

1 Introduction

Representation of data using a Kohonen map, followed by a cluster analysis. R software (Kohonen package) and Tanagra (Kohonen-Som component).

This tutorial complements the course material concerning the Kohonen map or Self-organizing map ([SOM 1], June 2017). In a first time, we try to highlight two important aspects of the approach: its ability to summarize the available information in a two-dimensional space; Its combination with a cluster analysis method for associating the topological representation (and the reading that one can do) to the interpretation of the groups obtained from the clustering algorithm. We use the R software and the “Kohonen” package (Wehrens et Buydens, 2007). In a second time, we carry out a comparative study of the quality of the partitioning with the one obtained with the K-means algorithm. We use an external evaluation i.e. we compare the clustering results with pre-established classes. This procedure is often used in research to evaluate the performance of clustering methods. It takes on its meaning when it is applied to artificial data where the true class membership is known. We use the K-Means and Kohonen-Som components of Tanagra.

This tutorial is based on the Shane Lynn's article on the R-bloggers website (Lynn, 2014). I completed it by introducing the intermediate calculations to better understand the meaning of the charts, and by conducting the comparative study.

2 Kohonen map with R (“Kohonen” package)

2.1 Dataset

We use the famous WAVEFORM dataset (Breiman and al., 1984), which is available on the UCI repository¹. We have 21 descriptors. Two modifications are introduced in this part of the tutorial: because we are in an unsupervised learning task, we have remove the class attribute; we drawn a random sample of 1500 cases from the 5000 available instances. This dataset is especially interesting because (1) we know the true number of clusters, (2) we know also the relevant variables (the first and last variables do not play a role in the discrimination between the classes).

¹ [https://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+\(Version+1\)](https://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+(Version+1))

2.2 Data importation and preparation

First, we load and inspect the dataset.

```
#modifying the default directory
setwd("../votre dossier de travail...")

#loading the data file, tab-delimited text file
D <- read.table("waveform_som_1.txt",sep="\t",dec=".",header=T)

#calculating and displaying the descriptive statistics
print(summary(D))
```

We check above all that there are no anomalies in the dataset.

v1		v2		v3		v4			
Min.	:-2.930000	Min.	:-3.1000	Min.	:-4.0500	Min.	:-2.6100		
1st Qu.	:-0.670000	1st Qu.	:-0.4300	1st Qu.	:-0.1425	1st Qu.	:-0.0925		
Median	: 0.020000	Median	: 0.2800	Median	: 0.6800	Median	: 0.9100		
Mean	: 0.004287	Mean	: 0.2921	Mean	: 0.6648	Mean	: 0.9548		
3rd Qu.	: 0.710000	3rd Qu.	: 1.0100	3rd Qu.	: 1.4400	3rd Qu.	: 1.9600		
Max.	: 3.310000	Max.	: 3.8800	Max.	: 4.7200	Max.	: 5.7500		
v5		v6		v7		v8		v9	
Min.	:-2.850	Min.	:-2.7600	Min.	:-2.200	Min.	:-1.910	Min.	:-2.370
1st Qu.	: 0.020	1st Qu.	: 0.4975	1st Qu.	: 1.030	1st Qu.	: 1.360	1st Qu.	: 1.367
Median	: 1.110	Median	: 1.7800	Median	: 2.470	Median	: 2.735	Median	: 2.810
Mean	: 1.309	Mean	: 1.9371	Mean	: 2.622	Mean	: 2.632	Mean	: 2.639
3rd Qu.	: 2.560	3rd Qu.	: 3.3025	3rd Qu.	: 4.223	3rd Qu.	: 3.920	3rd Qu.	: 3.870
Max.	: 6.110	Max.	: 6.7500	Max.	: 8.420	Max.	: 6.870	Max.	: 6.550
v10		v11		v12		v13		v14	
Min.	:-1.790	Min.	:-1.480	Min.	:-1.69	Min.	:-2.610	Min.	:-1.970
1st Qu.	: 1.887	1st Qu.	: 2.058	1st Qu.	: 1.97	1st Qu.	: 1.570	1st Qu.	: 1.380
Median	: 3.050	Median	: 3.175	Median	: 3.01	Median	: 2.955	Median	: 2.730
Mean	: 3.000	Mean	: 3.362	Mean	: 3.05	Mean	: 2.743	Mean	: 2.658
3rd Qu.	: 4.082	3rd Qu.	: 4.600	3rd Qu.	: 4.21	3rd Qu.	: 3.970	3rd Qu.	: 3.953
Max.	: 6.840	Max.	: 9.060	Max.	: 7.32	Max.	: 7.040	Max.	: 7.750
v15		v16		v17		v18		v19	
Min.	:-2.290	Min.	:-2.480	Min.	:-2.880	Min.	:-4.080	Min.	:-3.5000
1st Qu.	: 1.100	1st Qu.	: 0.610	1st Qu.	: 0.020	1st Qu.	: 0.000	1st Qu.	:-0.1700
Median	: 2.420	Median	: 1.825	Median	: 1.095	Median	: 0.955	Median	: 0.6150
Mean	: 2.659	Mean	: 2.002	Mean	: 1.308	Mean	: 1.001	Mean	: 0.6552
3rd Qu.	: 4.250	3rd Qu.	: 3.330	3rd Qu.	: 2.553	3rd Qu.	: 1.950	3rd Qu.	: 1.4300
Max.	: 8.400	Max.	: 7.090	Max.	: 6.610	Max.	: 5.110	Max.	: 5.2800
v20		v21							
Min.	:-3.570	Min.	:-3.45000						
1st Qu.	:-0.300	1st Qu.	:-0.68250						
Median	: 0.345	Median	:-0.06500						
Mean	: 0.368	Mean	:-0.01835						
3rd Qu.	: 1.093	3rd Qu.	: 0.67250						
Max.	: 3.700	Max.	: 4.01000						

We standardize the variables with the `scale()` command.

```
#Z value
Z <- scale(D,center=T,scale=T)
```

Thus, we are ready to perform the analysis.

2.3 Configuration and learning process

Configuration and launching the learning process. We must first install and load the package "Kohonen" before starting the learning process with the `som()` function.

```
#kohonen library
library(kohonen)

#SOM
set.seed(100)
carte <- som(Z,grid=somgrid(15,10,"hexagonal"))
```

We asked a hexagonal grid of size 15 x 10. For this kind of grid, a circular neighborhood shape is used (Figure 1).

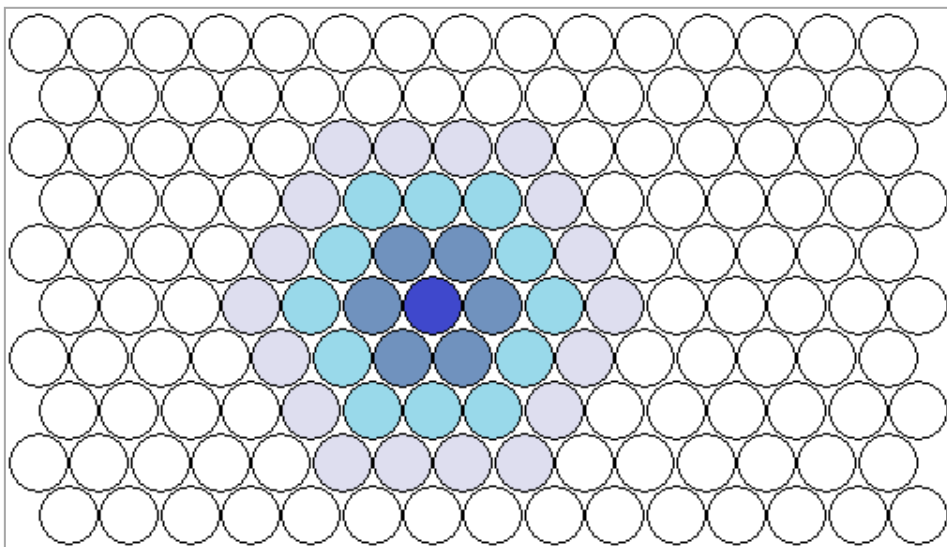


Figure 1 – Hexagonal grid (15 x 10) and circular neighborhood shape

The results are displayed with the `summary()` command...

```
#summary
print(summary(carte))
```

... they are rather short.

```
som map of size 15x10 with a hexagonal topology.
Training data included; dimension is 1500 by 21
Mean distance to the closest unit in the map: 6.441387
```

Structure of the grid. We must access to the properties of the object to obtain details about the structure of the grid.

```
#structure of the grid
print(carte$grid)
```

We have an object of class `somgrid`. It holds various properties.

```

$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
...
[149,] 14.0  8.6602540
[150,] 15.0  8.6602540

$xdim
[1] 15

$ydim
[1] 10

$topo
[1] "hexagonal"

$n.hood
[1] "circular"

attr(,"class")
[1] "somgrid"

```

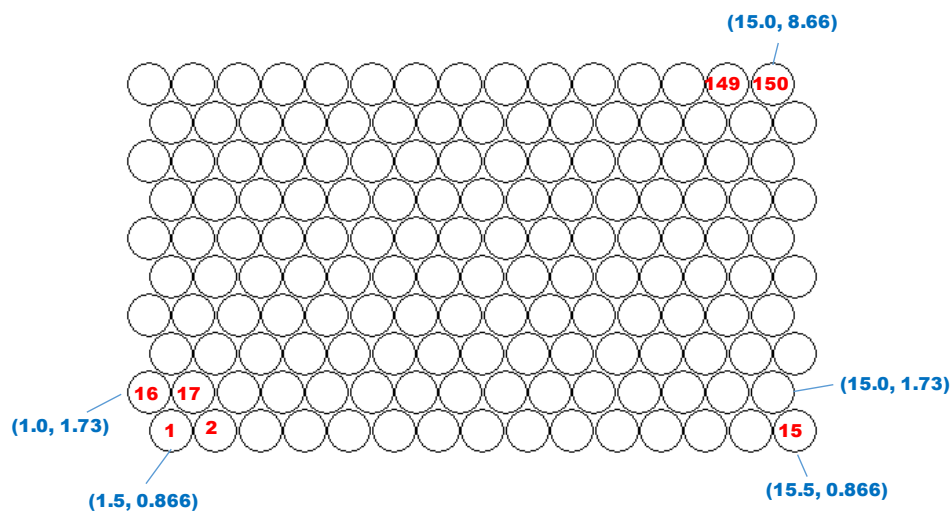


Figure 2 - Number (in red) and coordinates (in blue) of the cells into the grid

The property `$pts` draws our attention. The cells are numbered from 1 to 150 (15 rows x 10 columns). A row (x) and column (y) coordinates are associated to each cell (Figure 2). It is therefore possible to calculate descriptive statistics relating to the cells by positioning them into the representation space. Even if it is not very common, one could, for example,

calculate a variable factor map to identify correlations in the topological map. We will see that below (page 10).

2.4 Graphical representation

Graphical representations are one of the attractive aspects of Kohonen maps. We detail some of them in this section.

Progression of the learning process. This graph enables to appreciate the convergence of the algorithm. It shows the evolution of the average distance to the nearest cells in the map. For our dataset (Figure 3), after a strong decreasing, we have not a significant improvement from (nearly) 60 iterations. By default, the procedure requests $RLEN = 100$ iterations. We should increase $RLEN$ if the curve continues to decrease. This is not required here.

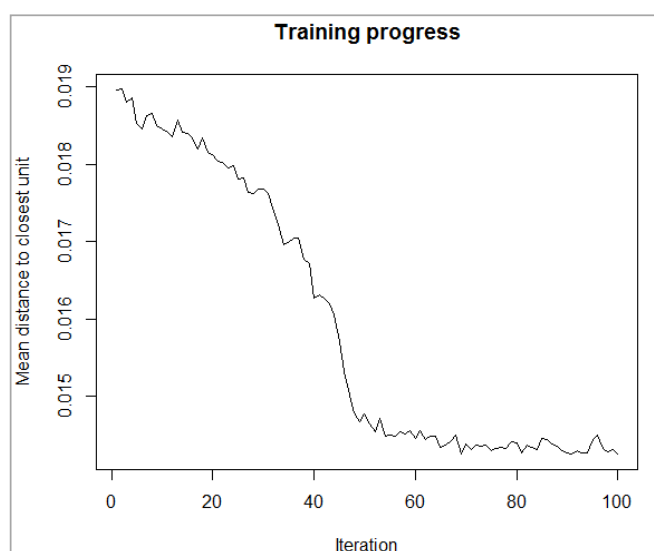


Figure 3 – Training progress

Count plots. The number of instances into the cells are used to identify high-density areas. The "Kohonen" package for R allows you to specify the color function to use. We define the function `degrade.bleu ()` [range of blue].

```
#color range for the cells of the map
degrade.bleu <- function(n){
  return(rgb(0,0.4,1,alpha=seq(0,1,1/n)))
}
```

The function takes "n" as input, it corresponds to the number of colors to be produced. We generate a set of blue colors more or less opaque using `rgb()`. The higher is the number "n", the darker is the color. Then, we plot the map with the `plot ()` command by specifying the right options (Figure 4).

```
#count plot
plot(carte,type="count",palette.name=degrade.bleu)
```

Ideally, the distribution should be homogeneous. The size of the map should be reduced if there are many empty cells. Conversely, we must increase it if areas of very high density appear (Lynn, 2014).

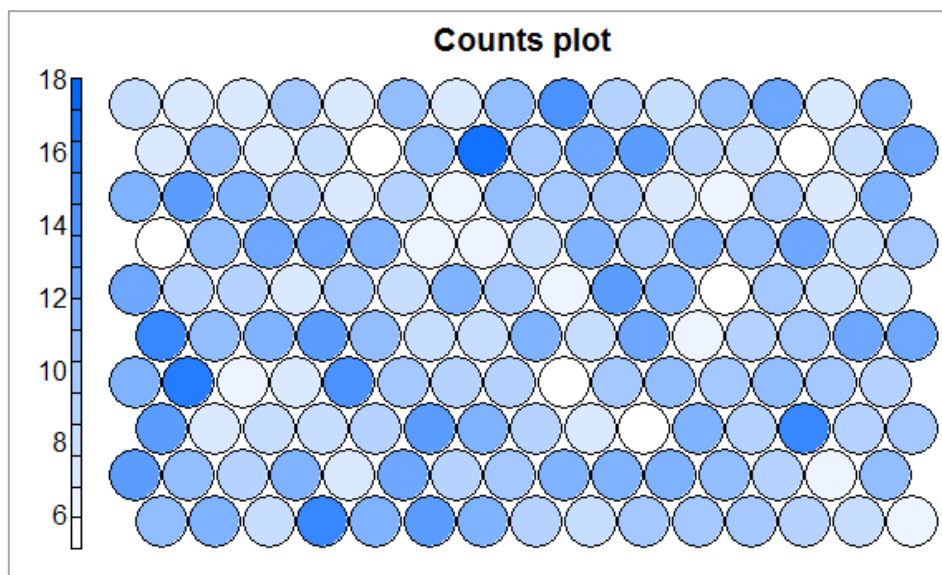


Figure 4 – Counts plot

Details of the results can be accessed with the properties of the object `somgrid`. The property « `$unit_classif` » points the order number of the cells to which is assigned the individuals.

```
# cell membership for each individual
print(carte$unit.classif)
```

The first instance is assigned to the cell n° 83, the second one to the n° 6, ..., the last one to the n°74.

```
[1] 83 6 26 53 20 86 99 120 52 3 146 82 81 120 68 145 112 21 109
21 139 148 82 36 123 89
[27] 24 46 52 57 1 127 99 114 148 16 150 41 21 104 102 126 11 59 46
102 64 130 64 77 118 25
...
[1457] 58 78 107 26 7 144 47 57 127 43 83 141 78 19 102 62 7 76 126
7 58 69 125 76 43 6
[1483] 12 27 144 102 13 144 98 36 103 101 93 106 137 24 42 146 138 74
```

We can calculate the number of instance in each cell using the `table()` command.

```
# number of instances assigned to each node
nb <- table(carte$unit.classif)
print(nb)
```

There are 11 instances into the cell n°1, 12 into n°2, ..., 12 into n°150.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
11	12	8	16	12	14	12	9	8	10	10	10	9	8	6	14	11	9	12
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
7	13	9	10	12	12	12	11	9	6	11	14	7	8	8	9	14	12	9
39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57
7	5	12	9	16	9	10	12	17	6	7	15	10	9	9	5	10	11	10
58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76
11	10	9	16	11	12	14	11	8	8	12	8	13	6	9	10	13	13	13
77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
9	9	7	10	8	12	10	6	14	12	5	10	8	8	5	11	13	13	12
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114
6	6	8	12	10	12	11	13	8	10	12	14	12	9	7	9	6	11	10
115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133
10	7	6	10	7	12	7	11	7	8	5	11	18	10	13	14	9	8	5
134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150		
8	13	8	7	7	10	7	11	7	11	15	9	8	11	13	7	12		

We can also check if there are empty nodes by counting the number of values into the vector generated by the `table()` command.

```
#check if there are empty nodes
print(length(nb))
```

We have **150** values. All nodes contain at least one observation.

Neighbour distance plot. Called "U-Matrix" (unified distance matrix), it represents a self-organizing map (SOM) where the Euclidean distance between the codebook vectors of neighboring neurons is depicted in a range of colors.

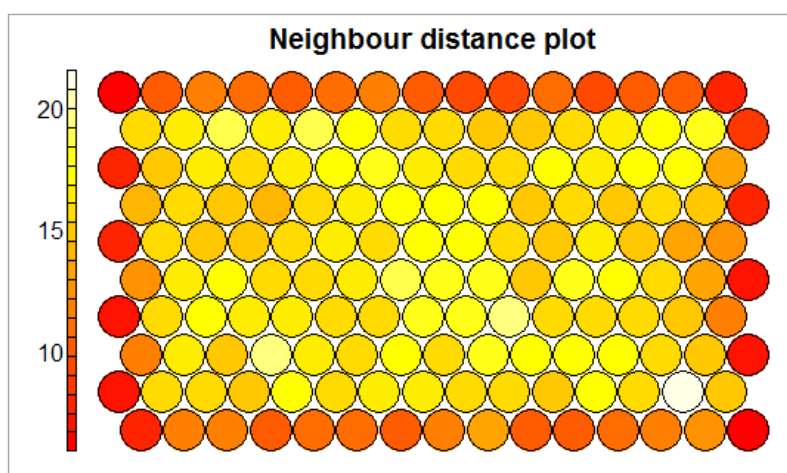


Figure 5 – Neighbour distance plot

The chart is obtained using the following command:

```
#plot distance to neighbours
plot(carte,type="dist.neighbours")
```

According to the package documentation, the nodes that form the same group tend to be close. Border areas are bounded by nodes that are far from each other. In our chart (Figure 5), the nodes which are close to the others are dark-colored. We observe that they are concentrated on the ends of the map. This leaves hope for a good separation of the groups in the typology.

2.5 Graphical representation – Influence of the variables

A section is devoted to this type of chart because it allows to establish the role of variables in the definition of the different areas that comprise the topological map. This is important for the interpretation of the results. This is especially true when we combine the new data representation system with a typology provided by a clustering algorithm (section 2.7).

Codebook vectors. This chart represents the vector of weights in a pie chart for each cell of the map.

```
#codebooks - nodes pattern
plot(carte,type="codes",codeRendering = "segments")
```

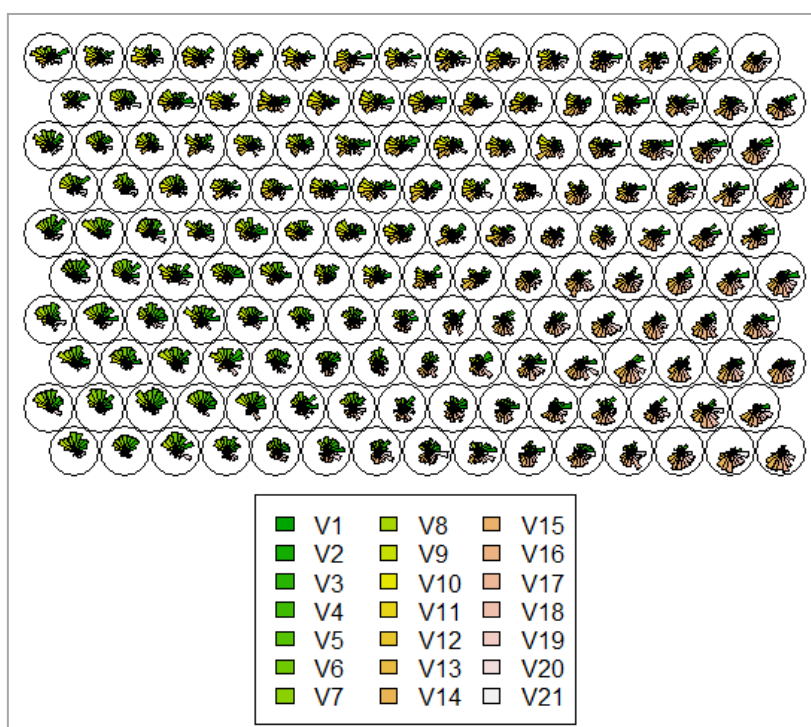


Figure 6 – Codebooks chart

It enables to distinguish at a glance the nature of the different areas of the map regarding the variables (Figure 6). We note that the southwestern part is rather characterized by the high values of the first variables (in green); the north is rather associated with the

intermediate variables (in yellow), the southeastern part would be relative to the last variables (in pink). We detail the codebooks for the two first cells.

```
#codebooks values for the two first cells
print(carte$codes[1:2,])
```

The values of the variables are comparable from one node to another. But also, because we have standardized them, they are also comparable within each node.

```
      v1      v2      v3      v4      v5      v6      v7
[1,] -1.402508  0.7030893  1.028195  1.112326  1.69281  1.979410  1.201450
[2,] -1.317920  0.2778695  1.100380  1.466021  1.23953  1.316825  1.205971
      v8      v9      v10      v11      v12      v13
[1,]  1.1301033  1.0421593 -0.04541570 -1.156651 -1.7364859 -1.899917
[2,]  0.8252693  0.3755839 -0.03180923 -0.887046 -0.8118423 -1.406320
      v14      v15      v16      v17      v18      v19
[1,] -1.9378001 -0.9810732 -1.031163 -0.8476767 -0.8018206 -0.01620569
[2,] -0.8989741 -0.9614553 -0.966065 -0.6904285 -0.7548960 -0.53788446
      v20      v21
[1,] -0.7937134  0.2506806
[2,] -0.7006180 -2.0250856
```

It seems that the southwest part of the map - cells n°1 and 2 by starting from the bottom left (Figure 2) - are characterized by high values for V₅, V₆ and V₇; and low values for V₁₂, V₁₃ and V₁₄.

Heatmaps. Even it is attractive, we realize that the "codebook" chart is difficult to read when the number of nodes and variables increases. Rather than making a single chart for all the variables, we can make a graph for each variable, trying to highlight the contrasts between the high and low value areas. This univariate description is easier to understand. We use the values of the "codebooks" to build the graphs that we set together into one frame. The color scheme used combines red colors (respectively blue) with high values (low values). We use the `coolBlueHotRed()` function for that (Wehrens, 2015).

```
#colors function for the charts
coolBlueHotRed <- function(n, alpha = 1) {
  rainbow(n, end=4/6, alpha=alpha)[n:1]
}

#plotting the heatmap for each variable
par(mfrow=c(6,4))
for (j in 1:ncol(D)){
  plot(carte,type="property",property=carte$codes[,j],palette.name=coolBlueHotRed,main=colnames(D)[j],cex=0.5)
}
par(mfrow=c(1,1))
```

The influence areas appear clearly (Figure 7).

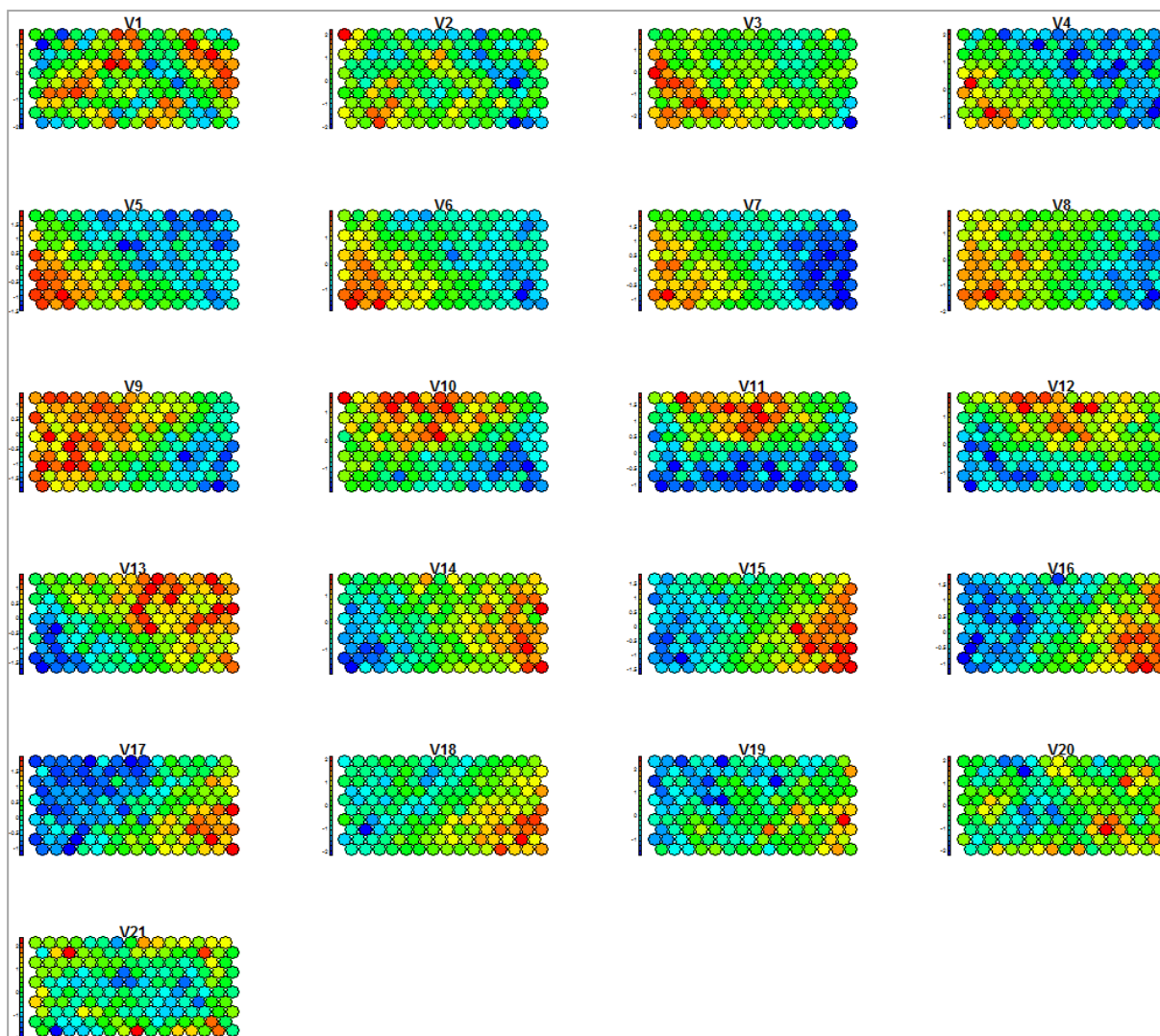


Figure 7 – Areas for high values (red) and low values (blue) for each variable

We distinguish approximately 3 areas which will be confirmed by the cluster analysis below (section 2.7): southwest, characterized by high values for V3 to V8; north for V10 to V12; southeast for V14 to V18.

Variable factor map. This kind of chart is very popular in principal component analysis (PCA), it is not usual in the Kohonen map context. However, we have all resources for achieving the calculations: we have the coordinates (x, y) of each cell (Figure 2); their weights (number of instances, Figure 4), and the value for each variable from the codebook.

We calculate the weighted correlation for each column of the codebook:

```
#correlation based on the coordinates (x,y) of the map
#v: variable (one column of the codebook), w: weight, grille: kohonen map
weighted.correlation <- function(v,w,grille){
  x <- grille$grid$pts[,"x"]
```

```

y <- grille$grid$pts[,"y"]
mx <- weighted.mean(x,w)
my <- weighted.mean(y,w)
mv <- weighted.mean(v,w)
numx <- sum(w*(x-mx)*(v-mv))
denomx <- sqrt(sum(w*(x-mx)^2))*sqrt(sum(w*(v-mv)^2))
numy <- sum(w*(y-my)*(v-mv))
denomy <- sqrt(sum(w*(y-my)^2))*sqrt(sum(w*(v-mv)^2))
#correlation for the two axes
res <- c(numx/denomx,numy/denomy)
return(res)
}

#correlations for all the columns of the codebook
CORMAP <- apply(carte$codes,2,weighted.correlation,w=nb,grille=carte)
print(CORMAP)

```

We obtain a pair of values for each variable:

	v1	v2	v3	v4	v5	v6	v7
[1,]	0.098092995	-0.3427152	-0.5881500	-0.6869289	-0.7590438	-0.8356554	-0.9120511
[2,]	0.007083712	-0.2025105	-0.2840669	-0.4148091	-0.4816572	-0.3385431	-0.1949636
	v8	v9	v10	v11	v12	v13	v14
[1,]	-0.89006908	-0.8006609	-0.4758664	-0.08393847	0.3694334	0.7546540	0.8776806
[2,]	0.02609795	0.3339969	0.6452938	0.77856603	0.7116477	0.4976912	0.1886124
	v15	v16	v17	v18	v19	v20	v21
[1,]	0.92361028	0.8841228	0.8006326	0.7578706	0.6278790	0.3548972	-0.001883609
[2,]	-0.08297735	-0.1824730	-0.3346339	-0.3135307	-0.3403646	-0.1477782	0.191807627

We insert them into a scatter plot:

```

#graphical representation of the variable factor map
plot(CORMAP[1,],CORMAP[2,],xlim=c(-1,1),ylim=c(-1,1),type="n")
lines(c(-1,1),c(0,0))
lines(c(0,0),c(-1,1))
text(CORMAP[1,],CORMAP[2,],labels=colnames(Z),cex=0.75)
symbols(0,0,circles=1,inches=F,add=T)

```

The chart (Figure 8) is very similar to the variable factor map obtained from the principal component analysis (PCA) on the same dataset. The chart looks like a heart (inverted here). I have always wondered if the authors (Breiman et al., 1984) have done it intentionally by generating the data.

We agree that the variable factor map is first as an indicative tool in this framework. There are no mathematical meanings to the calculations, especially since the topological map is supposed to be able to represent nonlinear patterns. Nevertheless, this kind of synthetic chart is otherwise easier to inspect than codebooks (Figure 6) and heatmaps (Figure 7) when the number of variables and nodes increases. In our example, the conclusions of the variable factor map are consistent with the above representations.

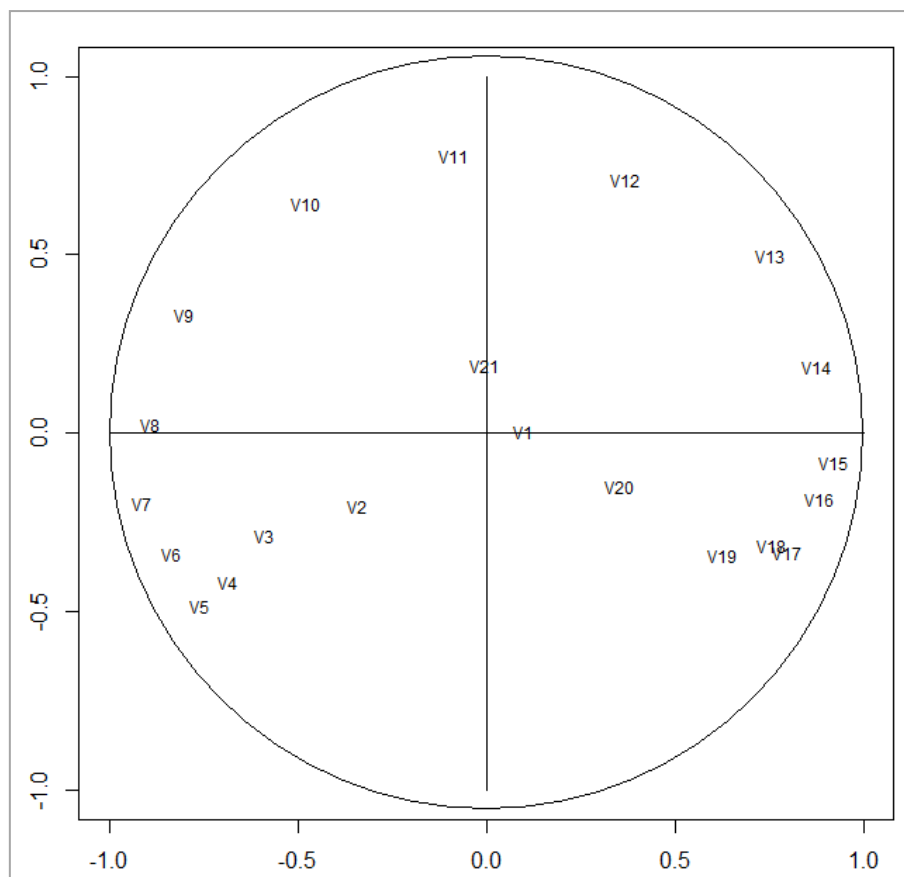


Figure 8 – Variable factor map for the Kohonen network

2.6 Relevance of the variables

Confirming the suggestions of graphical representations with statistical indicators is always a good thing. Strong contrasts highlight the importance of variables in the definition of the different areas into the “heatmaps” charts (Figure 7). We can convey this idea using an indicator such as the variance. Thus, it is possible to rank the influence of the variables.

Two characteristics allow us to do this: the variables are standardized, their influences are directly comparable; the codebooks vectors of the cells correspond to an estimation of the conditional averages, calculating their variance for each variable is equivalent to estimating the between-node variance of the variable, and hence their relevance.

We calculate the weighted variance for each variable (weighted by the number of instances within each cell).

```
#spread of the codebooks
#variance weighed by the cell size
sigma2 <- sqrt(apply(carte$codes,2,function(x,effectif){m<-sum(effectif*(x-
weighted.mean(x,effectif))^2)/(sum(effectif)-1)},effectif=nb))
```

```
# printing according to a decreasing order
print(sort(sigma2,decreasing=T))
```

The relevant variables (because they induce the strongest contrasts between the cells of the map) appear first:

v15	v7	v17	v8	v6	v13	v11	v14
0.8900360	0.8797254	0.8603220	0.8540305	0.8538536	0.8527706	0.8518798	0.8518771
v16	v5	v10	v9	v12	v18	v4	v19
0.8508026	0.8461657	0.8437845	0.8435556	0.8198914	0.8161934	0.8141638	0.8067165
v1	v21	v2	v20	v3			
0.8010057	0.7907470	0.7904047	0.7895682	0.7713664			

Extreme variables (V1 to V3) and (V19 to V21) are the less influential i.e. the conditional averages are homogeneous across the whole map. These results confirm what we found in the different graphical representations seen earlier. But numerical indicators are more convenient for the processing of large datasets.

2.7 Cluster analysis from the map – Two-step clustering

Clustering of nodes. Kohonen map is a kind of cluster analysis, where adjacent cells have similar codebooks. We can start from the preclusters (the cells of the map) provided by the SOM algorithm to perform a cluster analysis. The HAC (hierarchical agglomerative clustering) is often used in this context.

Below, we calculate the pairwise distance between the cells (using the codebooks) [**dist**], then we perform a HAC [**hclust**] using the Ward's method [**method = "ward.D2"**].

```
#distance matrix between the cells
dc <- dist(carte$codes)

#hac - the option "members" is crucial
cah <- hclust(dc,method="ward.D2",members=nb)
plot(cah,hang=-1,labels=F)

#visualizing the 3 clusters into the dendrogram
rect.hclust(cah,k=3)
```

The option "**members**" is crucial. Indeed, the number of instances related to each node is not the same i.e. the nodes have not identical weights. We must take into account it when we calculate the Ward's criterion.

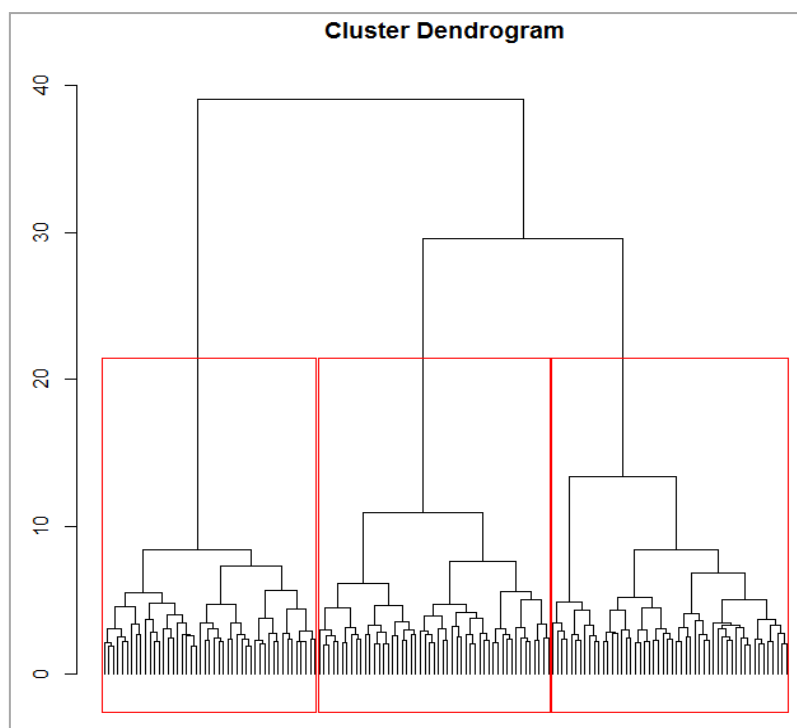


Figure 9 - Dendrogram

A partitioning in 3 clusters seems justified in view of the dendrogram (Figure 9).

We create a cluster membership variable (**cutree**):

```
#cutting in 3 clusters
groupes <- cutree(cah,k=3)
print(groupes)
```

The variable 'groupes' enables to identify the cluster membership of each node of the Kohonen map.

```
[1] 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 2 2
[42] 2 2 2 2 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 3 3 3 2 2 2 2 2 2 1 1 1 1 3 1 3
[83] 3 3 3 2 2 2 2 2 1 1 1 3 3 3 3 3 3 2 2 2 2 2 1 1 3 3 3 3 3 3 3 3 3 3 2 2 2 1 3 3
[124] 3 3 3 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2
```

The first node n°1 belongs to the **cluster 1**, the second one also, ..., the last node of the map (n°150) belongs to the **cluster 2**.

We can visualize the groupings into the map.

```
#visualizing the clusters into the map
plot(carte,type="mapping",bgcol=c("steelblue1","sienna1","yellowgreen")[groupes])
add.cluster.boundaries(carte,clustering=groupes)
```

The three areas identified in the various previous charts are clearly highlighted. And we know how to interpret them now.

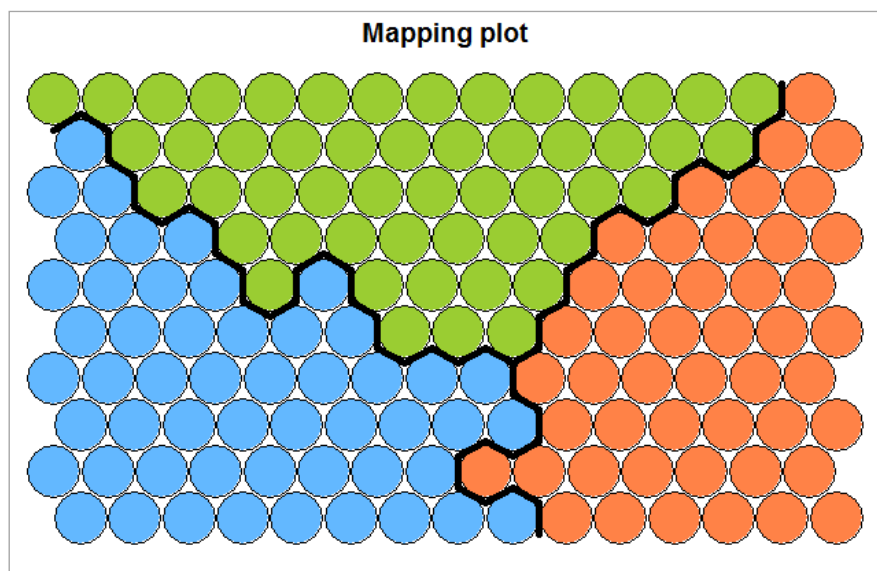


Figure 10 - Representation of the clusters into the map

Note: The adjacent nodes are grouped in the same cluster. This seems normal considering the properties of the map (adjacent nodes have similar codebooks). But anomalies may appear sometimes, mainly because of the constraints inherent in the clustering algorithm.

Cluster membership of the individuals. The cells of the map are associated to the cluster. But, usually, we are interested in the cluster membership of the individuals. We obtain this in a two-step process: we assign first the individuals to a cell, then we identify the cluster associated to this cell.

```
#assign each instance to its cluster
ind.groupe <- groupes[carte$unit.classif]
print(ind.groupe)
```

The individual n°1 is associated to the **cluster 3**, the n°2 to the **cluster 1**, ..., the n°1500 to the **cluster 2**.

```
[1] 3 1 2 1 1 2 3 2 1 1 3 3 1 2 3 3 3 1 3 1 3 3 3 1 3 2 2 1 1 2 1
...
[1477] 2 3 3 1 2 1 2 2 3 2 2 3 3 1 2 2 1 1 3 2 2 3 3 2
```

Visualizing the clusters into the original representation space. Since we know the cluster membership of individuals and the variables explaining the grouping, it is possible to represent them in a three-dimensional space formed from the main variables defining the 3 areas identified previously (sections 2.5 and 2.6), i.e. V7, V11 and V15.

We use the Karline Soetaert's "[plot3D](#)" package.

```
#representation of the individuals and their group membership
#in a 3D representation space
library(plot3D)
points3D(D$V7,D$V11,D$V15,colvar=ind.groupe,col=c("steelblue1","sienna1","yellowgreen"),phi=35,theta=70)
```

We easily distinguish the 3 clusters (Figure 11).

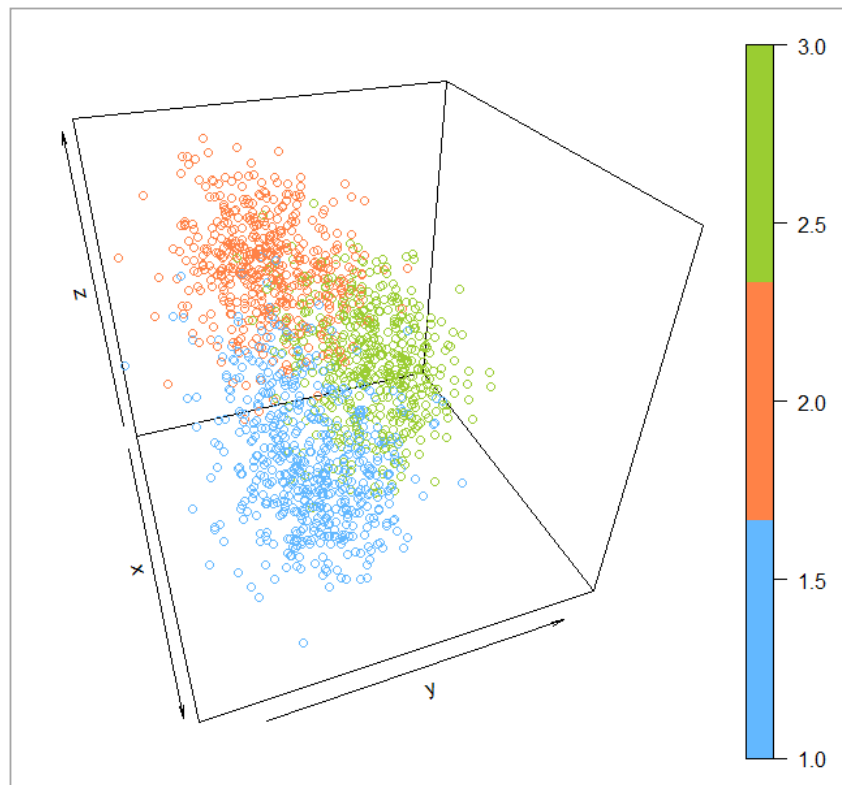


Figure 11 – Clusters in a 3D representation space (V7, V11 and V15)

By means of the “plot3Drgl” package of the same author, we can generate an interactive graphical representation.

```
#animated 3D graphical representation
library(plot3Drgl)
plotrgl(lighting = T)
```

A specific window appears. I admit to having a lot of fun by turning the graphic in all directions and playing with the zoom (Figure 12). As we can see, the possibilities of data representation, and therefore data analysis, are attractive.

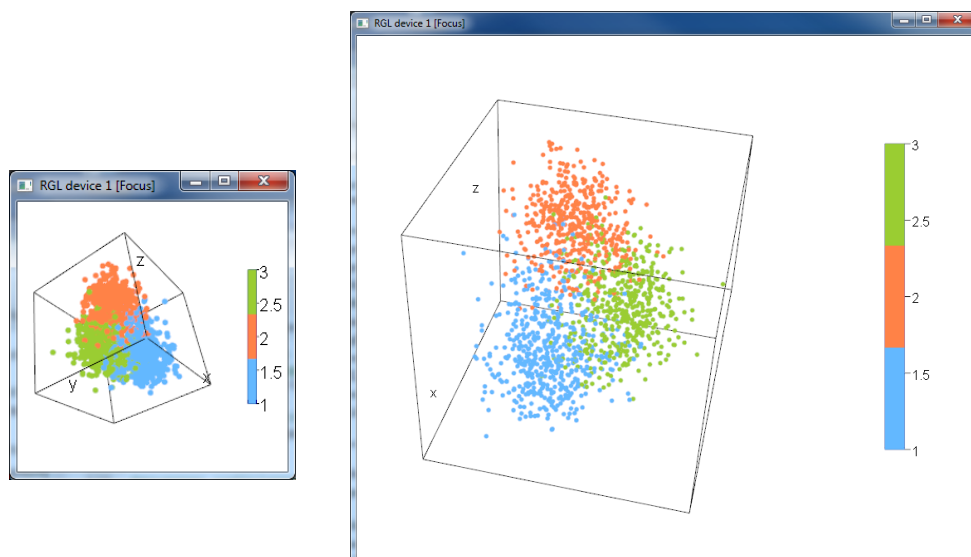


Figure 12- Interactive 3D representation

3 External evaluation - SOM vs. K-MEANS with Tanagra

Is the combined (SOM + HAC) approach is efficient? We chose to compare the performance of (SOM + HAC) with the state-of-the-art K-means to check it. The comparison is based on an external evaluation scheme i.e. the true class membership of the individuals is known, we want to create clusters which are the most consistent to this reference.

We go to the basics in this section. The implementation of the Kohonen maps under Tanagra is described in a previous tutorial (SOM 2, 2009). The reader may refer to it for the construction step by step and the settings of the data mining diagram.

3.1 Dataset

We use the whole 5000 instances of the [WAVEFORM](#) dataset here ([waveform_som_2.xls](#)). The class attribute WAVE (last column) is not used for the construction of the clusters, but is used for their evaluation. Because, the WAVEFORM is an artificial dataset, it is appropriated for this type of evaluation scheme.

3.2 SOM + CAH with Tanagra

We import the dataset², then we define the input variables with the DEFINE STATUS tool (input: V1 to V21). We add the KOHONEN-SOM component into the diagram. We set the size of the map (15 rows and 10 columns) (1), and we standardize the variables (2).

² <http://data-mining-tutorials.blogspot.fr/2010/08/tanagra-add-in-for-office-2007-and.html>

Kohonen's Self Organisation Map parameters

Parameters Results

MAP Structure

Row size : 15

Col size : 10

Distance normalization

None

Variance

Starting learning rate : 0.05

Seed random generator

Random

Standard

OK Cancel Help

TANAGRA 1.4.50 - [Kohonen-SOM 1]

File Diagram Component Window Help

Default title

Dataset (wavefrom_som_2.xls)

Define status 1

Kohonen-SOM 1

Results

MAP Topology

	1	2	3	4	5	6	7	8	9	10
1	36	36	34	49	36	33	36	35	33	46
2	33	40	33	23	41	42	28	36	31	34
3	42	27	28	39	36	23	32	43	34	26
4	32	21	29	33	30	32	24	29	38	32
5	29	42	32	24	33	29	31	37	22	55
6	49	29	44	27	34	26	28	24	39	37
7	42	39	27	33	33	19	25	40	33	43
8	38	38	29	36	34	8	32	21	18	45
9	33	32	33	18	43	25	27	42	37	43
10	32	34	28	36	29	26	27	37	33	44
11	39	33	39	34	26	34	31	38	26	52
12	36	25	24	23	34	22	27	31	32	30
13	33	39	30	25	36	34	25	33	36	43
14	31	32	28	26	36	28	27	37	38	50
15	42	36	32	45	42	42	37	28	43	42

MAP Quality

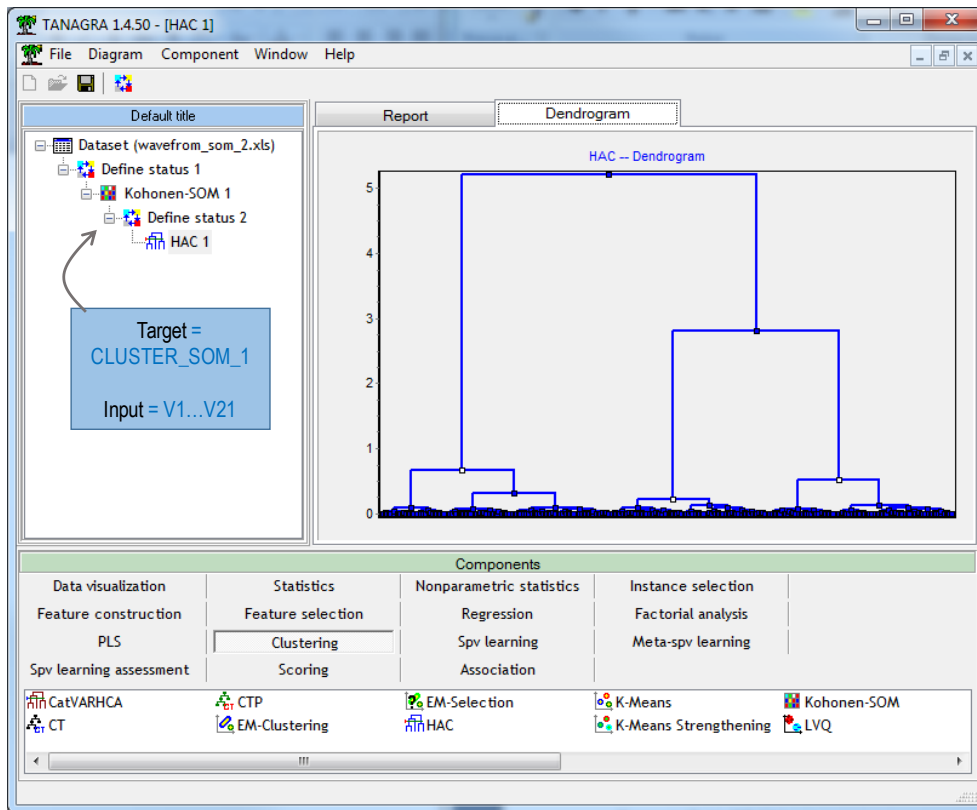
Ratio explained 0,5992

Components

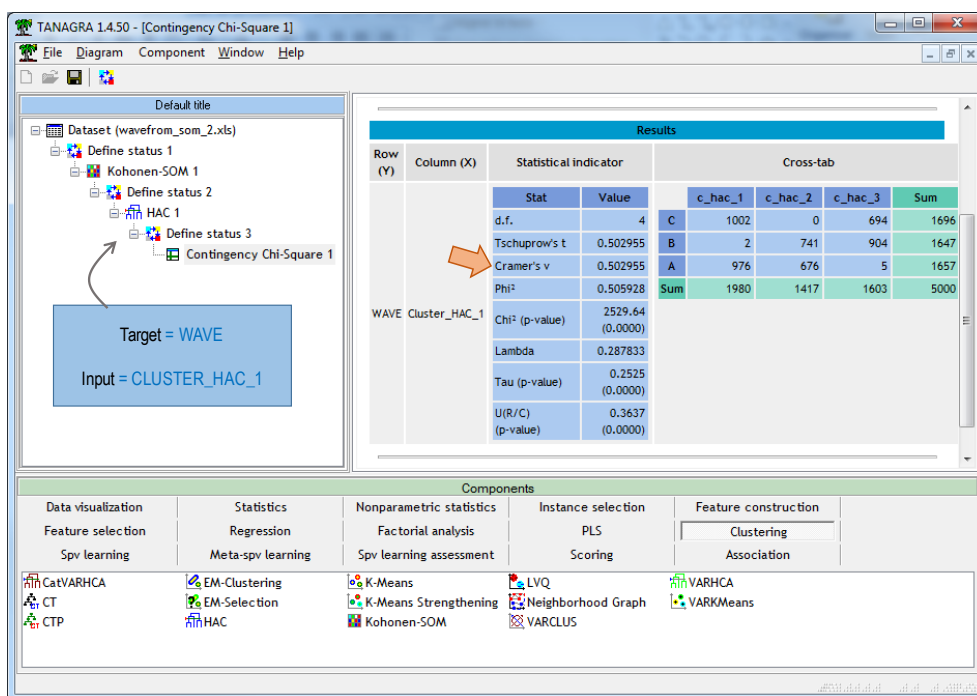
Data visualization	Statistics	Nonparametric statistics	Instance selection	Feature construction
Feature selection	Regression	Factorial analysis	PLS	Clustering
Spv learning	Meta spv learning	Spv learning assessment	Scoring	Association
CatVARHCA	EM-Clustering	K-Means	LVQ	VARHCA
CT	EM-Selection	K-Means Strengthening	Neighborhood Graph	VARKMeans
CTP	HAC	Kohonen-SOM	VARCLUS	

Tanagra displays the number of instances into each cell. It provides also an indication about the quality of the map, this is the proportion of variance explained. We have 59.92% for our analysis.

To launch the cluster analysis (HAC) from the pre-clusters defined by the topological map, we add the HAC component into the diagram by specifying the used variables with a second DEFINE STATUS component: TARGET corresponds to the pre-clusters of KOHONEN-SOM (CLUSTER_SOM_1), INPUT correspond to the descriptors (V1 to V21). We ask the creation of 3 clusters since we know the true solution in advance. In any case, if we let the tool to detect automatically the right number of cluster, it would have proposed this number of clusters for this dataset.



To evaluate the performance of the approach, we compute a cross-tabulation of the computed cluster variable and the observed class attribute. Then we calculate the [Cramer's V](#) criterion. The diagram is completed in the following manner.



$V = 0.5029$. This is a reference value. Let us see if K-Means algorithm can do better.

3.3 K-Means with Tanagra

[K-Means](#) algorithm is a well-known approach. We complete the diagram as follows.

The image shows two screenshots from the Tanagra 1.4.50 software. On the left is the 'K-Means parameters' dialog box. On the right is the main Tanagra interface showing a workflow diagram and a results window.

K-Means parameters dialog:

- Parameters tab:
 - Number of clusters: 3
 - Max iterations: 10
 - Number of trials: 5
 - Distance normalization: Variance
 - Average computation: Mc Queen
 - Seed random generator: Standard
- Results tab: (empty)

Main Tanagra Interface:

- Workflow diagram: Dataset (wavefrom_som_2.xls) -> Define status 1 -> Kohonen-SOM 1 -> Define status 2 -> HAC 1 -> Define status 3 -> Contingency Chi-Square 1 -> K-Means 1.
- Results window:
 - Global evaluation:**

Within Sum of Squares	62877.3406
Total Sum of Squares	105000.0000
R-Square	0.4012
 - Cluster size and WSS:**

Clusters	3		
Cluster	Description	Size	WSS
cluster n°1	c_kmeans_1	1784	22617.6476
cluster n°2	c_kmeans_2	1420	17389.7084
cluster n°3	c_kmeans_3	1796	22869.9846
- Components panel: Clustering, K-Means, K-Means Strengthening.

Again, we compare the computed cluster variable with the class attribute WAVE.

The image shows the Tanagra 1.4.50 interface for a 'Contingency Chi-Square 2' analysis. The workflow diagram shows 'K-Means 1' leading to 'Contingency Chi-Square 2'. A box indicates 'Target = WAVE' and 'Input = CLUSTER_KMEANS_1'.

Contingency Chi-Square 2 Results:

Row (Y)	Column (X)	Statistical indicator		Cross-tab				
		Stat	Value	c_kmeans_1	c_kmeans_2	c_kmeans_3	Sum	
WAVE	Cluster_KMeans_1	d.f.	4	C	0	725	971	1696
		Tschuprow's t	0.501076	B	959	688	0	1647
		Cramer's v	0.501076	A	825	7	825	1657
		Phi²	0.502155	Sum	1784	1420	1796	5000
		Chi² (p-value)	2510.78 (0.0000)					
	Lambda	0.290254						
	Tau (p-value)	0.2513 (0.0000)						
	U(R/C) (p-value)	0.3640 (0.0000)						

The components panel includes: Data visualization, Statistics, Nonparametric statistics, Instance selection, Feature construction, Feature selection, Spv learning assessment, Factorial analysis, Scoring, Association, Clustering, Spv learning, and Meta-spv learning.

Cramer's V is now equal to **0.5010**.

Both approaches are equivalent when compared with the true class membership. But, the association between (SOM + HAC) combines two advantages (1) the dimensionality reduction ability of the SOM algorithm which enables to visualize the data into a two-dimensional representation space proposed by the topological map (SOM 1, 2017); (2) the inherent qualities of the HAC which allow to propose scenarios of nested solutions and a tool, the dendrogram, for evaluating their reliability (CAH, 2017). The interpretation of the results is made easier.

4 Conclusion

Kohonen maps are both a technique for dimensionality reduction, visualization, and clustering (at least, it allows to establish a preclustering of data). In this tutorial, I tried to highlight the advantages of the approach, emphasizing the different possibilities of interpretations with the various maps, also showing the power of its association with the Hierarchical Agglomerative Clustering (HAC) in an unsupervised learning process.

5 References

(CAH, 2017) Rakotomalala R., "Hierarchical Agglomerative Clustering (slides)", [Tanagra Tutorial](#), June 2017.

(Lynn, 2014) Lynn S., "Self-Organising Maps for Customer Segmentation using R", [R-bloggers](#), February 2014.

(SOM 1, 2017) Rakotomalala R., "Self-organizing map (slides)", [Tanagra Tutorial](#), June 2017.

(SOM 2, 2009) Rakotomalala R., "Self-organizing map (SOM)", [Tanagra Tutorial](#), July 2009.

(Wehrens, 2015) Wehrens R., "Package '[kohonen](#)'", September 2015.

(Wehrens et Buydens, 2007) Wehrens R., Buydens L., "Self and Super Organising Maps in R: the kohonen package", [Journal of Statistical Software](#), (21) 5, 2007.