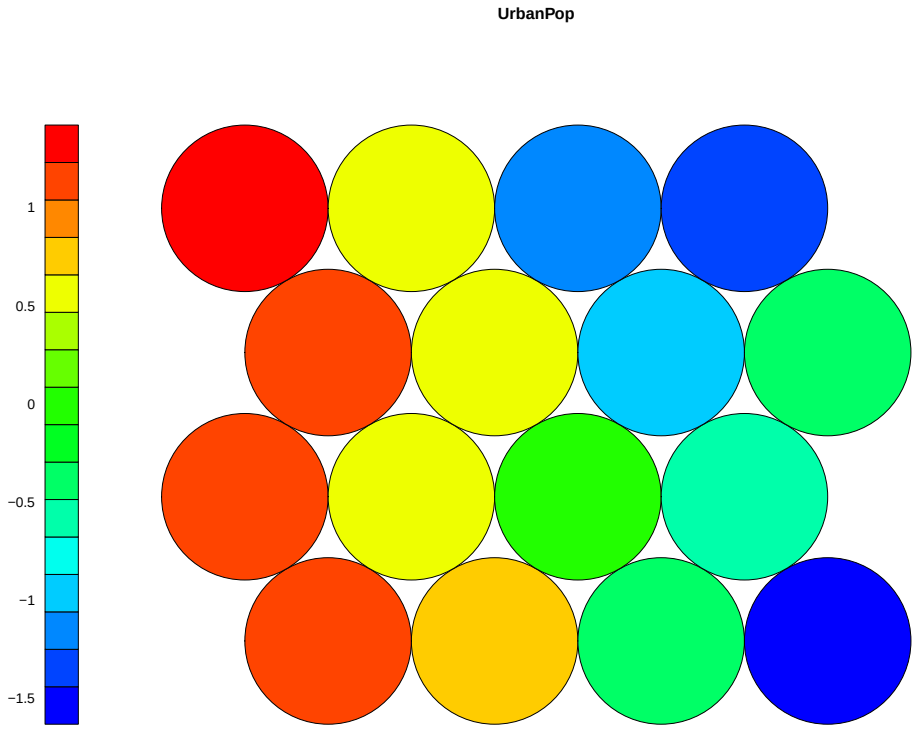
```
codes <- som_model$codes[[1]]

som_model$data <- data.frame(som_model$data)
for (i in 1:4){
  plot(som_model, type = "property", property = codes[,i], main=names(som_model$data)[i],palette.name=co
  }
```

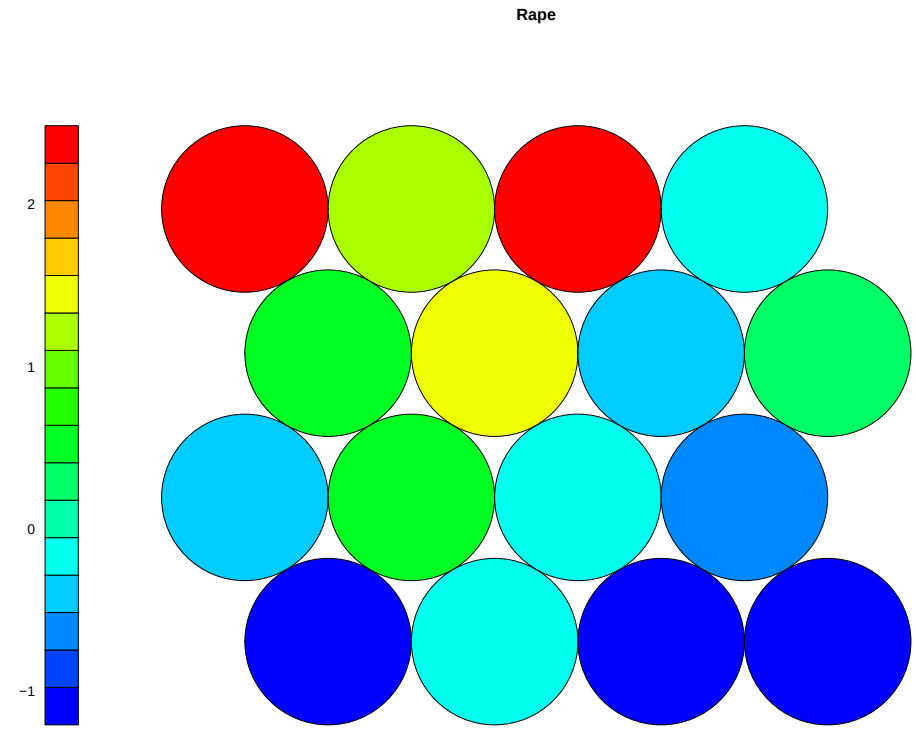
Murder



Assault



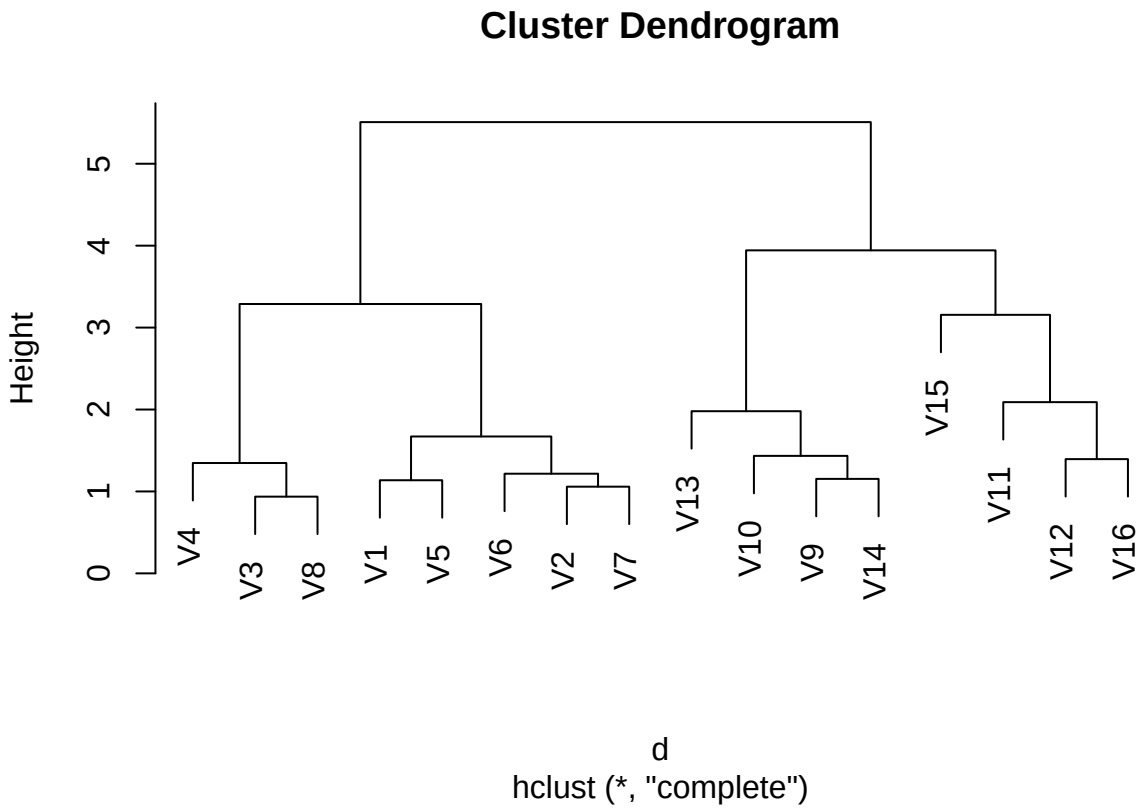




We find the distance using the SOM_Model codes and observe a clear distinction between the height of the

dendrogram between the 2 clusters. This helps us find the point for the cutoff and also helps us easily cluster the dat into 3 distinct groups which was difficult in the case of Plain Heirarchical Clustering using Eucledian Distance.

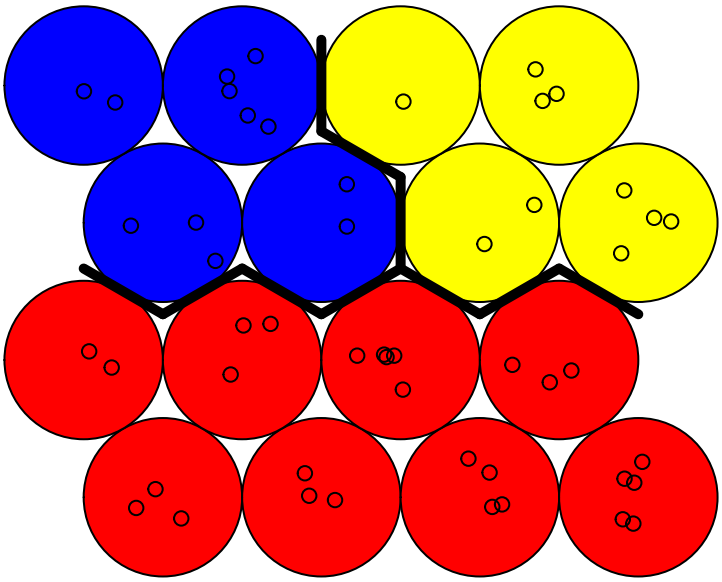
```
d <- dist(codes)
hc <- hclust(d)
plot(hc)
```



We now plot the SOM amd specify the cutoff point at a location where we can observe the approximate cutoff in the Dendrogram plotted above.

By specifying the cutoff point obtained from the plot above and clustering the data at that height gives us the approximate distribution of the data into 3 clusters.

```
som_cluster <- cutree(hc,h=3.7)
# plot these results:
my_pal <- c("red","blue","yellow")
my_bhcol <- my_pal[som_cluster]

{plot(som_model,type="mapping",col="black",bgcol = my_bhcol)
  add.cluster.boundaries(som_model,som_cluster)}
```



Comparison Of Heirarchical Clustering To Self Organization Map.

Heirarchical CLustering: 1) Heirarchical CLustering works by finding the measure of similarity between the data points and then correlating the data points based on Heirarchy. 2) It works well on Data that have Heirarchical Structure and we want to recover this Heirarchy. 3) It doesnot work well on Non Heirarchical or continuous data. 4) It is more imformative than K-Means or any such algorithm in that it doesnot return a flat cluster as in kmeans or other forms of Clustering Analysis. 5) It is Flexible with respect to proximity measure and can use many measures to visualize and evaluate its Results. 5) It is helpful in being able to visualize the data in the form of Dendrogram and also helps us divide the data into various clusters by using cuttree. We can also cluster and cut the data not only by the height but also by specifying the number of clusters that we want the data to be divided into. 6) Heirarchical clustering can basically be summarized as a non-iterative single step Greedy Algorithm which happens to be its major drawback as being greedy we can optimize the current step's task but we cannot guarentee the best partition in the future. 7) Incase there is a Non-Heirarchical Data or a Semi Heirarchical Data with not clear Heirarchy the Algorithm doesnot provide Best Results. 8) Missing Data causes the Heirarchical Cluster to fail. 9) A lot of decisions are arbitrary and there is no strong decision. 10) Incase a data is assigned to a given cluster we cannot go back and apply the instance to another cluster even though it might suite the other cluster better. 11) Not suitable for Very Very Large Dataset. 12) Order of Data and also Initial seeds have Impact on Results. 13) Sensitive to outliers. So need to scale data repeatedly.

Self Organization Map: 1) SOM can be summarized as basically a Neural Netowrk that maps a High Dimensional Input Data to a two dimsional Ouput space. 2) It consists of Nodes (2-D Rectangular or Hexagonal Grids). These units represents certain vectors in the data space that can be initialized. Vectors from dataset are presented to map in Random Order. The unit with the highest response to chosen vector is declared winner and the neighbourhood units are adapted to be more responsive to the present unit. Eucledian or other Gaussian Distance measures are used to determine neighbourhood. The neighbourhood function is declined during training as well as learning rate adapts to overall process. 3) It doesnot build a

Generative Model for the data. 4) It relies on predefined distance in feature space. 5) Not Intuitive without visualizing using other tools like Dendrogram. 6) Doesnot work well for categorical Data and even worst for Mixed Data. 7) Useful For Analysing Large Non intuitive data having cut points at the same / similar level. Hence widely used in Gene Mapping / Financial Data. 8) It is a Black Box process where the weights generated to map the SOM to the feature space is not very intuitively understandable to the user. 9) The nearby points should behave in a similar sense. Because the SOM tries to map the neighbourhood feature space in a way similar to the present feature. Hence Categorical data cannot be mappped as effectively. 10) Maps/ Clusters very large dataset with eese.

Comparison: 1) Hclust works well on Categorical Data. 2) SOM works well on Continuous Data. 3) Hclust Not Very Good to categorize very large, complex clusters as using Eucledian distnce causes loss of generality, data. 4) SOM Maps complex data very well as using the concept of Neural Networks it generates a feature space that maps the original data very well which can be used to further classify datasets. 2) Hclust used to visualize clusters. 5) SOM used for Dimensionality Reduction and mapping large dimension to 2-D space.