1 Objectif

Représentation des données à l'aide d'une carte de Kohonen, suivie d'une classification automatique. Logiciel R (package kohonen) et Tanagra (composant Kohonen-Som).

Ce tutoriel vient compléter le support de cours consacré aux « Cartes auto-organisatrices de Kohonen » (SOM 1, 2016). Le premier objectif est de mettre en lumière deux aspects importants de l'approche : sa capacité à résumer l'information disponible dans un espace à deux dimensions ; son couplage avec une méthode de classification automatique permettant d'associer la représentation topologique (et la lecture que l'on peut en faire) à l'interprétation des groupes issus de la typologie. Nous utiliserons le logiciel R et le package « kohonen » (Wehrens et Buydens, 2007).

Dans un deuxième temps, nous effectuerons une étude comparée de la qualité de la segmentation avec les K-Means, qui fait figure de référence, en procédant à une validation externe c.-à-d. en confrontant les regroupements proposés par les approches avec une classification préétablie. Cette procédure est souvent utilisée en recherche pour évaluer les performances des méthodes de clustering. Elle prend tout son sens lorsqu'on l'applique sur des données artificielles où l'on connaît – parce que générée sciemment – la bonne typologie. Nous utiliserons les composants K-Means et Kohonen-Som de Tanagra.

Rendons à César ce qui lui appartient, ce tutoriel est en partie inspiré de l'article de Shane Lynn, accessible sur le site R-bloggers (Lynn, 2014). Je me suis évertué à le compléter en introduisant les calculs intermédiaires permettant de mieux saisir le sens des graphiques, et en réalisant l'étude comparative.

2 Les cartes de Kohonen avec R (package « kohonen »)

2.1 Données

Les fameuses données WAVEFORM décrites dans l'ouvrage CART (Breiman et al., 1984) sont accessible sur le dépôt UCI[1]. La base est composée de 21 descripteurs. Deux modifications ont été introduites dans cette partie du tutoriel : puisque nous sommes dans le cadre d'une classification non-supervisée, la variable cible a été supprimée ; un échantillon de 1500 observations a été extrait au hasard parmi les 5000 disponibles. Cette

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

base est intéressante dans notre contexte parce que (1) nous savons qu'elle est composée de trois classes, (2) nous connaissons les variables pertinentes (les premières et dernières variables ne jouent aucun rôle dans la discrimination).

2.2 Importation et préparation des données

Premières étapes toujours dans toute analyse, nous chargeons les données et nous affichons les statistiques descriptives.

```
#modification du dossier par défaut
setwd("… votre dossier de travail…")
#chargement – fichier texte avec séparateur tabulation
D <- read.table("waveform_som_1.txt",sep="\t",dec=".",header=T)
#calcul et affichage des statistiques descriptives
print(summary(D))
```

Nous vérifions surtout que les données sont dépourvues d'anomalies.

```
       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
```

Pour évacuer les problèmes d'échelle, nous centrons et réduisons les données à l'aide de la commande **scale()**

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

Nous sommes parés pour lancer les traitements.

2.3 Paramétrage et apprentissage de la carte

Paramètres et apprentissage. Il nous faut au préalable installer et charger le package « kohonen » avant de lancer l'apprentissage avec la fonction **som()**.

```
#kohonen library
library(kohonen)

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

Nous avons demandé une grille hexagonale de taille 15 x 10 avec, option par défaut de la fonction som() pour ce type d'architecture, une structure de voisinage circulaire (Figure 1).

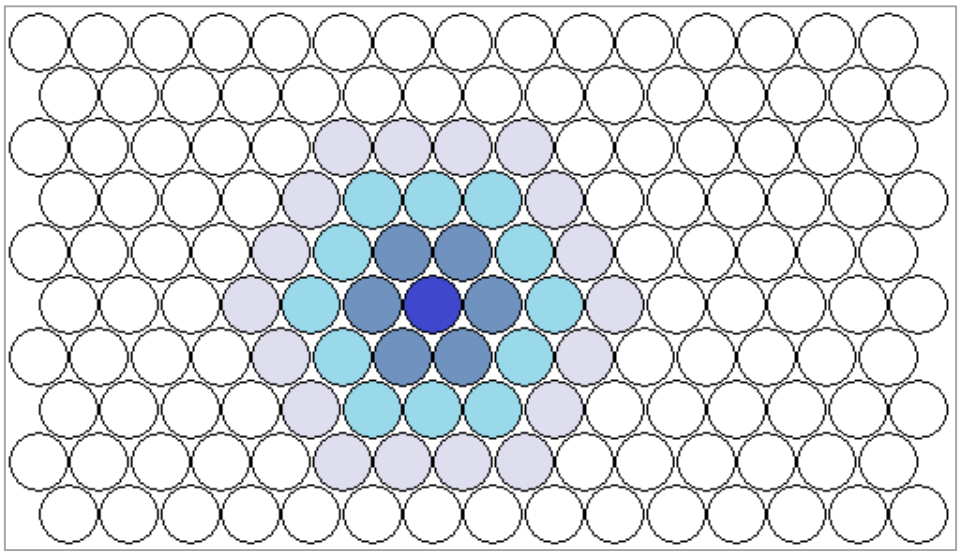


Figure 1 - Carte hexagonale 15 x 10 avec une structure de voisinage circulaire

L'affichage des informations via summary()…

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

… est plutôt succincte.

```
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
```

Architecture de la grille. Il faut aller dans les propriétés de l'objet pour disposer d'informations plus détaillées, notamment concernant l'architecture de la grille.

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

L'objet est de la classe **somgrid,** il possède plusieurs champs.

```
$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"
```

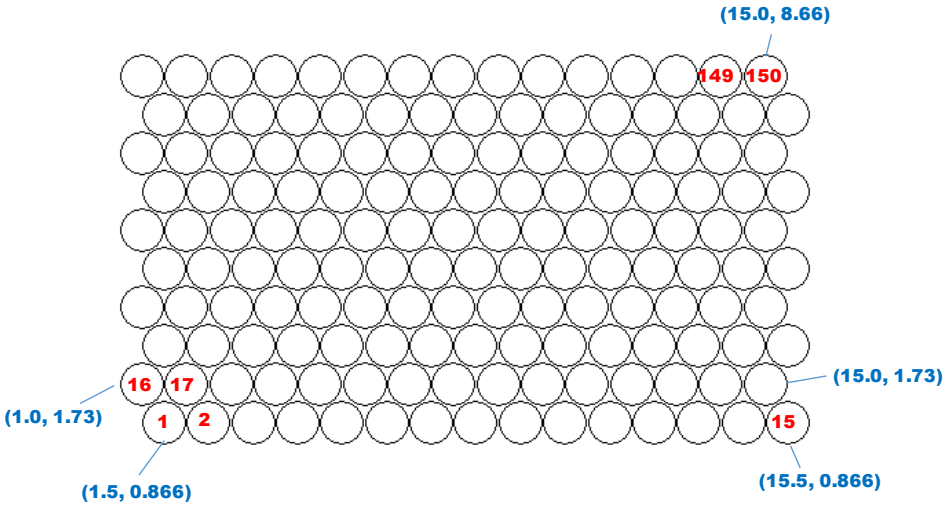


Figure 2 - Numéro (en rouge) et coordonnées (en bleu) des nœuds de la carte topologique

Le champ « **$pts** » attire notre attention. Les nœuds de la carte sont indexés de 1 à 150 (15 lignes x 10 colonnes = 150 cellules). Une coordonnée ligne (**x**) et colonne (**y**) est associée à chacun d'entre eux (Figure 2). Il est donc possible de calculer des statistiques descriptives relatives aux nœuds en les situant dans l'espace de représentation. Même si ce n'est pas très courant, on pourrait par exemple calculer un cercle de corrélation pour identifier les trajectoires dans la carte topologique. Nous y reviendrons plus bas (page 11).

2.4 Représentations graphiques

Les représentations graphiques constituent un des aspects séduisants des cartes topologiques. Nous en détaillons quelques unes dans cette section.

Progression de l'apprentissage. Ce graphique permet d'apprécier la convergence de l'algorithme. Il indique l'évolution de la distance moyenne au nœud le plus proche dans la carte. Dans notre exemple (Figure 3), après une forte décroissance, nous arrivons sur un plateau dans la partie finale. Des itérations supplémentaires ne sont pas nécessaires (par défaut, la procédure demande RLEN = 100 itérations c.-à-d. l'ensemble de la base est présenté 100 fois au réseau durant l'apprentissage). Il faudrait augmenter RLEN si la courbe continue à décroitre.

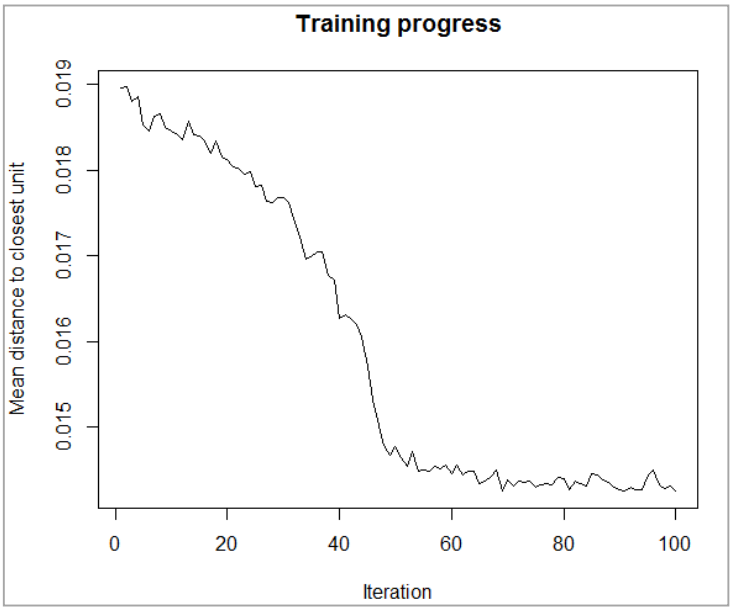


Figure 3 - Progression de l'apprentissage

Effectifs dans les nœuds. Les effectifs dans les nœuds permettent d'identifier les zones à forte densité. Le package « kohonen » permet de spécifier le jeu de couleurs à utiliser. Nous définissons la fonction **degrade.bleu ()** [dégradé de bleu].

```
#jeu de couleurs pour les nœuds de la carte
degrade.bleu <- function(n){
  return(rgb(0,0.4,1,alpha=seq(0,1,1/n)))
}
```

La fonction prend en entrée « n » qui correspond au nombre de couleurs à produire. Nous générons un ensemble de couleurs bleues plus ou moins opaques à l'aide de **rgb()**. Plus la couleur est foncée, plus les effectifs sont élevés.

Il reste à afficher la carte avec plot() en spécifiant les options adéquates (Figure 4).

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

Dans l'idéal, la répartition devrait être assez homogène. La taille de la carte doit être réduite s'il y a de nombreuses cellules vides. A contrario, elle doit être augmentée si des zones de très forte densité apparaissent (Lynn, 2014).

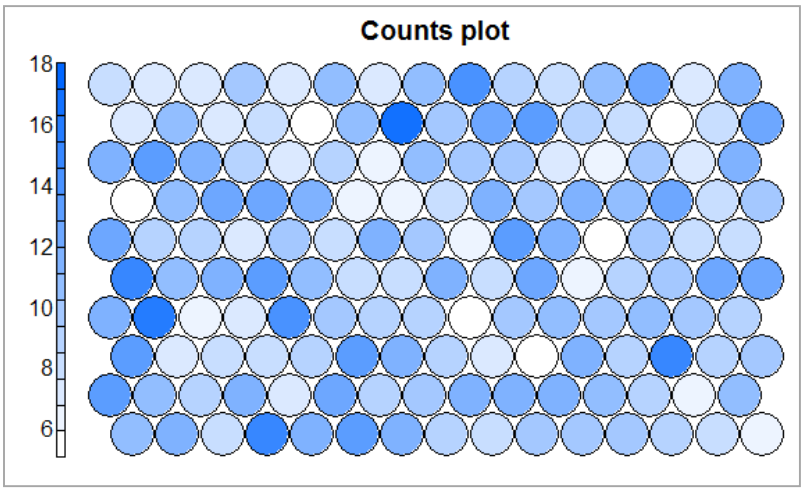


Figure 4 - Effectifs dans les nœuds

Le détail est accessible avec les champs de l'objet somgrid. Le champ « **$unit_classif** » indique les numéros des nœuds auxquels sont affectés les individus.

```
#noeud d'appartenance des observations
print(carte$unit.classif)
```

Le premier individu est affecté au nœud n° 83, le second au n° 6, …, le dernier au 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
```

Nous pouvons ainsi comptabiliser les individus dans chaque nœud avec **table()**.

```
#nombre d'observations affectés à chaque noeud
nb <- table(carte$unit.classif)
print(nb)
```

Il y a 11 individus dans le nœud n°1, 12 dans le n°2, …, 12 dans le 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
```

Nous pouvons aussi vérifier s'il y a des nœuds vides en comptant le nombre de valeurs dans le vecteur engendré par table().

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

Nous avons **150** valeurs. Tous les nœuds sont composés d'au moins une observation.

Distance au voisinage. Appelé « graphique U-Matrice », il indique la somme des distances aux voisins immédiats pour chaque nœud.

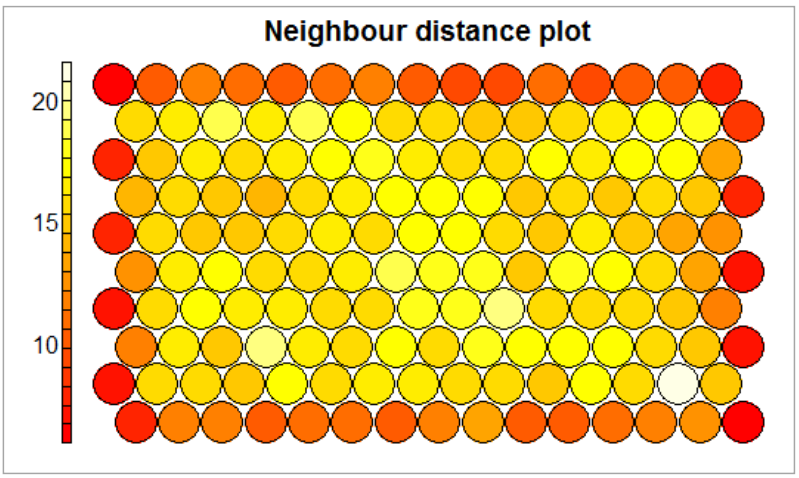


Figure 5 - Distance au voisinage

Le graphique est produit par la commande suivante :

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

Dixit la documentation du package, des nœuds formant la même classe ont tendance à être proches. Les zones frontières sont délimitées par des nœuds éloignés les uns des autres. Dans notre graphique (Figure 5), les nœuds peu distants des autres sont de couleur foncée. Nous constatons qu'ils sont concentrés sur les extrémités de la carte. Ce qui laisse espérer une bonne séparation des classes lors de la typologie.

2.5 Représentations graphiques – Rôle des variables

Une section à part est consacrée à ce type de graphique car il permet d'établir le rôle des variables dans la définition des différentes zones qui composent la carte topologique. Il est crucial dans l'interprétation des résultats. Il le sera d'autant plus lorsque nous combinons le nouveau système de représentation des données avec une typologie produite par un algorithme de classification automatique (section 2.7).

Codebook vectors. Ce graphique représente les vecteurs de poids (le *profil*) de chaque nœud dans un diagramme circulaire.

```
#codebooks - profils des noeuds
plot(carte,type="codes",codeRendering = "segments")
```

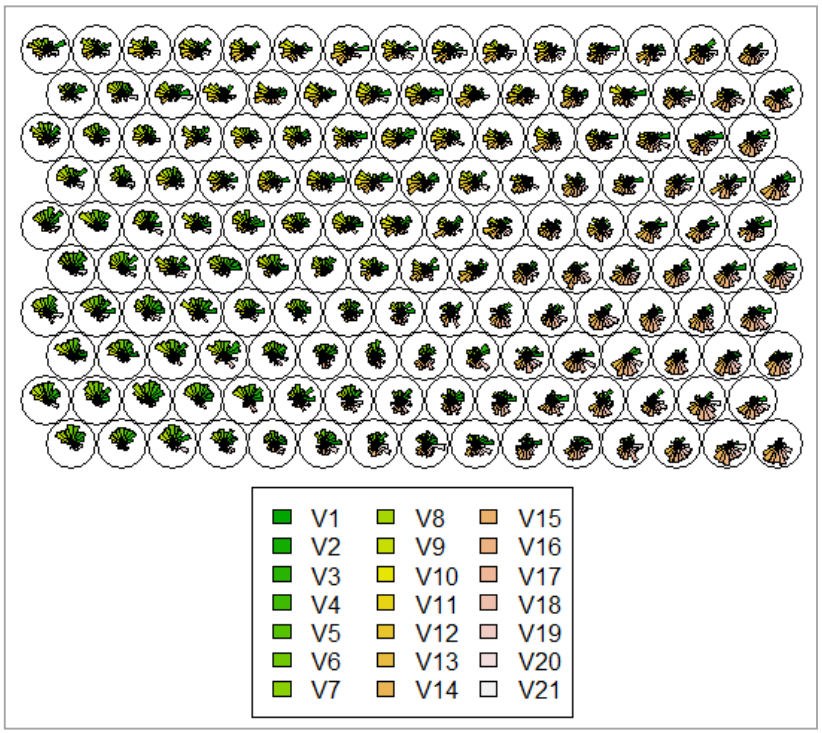


Figure 6 - Graphique "codebooks"

Il permet de distinguer en un coup d'œil la nature des différentes zones de la carte au regard des variables actives (Figure 6). Avec des yeux d'aigle, on peut se rendre compte que la partie sud-ouest est plutôt caractérisée par les valeurs élevées des premières variables (en vert), le nord est plutôt associé aux variables intermédiaires (en jaune), la partie sud-est serait relative aux dernières variables (en rose).

Nous affichons les codebooks pour les deux premiers nœuds.

```
#tableau des codebooks pour les deux premiers noeuds
print(carte$codes[1:2,])
```

Les valeurs des variables sont comparables d'un nœud à l'autre. Mais aussi, parce que nous les avons centrées et réduites, elles le sont également à l'intérieur de chaque nœud.

```
            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
```

On peut difficilement disserter à partir de deux nœuds, mais il semble que le coin sud-ouest à son extrémité – les cellules n°1 et 2 sont les premières en partant du bas et à gauche de la carte (Figure 2) – soit caractérisé par les valeurs élevées (resp. faibles) des variables V5, V6, V7 (resp. V12, V13, V14).

Graphiques heatmaps. Pour séduisant qu'il soit, on se rend bien compte que le graphique « codebook » est difficile à déchiffrer lorsque le nombre de nœuds et de variables augmente. Plutôt que de réaliser un seul graphique pour l'ensemble des variables, nous pouvons réaliser un graphique pour chaque variable, en essayant de mettre en évidence les contrastes entre les zones à valeurs élevées et faibles. Cette description univariée est plus facile à appréhender. Nous utilisons les valeurs des « codebooks » pour construire les graphiques que nous faisons tenir dans un seul cadre. Le jeu de couleurs utilisé associe le rouge (resp. bleu) aux valeurs élevées (resp. faibles). C'est le rôle de la fonction coolBlueHotRed() (Wehrens, 2015).

```
#jeu de couleurs pour le graphique
coolBlueHotRed <- function(n, alpha = 1) {
```

```
  rainbow(n, end=4/6, alpha=alpha)[n:1]
}
#graphique pour chaque 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))
```

Les zones d'influence apparaissent clairement (Figure 7).

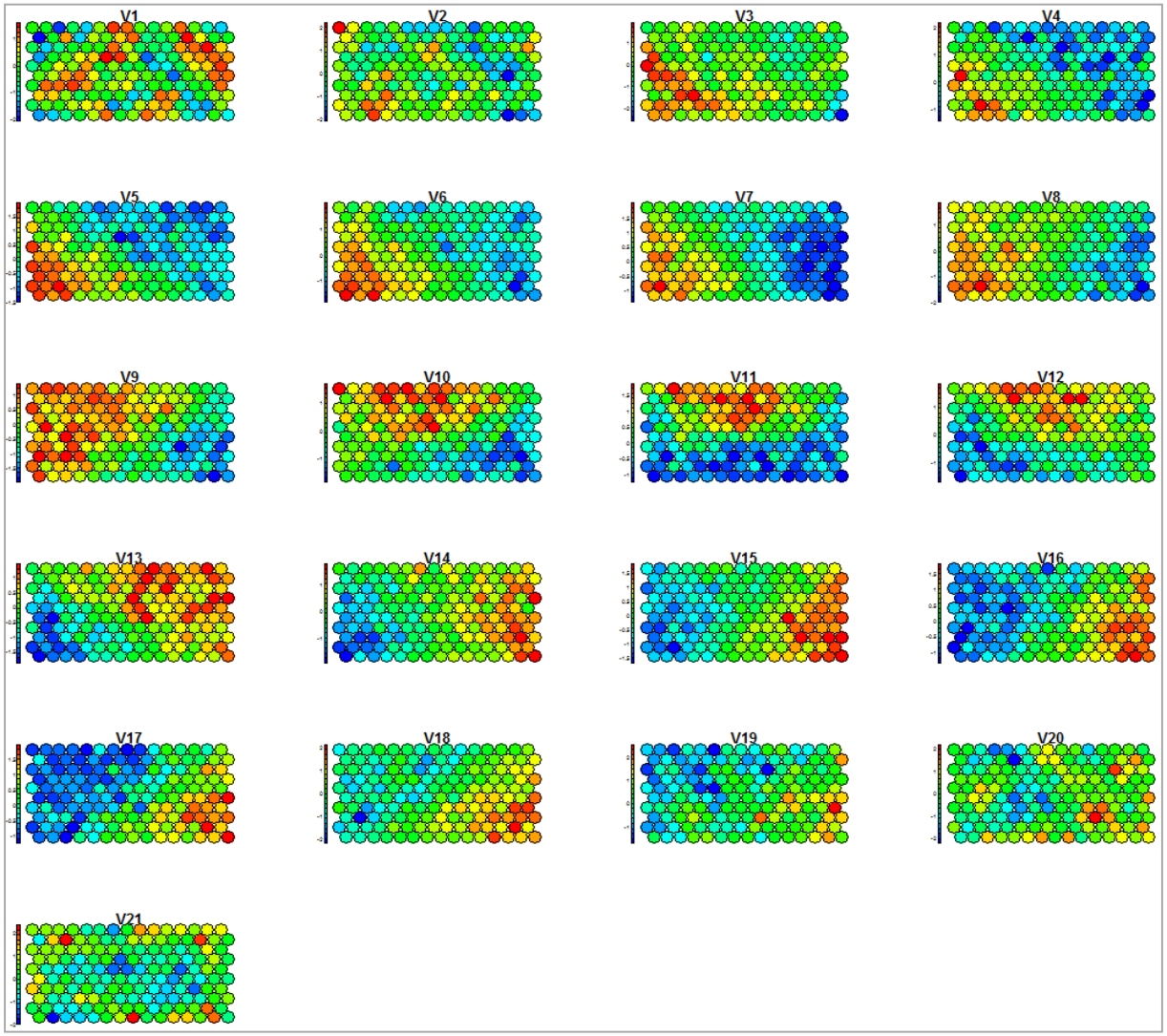


Figure 7 - Zones de valeurs élevées (rouge) et faible (bleu) pour chaque variable

Nous distinguons *approximativement* 3 zones qui seront confirmées par la classification automatique (section 2.7) : sud-ouest, caractérisée par les valeurs élevées des variables V3 à V8 ; nord par V10 à V12 ; sud-est par V14 à V18.

Cercle des corrélations. Ce type graphique, très populaire en analyse en composantes principales (ACP), n'est pas usuel dans les cartes de Kohonen. Pourtant, nous disposons de tous les éléments nécessaires au calcul : à chaque nœud est associée une coordonnée (x,y) (Figure 2), une pondération (effectifs, Figure 4) et une valeur issu du codebook.

Nous calculons la corrélation pondérée pour chaque colonne du codebook.

```
#corrélation avec les axes (x,y) de la carte
#v : variable (une colonne du codebook), w : poids, grille : carte de Kohonen
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)
}

#calcul des corrélations pour l'ensemble des colonnes du codebook
CORMAP <- apply(carte$codes,2,weighted.correlation,w=nb,grille=carte)
print(CORMAP)
```

Nous obtenons une paire de valeurs pour chaque 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
```

Nous les insérons dans un graphique :

```
#représentation graphique du cercle des corrélations
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)
```

La représentation (Figure 8) n'est pas sans rappeler le cercle des corrélations issue d'une ACP sur les mêmes données. Je trouve d'ailleurs que le graphique ressemble énormément à

un cœur (inversé ici). Je me suis toujours demandé si les auteurs l'ont fait sciemment en générant les données.

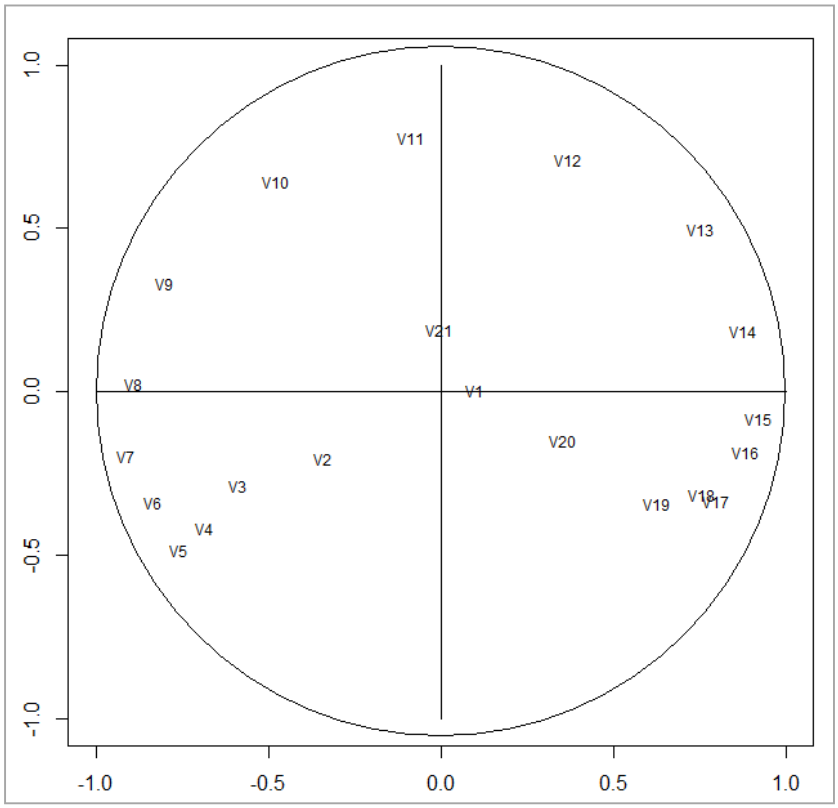


Figure 8 - Cercle des corrélations pour une carte topologique

Nous sommes bien d'accord, le cercle de corrélation se présente avant tout comme un outil indicatif dans ce cadre. Il n'y a pas de significations mathématiques aux calculs, d'autant plus que la carte topologique est censée pouvoir représenter des formes non-linéaires. Il n'en reste pas moins que ce type de graphique synthétique est autrement plus facile à inspecter que les codebooks (Figure 6) et les heatmaps (Figure 7) lorsque le nombre de variables et de nœuds augmente. Dans notre exemple, les conclusions du cercle des corrélations sont cohérentes avec les représentations sus-citées.

2.6 Pertinence des variables

Corroborer les suggestions des représentations graphiques avec des marqueurs statistiques est toujours une bonne chose. Les contrastes forts mettent en lumière l'importance des variables dans la définition des différentes zones dans le graphique des « heatmaps » (Figure 7). Nous pouvons traduire la même idée à l'aide d'un indicateur de dispersion telle que la variance, il devient alors possible de hiérarchiser leur rôle.

Deux éléments nous permettent de le faire : les variables sont centrées et réduites, leurs influences sont directement comparables ; les vecteurs codebooks des nœuds correspondent à des estimations des moyennes conditionnelles, calculer leur dispersion pour chaque variable revient à estimer la variance inter-nœuds et, par là, leur pouvoir explicatif.

Nous calculons la variance pondérée par les effectifs des nœuds.

```
#dispersion des codebooks
#variance pondérée par les effectifs
sigma2     <-     sqrt(apply(carte$codes,2,function(x,effectif){m<-sum(effectif*(x-
weighted.mean(x,effectif))^2)/(sum(effectif)-1)},effectif=nb))
#affichage par ordre d'importance décroissante
print(sort(sigma2,decreasing=T))
```

Les variables importantes (parce qu'induisent les plus forts contrastes) apparaissent dans les premières positions :

```
      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
```

Les variables extrêmes (V1 à V3) et (V19 à V21) sont les moins influentes c.-à-d. les moyennes conditionnelles tendent à être homogènes sur l'ensemble de la carte. Ces résultats confirment ce que nous avions constatés dans les différents graphiques vus précédemment. Mais les indicateurs numériques se prêtent mieux au traitement des grands ensembles de données.

2.7 Classification automatique à partir de la carte

Regroupement des nœuds. Pouvoir enchaîner avec une classification automatique est un des intérêts des cartes topologiques de Kohonen. En effet, nous disposons d'une représentation 2D des observations avec des zones et des proximités que l'on sait caractériser avec les variables. Les noeuds constituent un excellent point de départ pour un regroupement itératif en classes. La classification ascendante hiérarchique (CAH) est souvent mise à contribution dans ce contexte.

Nous calculons la distance entre les nœuds (entre les codebooks) dans la carte (**dist**), puis nous effectuons une CAH (**hclust**) avec le critère de Ward (carré de la distance entre centres de classes) (**method = « ward.D2 »**).

```
#matrice de distance entre les noeuds
dc <- dist(carte$codes)

#cah - attention à l'option members
cah <- hclust(dc,method="ward.D2",members=nb)
plot(cah,hang=-1,labels=F)

#matérialiser les 3 classes dans le dendrogramme
rect.hclust(cah,k=3)
```

L'option « **members** » est primordiale. En effet, les nœuds sont constitués d'individus. Ils n'ont pas tous le même poids. Il faut en tenir compte lors du calcul du critère de Ward.

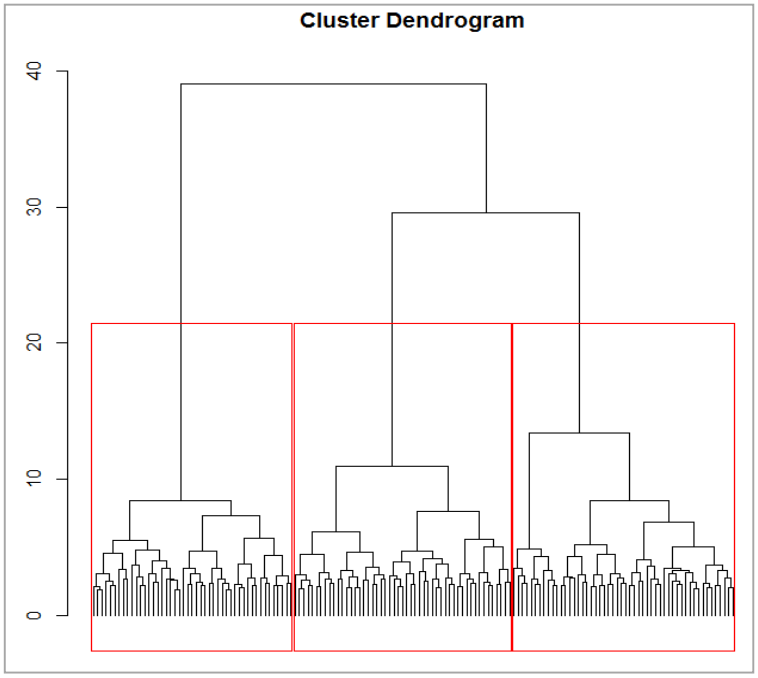


Figure 9 - Dendrogramme

Un regroupement en 3 classes paraît légitime à la vue du dendrogramme (Figure 9).

Nous créons les groupes sous R (**cutree**) :

```
#découpage en 3 classes
groupes <- cutree(cah,k=3)
print(groupes)
```

La variable 'groupes' permet d'associer les nœuds de la carte à leur groupe d'appartenance.

```
  [1] 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 2 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 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 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 3 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2
```

Le premier nœud n°1 appartient au *groupe 1*, le second aussi, …, le dernier nœud de la carte (n°150) appartient au *groupe 2*.

Il est possible de visualiser les regroupements dans la carte.

```
#visualisation des classes dans la carte topologique
plot(carte,type="mapping",bgcol=c("steelblue1","sienna1","yellowgreen")[groupes])
add.cluster.boundaries(carte,clustering=groupes)
```

Les 3 zones identifiées dans les différents graphiques précédents sont clairement mises en évidence. Et nous savons les interpréter directement maintenant.

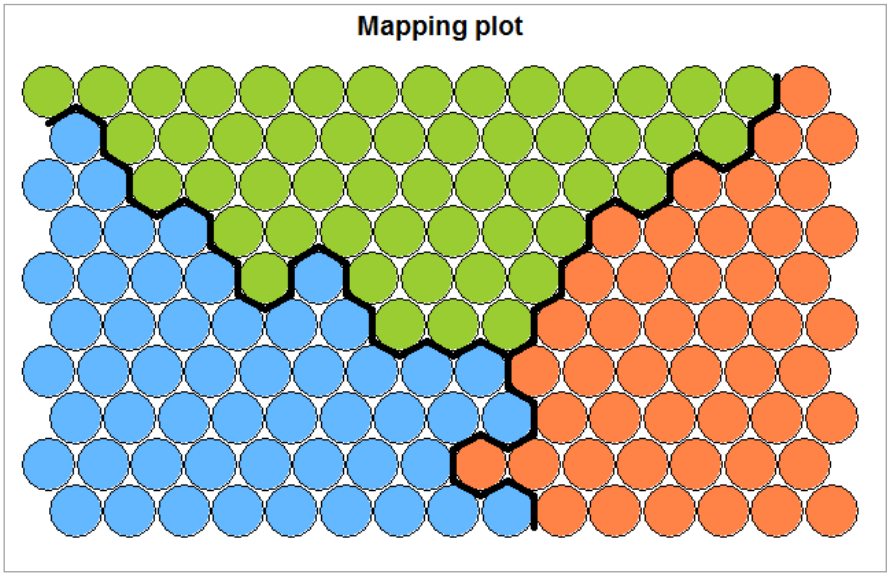


Figure 10 - Représentation des classes dans la carte topologique

Remarque : Les nœuds adjacents sont regroupés dans la même classe. Cela semble normal en considération des propriétés de proximités de la carte. Des hiatus peuvent apparaître parfois, principalement à cause des contraintes inhérentes à l'algorithme de classification automatique.

Groupe d'appartenance des individus. Les nœuds de la carte sont rattachés aux classes. Mais, habituellement, nous nous intéressons au rattachement des individus aux groupes lors d'un processus de classification. Nous l'obtenons en associant l'affectation des nœuds aux classes avec l'affectation des individus aux nœuds, soit :

```
#affecter chaque individu à sa classe
ind.groupe <- groupes[carte$unit.classif]
print(ind.groupe)
```

L'individu n°1 est associé au groupe 3, le n°2 au groupe 1, …, le n°1500 au groupe 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
```

Représentation des individus dans l'espace initial. Puisque nous connaissons le groupe d'appartenance des individus et les variables expliquant le partitionnement, il est possible de les représenter dans un espace à 3 dimensions formé à partir des principales variables désignant les 3 zones identifiées précédemment (sections 2.5 et 2.6), à savoir V7, V11 et V15.

Nous utilisons le package « plot3D » de Karline Soetaert.

```
#représentation des individus et leurs groupes d'appartenance dans un espace 3D
library(plot3D)
points3D(D$V7,D$V11,D$V15,colvar=ind.groupe,col=c("steelblue1","sienna1","yellowgreen"),phi=35,theta=70)
```

On distingue aisément les 3 groupes d'observations (Figure 11).

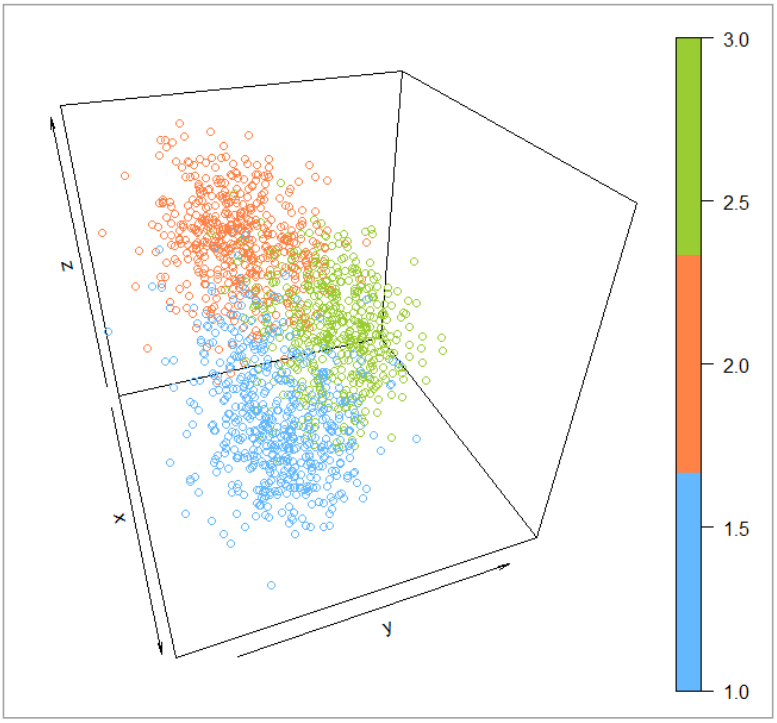


Figure 11 - Représentation des groupes dans un espace 3D (V7, V11 et V15)

Grâce au package « plot3Drgl » du même auteur, il est possible de produire un graphique dynamique permettant de régler interactivement la profondeur et la rotation du graphique.

```
#et en animé
library(plot3Drgl)
plotrgl(lighting = T)
```

Une fenêtre spécifique apparaît. J'avoue m'être beaucoup amusé en tournant le graphique dans tous les sens et en jouant sur le zoom (Figure 12).

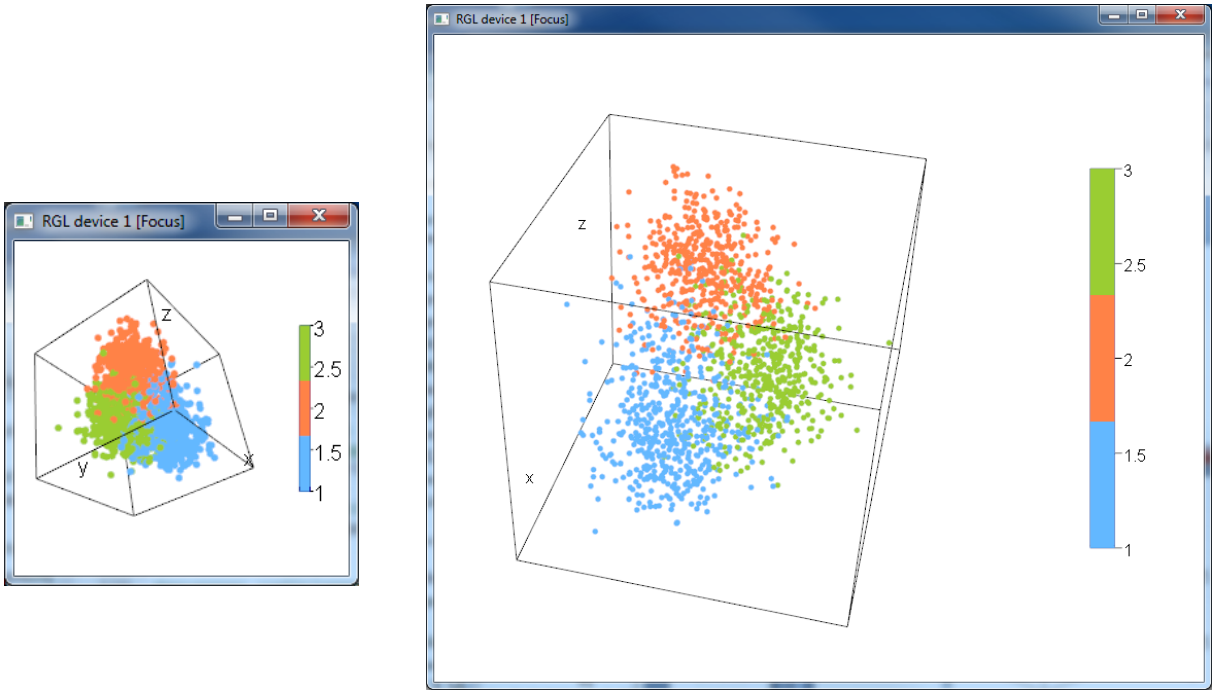


Figure 12- Représentation 3D interactive des groupes (niveaux de zoom différents)

Comme nous pouvons le constater, les possibilités de représentation des données, et par conséquent d'analyses, sont séduisantes et innombrables.

3 Validation externe – SOM vs. K-MEANS avec Tanagra

C'est très joli tout ça, mais est-ce que la démarche (SOM + CAH) est efficace ? Nous avons choisi de confronter la partition induite par SOM + CAH avec celle des K-MEANS pour le vérifier. Un juge impartial sous la forme d'une pré-classification sous-jacente à la génération des données permet d'évaluer la qualité du regroupement proposé. On parle de « validation externe » dans la littérature.

Nous allons à l'essentiel dans cette section. La mise en œuvre des cartes de Kohonen sous TANAGRA est détaillé dans un tutoriel antérieur (SOM 2, 2008). Le lecteur pourra s'y référer pour la construction pas à pas et le paramétrage du diagramme de traitements.

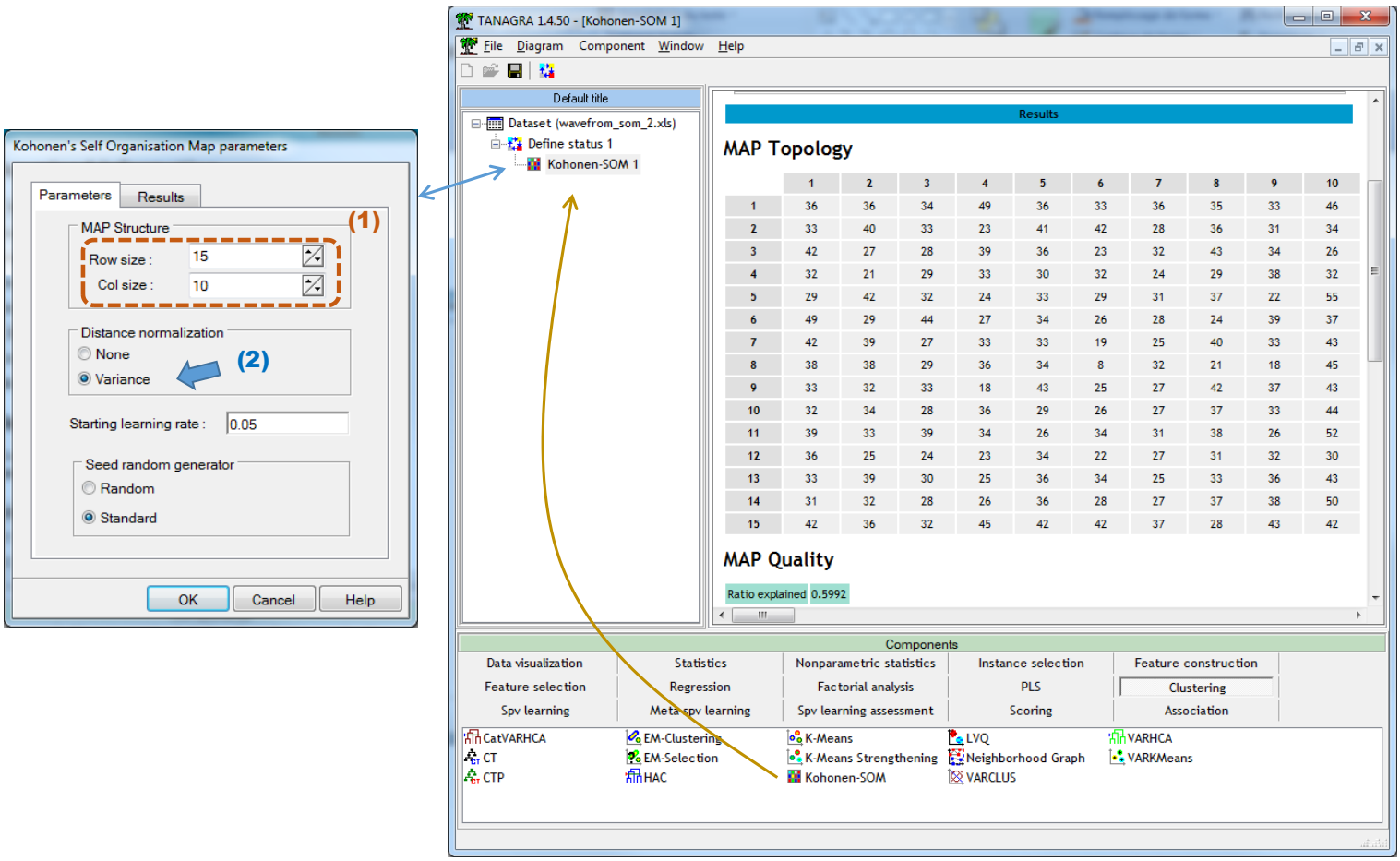
3.1 Données

Nous exploitons la totalité des 5000 observations base WAVEFORM de l'UCI cette fois-ci (**waveform_som_2.xls**). Nous conservons la variable supplémentaire (dernière colonne) WAVE qui correspond à l'attribut classe. Chaque individu appartient à un groupe prédéfini. Pour évaluer une méthode, nous regarderons dans quelle mesure la partition engendrée par l'algorithme de classification est cohérente avec la partition réelle à l'origine de la

génération des données. WAVEFORM étant une base artificielle (créée de toutes pièces, dont on connaît les propriétés), elle se prête à merveille à ce type d'évaluation.

3.2 SOM + CAH sous Tanagra

Après avoir importé les données[2] et sélectionner les variables actives à l'aide de l'outil DEFINE STATUS 1 (INPUT : V1 à V21), nous insérons l'outil KOHONEN-SOM dans le diagramme en spécifiant la taille de la carte (15 lignes x 10 colonnes) **(1)** et la standardisation des variables **(2)** lors du paramétrage.

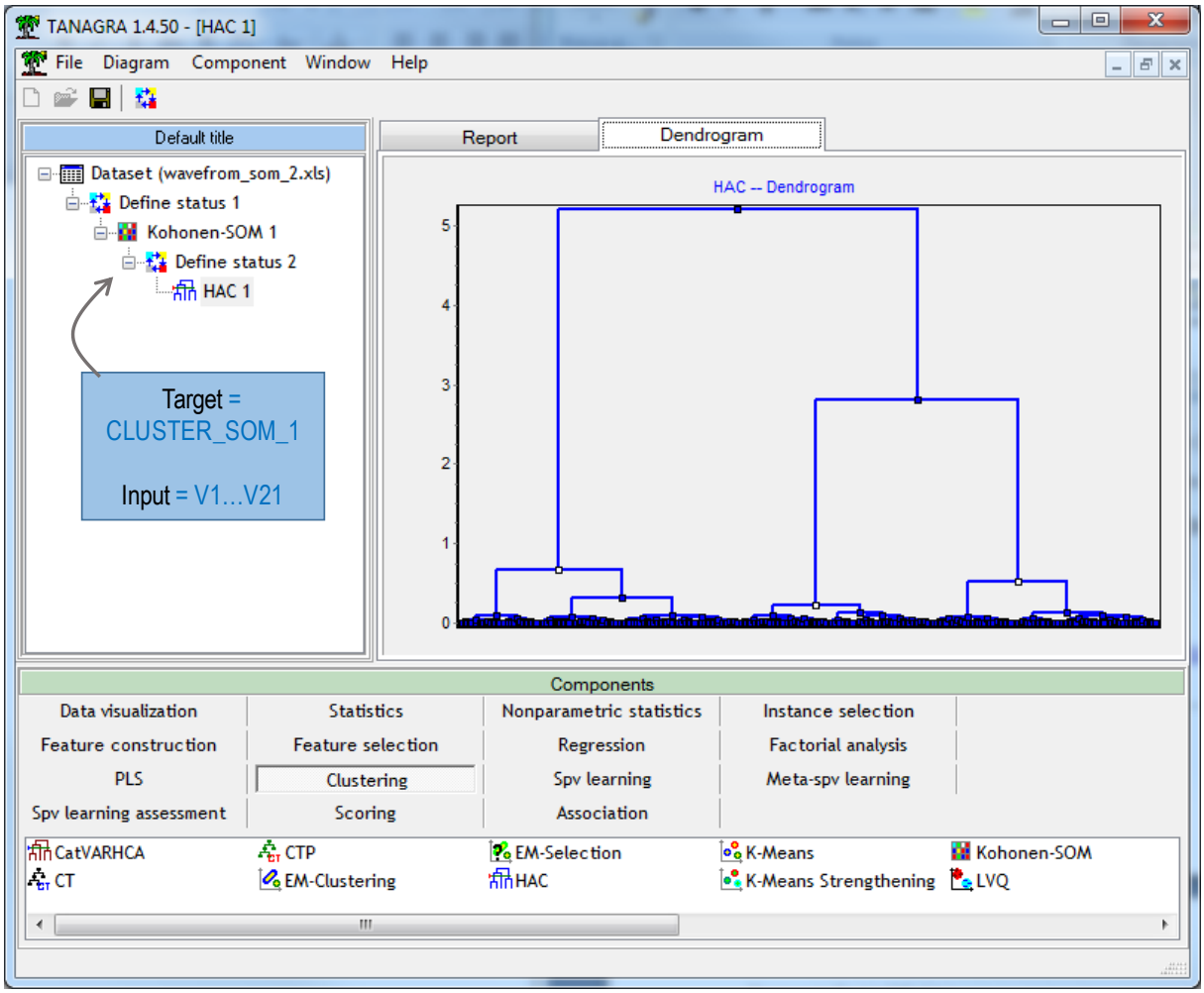


Tanagra affiche les effectifs dans une grille (*Note : il y a certainement un effort à faire au niveau de l'affichage – à voir dans les versions futures*). Il donne également une indication sur la qualité de la carte avec la proportion d'inertie expliquée, 59.92% pour notre exemple.

Pour lancer la classification ascendante hiérarchique à partir du pré-clustering induit par la carte topologique, nous introduisons le composant HAC dans le diagramme en spécifiant les

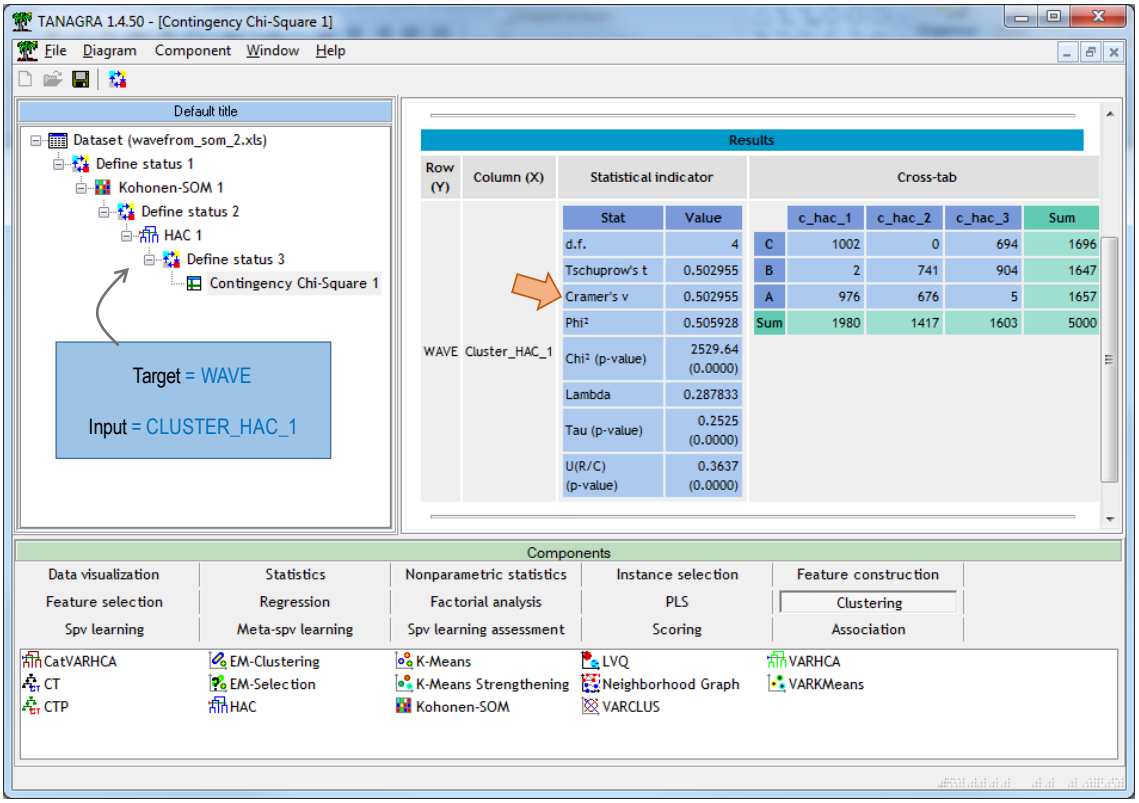
[2] Voir : http://tutoriels-data-mining.blogspot.fr/2010/08/ladd-in-tanagra-pour-excel-2007-et-2010.html

variables à manipuler dans DEFINE STATUS 2 : TARGET correspond à la partition produite par KOHONEN_SOM, représentée par la variable CLUSTER_SOM_1 générée automatiquement par le composant ; les descripteurs originels (V1 à V21) correspondent aux INPUT. Une partition en 3 classes est demandée lors du paramétrage du composant HAC (BEST CLUSTERS : DEFINE = 3). L'outil aurait de toute manière détecté automatiquement cette solution si nous ne l'avions pas spécifié.



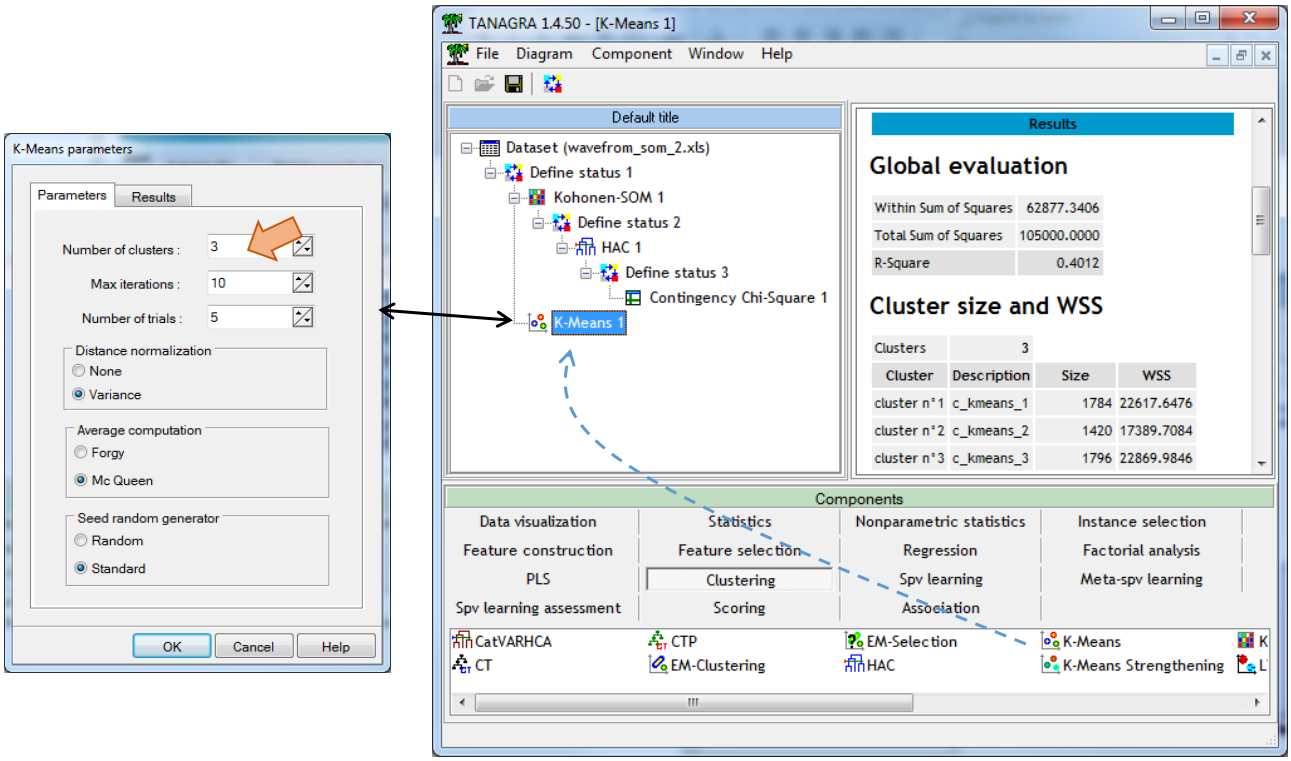
Pour évaluer la qualité de la partition de SOM + CAH, nous la confrontons à la pré-classification (la variable supplémentaire) WAVE dans un tableau croisé et nous calculons le V de Cramer. Le diagramme est complété comme suit.

Le V de Cramer est égal à **0.5029**. Nous ne pouvons rien dire d'autre pour l'instant.

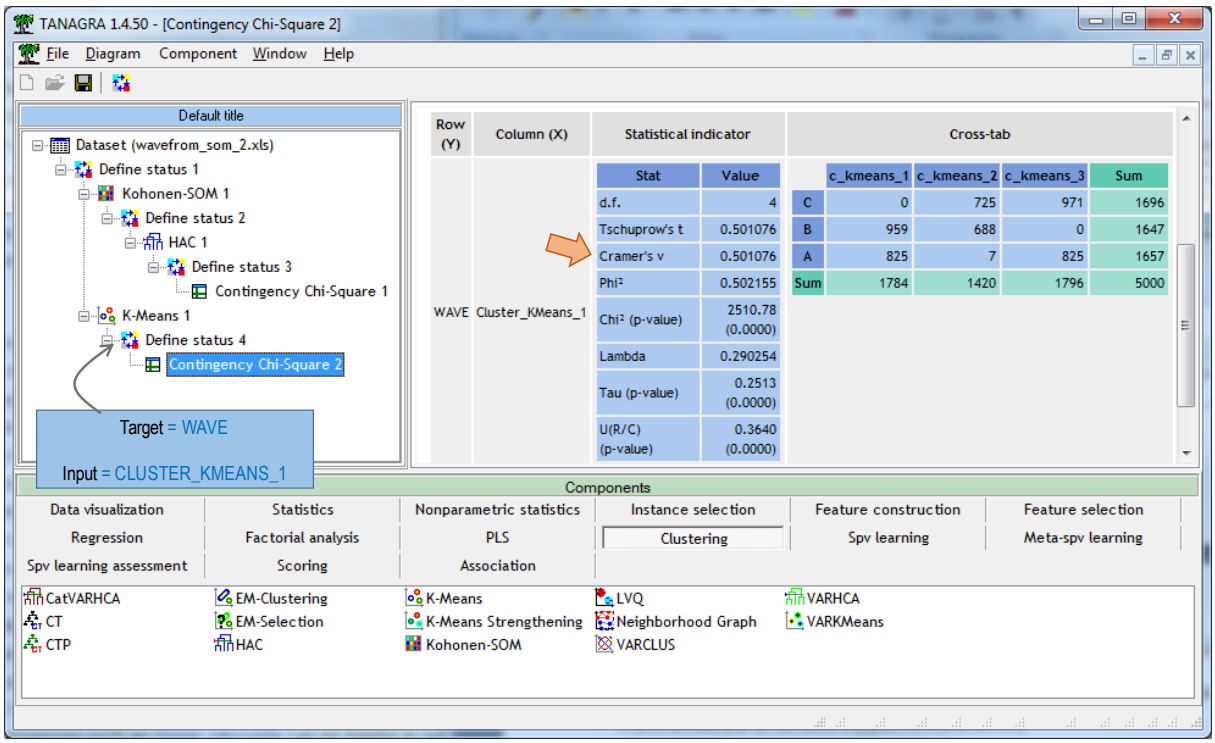


3.3 K-Means sous Tanagra

La méthode des K-Means fait référence en classification automatique. Voyons comment elle se comporte sur nos données.



De nouveau, nous confrontons la partition induite par l'approche avec la pré-classification matérialisée par la variable supplémentaire WAVE.



Le V de Cramer est égal à **0.5010**.

Les deux approches sont équivalentes lorsqu'elles sont confrontées à la classe d'appartenance originelle (WAVE). Nous étions déjà parvenus à une conclusion similaire (les deux méthodes sont proches) lorsque nous avions comparées les partitions engendrées par K-MEANS et (SOM + CAH) sur les mêmes données (SOM 2, 2008).

L'avantage de la combinaison (SOM + CAH) réside dans l'association (1) des atouts de la représentation des données dans un espace intermédiaire de dimension réduite proposé par la carte topologique, (2) des qualités inhérentes à la classification ascendante hiérarchique (CAH, 2016). Ainsi, nous disposons d'un arsenal performant de représentations graphiques et d'outils d'aide à l'interprétation. Le dendrogramme nous permet de définir des scénarios de solutions, attitude autrement plus réaliste que la prescription arbitraire d'un illusoire nombre de classes « optimal ».

4 Conclusion

Les cartes de Kohonen constituent à la fois une technique de réduction de dimensionnalité, de visualisation, et de classification automatique (tout du moins, permet d'établir un pré-

clustering des données). Dans ce tutoriel, j'ai essayé de valoriser les ressources de l'approche, en mettant l'accent sur les différentes possibilités d'illustrations des cartes, en montrant également la puissance de son association avec la classification ascendante hiérarchique (CAH) dans un processus d'apprentissage non-supervisé.

5 Références

(CAH, 2016) Rakotomalala R., "Classification Ascendante hiérarchique – Diapos", Tutoriel Tanagra, Juillet 2016.

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

(SOM 1, 2016) Rakotomalala R., "Les cartes auto-organisatrices de Kohonen - Diapos", Tutoriel Tanagra, Juillet 2016.

(SOM 2, 2008) Rakotomalala R., "Les cartes de Kohonen", Tutoriel Tanagra, Juillet 2008.

(Wehrens, 2015) Wehrens R., "Package 'kohonen'", Septembre 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.