Wine regions clustering

Miguel Ángel Juárez Garzón

Libraries

```
library("kohonen")
```

```
## Warning: package 'kohonen' was built under R version 4.0.5
```

Data Load

```
data("wines")
head(wines)
```

```
##      alcohol malic acid  ash ash alkalinity magnesium tot. phenols flavonoids
## [1,]   13.20       1.78 2.14           11.2       100         2.65       2.76
## [2,]   13.16       2.36 2.67           18.6       101         2.80       3.24
## [3,]   14.37       1.95 2.50           16.8       113         3.85       3.49
## [4,]   13.24       2.59 2.87           21.0       118         2.80       2.69
## [5,]   14.20       1.76 2.45           15.2       112         3.27       3.39
## [6,]   14.39       1.87 2.45           14.6        96         2.50       2.52
##      non-flav. phenols proanth col. int. col. hue OD ratio proline
## [1,]              0.26    1.28      4.38     1.05     3.40    1050
## [2,]              0.30    2.81      5.68     1.03     3.17    1185
## [3,]              0.24    2.18      7.80     0.86     3.45    1480
## [4,]              0.39    1.82      4.32     1.04     2.93     735
## [5,]              0.34    1.97      6.75     1.05     2.85    1450
## [6,]              0.30    1.98      5.25     1.02     3.58    1290
```

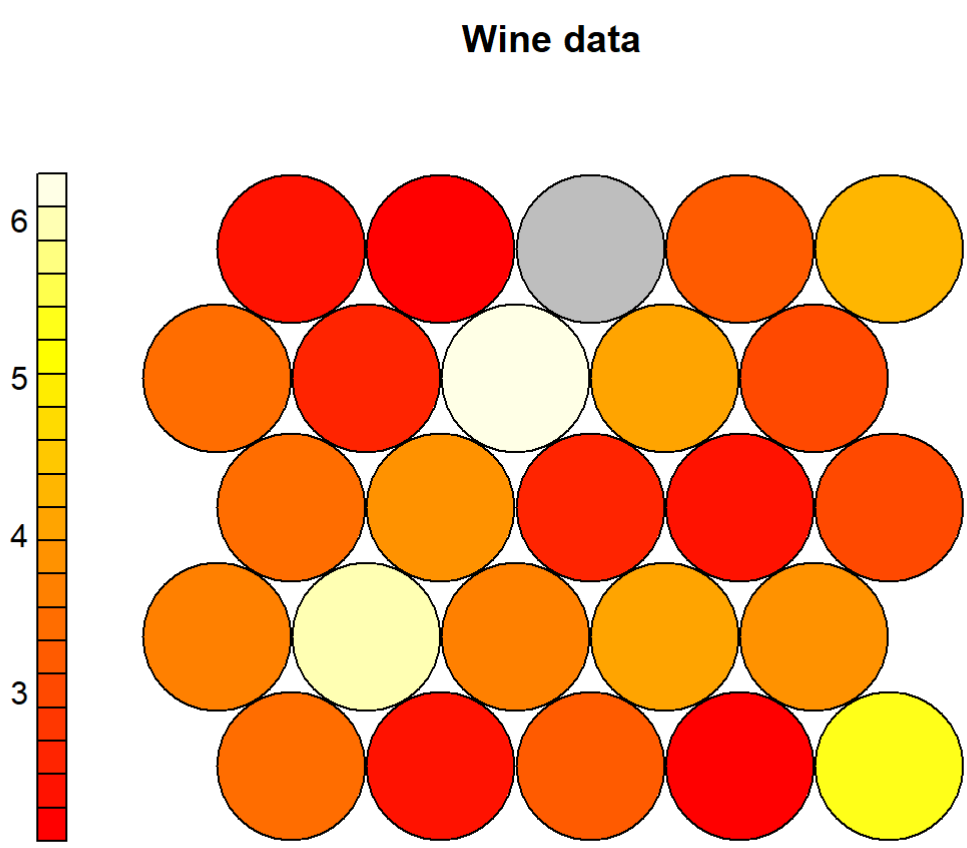
```
Swine<-scale(wines)
```

Exercise 1: Mesh definition

```
grid.wine <- somgrid(5,5, "hexagonal")
```

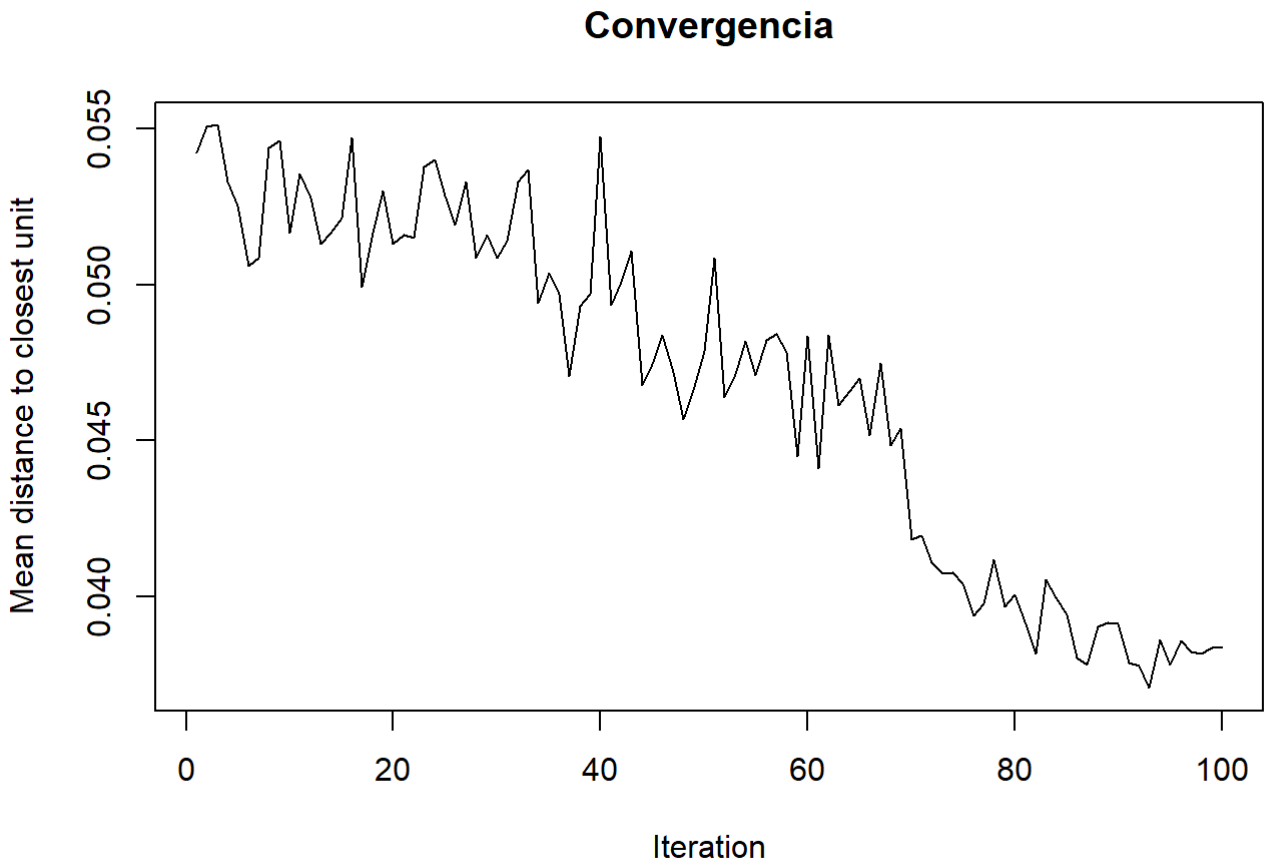
Exercise 2: Self-organizing map

```
som.wine <- supersom(data = Swine, grid = grid.wine)
plot(som.wine, type = "quality", main = "Wine data")
```



Convergence

```
plot(som.wine, type = "changes", main = "Convergencia")
```



We can observe how the algorithm effectively reduces the average distance to the nearest unit quickly and converges in a few steps.

Summary of SOM map

```
summary(som.wine)
```

```
## SOM of size 5x5 with a hexagonal topology and a bubble neighbourhood function.
## The number of data layers is 1.
## Distance measure(s) used: sumofsquares.
## Training data included: 177 objects.
## Mean distance to the closest unit in the map: 3.392.
```

```
print(som.wine$grid)
```

```
## $pts
##         x         y
##  [1,] 1.5 0.8660254
##  [2,] 2.5 0.8660254
##  [3,] 3.5 0.8660254
##  [4,] 4.5 0.8660254
##  [5,] 5.5 0.8660254
##  [6,] 1.0 1.7320508
##  [7,] 2.0 1.7320508
##  [8,] 3.0 1.7320508
##  [9,] 4.0 1.7320508
## [10,] 5.0 1.7320508
## [11,] 1.5 2.5980762
## [12,] 2.5 2.5980762
## [13,] 3.5 2.5980762
## [14,] 4.5 2.5980762
## [15,] 5.5 2.5980762
## [16,] 1.0 3.4641016
## [17,] 2.0 3.4641016
## [18,] 3.0 3.4641016
## [19,] 4.0 3.4641016
## [20,] 5.0 3.4641016
## [21,] 1.5 4.3301270
## [22,] 2.5 4.3301270
## [23,] 3.5 4.3301270
## [24,] 4.5 4.3301270
## [25,] 5.5 4.3301270
##
## $xdim
## [1] 5
##
## $ydim
## [1] 5
##
## $topo
## [1] "hexagonal"
##
## $neighbourhood.fct
## [1] bubble
## Levels: bubble gaussian
##
## $toroidal
## [1] FALSE
##
## attr(,"class")
## [1] "somgrid"
```

```
getCodes(som.wine)
```

```
##         alcohol  malic acid         ash ash alkalinity   magnesium tot. phenols
## V1  -0.70810408 -0.88126837 -1.22930181   -0.542759788 -0.71761693   0.71809540
## V2  -0.95689741 -0.45129493 -1.32739875   -0.073862782 -1.03759290  -0.40539514
## V3  -1.48110633 -0.93746863  0.26316168    1.040440251 -1.24746656  -0.04782964
## V4  -0.86363715 -0.65095598 -0.04335630    0.388866959 -0.85001608  -1.12655343
## V5  -0.47302996 -0.93092010 -1.55829022   -1.179832396 -0.13243053  -0.44627010
## V6  -0.62893819 -0.85471758 -1.24029236   -0.750072753  3.00400340  -0.08635069
## V7  -1.49114558  0.18633418  0.08231845    0.001052648 -0.46762771   0.76184682
## V8  -0.57512222 -0.06793824 -0.42749079    0.813435820 -0.56913025   0.14926010
## V9  -1.28188335  0.25600860  0.74287765    0.966207910 -0.64131034  -0.57469257
## V10 -0.72224028  0.06688795 -0.58599893   -0.348734912 -0.36395512  -1.15194798
## V11  0.96831064  0.24664205 -0.18361934   -0.922097565  0.73192976   0.95030515
## V12  0.10242470 -0.46676496  0.61746304    0.175430867 -0.20645484   0.59470810
## V13 -0.48880501  1.00634412  0.86092038    1.250518512  0.11823437   0.05160825
## V14  0.19911702  0.70677314  0.50327240    0.532858550  0.06290808  -1.03589755
## V15  0.34887510  2.09800544 -0.01078449    0.578918997 -0.73911294  -1.17393371
## V16  1.03028156 -0.59747120 -0.44308388   -1.528102884 -0.32082134   0.54614378
## V17  0.63287205 -0.44524830  0.67995419   -0.460450932  0.07657329   0.66144644
## V18 -0.23304604 -0.25549796  2.09907911    1.786738038  1.71243170   1.00931795
## V19 -0.29518607 -0.03327225  0.37666730    0.561481474  0.95441664  -1.14060844
## V20 -0.08372397  0.53679501 -0.08050368    0.563790191 -0.46158110  -1.00413292
## V21  1.29453158 -0.49415070  0.10609009   -0.948908816  0.59254388   1.49269730
## V22  1.00055494 -0.47793679  1.24858086   -0.327698171  1.16598355   0.80480291
## V23  0.32235933 -0.06676411  0.94208499    0.407864187  0.98344434   0.14369630
## V24  0.01935370  1.02019384  0.26187717    0.578154597  1.12529656  -1.20514344
## V25  0.94384781  0.64028537  0.61159672    0.682432361 -0.19356742  -0.52452319
##      flavonoids non-flav. phenols     proanth    col. int.    col. hue
## V1   0.62581622       -0.93053225  0.19576884 -0.537507780  1.01495350
## V2  -0.17561831       -0.29790546 -0.01703420 -0.985411212  0.22997276
## V3  -0.15131631        0.50964261 -0.29923702 -1.131791433  1.12667592
## V4  -0.46979851        1.06934286 -0.13910903 -0.891580426  0.70925798
## V5  -0.75412618        0.46561714 -1.61321094 -0.652934707  0.68718385
## V6  -0.02458092       -0.76176477  1.91722882 -0.815498572  0.83047310
## V7   0.72111678       -0.66051490  1.71110832 -0.924363000 -0.22842463
## V8   0.40453618       -0.14485752 -0.03173339 -1.003141857 -0.37200017
## V9  -0.13403948        1.41298602 -0.35931892 -0.860899208  0.11918692
## V10 -1.11804943        0.93274117 -0.94850336 -0.252045124 -0.70692692
## V11  0.98138885       -0.89253226  0.42772053  0.071301951 -0.08409739
## V12  0.59161345       -0.83269601  0.32946885 -0.424713977  0.70563338
## V13  0.17904981        0.39967698  0.05121921 -0.858407474  0.07543154
## V14 -1.45666358        0.99348539 -1.30368493 -0.001664767 -0.68790738
## V15 -1.37403459        1.39773616 -0.90629848  0.299606682 -0.94440487
## V16  0.79298693       -0.68177142  0.55831189  0.069975280  0.64678063
## V17  0.89930961       -0.39711957  0.73053529  0.055898245  0.64357302
## V18  1.18698197        0.09522731  0.48020236 -0.336797247  0.66120447
## V19 -0.82584631       -1.12594154 -0.92077456 -0.020862810 -0.92529614
## V20 -1.35298740        0.87930702 -0.85877175  1.556336693 -1.39716184
## V21  1.47923057       -0.69533760  0.98837952  0.953693737  0.58713733
## V22  0.84713757       -0.01229689 -0.02936765  0.219253264  0.86136889
## V23  0.39546536       -0.42556912 -0.07880044  0.460336732 -0.08174840
## V24 -0.95289387       -0.31993822 -0.59607471  1.848432370 -1.61655641
## V25 -1.12653950        1.09597986  0.24986889  2.232779642 -1.42689266
##       OD ratio    proline
## V1   0.4888055 -0.7478037
## V2   0.7557575 -0.8336213
```

```
## V3   0.4194136 -0.7394109
## V4  -0.4059349 -0.8990891
## V5  -0.9064182 -0.5835028
## V6   0.2265068  0.2436104
## V7   0.8395073 -0.6471722
## V8   0.5351187 -1.1755390
## V9   0.1153592 -0.6742561
## V10 -0.9989047 -0.4528960
## V11  1.0997132  0.6011253
## V12  1.0319576  0.1706328
## V13  0.4993262 -0.7772000
## V14 -0.7899125 -0.3042008
## V15 -1.2864802 -0.5230553
## V16  0.6512240  1.3021975
## V17  0.4641855  1.3001734
## V18  0.9559572  0.1489909
## V19 -1.6220888 -0.5613126
## V20 -1.3592037 -0.3025866
## V21  0.6026715  2.2120235
## V22  0.5324641  1.3141334
## V23 -0.2194400  0.5605888
## V24 -1.4184554 -0.2577044
## V25 -1.3125941 -0.3615702
```

```
print(som.wine$unit.classif)
```

```
##   [1] 16 17 21 18 21 16 22 16 16 21 16 16 16 21 22 22 22 21 11 11 12 11 12 12 18
##  [26] 17 16 22 16 17 21 17 22 22 12 22 17 16 11 11 11 11 11 11 11 11 11 17 21 16
##  [51] 17 21 22 11 17 11 17 21  5  5  5  5  1  4 12  1  2  5  6 10 12  8 18 12  5
##  [76]  1 10  6 13  1  1  3 15  7  1  4  3  9  3  4  4  4  2  1  6 19  1  1  7  1
## [101]  2  8  2  2  9  2  4  2  7  7  8  9  3  3  3  2  8 10  2  7 18 13  8  7  8
## [126]  8  9  8  9 19 19 19 19 10 10 15 15 15 14 14 14 14 15 24 14 15 15 20 24 24
## [151] 24 19 25 20 15 25 20 25 25 20 14 14 14 20 15 25 20 25 24 10 20 25 15 20 24
## [176] 24 25
```

Exercise 4: Number of assignments to neuron

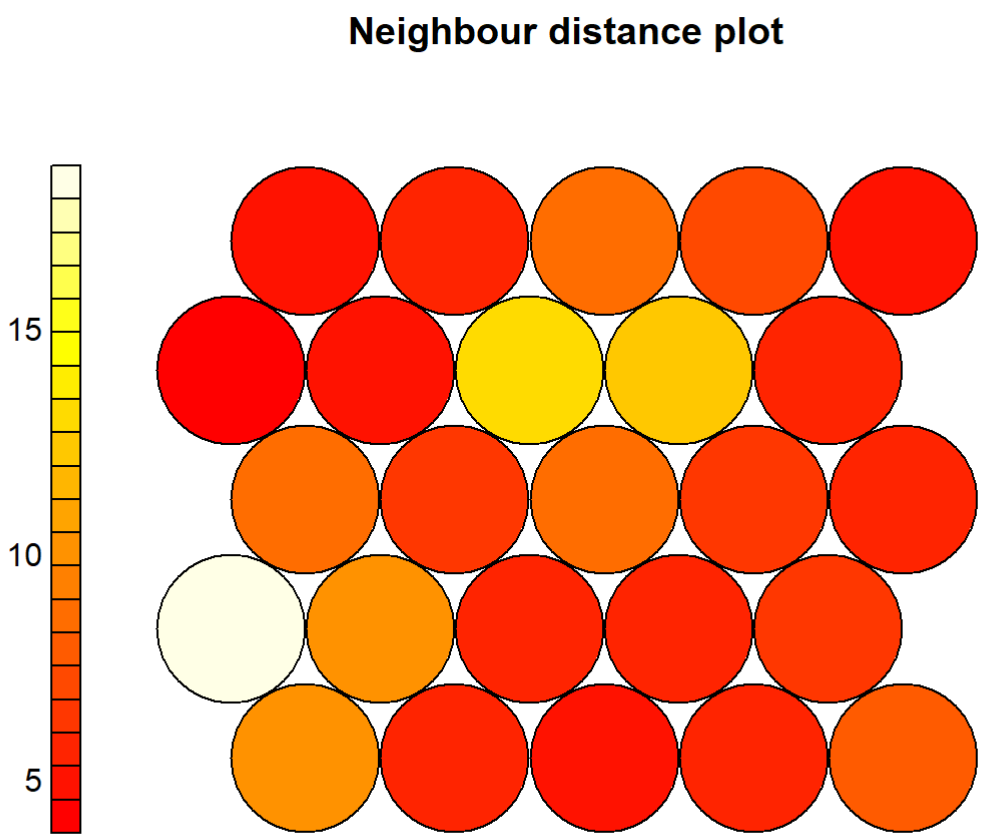
```
num_asig <- table(som.wine$unit.classif, dnn = "Neurona")
num_asig
```

```
## Neurona
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 24 25
## 10  9  6  6  6  3  6  8  5  6 14  7  2  8 10 11  9  4  6  8  9  9  7  8
```

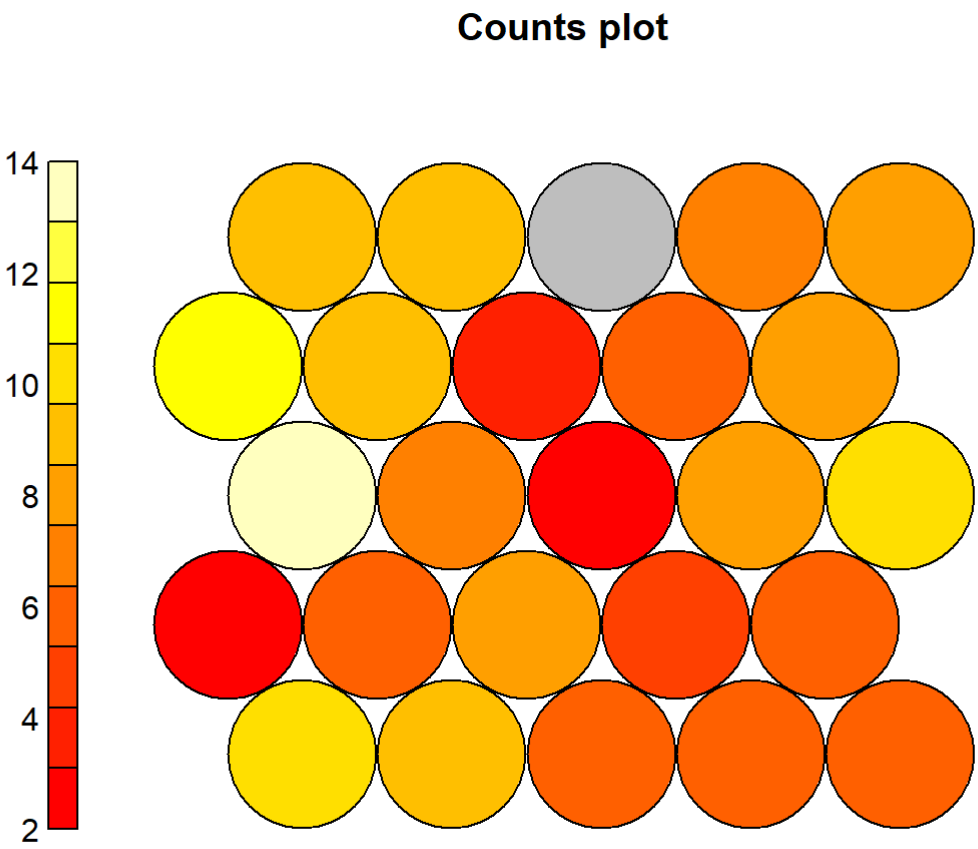
Based on this table, we can observe that there are regions with a higher number of assignments, while other regions have lower assignment values. This indicates the central points of each group in the regions with a higher number of assignments, while regions with fewer assigned instances serve as transition zones between groups. For example, neurons 1-2, 11, 15-16 would be central areas of the clusters, while neurons 6 or 18, among others, would be transition neurons.

Clusters Visualization

```
plot(som.wine, type="dist.neighbours")
```

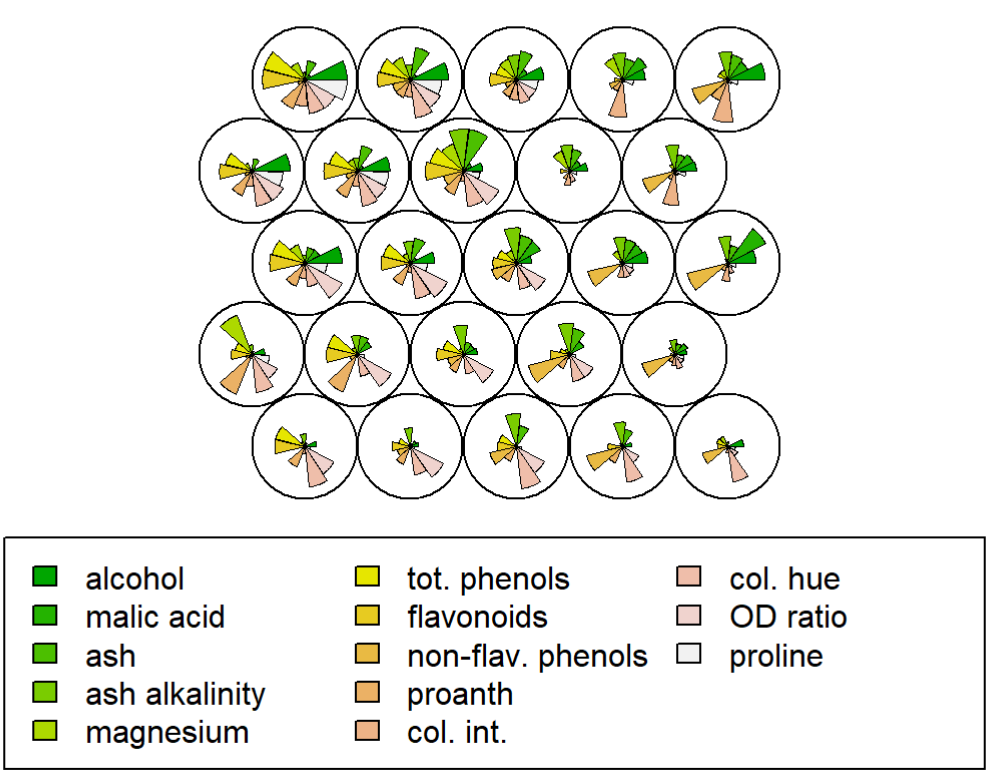



```
plot(som.wine, type="count")
```



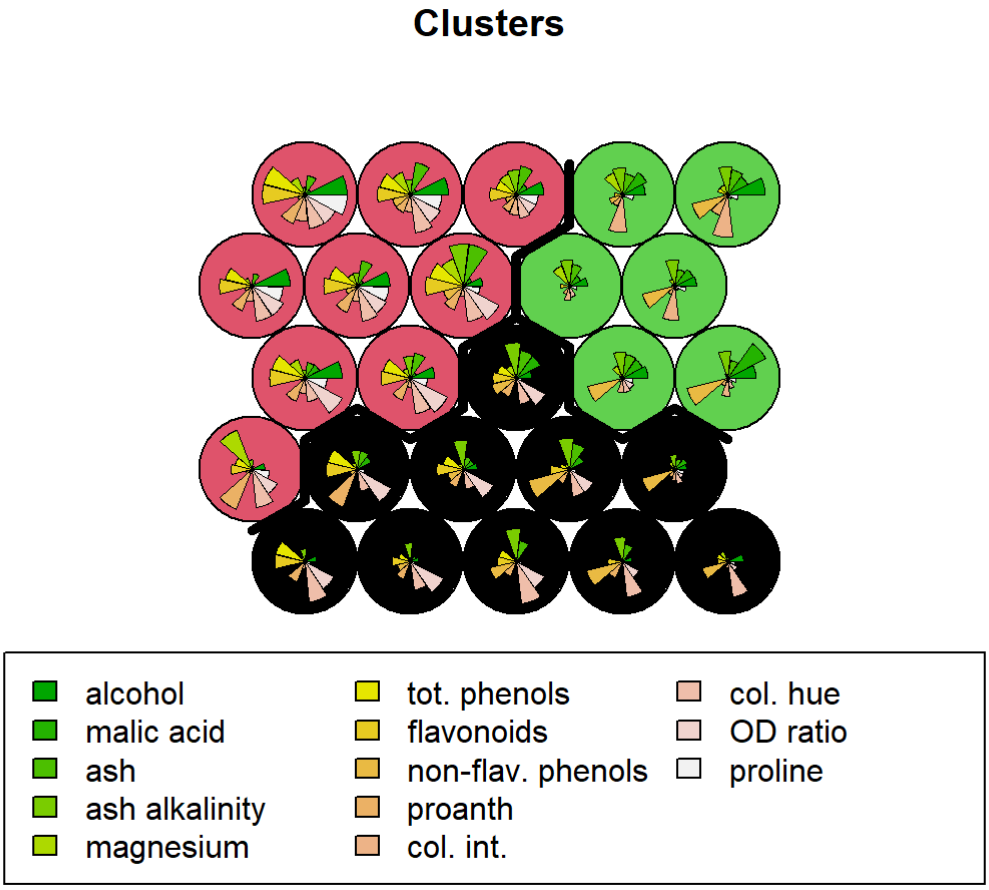
Exercise 5

```
plot(som.wine, type="codes")
```



This graph allows us to observe the values of each attribute in different neurons. However, there is an even more comprehensive graph that allows us to visualize the attributes along with the different groups.

```
som_cluster <- cutree(hclust(dist(getCodes(som.wine))), 3)

plot(som.wine, type="codes", bgcol = som_cluster, main = "Clusters", col=5)
add.cluster.boundaries(som.wine, som_cluster)
```



With this graph, we can observe the composition of each group:

1. The pink cluster exhibits high values for a significant number of attributes, particularly alcohol content, proline, OD ratio, color hue, and in some cases, total phenols or flavonoids.
2. The black cluster stands out for low values in most attributes, but higher values in OD ratio and color hue, and in some cases, non-flavonoid phenols.
3. The green cluster showcases medium values in attributes such as alcohol content, malic acid, ash, ash alkalinity, and magnesium. Additionally, it can be observed that it has high values in color intensity and non-flavonoid phenols.

2.1 SOM as a classifier

Train set

```
datawine<-read.csv("wine.csv",header=FALSE)
colnames(datawine)<-c("Cultivars","Alcohol","Malic_acid","Ash","Alcalinity_of_ash",
                      "Magnesium","Total_phenols","Flavanoids","Nonflavanoid_phenol
s",
                      "Proanthocyanins","Color_intensity","Hue","OD280_OD315","Prolin
e")
head(datawine)
```

	Cultivars <int>	Alcohol <dbl>	Malic_acid <dbl>	A… <dbl>	Alcalinity_of_ash <dbl>	Magnesi… <int>	Total_phenols <dbl>	Flava
1	1	14.23	1.71	2.43	15.6	127	2.80	

	Cultivars	Alcohol	Malic_acid	A…	Alcalinity_of_ash	Magnesi…	Total_phenols	Flava
	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	
2	1	13.20	1.78	2.14	11.2	100	2.65	
3	1	13.16	2.36	2.67	18.6	101	2.80	
4	1	14.37	1.95	2.50	16.8	113	3.85	
5	1	13.24	2.59	2.87	21.0	118	2.80	
6	1	14.20	1.76	2.45	15.2	112	3.27	

6 rows | 1-9 of 15 columns

```
train.wine = sample(nrow(datawine), 125)
Xtrain.wine = scale(datawine[train.wine, ])
Xtest.wine = scale(datawine[-train.wine, ],
center = attr(Xtrain.wine, "scaled:center"),
scale = attr(Xtrain.wine, "scaled:scale"))
vintages<-datawine[,1]
trainingdata = list(measurements = Xtrain.wine, vintages = vintages[train.wine])
testdata = list(measurements = Xtest.wine, vintages = vintages[-train.wine])

trainingdata$vintages
```

```
##   [1] 1 3 2 3 1 3 2 2 1 2 2 1 1 2 2 2 3 1 3 1 2 3 2 3 3 2 1 1 2 2 3 2 1 1 1 1 2
##  [38] 3 1 1 2 1 3 1 1 2 2 1 3 2 3 1 1 2 2 2 2 1 2 3 1 3 2 3 3 3 1 2 3 1 3 3 2 1
##  [75] 1 1 3 2 2 1 2 3 1 1 2 2 2 1 2 2 1 3 3 1 2 2 3 1 2 2 2 3 1 3 3 3 2 2 2 2 2
## [112] 2 2 2 3 3 3 3 1 2 2 2 2 3 2
```

```
testdata$vintages
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3
```

Exercise 1: Classification and error rates

```
som.wine <- supersom(data = trainingdata , grid = somgrid(5,5, "hexagonal"))
```

```
pred <- predict(som.wine, newdata = testdata)

tb <- table(pred$predictions$vintages, testdata$vintages)
tb
```

```
##
##      1  2  3
##   1 22  0  0
##   2  0 18  0
##   3  0  0 13
```

```
#Error rate
den_err <- (sum(sum(tb-diag(diag(tb)))) + length(pred$predictions$vintages) - sum(dia
g(tb)))
num_err <- length(pred$predictions$vintages)
error_rate <- den_err / num_err
error_rate
```

```
## [1] 0
```

We can observe that this self-organizing map (SOM) exhibits great accuracy, being able to correctly classify all instances in the validation set.

3. SOM for studying microbial comunities

```
commdata<-read.csv("community_structure.csv")
head(commdata)
```

	X <chr>	X2 <dbl>	X3 <dbl>	X16 <dbl>	X26 <dbl>	X52 <dbl>	
1	2009-11-03	0.0027418703	4.154349e-05	5.539132e-05	0.000000e+00	0.0005816088	2.0
2	2009-11-09	0.0013486449	0.000000e+00	8.990966e-05	0.000000e+00	0.0003896085	2.2
3	2009-11-16	0.0016629515	0.000000e+00	6.159080e-05	0.000000e+00	0.0004619310	2.3
4	2009-11-18	0.0005869034	2.445431e-05	0.000000e+00	3.260575e-05	0.0004564804	7.3
5	2009-11-23	0.0000000000	4.781807e-05	0.000000e+00	0.000000e+00	0.0001912723	0.0
6	2009-11-30	0.0000000000	0.000000e+00	0.000000e+00	0.000000e+00	0.0001114909	0.0

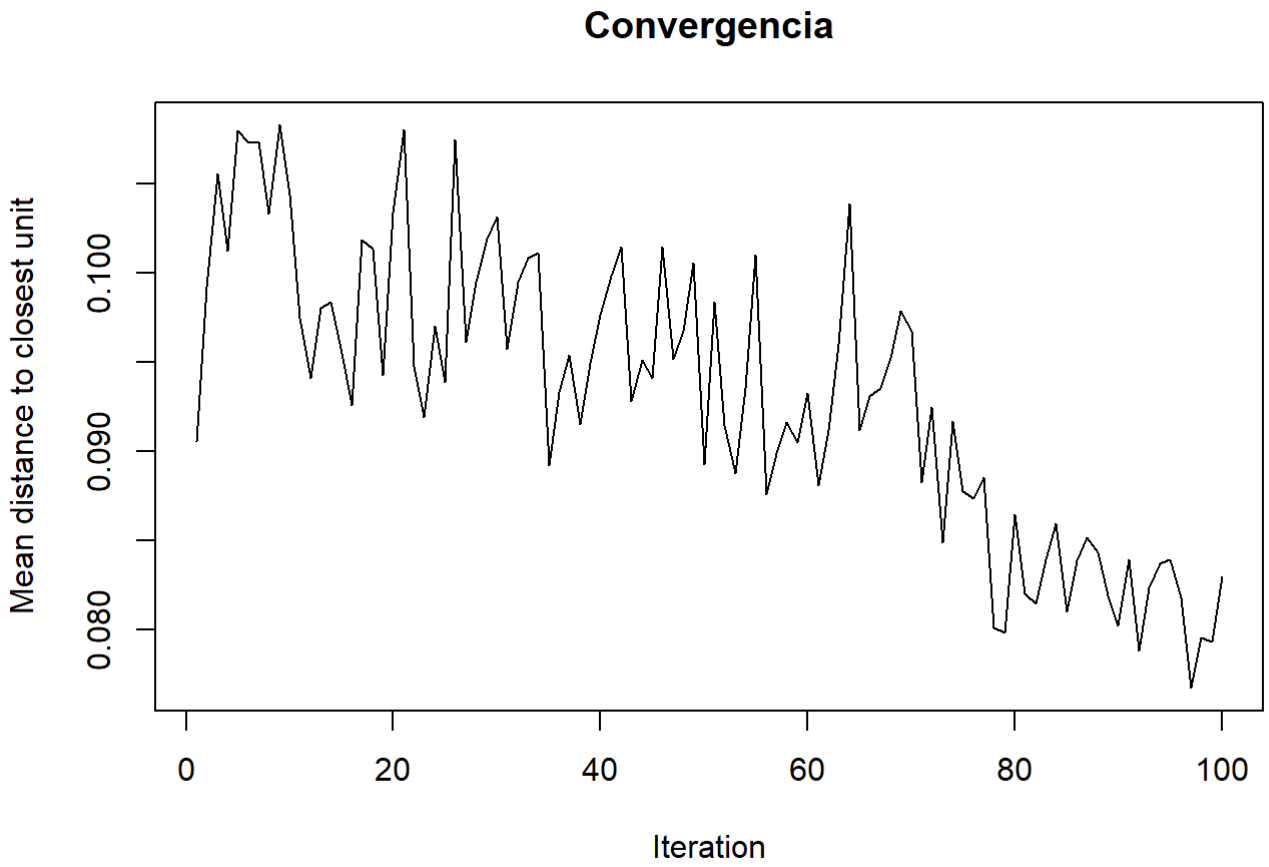
6 rows | 1-8 of 1675 columns

```
Scomm<-scale(commdata[-1])
```

Simple network 5x5

Firtsly, we will try a 5 x 5 mesh with hexagonal conformation

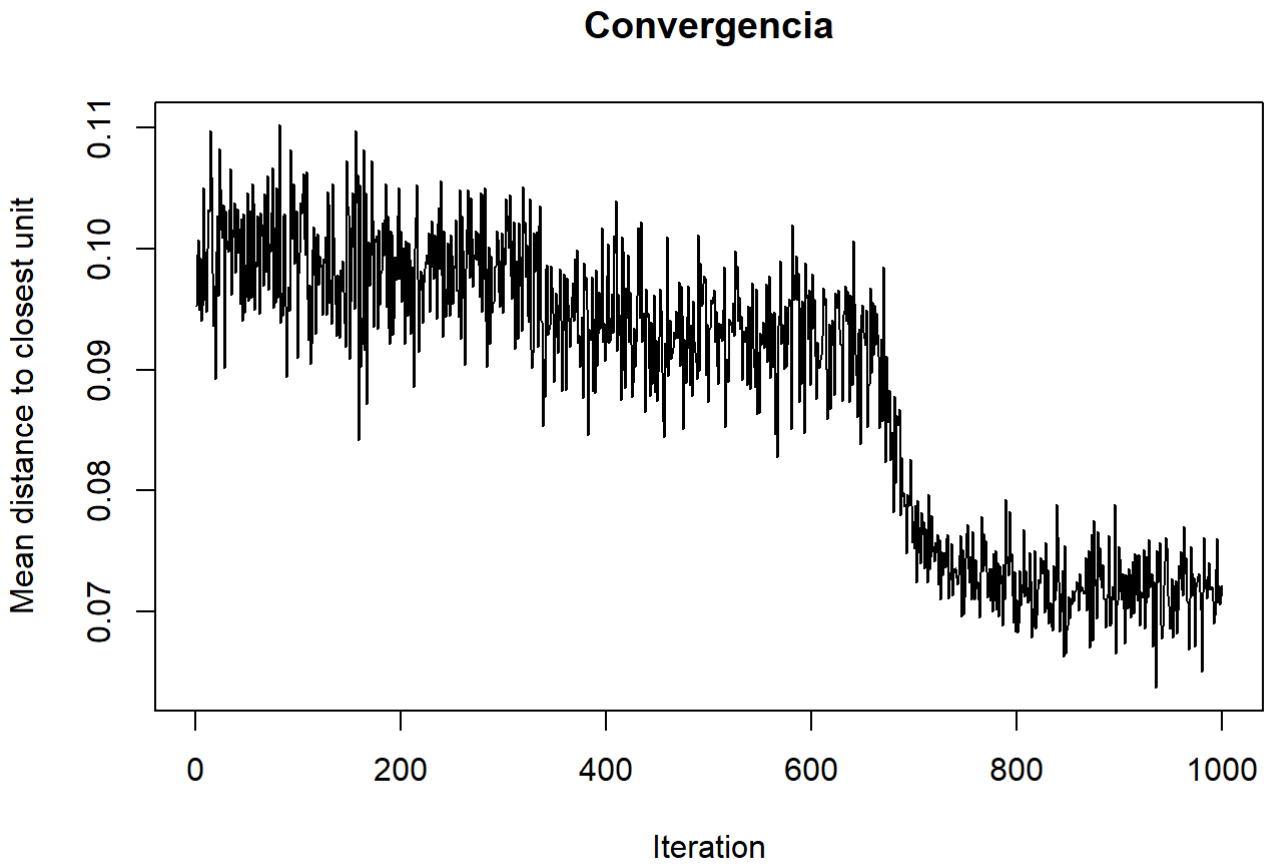
```
som.comm <- supersom(data = as.matrix(Scomm) , grid = somgrid(5,5, "hexagonal"))

plot(som.comm, type = "changes", main = "Convergencia")
```



We can observe that with just 100 iterations, convergence in the distances to the nearest unit begins to occur. However, there may still be room for improvement and further reduction in the distances between units.

Increasing iteration number

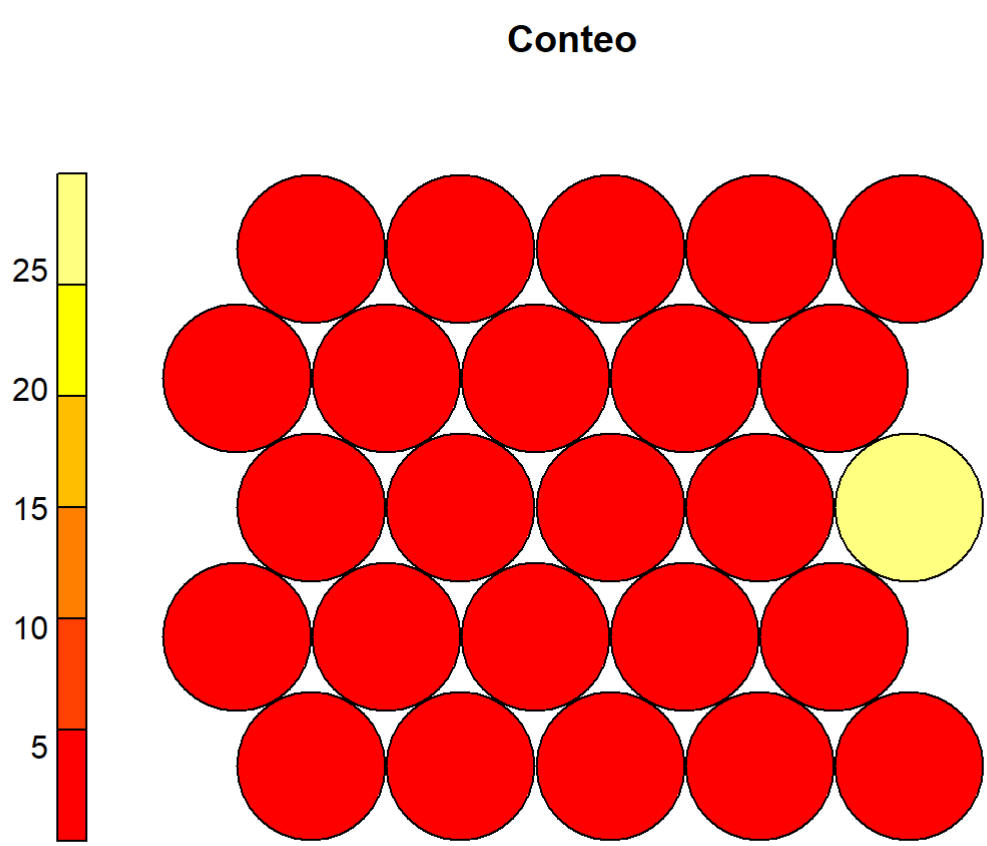
```
som.comm <- supersom(data = as.matrix(Scomm) , grid = somgrid(5,5, "hexagonal"), rlen
= 1000)

plot(som.comm, type = "changes", main = "Convergencia")
```



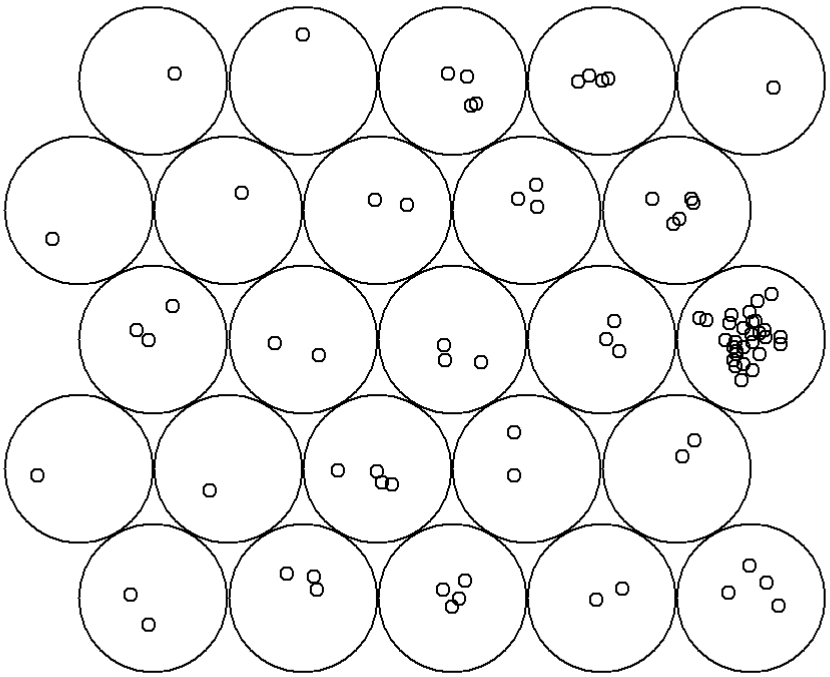
We can observe that by increasing the number of iterations to 1000, convergence is achieved around 700 iterations. Therefore, this increase in the number of iterations allows us to improve the network.

If we examine the various available representations of the grid, we can study how the instances are distributed.

```
plot(som.comm, type="count", main = "Conteo")
```

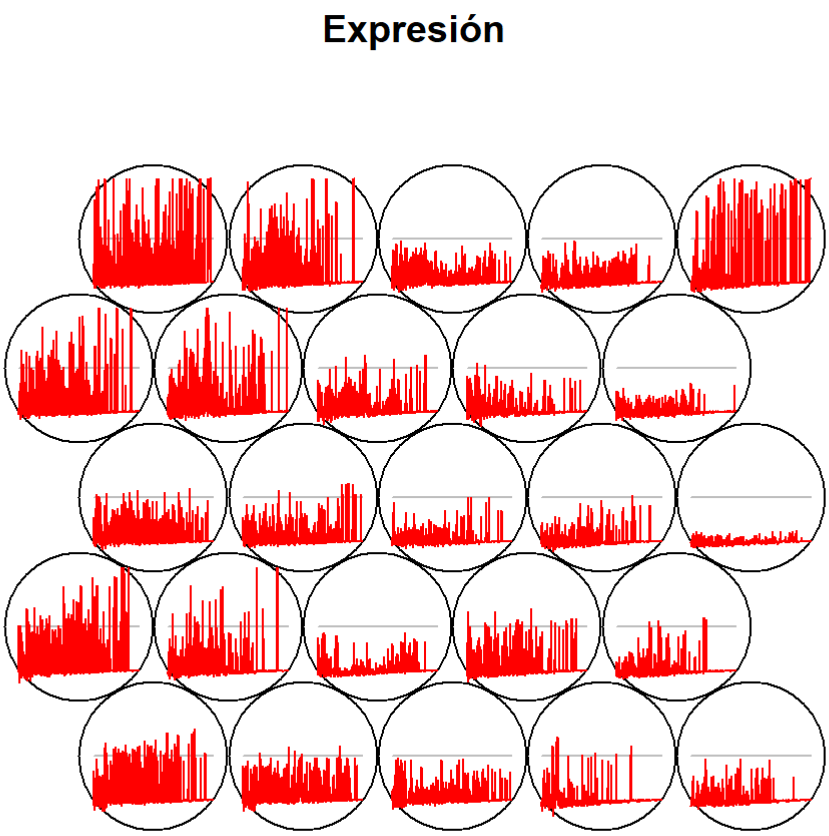



```
plot(som.comm, type="mapping", main = "Mapeado")
```



Based on these two graphs, we can observe that the majority of instances are clustered in a single neuron, while the remaining neurons contain a much smaller number of instances. However, we can also observe that in regions near the neuron with the highest number of instances, there are certain clusters of instances. This suggests that we could form groups around these clusters for our analysis.

```
plot(som.comm, type="codes", main = "Expresión")
```

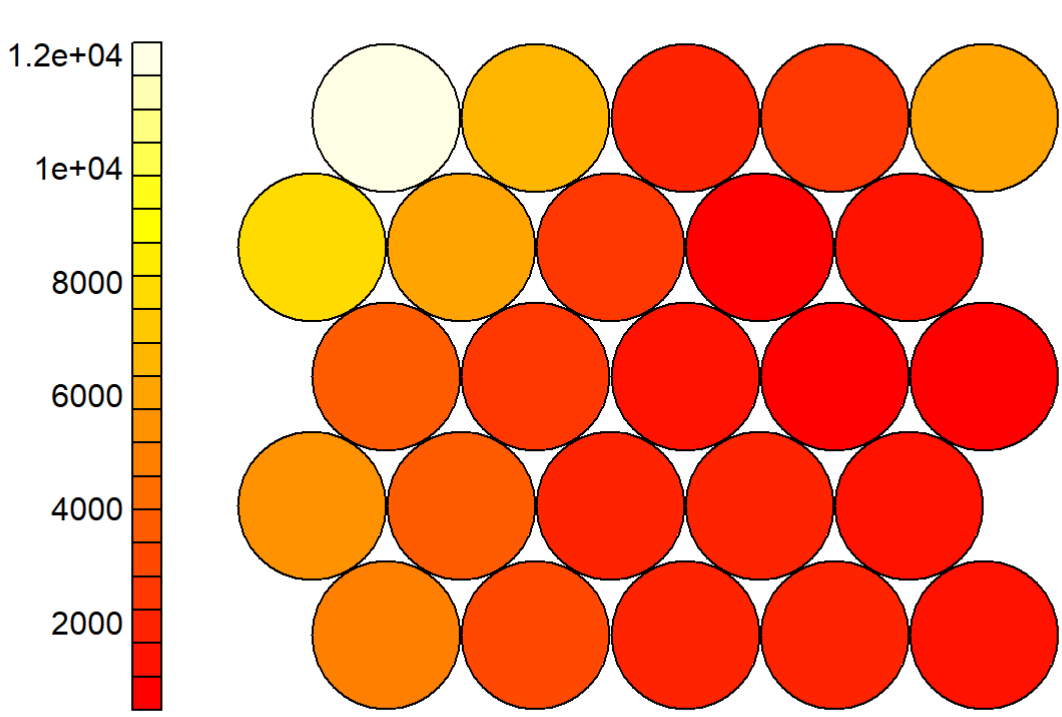


With this graph, we can observe the expression within each neuron of the grid. We can see that the neuron with the highest number of instances exhibits reduced expression in all genes.

Additionally, we can distinguish a region on the left side with elevated expression.

In general, we can observe a gradient of expression throughout the entire grid.

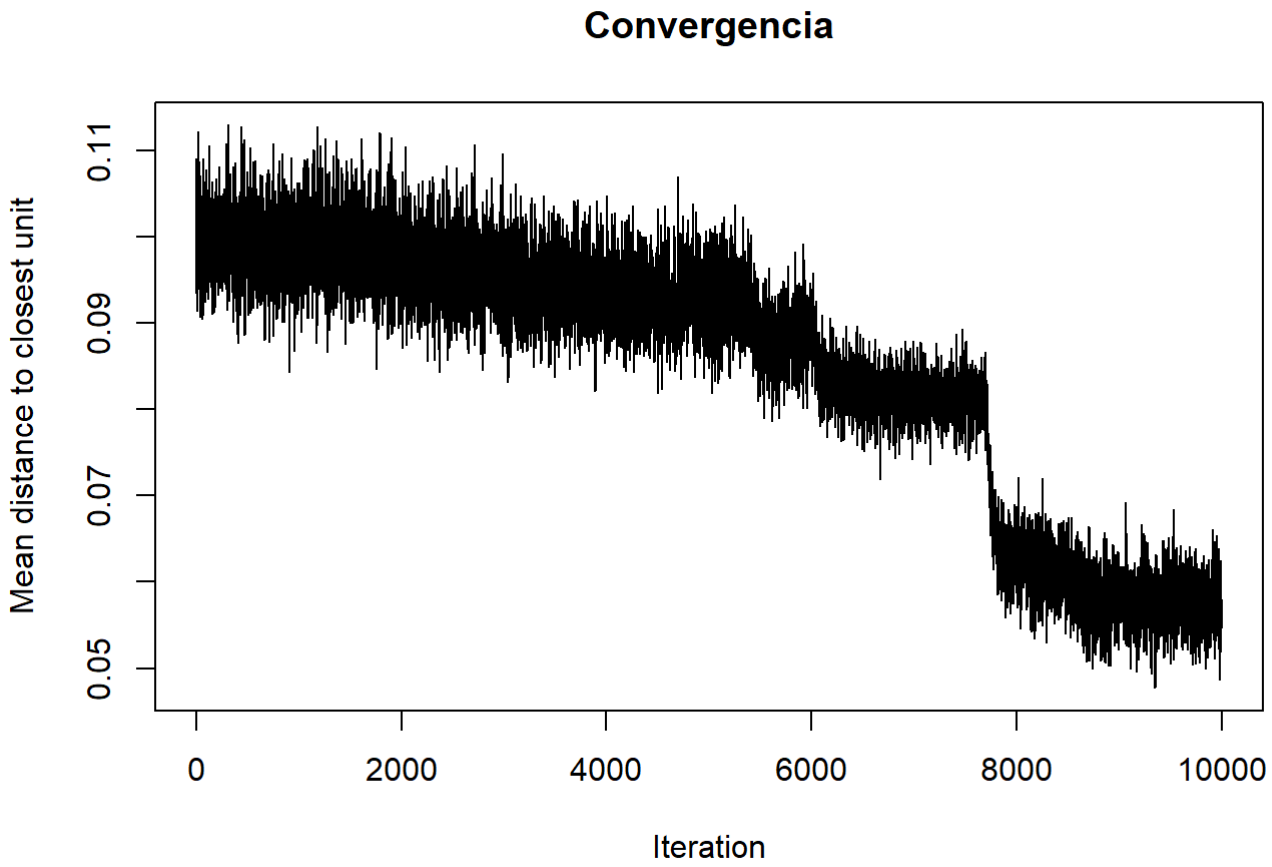
```
plot(som.comm, type="dist.neighbours", main = "Distancias")
```



By adding this graph to the previous ones, we can observe a separation between the neurons in regions of higher expression. This is logical since these regions exhibit the greatest deviation from the average expression levels

Increase neuron number

```
som.comm <- supersom(data = as.matrix(Scomm) , grid = somgrid(7,7, "hexagonal"), rlen
= 10000)

plot(som.comm, type = "changes", main = "Convergencia")
```

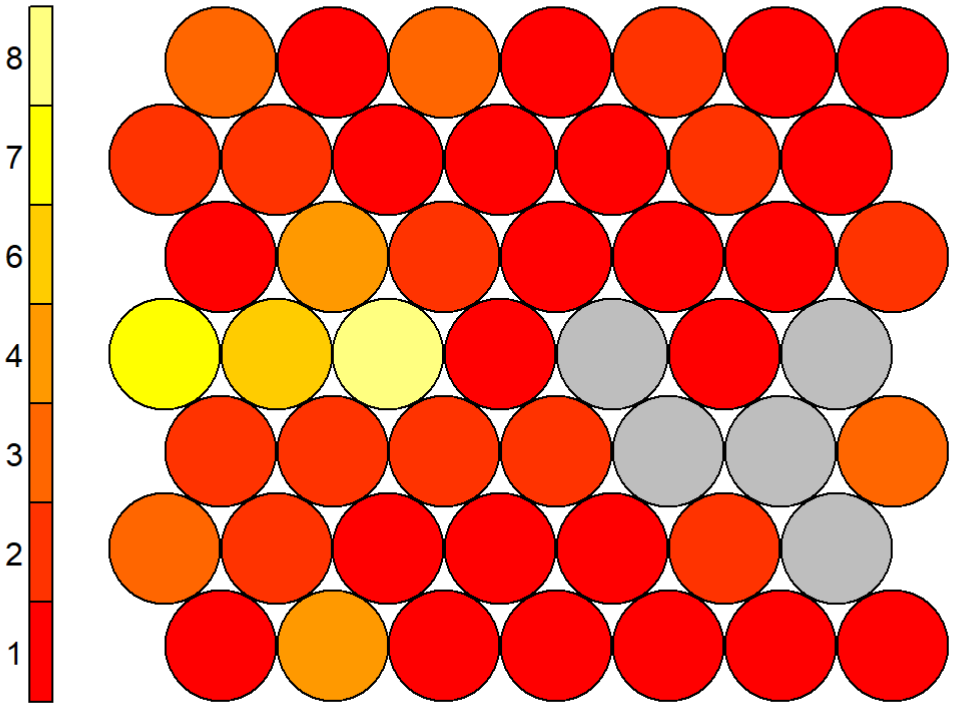


In this case, we increased the number of neurons to a 7x7 grid and increased the number of iterations, surpassing a significant step around 8000 iterations.

If we examine the various available representations of the grid, we can study how the instances are distributed.

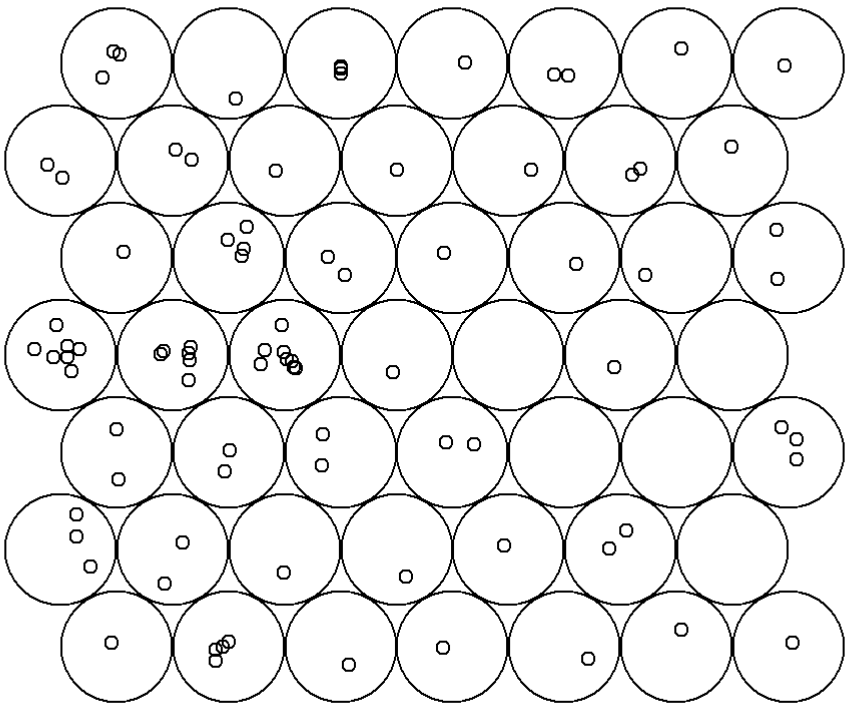
```
plot(som.comm, type="count", main = "Conteo")
```

Conteo




```
plot(som.comm, type="mapping", main = "Mapeado")
```

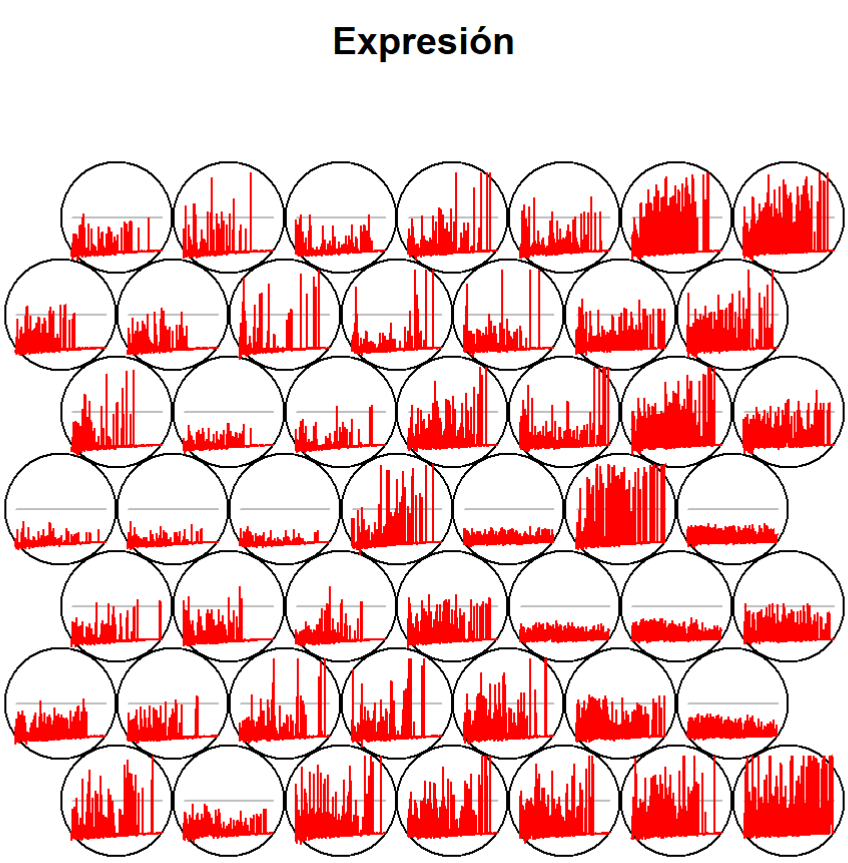
Mapeado



Based on these two graphs, we can once again observe that the majority of instances are clustered in a small group of neurons on the left side of the grid, while the remaining neurons contain a much smaller number of instances, with some neurons being left empty.

This suggests that such a high number of neurons may not be optimal for the number of instances we have, as we end up with unused neurons, partially due to the overcrowding of instances in just three or four neurons of the network.

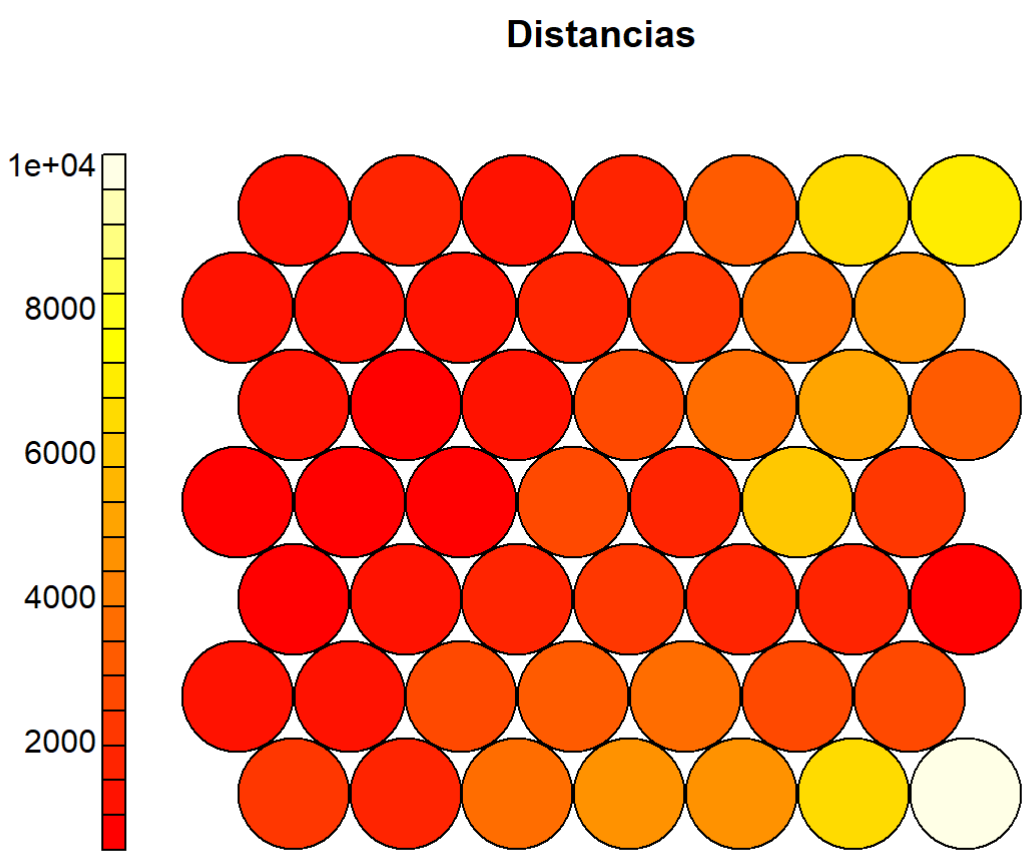
```
plot(som.comm, type="codes", main = "Expresión")
```



We can observe that the neurons with the highest number of instances once again exhibit reduced expression in all genes.

Additionally, in the corners of the right region, we find the points with the highest expression.

```
plot(som.comm, type="dist.neighbours", main = "Distancias")
```



By adding this graph to the previous ones, we can observe a separation between the neurons in the regions of higher expression that we described earlier, similar to the previous case.

Conclusions

Based on these results, we can conclude that the majority of the collected samples exhibit low expression in all genes, while there are some exceptions where expression is elevated, either in certain genes or across all genes.

These instances with higher expression are the ones that deviate the most from the average of the instances, and from them, we could form different expression groups, always keeping the large group with low expression that comprises the majority of the instances.