1 Objectif

Classement des individus supplémentaires à partir des résultats d'une typologie (méthode des K-Means).

Le déploiement est une étape importante du Data Mining. Dans le cas d'une typologie, il s'agit, après la construction des classes à l'aide d'un algorithme de classification automatique, d'affecter les individus supplémentaires aux groupes.

Cette phase de catégorisation vient naturellement après le processus de modélisation. La construction et l'interprétation des groupes nous permettent de dégager des caractéristiques et des comportements types. Lorsque apparaît un nouvel individu (un nouveau client pour une banque, un nouveau patient pour un centre hospitalier, etc.), le positionner par rapport aux groupes permet d'anticiper sur son attitude.

Mais le traitement des individus supplémentaires peut aussi servir à renforcer les résultats. Lorsqu'une sous population est connue pour son comportement atypique, la classer par rapport aux groupes construits sur le reste de la population renforce à la fois l'interprétation des groupes et la connaissance que l'on peut avoir des ces « niches » d'observations. On parle plus volontiers d'individus illustratifs dans ce cas.

Dans ce didacticiel, nous construisons tout d'abord les groupes à l'aide de la méthode des K-Means (méthode des centres mobiles). Puis, nous associons chaque individu supplémentaire à la classe qui lui est la plus proche au sens de la distance aux centres de classes. La méthode est viable car la technique utilisée pour classer l'individu supplémentaire est en accord avec la démarche de constitution des groupes lors de l'apprentissage. Ce n'est pas toujours bien compris. Si nous avions utilisé une classification ascendante hiérarchique avec la méthode du saut minimum, classer un nouvel individu à partir de la distance aux centres de classes n'est pas approprié. **La stratégie d'affectation doit être en adéquation avec la stratégie d'agrégation**.

Notre fichier est composé exclusivement de variables qualitatives. Nous devons donc passer par une phase préalable de préparation des variables[1]. Schématiquement, nous réaliserons successivement les opérations suivantes :

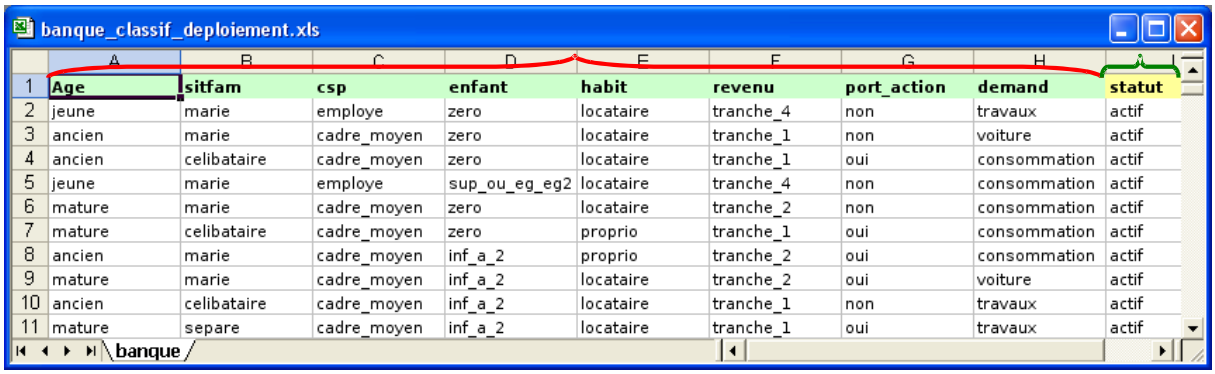
- Importer les données dans le logiciel ;
- Réaliser quelques statistiques descriptives, pour détecter d'éventuelles scories ;
- Dissocier les individus actifs, qui vont participer à la construction de la typologie, des individus supplémentaires, ceux que l'on souhaite simplement classer ;
- Projeter les individus dans un nouveau repère à l'aide d'une analyse des correspondances multiples ;
- Réaliser la classification automatique sur les individus actifs, nous utilisons la méthode des K-Means appliquée aux axes factoriels ;
- Interpréter les groupes ;
- Déployer c.-à-d. associer les individus supplémentaires aux groupes.

Nous utilisons **Tanagra 1.4.28** et **R 2.7.2** (avec le package Facto Miner pour l'analyse des correspondances multiples, http://factominer.free.fr/). Dans ce didacticiel, nos objectifs sont : (1) montrer comment réaliser ce type de tâche avec ces deux logiciels ; (3) comparer les résultats ; (3) en détaillant les commandes dans R, nous donnons une meilleure visibilité sur les calculs réalisés par Tanagra.

[1] Voir aussi : http://tutoriels-data-mining.blogspot.com/2008/03/k-means-sur-variables-qualitatives.html

2 Données

Notre fichier décrit les clients d'une banque. Nous disposons de 198 observations : 98 seront présentés à l'algorithme d'apprentissage, les 100 autres correspondent aux individus supplémentaires que nous souhaitons classer. Le fichier comporte 9 variables, toutes qualitatives, certaines ordinales. La colonne additionnelle « statut » spécifie le rôle de chaque individu : « actif » ou « supplémentaire ». Nous montrons ci-dessous les 10 premières observations du fichier (http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/banque_classif_deploiement.zip).

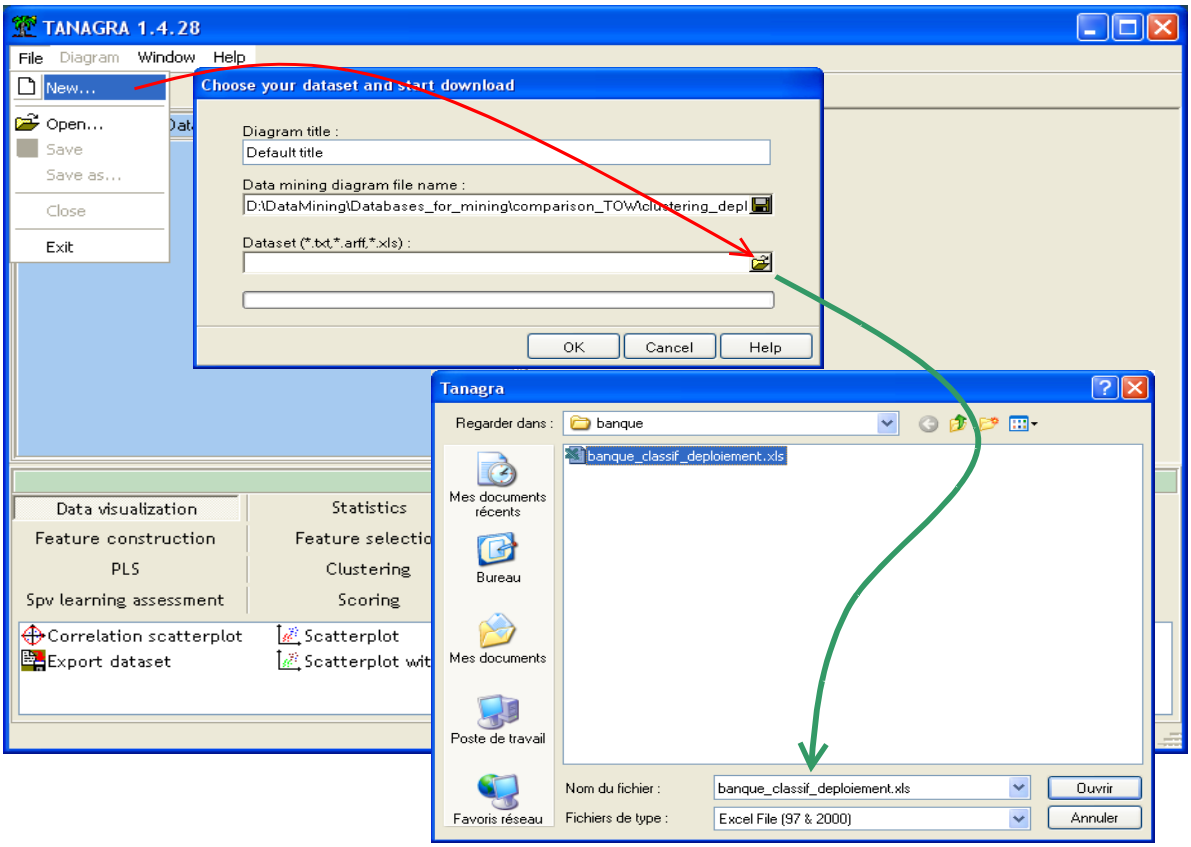


	Age	sitfam	csp	enfant	habit	revenu	port_action	demand	statut
2	jeune	marie	employe	zero	locataire	tranche_4	non	travaux	actif
3	ancien	marie	cadre_moyen	zero	locataire	tranche_1	non	voiture	actif
4	ancien	celibataire	cadre_moyen	zero	locataire	tranche_1	oui	consommation	actif
5	jeune	marie	employe	sup_ou_eg_eg2	locataire	tranche_4	non	consommation	actif
6	mature	marie	cadre_moyen	zero	locataire	tranche_2	non	consommation	actif
7	mature	celibataire	cadre_moyen	zero	proprio	tranche_1	oui	consommation	actif
8	ancien	marie	cadre_moyen	inf_a_2	proprio	tranche_2	oui	consommation	actif
9	mature	marie	cadre_moyen	inf_a_2	locataire	tranche_2	oui	voiture	actif
10	ancien	celibataire	cadre_moyen	inf_a_2	locataire	tranche_1	non	travaux	actif
11	mature	separe	cadre_moyen	inf_a_2	locataire	tranche_1	oui	travaux	actif

3 K-Means et déploiement avec TANAGRA

3.1 Création d'un diagramme et importation des données

Tanagra sait manipuler directement les fichiers au format Excel (XLS). Après avoir démarré le logiciel, nous actionnons le menu FILE/NEW pour créer un nouveau diagramme, nous sélectionnons notre fichier BANQUE_CLASSIF_DEPLOIEMENT.XLS.

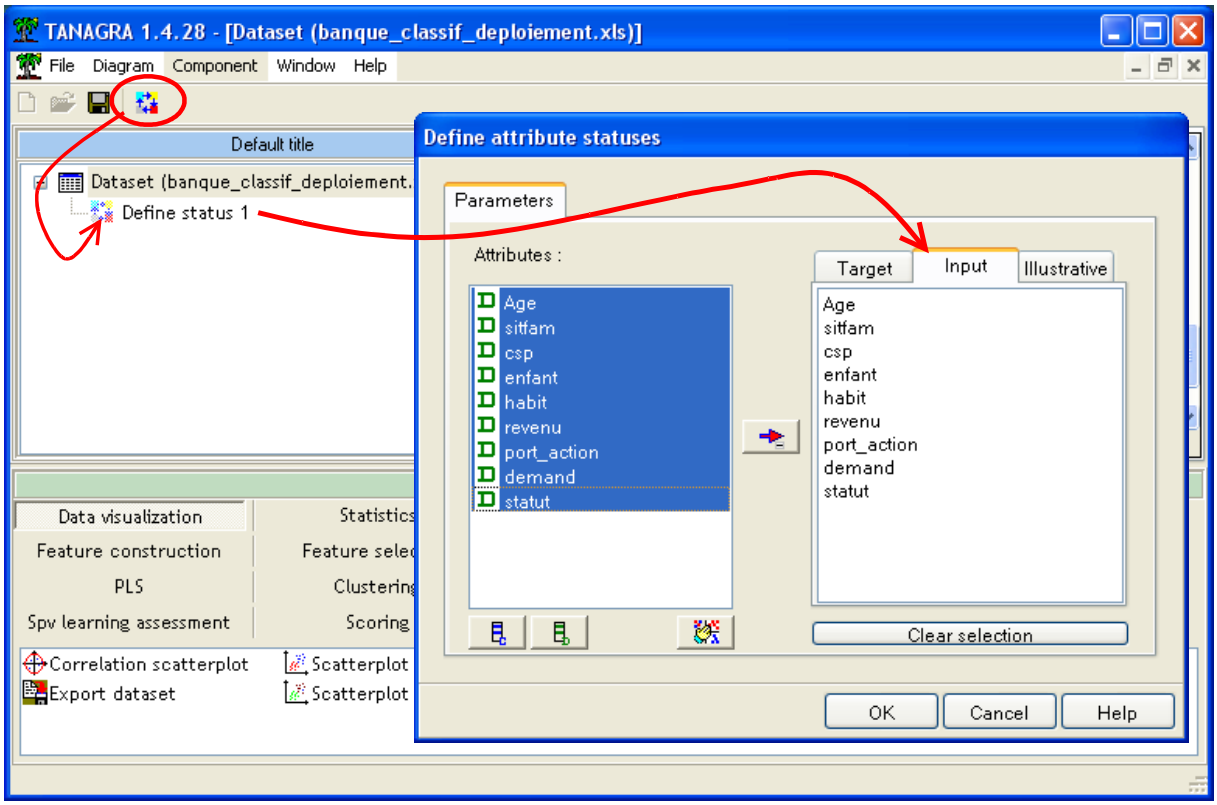


Un diagramme est créé, le fichier importé. Il comporte 198 observations et 9 colonnes[2].

3.2 Statistiques descriptives

Nous calculons les histogrammes de fréquences sur l'ensemble des colonnes. Pour « statut », nous souhaitons seulement comptabiliser le nombre d'observations actives et supplémentaires. Pour les autres variables qui participeront à l'élaboration des classes, il s'agit avant tout de détecter d'éventuelles situations à problèmes, comme la présence de modalités très rares, susceptibles de perturber les résultats. Dans notre fichier, on se méfiera principalement de la modalité « CSP = retraité » qui ne comporte que 4 observations par exemple.

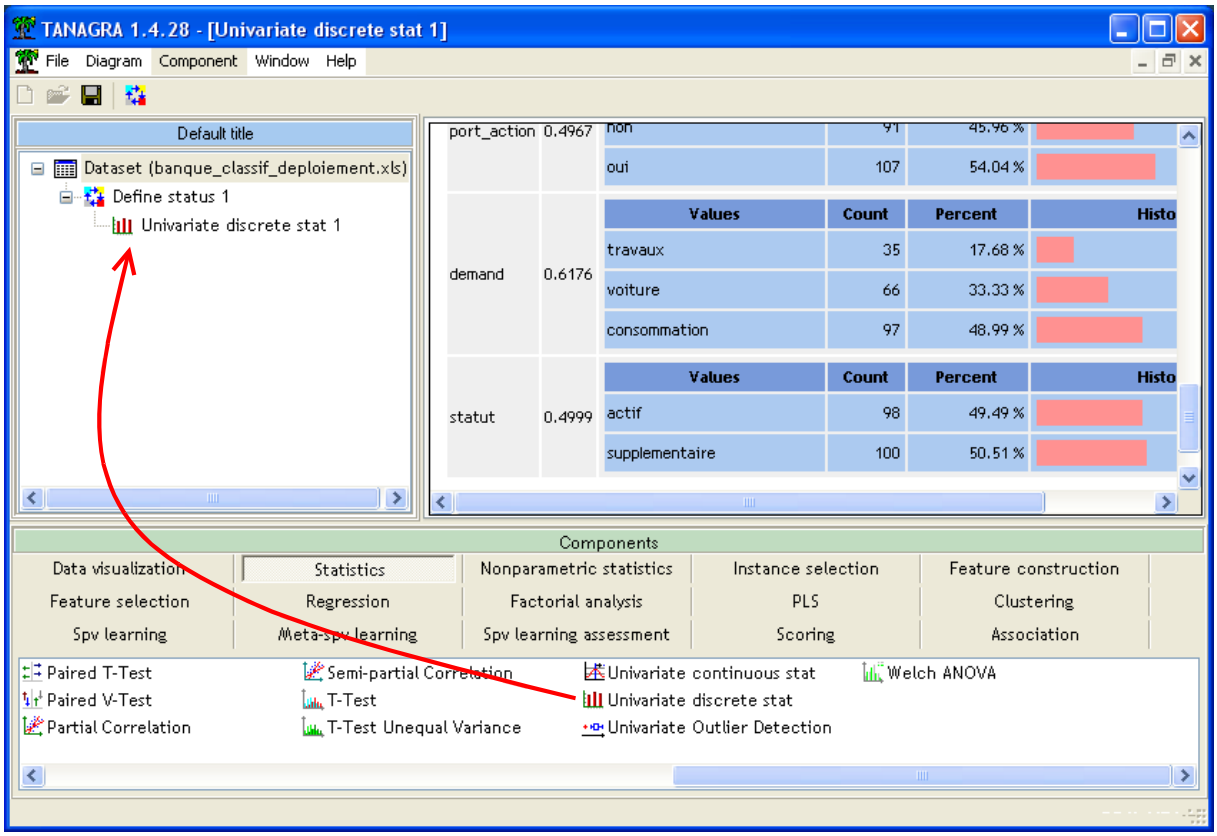
Nous insérons le composant DEFINE STATUS dans le diagramme. Nous plaçons la totalité des colonnes en INPUT.



Puis nous introduisons le composant UNIVARIATE DISCRETE STAT (onglet STATISTICS).

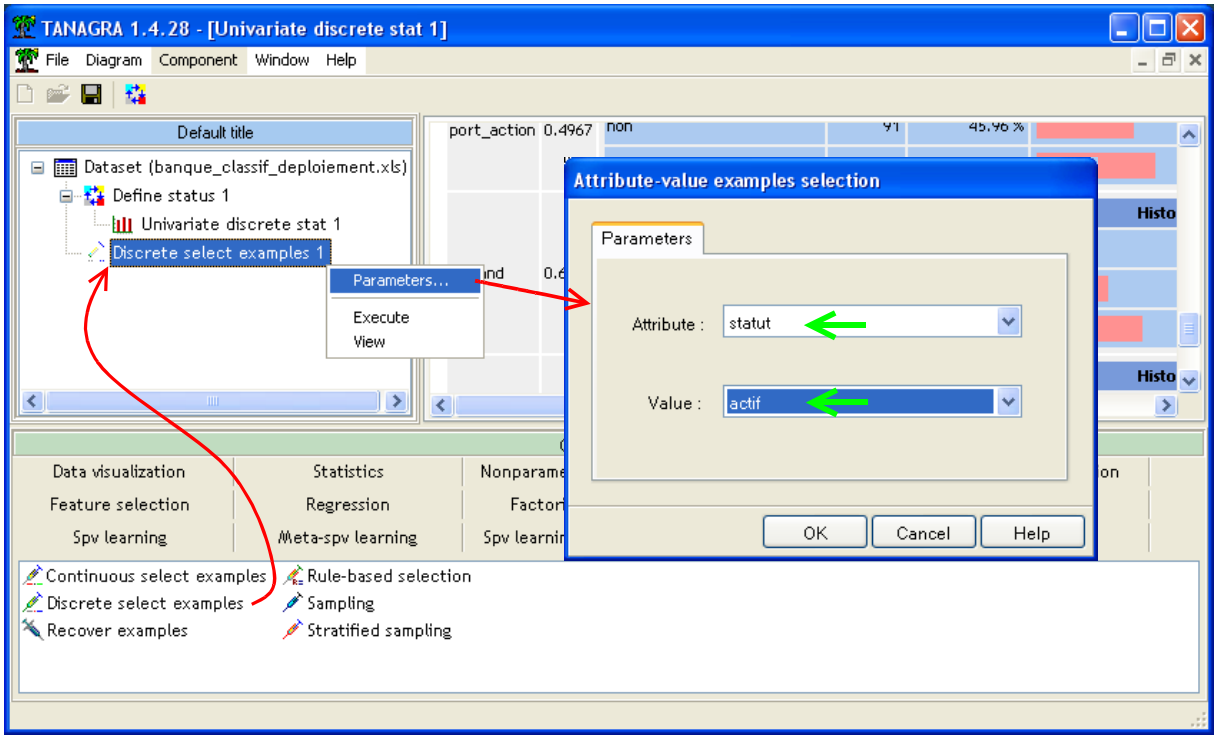
Nous retiendrons principalement qu'il y a 98 observations actives et 100 observations supplémentaires à classer dans le fichier. C'est ce que nous indique la distribution de la variable « statut ».

[2] Voir http://tutoriels-data-mining.blogspot.com/2008/03/importation-fichier-xls-excel-mode.html pour l'importation directe des fichiers Excel. Envoyer les données à partir d'Excel via une macro complémentaire est aussi possible : http://tutoriels-data-mining.blogspot.com/2008/03/importation-fichier-xls-excel-macro.html



3.3 Sélection des individus actifs

Dissocions maintenant les individus actifs des individus supplémentaires. Nous introduisons le composant DISCRETE SELECT EXAMPLES (onglet INSTANCE SELECTION). Nous actionnons le menu contextuel PARAMETERS. Nous le paramétrons de la manière suivante.

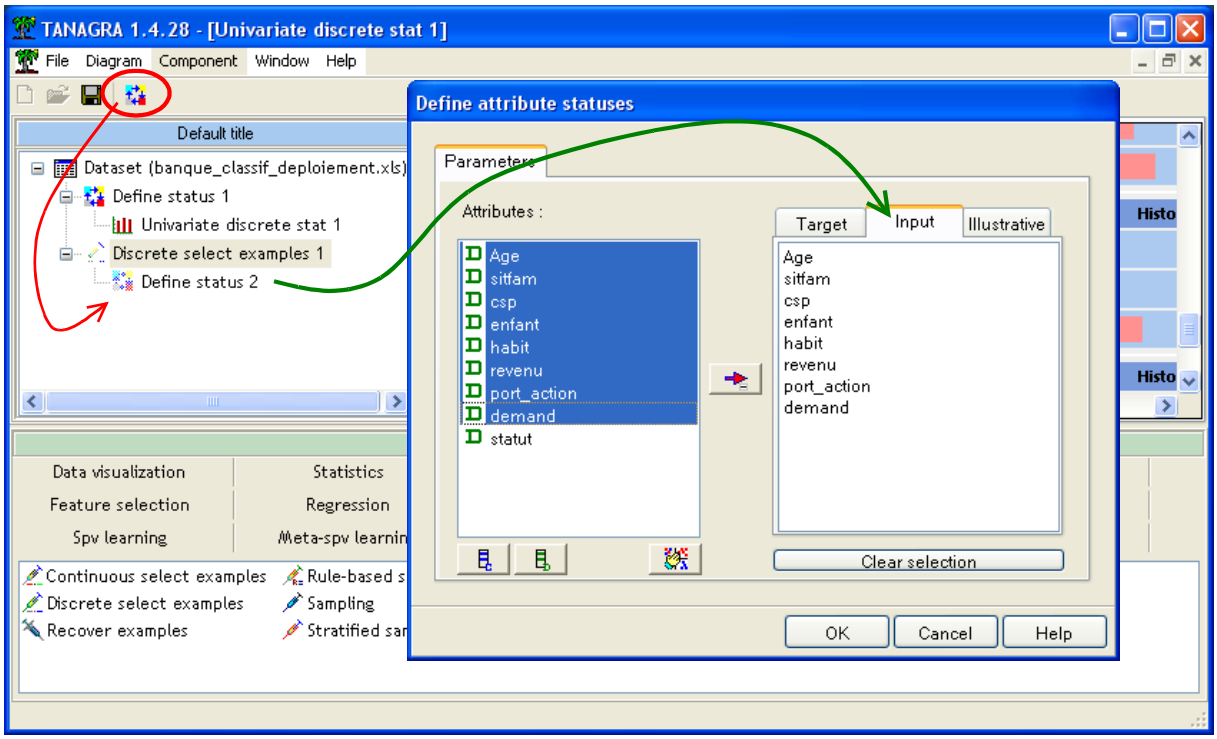


Après validation et activation du menu VIEW, Tanagra nous indique que, sauf contre ordre, 98 observations seront dorénavant sélectionnées pour les calculs dans cette branche du diagramme.

3.4 Analyse des correspondances multiples

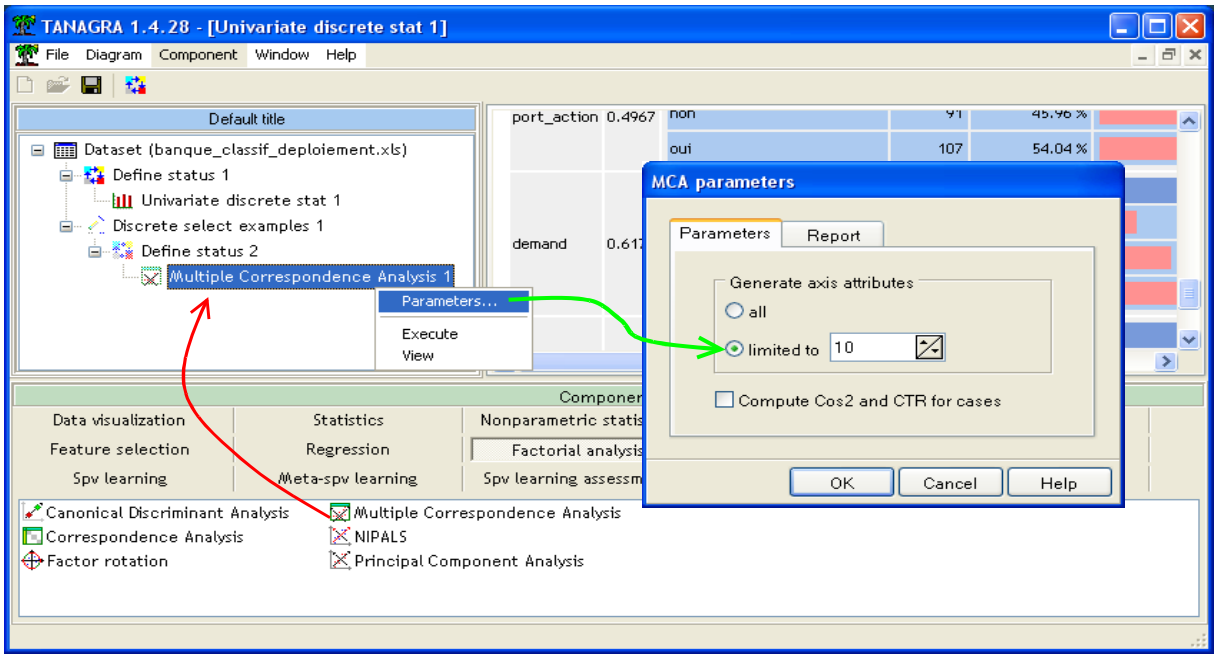
Il n'existe pas de technique sachant créer une typologie directement à partir de variables qualitatives dans Tanagra. La stratégie, très usitée, que nous mettons en place est la suivante : nous projetons les observations dans un nouvel espace à l'aide de l'analyse des correspondances multiples[3], nous pouvons dès lors réaliser une classification dans ce nouvel espace de représentation en utilisant la métrique euclidienne usuelle. Premier avantage pratique à cette approche, les calculs sont réalisables avec la grande majorité des logiciels, c'est déjà pas mal. Mais il y a aussi un avantage méthodologique. En ne présentant que les « q » premiers facteurs issus de l'analyse factorielle à l'algorithme de classification, nous ne retenons que l'information « essentielle », nous évacuons celles assimilables à du bruit. La qualité du partitionnement n'en sera que meilleure.

Nous introduisons un nouveau composant DEFINE STATUS, nous plaçons en INPUT les variables d'analyse (AGE...DEMAND).

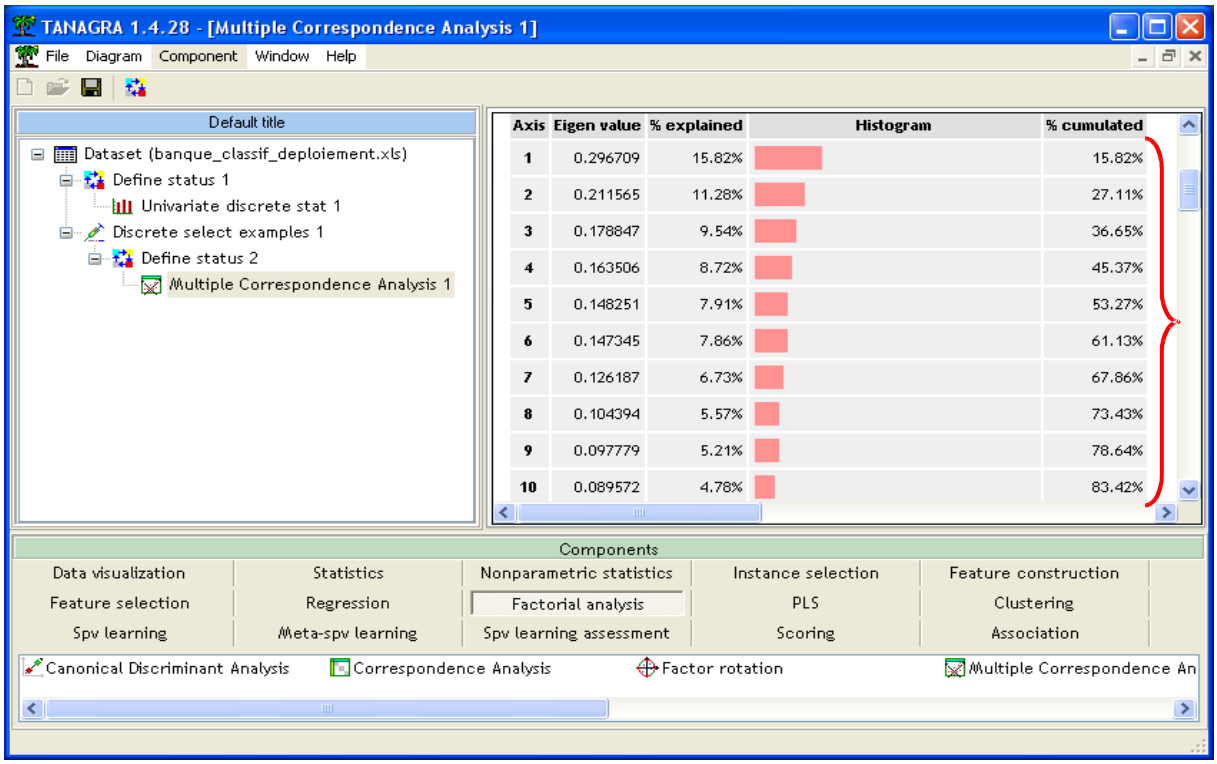


Nous insérons le composant MULTIPLE CORRESPONDANCE ANALYSIS (onglet FACTORIAL ANALYSIS). Nous le paramétrons (menu PARAMETERS) de manière à produire les 10 premiers axes factoriels.

[3] Dans notre configuration, l'AFCM correspond essentiellement à une phase de préparation de variables, Nous ne nous attarderons pas sur la lecture et l'interprétation des résultats. Pour le lecteur désireux d'approfondir ces questions, nous conseillons le tutoriel suivant, inspiré de l'ouvrage de M. Tenenhaus (2007) -- http://tutoriels-data-mining.blogspot.com/2008/03/afcm-races-canines.html



Nous obtenons les résultats en cliquant sur VIEW, les 10 premiers facteurs traduisent 83.42% de l'information disponible (pourcentage d'inertie).

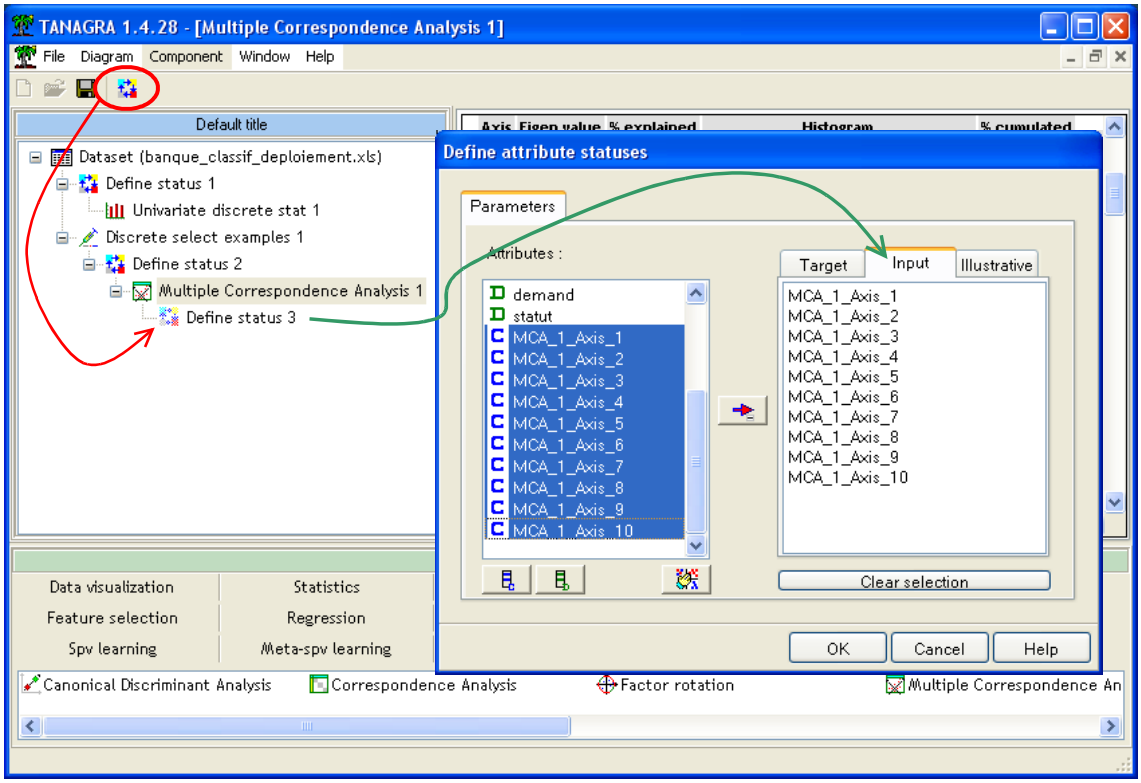


Deux éléments importants, que nous exploiterons par la suite, doivent retenir notre attention :

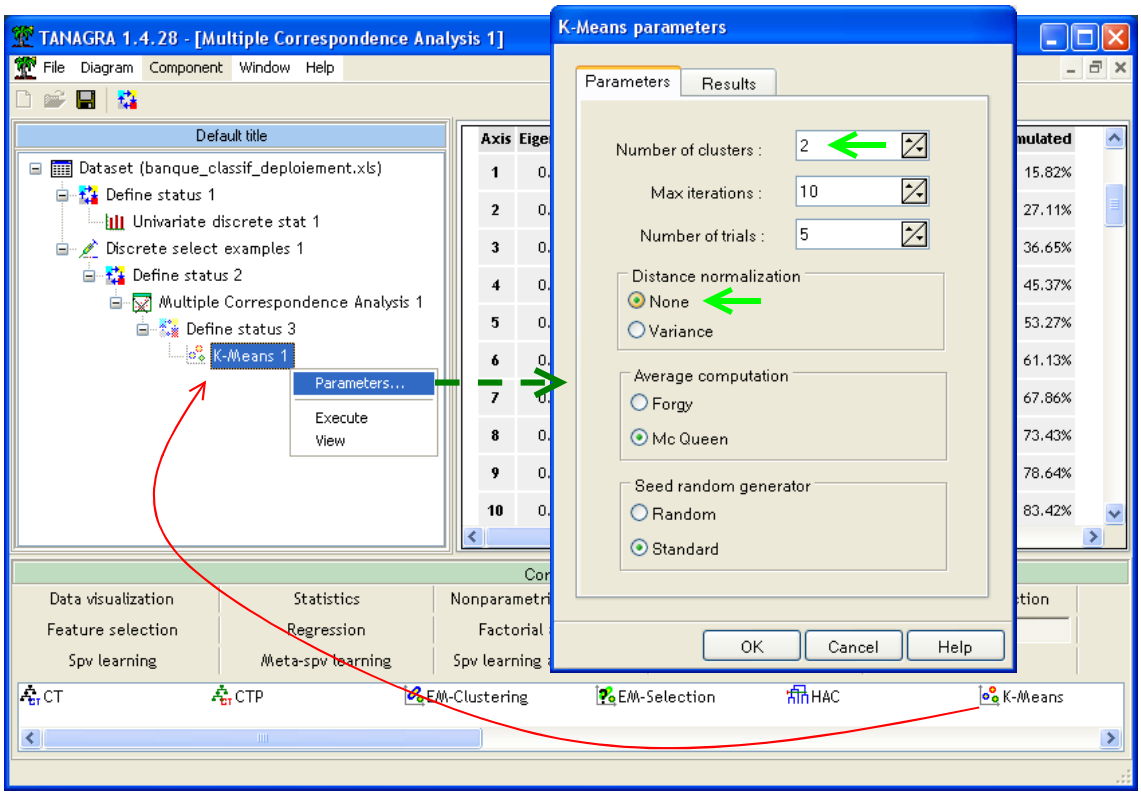
1. **Tanagra produit automatiquement 10 nouvelles variables**, elles seront disponibles en aval dans la branche du diagramme ;
2. **Ces nouvelles variables sont calculées sur la totalité du fichier**, elles couvrent à la fois les individus actifs et supplémentaires, avec une distinction forte : les individus actifs ont servi à élaborer les formules de projection, les individus illustratifs sont simplement projetés dans l'espace des axes factoriels.

3.5 K-Means sur les axes factoriels

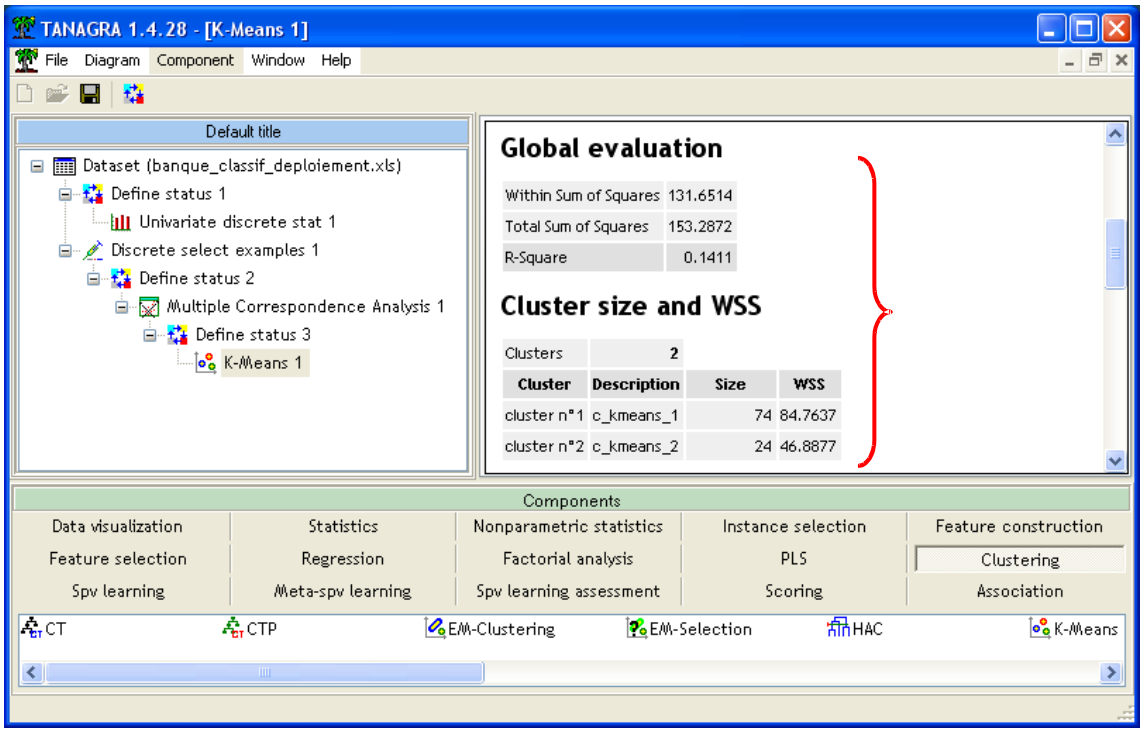
Nous souhaitons lancer la méthode des centres mobiles sur les axes factoriels. Nous insérons le composant DEFINE STATUS, nous plaçons en INPUT les facteurs MCA_1_AXIS_1 à MCA_1_AXIS_10.



Puis, nous insérons le composant K-Means (onglet CLUSTERING) : nous demandons une partition en 2 classes, aucune normalisation n'est effectuée c.-à-d. la distance euclidienne simple est utilisée.

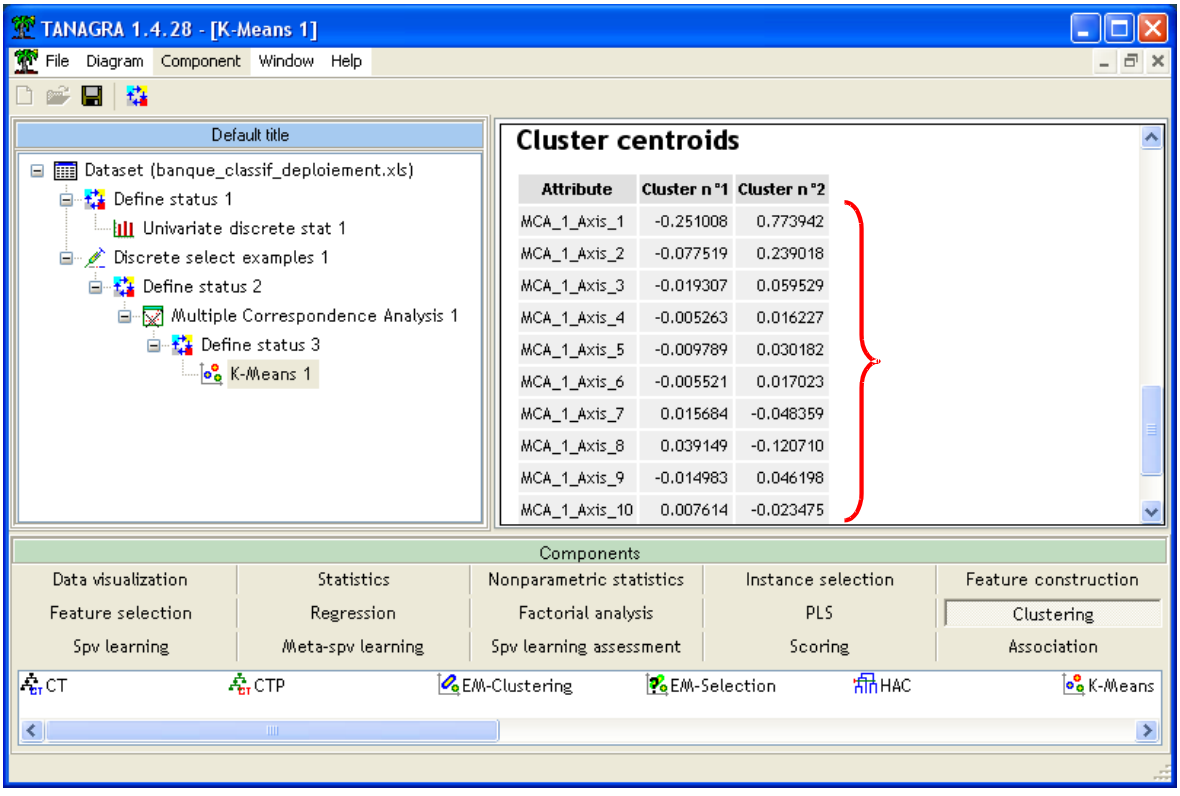


Nous actionnons le menu VIEW. Plusieurs informations arrivent.



Le premier groupe comporte 74 observations, la seconde 24. L'inertie intra classes est égale à 131.6514 (84.7637 + 46.8877). Nous retiendrons ces résultats pour les comparer avec ceux de R.

Dans la partie basse de la fenêtre, nous disposons des centres de classes calculés sur l'espace des axes factoriels, utilisés lors de la construction de la typologie. Ils donnent une idée du positionnement des groupes dans l'espace des facteurs. Mais ils ne sont pas très utilisables pour l'interprétation des groupes.



Ici également, nous devons retenir une autre information très importante : si **Tanagra** construit effectivement la typologie à partir des individus actifs, il **a affecté automatiquement chaque individu supplémentaire à un des groupes en se basant sur la distance aux centres de classes**. De fait, une nouvelle variable est créée, disponible en aval du composant, il s'agit d'une variable indicatrice de l'appartenance au groupes pour la totalité des individus.

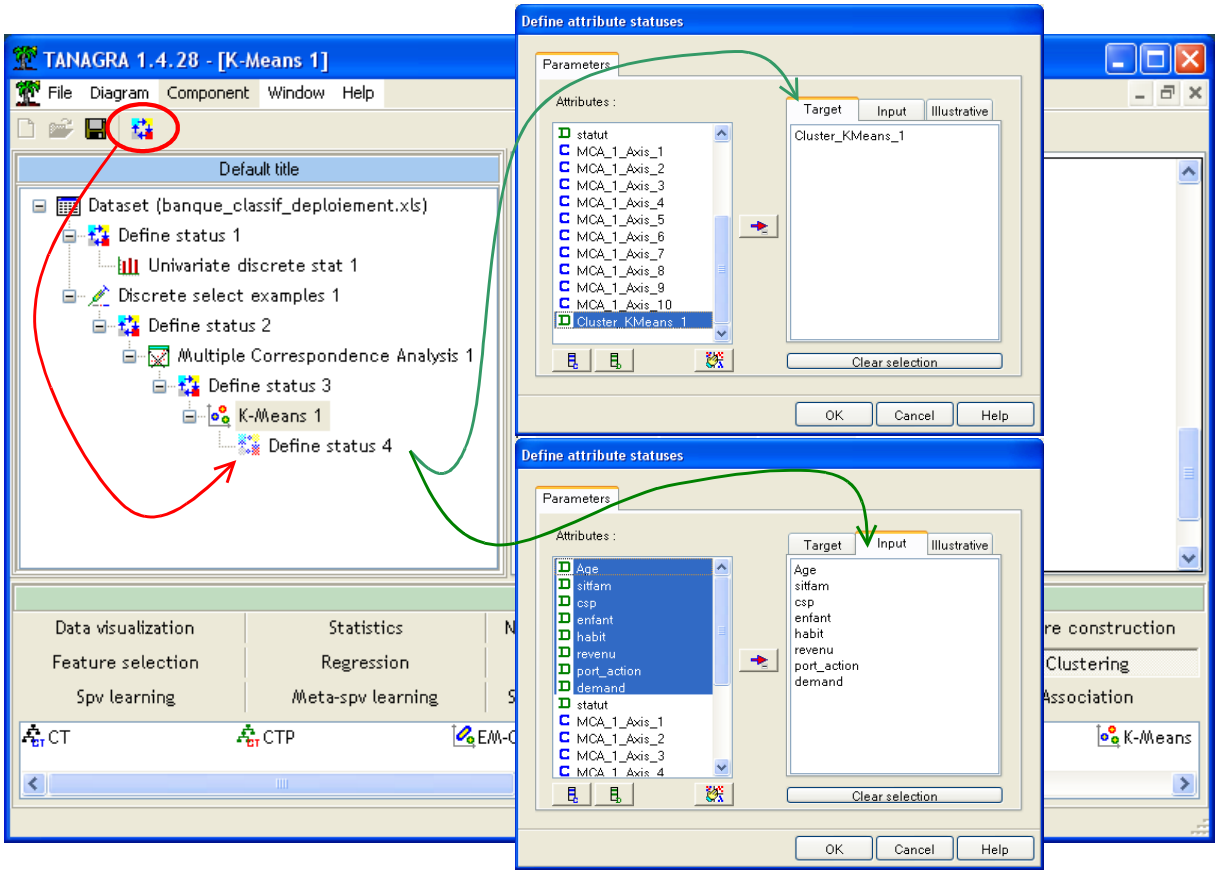
Nous exploitons cette nouvelle variable pour interpréter les classes issues de la typologie.

3.6 Interprétation des groupes

3.6.1 Statistiques descriptives comparatives

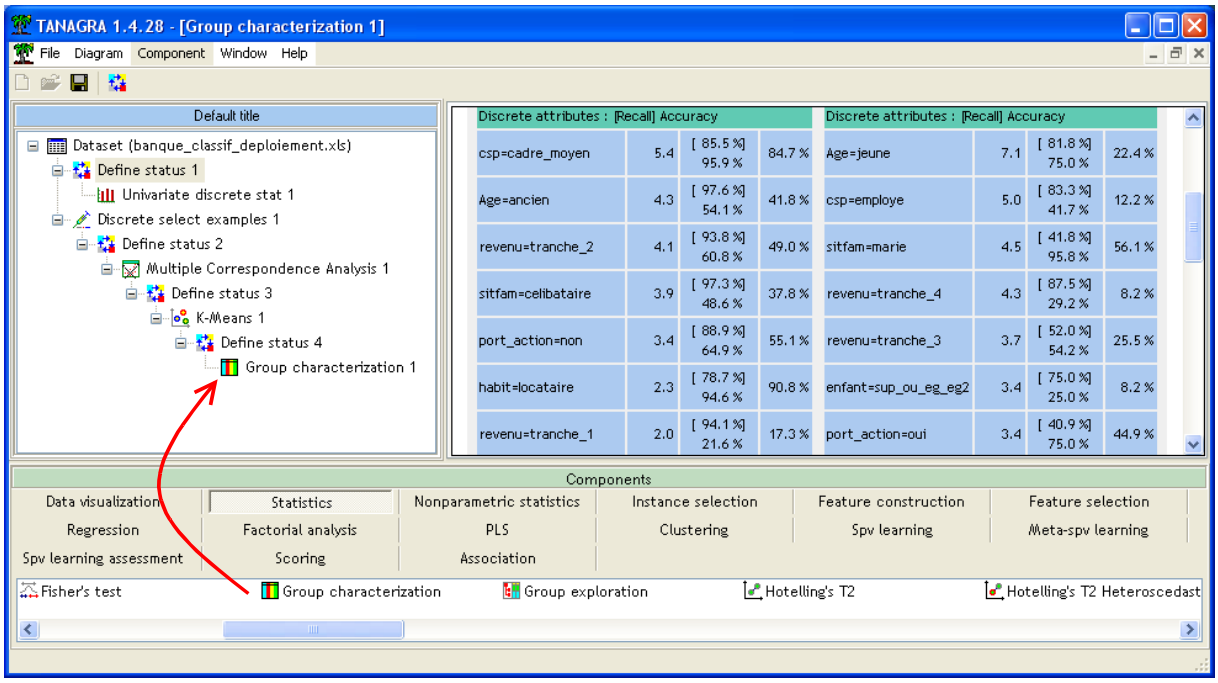
Les statistiques descriptives conditionnelles donnent d'excellentes indications sur la teneur des classes. Concernant les variables catégorielles, il s'agit principalement de comparer la fréquence des modalités dans chaque classe.

Nous introduisons le composant DEFINE STATUS dans le diagramme. Nous plaçons en TARGET la variable indicatrice des classes CLUSTER_KMEANS_1, en INPUT les descripteurs (AGE...DEMAND).



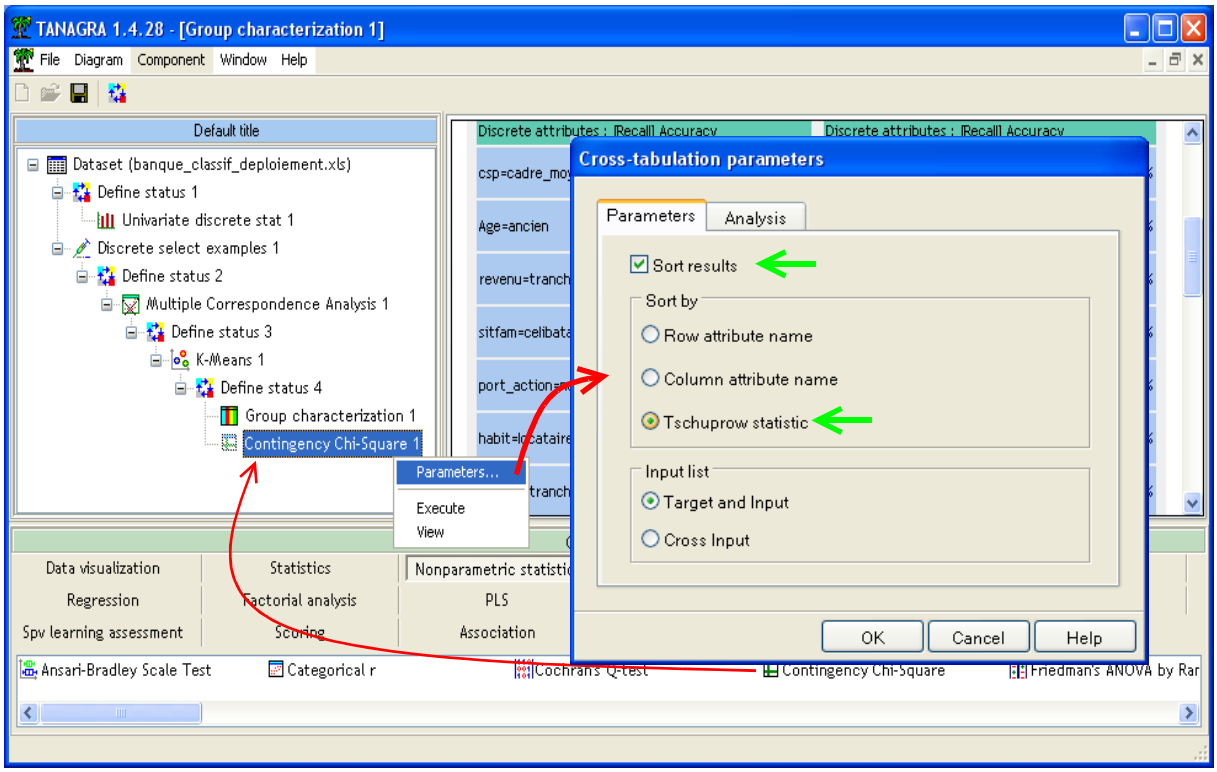
Nous introduisons ensuite le composant GROUP CHARACTERIZATION (onglet STATISTICS). Nous cliquons sur VIEW.

Le groupe des 74 individus semble principalement lié aux cadres moyens, un peu plus âgés que la moyenne, à haut revenu (tranche 1 et 2) et célibataire. Le second groupe des 24 individus semble plutôt correspondre aux jeunes employés, mariés, à faible revenu (tranche 3 et 4).

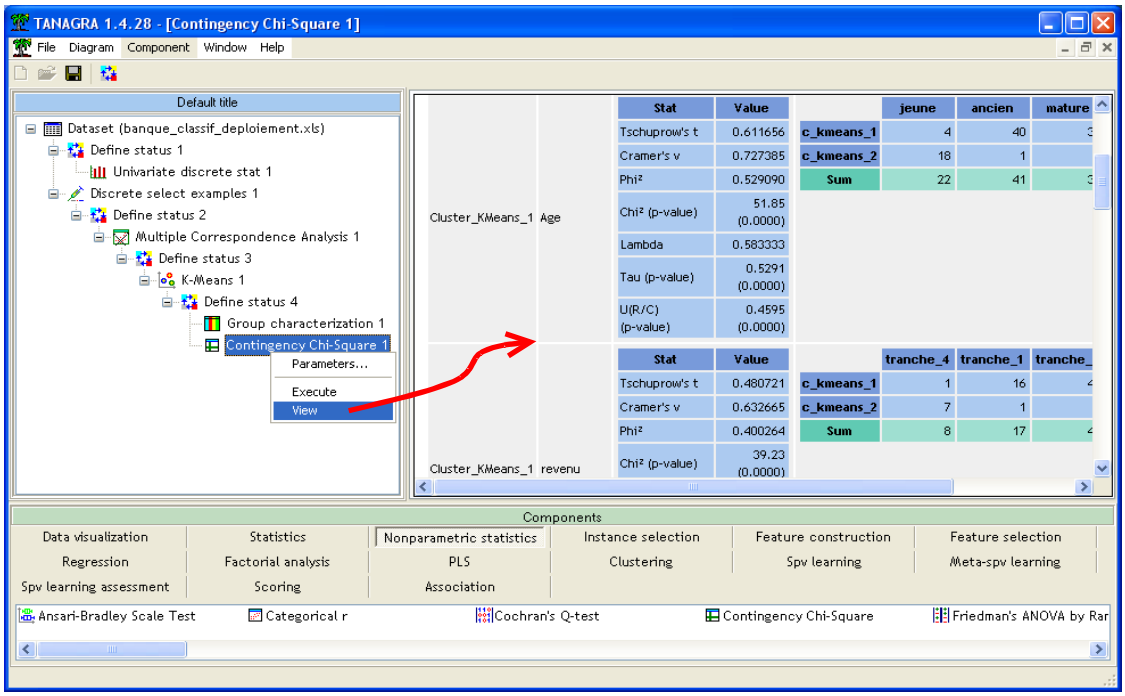


3.6.2 Croisement cluster et variables catégorielles actives

Une autre manière de comprendre les classes est de les croiser avec les variables actives. Nous introduisons le composant CONTINGENCY CHI-SQUARE (onglet NONPARAMETRIC STATISTICS). Nous le paramétrons pour que les liaisons les plus significatives apparaissent en premier, cela revient à trier les résultats selon le t de Tschuprow décroissant.



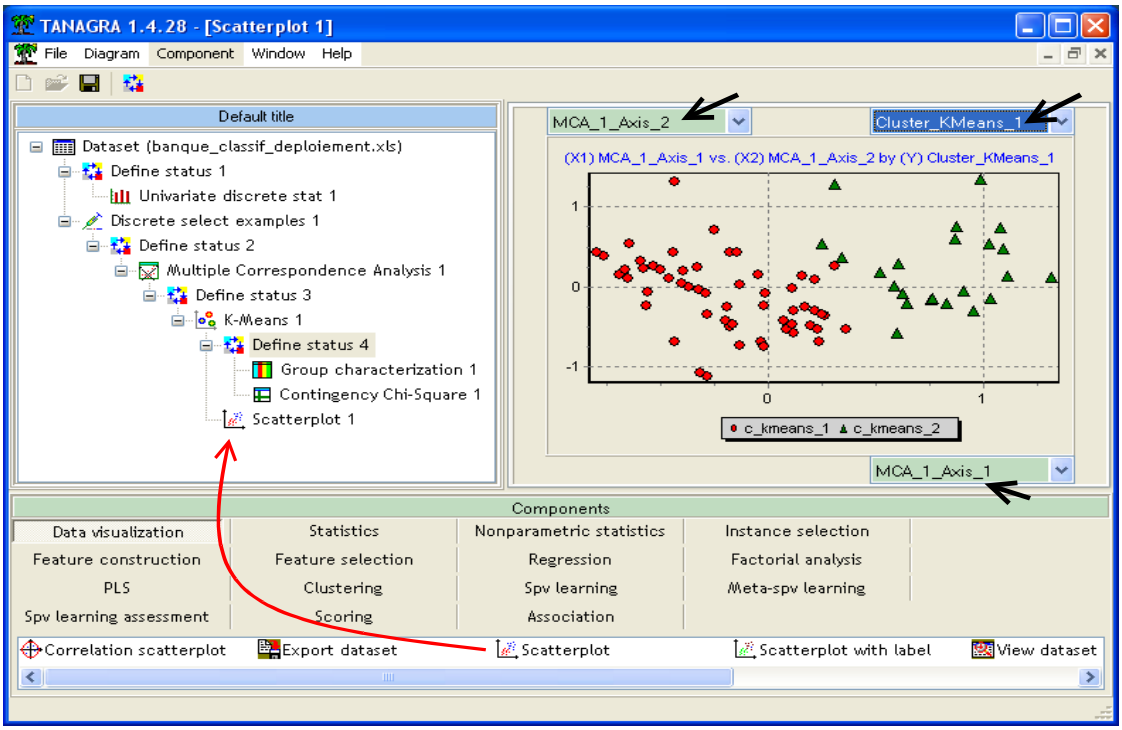
Nous actionnons VIEW.



Nous constatons que l'âge est la variable la plus liée aux clusters : le groupe 1 correspond aux anciens et matures, le groupe 2 aux jeunes. Les variables suivantes sont le revenu, la CSP, la situation familiale, etc. Bien évidemment, les résultats présentés ici ne peuvent pas être en contradiction avec ceux du composant GROUP CHARACTERIZATION. Mais, au lieu d'une analyse par modalité, nous avons une analyse par variable.

3.6.3 Graphique « Nuage de points »

Enfin, dernier outil pour interpréter les groupes, nous utilisons la projection des observations dans les plans factoriels. Cela nous impose de savoir interpréter correctement les axes, ce n'est pas toujours évident. Mais lorsque nous y parvenons, les résultats sont souvent très intéressants.



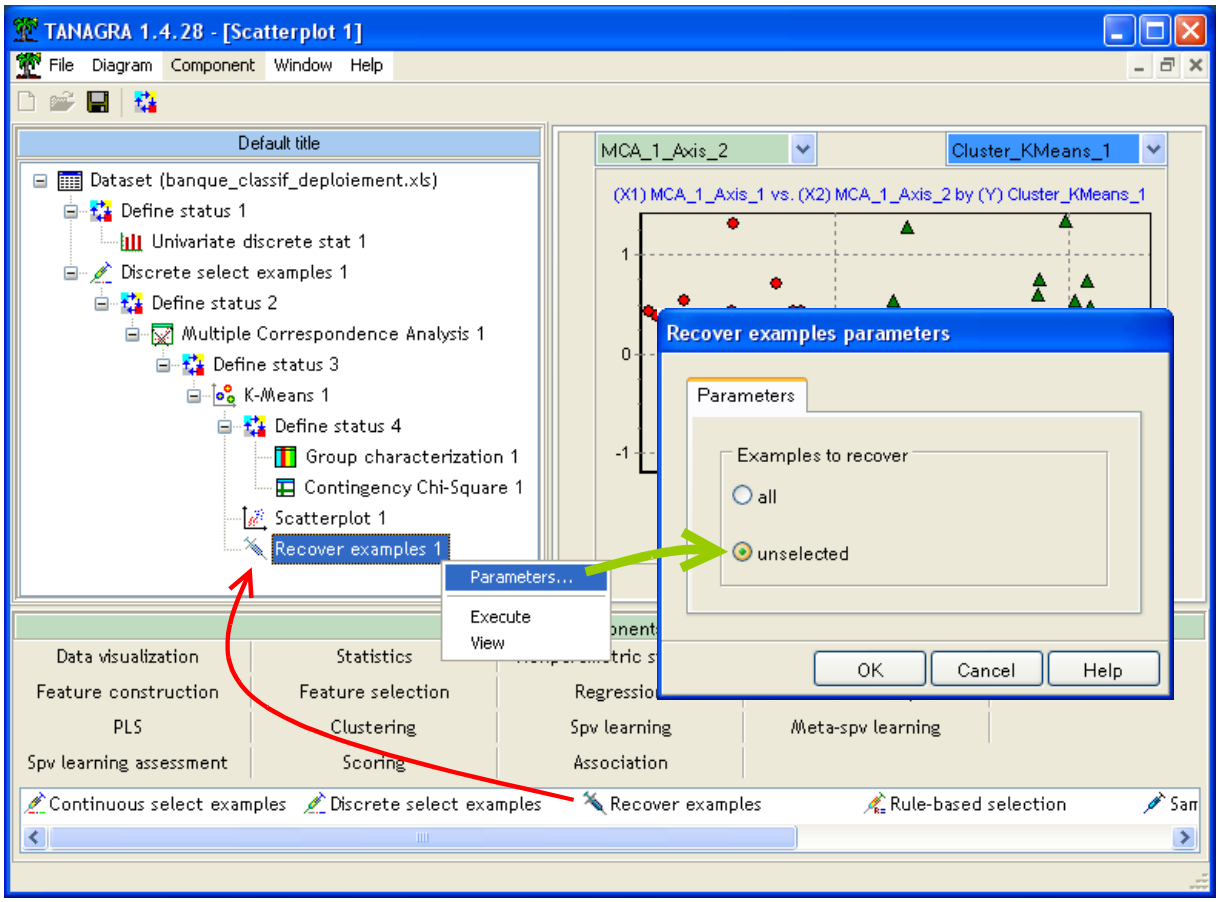
Nous introduisons le composant SCATTERPLOT (nuage de points, onglet DATA VISUALIZATION), nous mettons en abscisse le premier axe MCA_1_AXIS_1, en ordonnée le second MCA_1_AXIS_2, puis nous illustrons les points selon la classe d'appartenance CLUSTER_KMEANS_1. Nous constatons que les groupes sont quasiment parfaitement discernables sur le premier axe factoriel.

En revenant sur l'analyse factorielle, nous constaterons que le premier axe (colonne des contributions) est principalement défini par l'âge, la situation familiale et le revenu.

Toutes ces analyses sont autant de points de vue sur un seul et même résultat. Le fait qu'elles convergent ne peut que raffermir les interprétations.

3.7 Récupération des observations « à classer »

Tout au long du processus ci-dessus, Tanagra a réalisé en sous-main les opérations adéquates sur les individus supplémentaires : projection sur les axes factoriels, affectation des individus aux groupes. Il est temps maintenant de mettre en avant ces individus pour voir ce qu'il en est. Le composant RECOVER EXAMPLES (onglet INSTANCE SELECTION) permet de récupérer les individus non sélectionnés. Nous l'insérons dans le diagramme, nous actionnons le menu contextuel PARAMETERS. Nous constatons que nous pouvons mettre au premier plan, soit la totalité des individus, soit les individus non sélectionnés précédemment, ils correspondent aux individus supplémentaires pour nous. Nous choisissons cette seconde option.

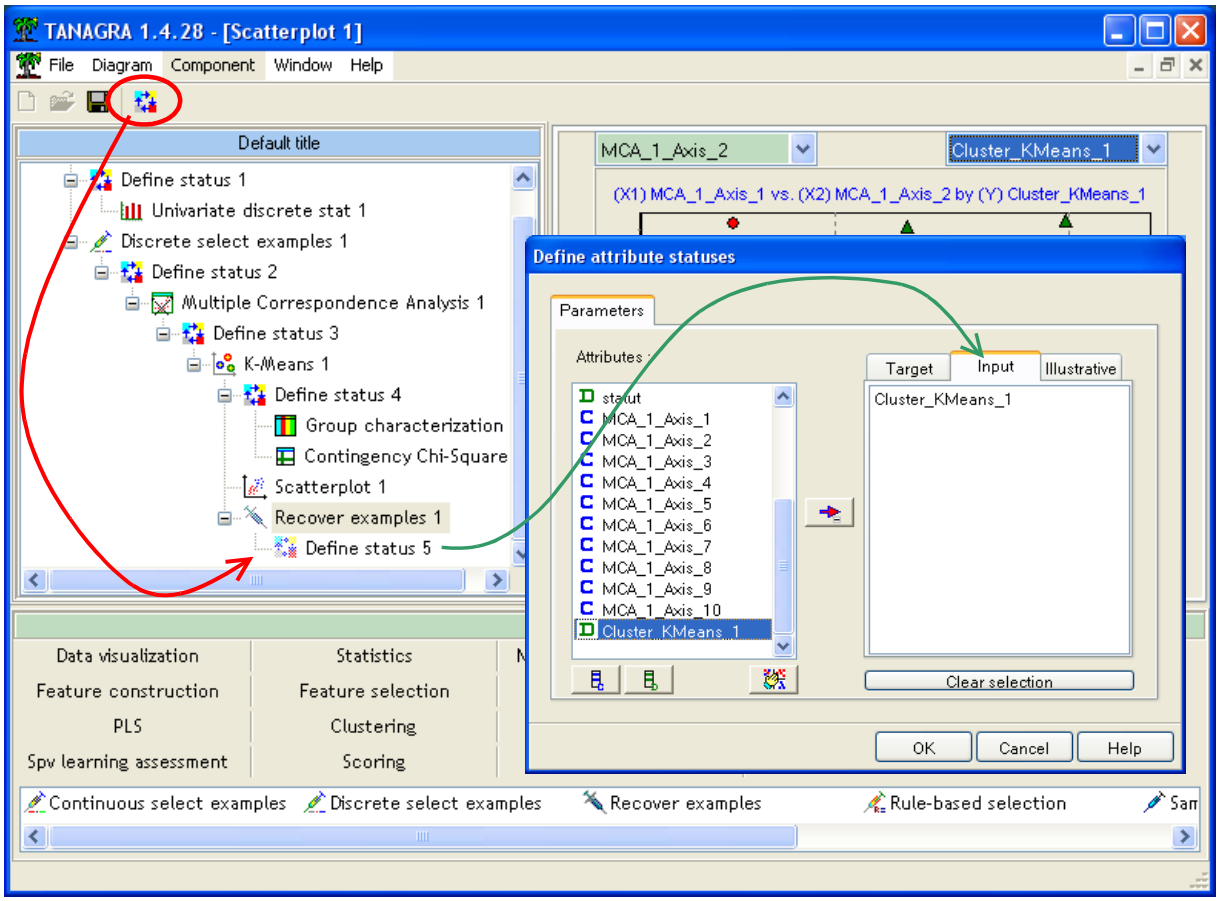


Un clic sur VIEW nous annonce que 100 individus sont maintenant sélectionnés, soit les 100 individus étiquetés « supplémentaires » dans le fichier initial.

3.8 Distribution des classes chez les individus supplémentaires

Une première vérification consiste à calculer la distribution de fréquence des classes chez les individus supplémentaires. Dans notre analyse, la partition « actif » - « supplémentaire » a été réalisée au hasard, on s'attend à ce que la proportion des groupes soit à peu près identique à celle de l'apprentissage[4] qui était, rappelons-le, 74 individus pour le groupe 1, 24 pour le groupe 2.

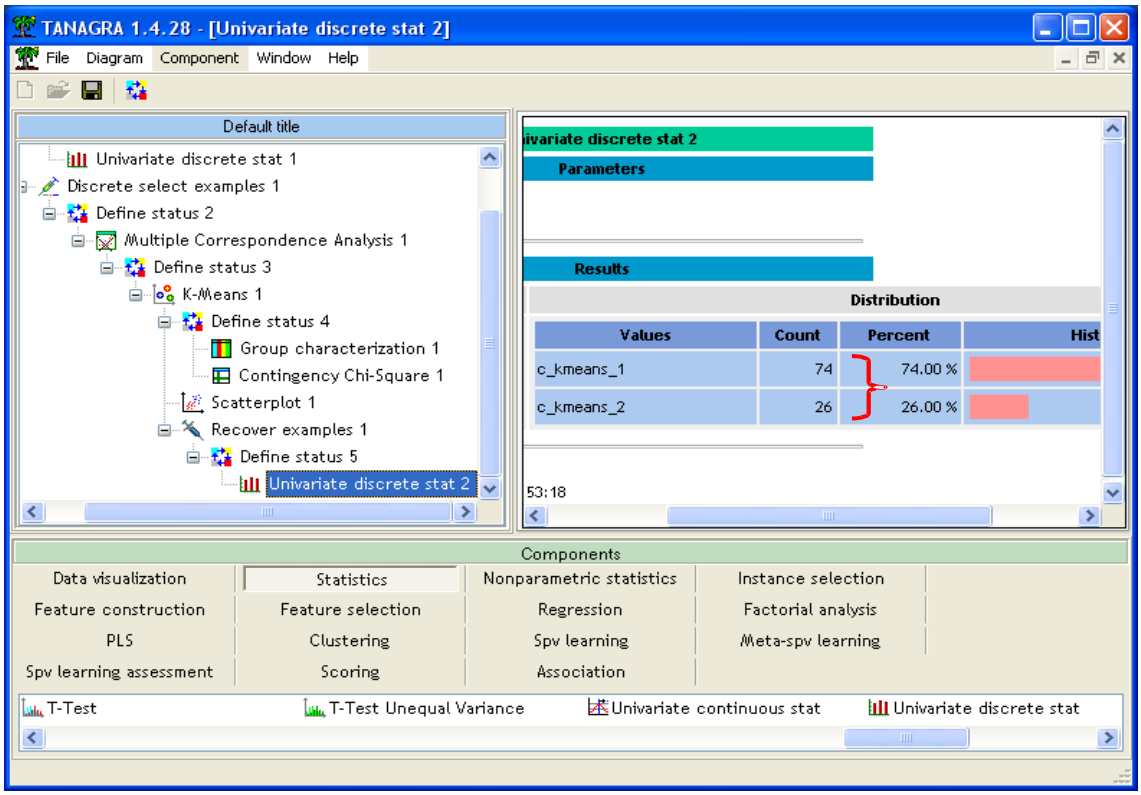
Nous insérons le composant DEFINE STATUS, nous plaçons en INPUT la variable indicatrice des clusters (CLUSTER_KMEANS_1).



Nous introduisons le composant UNIVARIATE DISCRETE STAT (onglet STATISTICS).

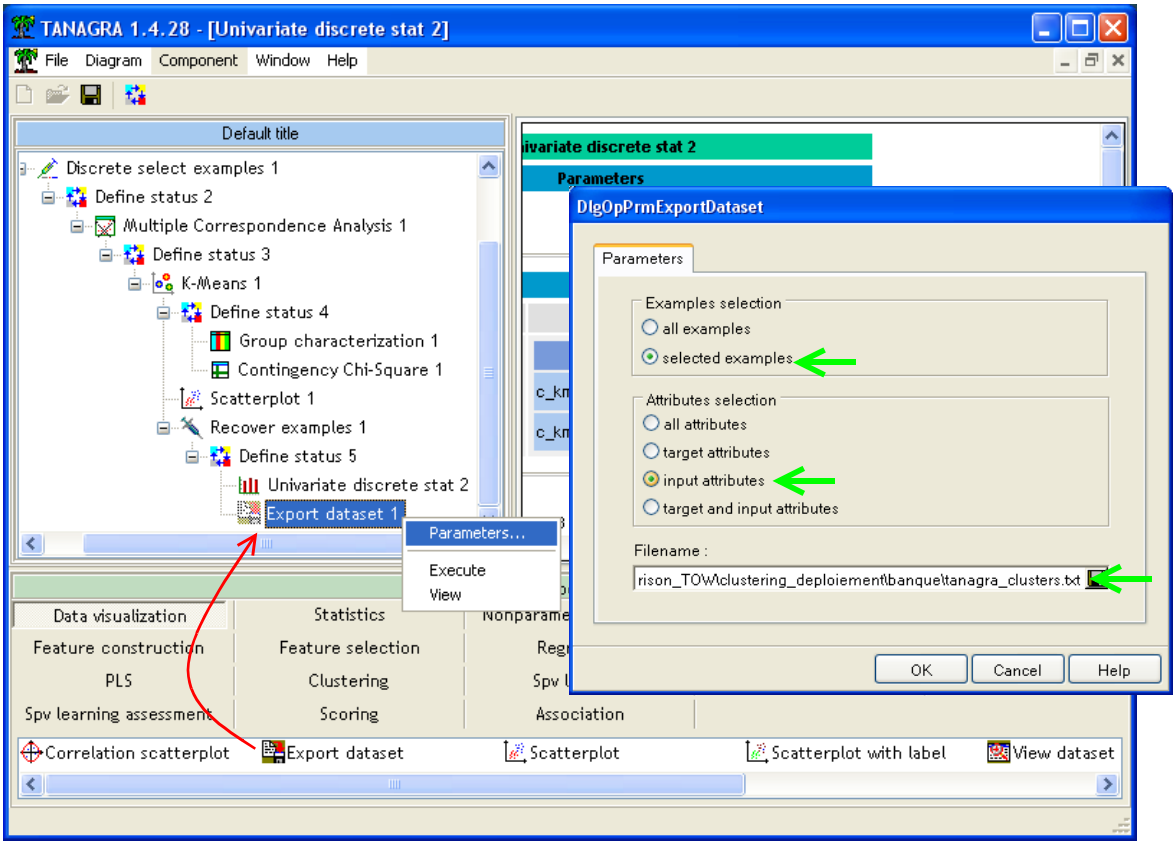
Les proportions sont à peu près similaires, 74 observations correspondent au groupe 1, 26 au second.

[4] Dans le cas où les individus supplémentaires ne correspondent pas à un tirage au hasard, mais plutôt à une sous population particulière identifiée (les clients d'une banque située dans une région reculée par exemple), ce raisonnement n'a pas lieu d'être. Il se peut que la totalité de ces individus correspondent essentiellement à l'un des groupes.



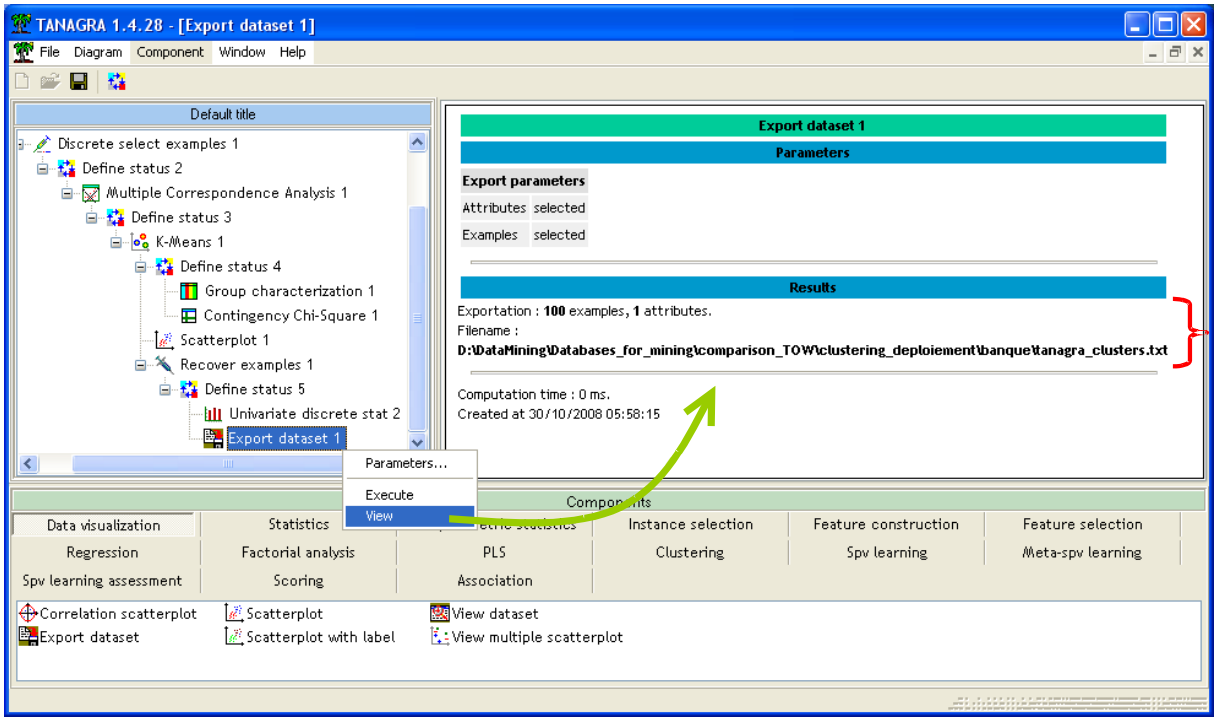
3.9 Exportation des classes des individus « à classer »

Il nous reste à exporter cette nouvelle colonne pour les individus supplémentaires. L'idée est de confronter les affectations avec ceux de R dans la suite de didacticiel. Nous utilisons pour cela le composant EXPORT DATASET (onglet DATA VISUALIZATION).



Nous le paramétrons (menu PARAMETERS) de manière à ce que : seuls les individus sélectionnés soient exportés (les 100 individus supplémentaires pour nous) et la variable INPUT (dans le DEFINE STATUS directement en AMONT) soient exportés. Nous pouvons spécifier le nom du fichier, nous choisissons TANAGRA_CLUSTERS.TXT.

Après exécution (clic sur VIEW), le composant nous indique qu'un fichier de 100 observations avec 1 variable a bien été produit.



4 K-Means et déploiement avec R

L'objectif est de reproduire exactement la démarche précédente avec R, puis de confronter les résultats. Corollaire à cela, en détaillant les opérations dans R, nous donnons une meilleure visibilité sur les calculs réalisés en interne par Tanagra.

Importation des données. Le package **xlsReadWrite** permet de manipuler directement les fichiers au format XLS. Après avoir chargé la librairie avec la commande **library(.)**, l'instruction **read.xls(.)** importe les données.

```
#charger les données
library(xlsReadWrite)
setwd("D:/DataMining/Databases_for_mining/comparison_TOW/clustering_deploiement/banque")
donnees <- read.xls(file="banque_classif_deploiement.xls",sheet=1)
summary(donnees)
```

La commande **summary(.)** calcule la distribution de fréquences pour chaque colonne. On retiendra surtout qu'il y a bien 98 individus actifs et 100 individus supplémentaires dans ce fichier.

```
> summary(donnees)
     Age              sitfam              csp                 enfant
 ancien:74   celibataire: 62   cadre_moyen:171   inf_a_2       : 73
 jeune :42   marie       :111   employe     : 23   sup_ou_eg_eg2: 18
 mature:82   separe      : 25   retraite    :  4   zero          :107

       habit           revenu   port_action          demand
 locataire:181   tranche_1:38   non: 91     consommation:97
 proprio  : 17   tranche_2:94   oui:107     travaux      :35
                 tranche_3:46               voiture      :66
                 tranche_4:20
            statut
 actif         : 98
 supplementaire:100
```

Scission individus actifs et supplémentaires. Plutôt que de scinder le data.frame en 2 parties, opération usuelle avec R, nous allons créer deux vecteurs indiquant le numéro d'observations dans chaque sous-ensemble de données.

```
#index des individus actifs et illustratifs
id.actif <- which(donnees$statut=="actif")
id.illus <- which(donnees$statut=="supplementaire")
print(id.actif)
print(id.illus)
```

Nous obtenons ainsi

```
> print(id.actif)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
[23] 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
[45] 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
[67] 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88
[89] 89 90 91 92 93 94 95 96 97 98
> print(id.illus)
  [1]  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114
 [17] 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130
 [33] 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146
 [49] 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
 [65] 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178
 [81] 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194
 [97] 195 196 197 198
```

Ce qui est conforme à l'organisation de notre fichier, les 98 premières observations correspondent aux individus actifs, les autres constituent les individus supplémentaires.

Analyse des correspondances multiples. Nous utilisons la procédure MCA du package FACTOMINER (http://factominer.free.fr/) pour réaliser l'analyse des correspondances multiples. Elle fonctionne de manière inhabituelle pour R. Plutôt que de surcharger la méthode **predict(.)** pour projeter les individus supplémentaires dans le repère factoriel, MCA préfère que nous lui envoyons la totalité des données accompagnée de deux vecteurs d'index, l'une correspondant aux numéros des individus actifs, la seconde aux individus supplémentaires. D'où l'utilisation de la fonction **which(.)** ci dessus pour élaborer ces vecteurs. Ce n'est pas très usuel pour R, mais une fois que l'on a compris, c'est finalement très pratique, c'est ce qui importe.

Les coordonnées factorielles sont disponibles dans l'objet modèle : **indcoord** (resp. **$ind.sup$coord**) décrit les coordonnées des individus actifs (resp. supplémentaires). Tout comme dans Tanagra, nous avions demandé 10 axes.

```
#chargement de la librairie FactoMineR
library(FactoMineR)

#acm, avec partition individus actifs et supplémentaires
#la colonne statut n'est pas utilisée
#les nb.axes premiers axes sont demandés
nb.axes <- 10
X <- donnees[,-9]
modele.acm <- MCA(X,ncp=nb.axes,ind.sup=id.illus,graph=FALSE)
```

Nous pouvons par exemple récupérer, et comparer aces ceux de Tanagra, les valeurs propres et le pourcentage d'inertie expliquée par les axes.

```
> print(modele.acm$eig)
       eigenvalue percentage of variance cumulative percentage of variance
dim 1  2.967092e-01           1.582449e+01                          15.82449
dim 2  2.115647e-01           1.128345e+01                          27.10794
dim 3  1.788475e-01           9.538532e+00                          36.64647
dim 4  1.635057e-01           8.720303e+00                          45.36678
dim 5  1.482514e-01           7.906743e+00                          53.27352
dim 6  1.473455e-01           7.858426e+00                          61.13194
dim 7  1.261867e-01           6.729958e+00                          67.86190
dim 8  1.043939e-01           5.567677e+00                          73.42958
dim 9  9.777875e-02           5.214866e+00                          78.64445
dim 10 8.957172e-02           4.777158e+00                          83.42160
dim 11 8.770947e-02           4.677838e+00                          88.09944
dim 12 7.725772e-02           4.120412e+00                          92.21985
dim 13 5.744899e-02           3.063946e+00                          95.28380
dim 14 5.055581e-02           2.696310e+00                          97.98011
dim 15 3.787292e-02           2.019889e+00                         100.00000
dim 16 4.583556e-32           2.444563e-30                         100.00000
dim 17 5.319818e-33           2.837236e-31                         100.00000
dim 18 3.955367e-33           2.109529e-31                         100.00000
dim 19 3.822916e-33           2.038889e-31                         100.00000
dim 20 1.572476e-33           8.386540e-32                         100.00000
dim 21 1.228609e-33           6.552579e-32                         100.00000
dim 22 1.025267e-33           5.468090e-32                         100.00000
dim 23 5.070372e-34           2.704198e-32                         100.00000
```

Les résultats sont identiques (heureusement, le calcul est déterministe). R affiche toutes les valeurs propres, y compris celles qui sont nulles.

K-Means sur les axes factoriels. L'étape suivante consiste à lancer l'algorithme des K-Means sur les individus actifs, dans le nouvel espace décrit par les 10 premiers axes factoriels. Nous demandons 2 classes.

```
#k-means sur les axes de l'ACM (individus actifs)
nb.classes <- 2
set.seed(10)
modele.kmeans <- kmeans(modele.acm$ind$coord,centers = nb.classes,algorithm="MacQueen",iter.max=40)
print(modele.kmeans)
```

R nous fournit.

```
K-means clustering with 2 clusters of sizes 24, 74

Cluster means:
       Dim 1       Dim 2       Dim 3       Dim 4       Dim 5       Dim 6
1  0.7739419  0.23901805 -0.05952894 -0.016227008  0.03018225 -0.017023200
2 -0.2510082 -0.07751937  0.01930668  0.005262814 -0.00978884  0.005521038
        Dim 7       Dim 8       Dim 9       Dim 10
1  0.04835855 -0.12070994  0.04619789 -0.023475007
2 -0.01568386  0.03914917 -0.01498310  0.007613516

Clustering vector:
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 1  2  2  1  2  2  2  2  2  2  2  2  2  2  2  2  2  1  2  2  2  1  1  2  2
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
 2  2  2  2  2  2  2  2  1  2  2  1  2  2  1  2  2  2  2  2  2  1  1  1  2  1
53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
 1  2  2  2  2  2  2  2  1  2  1  1  2  2  2  2  1  2  2  1  1  2  2  2  2  1
79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
 2  2  1  2  2  2  2  2  2  2  2  2  1  2  2  1  2  2  1  2

Within cluster sum of squares by cluster:
[1] 46.88773 84.76367

Available components:
[1] "cluster"  "centers"  "withinss" "size"
```

Nous obtenons deux groupes avec respectivement 74 et 24 observations. Nous suspectons avoir obtenu la même partition que Tanagra, en effet l'inertie intra classes est identique, soit

```
> sum(modele.kmeans$withinss)
[1] 131.6514
```

Nous pouvons également comparer les centres de classes décrites dans le repère factoriel.

```
#récupération des centres de classes dans l'espace des axes factoriels
centres <- modele.kmeans$centers
numero <- seq(from=1,to=nb.classes)
rownames(centres) <- paste("clus_",numero,sep="")
print(centres)
```

Nous obtenons les valeurs ci-dessous. Nous pouvons dire à ce stade que le cluster n°1 (resp. n°2) de Tanagra correspond exactement à clus_2 (resp. clus_1) de R.

```
> print(centres)
            Dim 1       Dim 2       Dim 3       Dim 4       Dim 5
clus_1  0.7739419  0.23901805 -0.05952894 -0.016227008  0.03018225
clus_2 -0.2510082 -0.07751937  0.01930668  0.005262814 -0.00978884
              Dim 6       Dim 7       Dim 8       Dim 9       Dim 10
clus_1 -0.017023200  0.04835855 -0.12070994  0.04619789 -0.023475007
clus_2  0.005521038 -0.01568386  0.03914917 -0.01498310  0.007613516
```

Déploiement. Nous devons définir plusieurs fonctions pour affecter les classes aux individus supplémentaires. Tout d'abord, nous définissons la fonction **dist_euclidienne(.)** qui calcule la distance euclidienne entre un centre de référence et un individu à classer.

```
#fonction distance euclidienne entre deux lignes
dist_euclidienne <- function(ref,a_classer){
 dist <- sum((ref-a_classer)^2)
 return(dist)
}
```

Ensuite, toujours pour un individu à classer, nous appliquons cette fonction à l'ensemble des centres de classes, nous détectons l'indice de celui qui est le plus proche. C'est le rôle de la fonction **le_plus_proche(.)**.

```
#pour une observation, calculer le min. de distance, et renvoyer l'indice
le_plus_proche <- function(barycentres=centres,a_classer){
  vec <- apply(barycentres,1,dist_euclidienne,a_classer)
  #renvoyer l'indice
  indice <- which.min(vec)
  return(indice)
}
```

Enfin, dernière étape, nous devons réitérer l'opération pour chaque individu supplémentaire, en utilisant leur description dans l'espace factoriel. Nous obtenons en sortie un vecteur indiquant la classe d'appartenance de chaque individu supplémentaire[5].

```
#enchaîner tout cela pour chaque individu supplémentaire
resultat <- apply(modele.acm$ind.sup$coord,1,le_plus_proche,barycentres=centres)
resultat <- as.factor(resultat)
print(summary(resultat))
```

La distribution des classes sur les individus supplémentaires est identique à celle de Tanagra.

```
> print(summary(resultat))
 1  2
26 74
```

Confrontation des classes affectées en déploiement, Tanagra et R. Les distributions sont les mêmes. Mais rien ne dit que chaque individu supplémentaire est classé de la même manière par les deux logiciels. Pour s'en assurer, le plus simple est de confronter les deux classements. Nous chargeons donc le fichier exporté par Tanagra indiquant les classes d'appartenance des individus supplémentaires. Nous le croisons avec celui de R.

```
#*****************************************
#croiser les résultats avec ceux de Tanagra
#*****************************************
#renuméroter les individus de 1 à 100
names(resultat) <- 1:100
#charger les clusters de Tanagra
tanagra.clus <- read.table(file="tanagra_clusters.txt",header=T)
#confronter
croisement <- table(resultat,tanagra.clus$Cluster_KMeans_1)
print(croisement)
```

Plus de doute, les deux classements concordent parfaitement :

```
> print(croisement)

resultat c_kmeans_1 c_kmeans_2
       1          0         26
       2         74          0
```

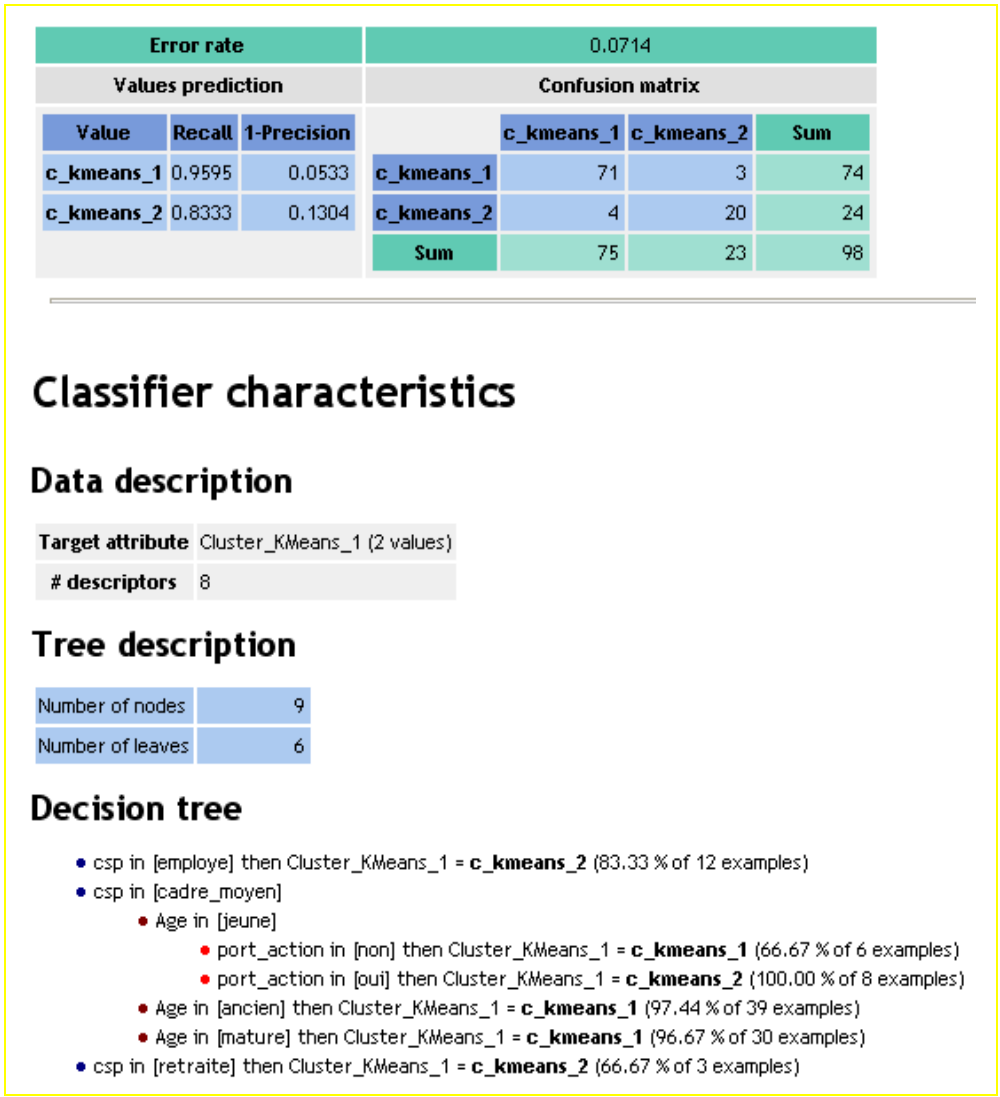
[5] Bon là effectivement, si on n'est pas très familiarisé avec R, l'instruction **apply(.)** peut paraître très mystérieuse… C'est le seul barrage que je vois à l'utilisation de R. Nous devons faire l'apprentissage d'un langage de programmation lorsque l'on souhaite l'exploiter pleinement. Pour certains, ce n'est pas un problème, pour d'autres c'est rédhibitoire.

5 Conclusion

Nous décrivons dans ce didacticiel une approche possible du déploiement en apprentissage non supervisé. Curieusement, ce problème est souvent éludé dans la littérature. Pourtant, ce ne sont pas les applications pratiques qui manquent.

Nous avons mis l'accent sur la cohérence entre la méthode de constitution des classes (K-Means) et la stratégie d'affectation des nouveaux individus (distance aux centres de classes). Parfois, assurer cette adéquation n'est pas possible. On peut alors se tourner vers des démarches plus pragmatiques en mixant apprentissage non supervisé et supervisé c.-à-d. (1) construire les classes à l'aide d'une méthode de typologie quelconque ; (2) utiliser une technique supervisée, les arbres de décision par exemple, pour élaborer un modèle d'affectation permettant d'associer à un individu une classe à partir de sa description dans l'espace originel.

Dans notre exemple, nous avons utilisé la méthode C4.5 pour prédire les clusters à partir des variables de la base, nous obtenons le modèle d'affectation suivant.



Error rate	0.0714
Values prediction	Confusion matrix

Value	Recall	1-Precision
c_kmeans_1	0.9595	0.0533
c_kmeans_2	0.8333	0.1304

	c_kmeans_1	c_kmeans_2	Sum
c_kmeans_1	71	3	74
c_kmeans_2	4	20	24
Sum	75	23	98

Classifier characteristics

Data description

Target attribute	Cluster_KMeans_1 (2 values)
# descriptors	8

Tree description

Number of nodes	9
Number of leaves	6

Decision tree

- csp in [employe] then Cluster_KMeans_1 = **c_kmeans_2** (83.33 % of 12 examples)
- csp in [cadre_moyen]
  - Age in [jeune]
    - port_action in [non] then Cluster_KMeans_1 = **c_kmeans_1** (66.67 % of 6 examples)
    - port_action in [oui] then Cluster_KMeans_1 = **c_kmeans_2** (100.00 % of 8 examples)
  - Age in [ancien] then Cluster_KMeans_1 = **c_kmeans_1** (97.44 % of 39 examples)
  - Age in [mature] then Cluster_KMeans_1 = **c_kmeans_1** (96.67 % of 30 examples)
- csp in [retraite] then Cluster_KMeans_1 = **c_kmeans_2** (66.67 % of 3 examples)

Le modèle n'est pas exact, 7 individus ne sont pas ré affectés dans leur groupe originel. C'est un problème. Mais l'approche a le mérite de la simplicité.