

1 Subject

Classification of new examples from the results of a clustering method (K-Means).

The deployment is an important step of the Data Mining framework. In the case of a clustering, after the construction of clusters with a learning algorithm, we want to determine to which particular cluster (group) a new unlabelled instance belongs.

In this tutorial, we use the K-Means algorithm. We assign each new instance to the group which is closest using the distance to the center of groups. The method is fair because the technique used to assign a group in the deployment phase is consistent with the learning algorithm. It is not true if we use another learning algorithm e.g. HAC (hierarchical agglomerative clustering) with de single linkage aggregation rule. The distance to the center of groups is inadequate in this context. Thus, **the classification strategy must be consistent with the learning strategy.**

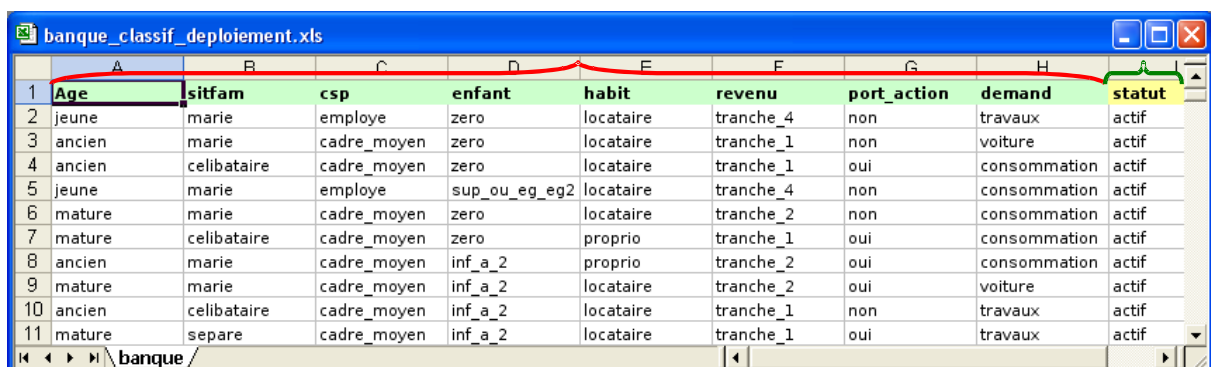
All the descriptors are discrete in our dataset. The K-Means algorithm does not handle directly this kind of data. We must transform them. We use a multiple correspondence analysis algorithm (see also <http://data-mining-tutorials.blogspot.com/2008/11/k-means-algorithm-on-discrete.html>). In this tutorial, we use the following steps:

- Data Importation;
- Computing some descriptive statistics, in order to detect potential anomalies;
- Partitioning the dataset into training set and test set, this last one is used during the deployment phase;
- Computing the coordinates of examples in the new representation space generated by the multiple correspondence analysis;
- Applying the K-Means algorithm on the latent variables;
- Interpreting groups using descriptive statistics;
- Deploying i.e. determining the group membership of each instance from the test set.

We use **Tanagra 1.4.28** and **R 2.7.2** (with the Facto Miner package for the multiple correspondence analysis - MCA; <http://factominer.free.fr/>). In this tutorial, we want (1) to show how to perform this kind of task with these tools; (2) compare the results; (3) by giving the details of commands with R, we can explain better the internal computation of Tanagra.

2 Dataset

The dataset describes 198 bank customers: 98 are used during the learning phase, 100 during the deployment. There are 9 discrete variables. The additional column "STATUT" gives the status of each example. We show here the 10 first examples of the database (http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/banque_classif_deploiement.zip).

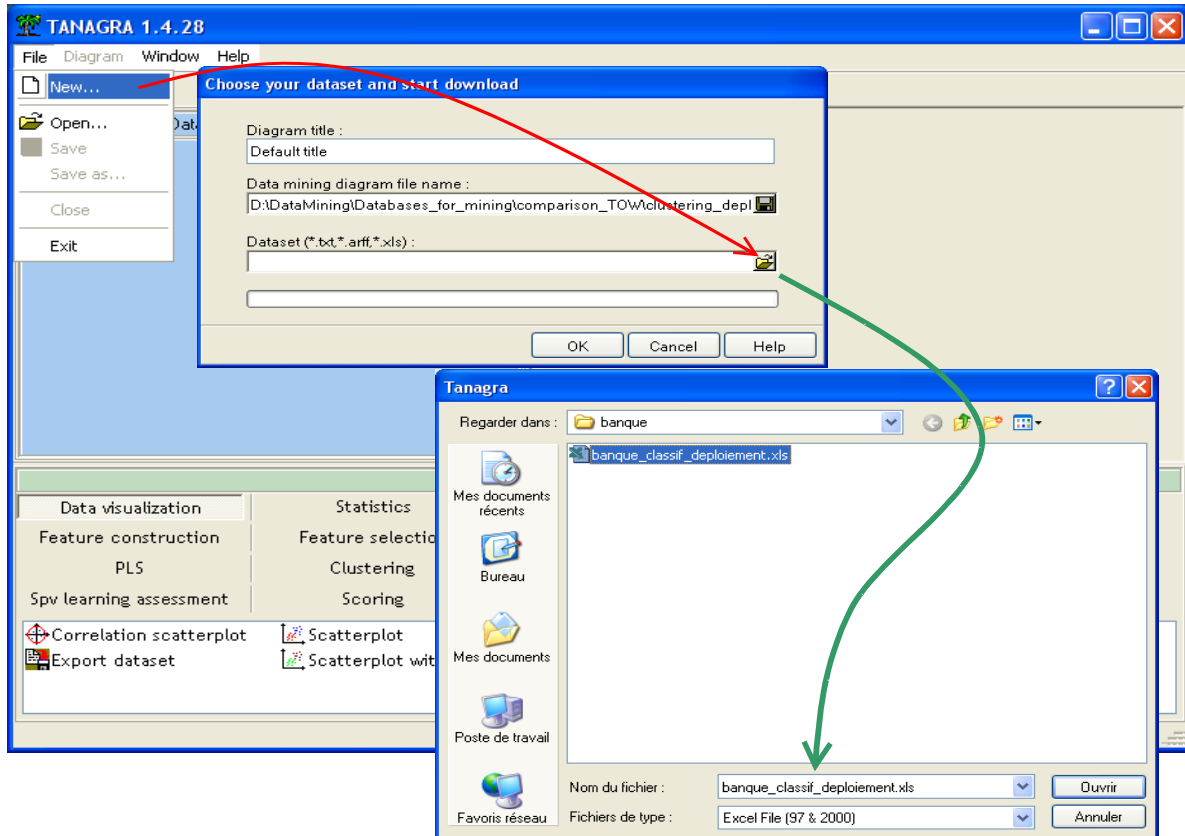


	A	B	C	D	E	F	G	H	I
1	Age	sitfam	csp	enfant	habit	revenu	port_action	demand	statut
2	jeune	marie	employe	zero	locataire	tranche_4	non	travaux	actif
3	ancien	marie	cadre_moyen	zero	locataire	tranche_1	non	voiture	actif
4	ancien	celibataire	cadre_moyen	zero	locataire	tranche_1	oui	consommation	actif
5	jeune	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 and deployment with TANAGRA

3.1 Data importation and diagram creation

We can directly import an Excel file (XLS). After launching Tanagra, we activate the FILE/NEW menu in order to create a new diagram. We select the BANQUE_CLASSIF_DEPLOIEMENT.XLS data file.



A new diagram is automatically created. The dataset is loaded. We have 198 examples and 9 columns¹.

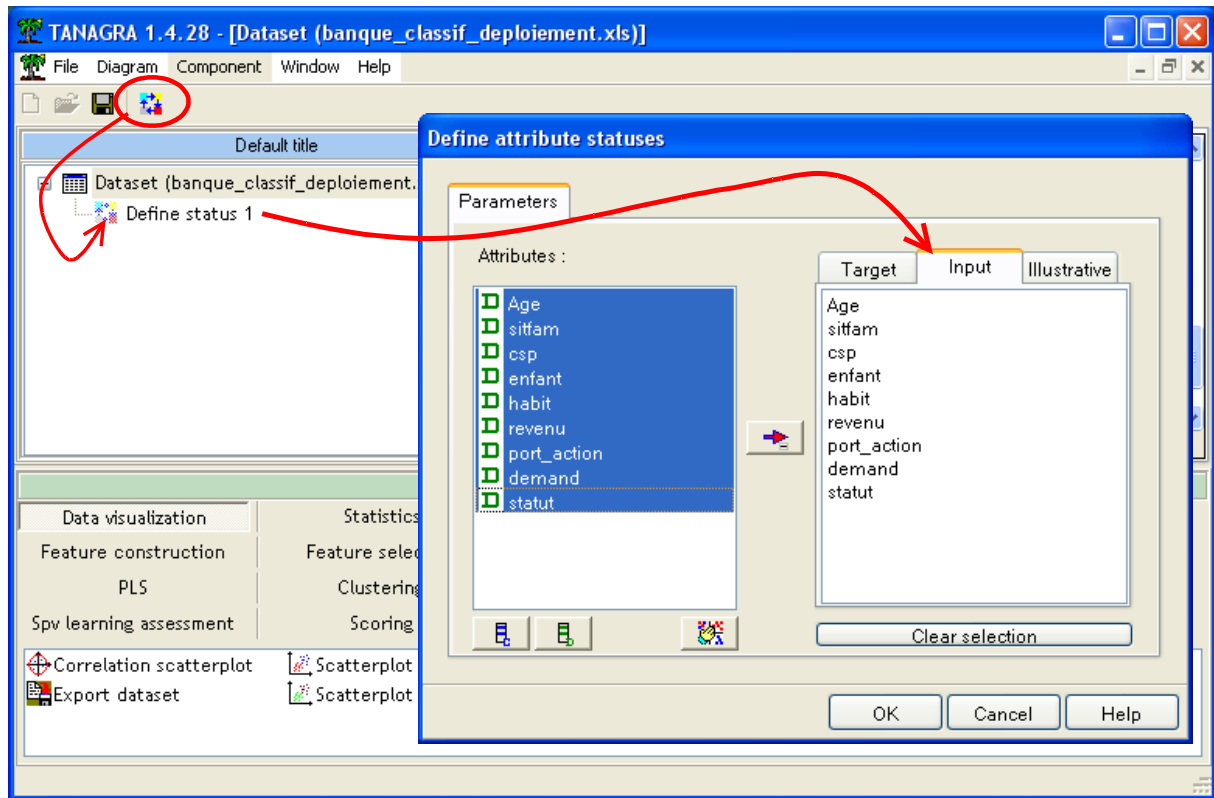
3.2 Descriptive statistics

We calculate the histograms for each column. For "STATUT", we wish only count the number of examples for the training and the test set. For other variables used for the analysis, we want mainly detect possible problems.

For instance, the value "CSP = RETRAITE" can cause problem during the MCA analysis because there are very few examples (4). We keep in mind this particularity during our analysis.

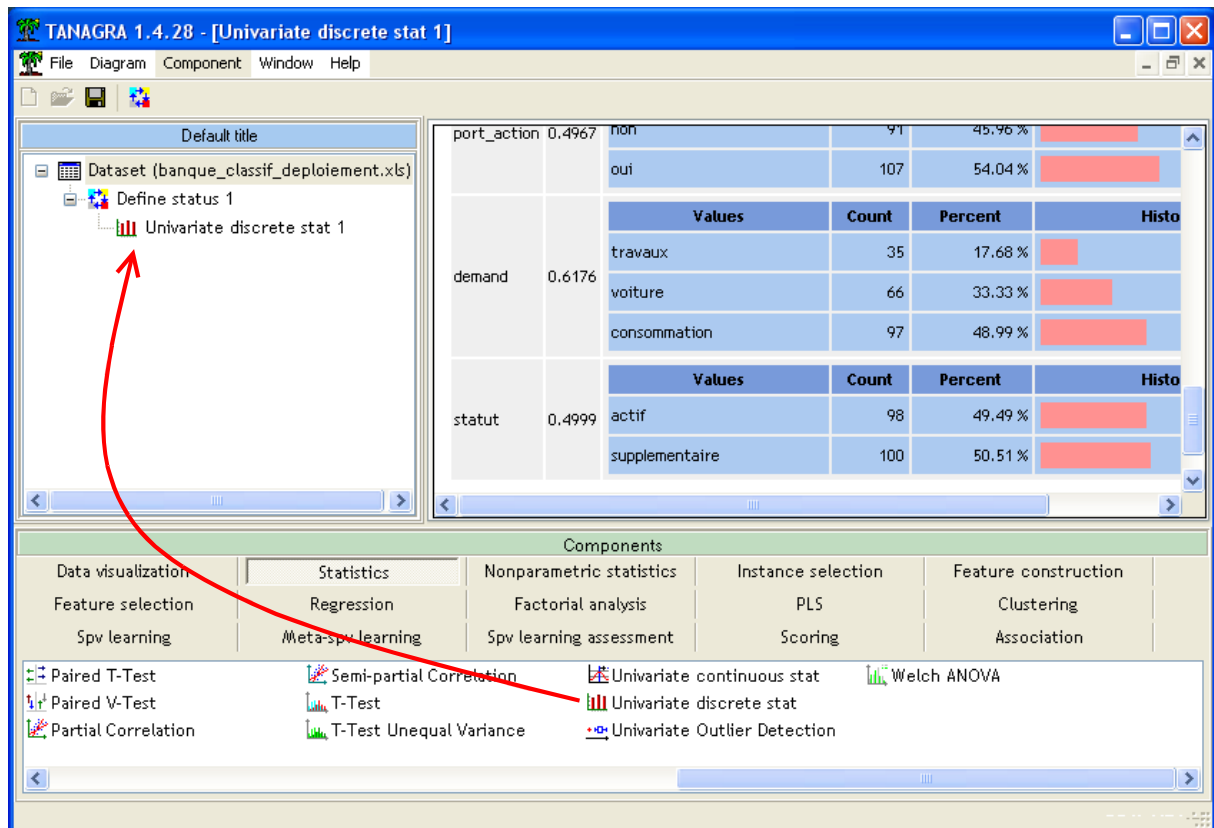
We add the DEFINE STATUS component into the diagram. We set all the columns as INPUT.

¹ See <http://tutoriels-data-mining.blogspot.com/2008/03/importation-fichier-xls-excel-mode.html> for the direct importation of XLS file. We can also send the dataset from Excel to Tanagra using an add-in : <http://tutoriels-data-mining.blogspot.com/2008/03/importation-fichier-xls-excel-macro.html>



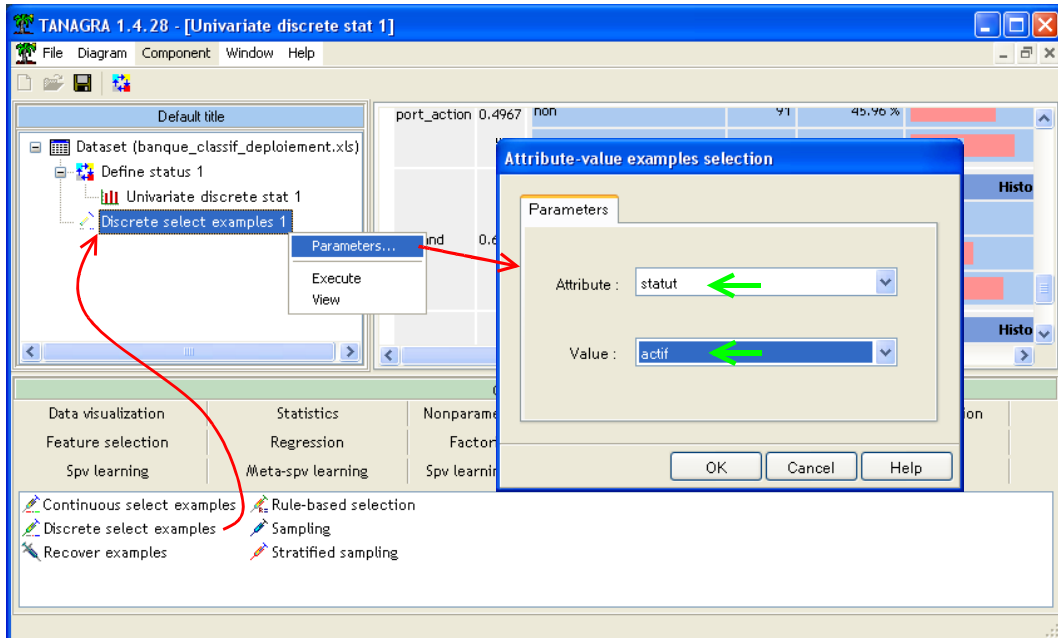
We insert the UNIVARIATE DISCRETE STAT (STATISTICS tab) into the diagram.

The main result is there are indeed 98 active examples for the analysis and 100 examples to classify (see STATUT).



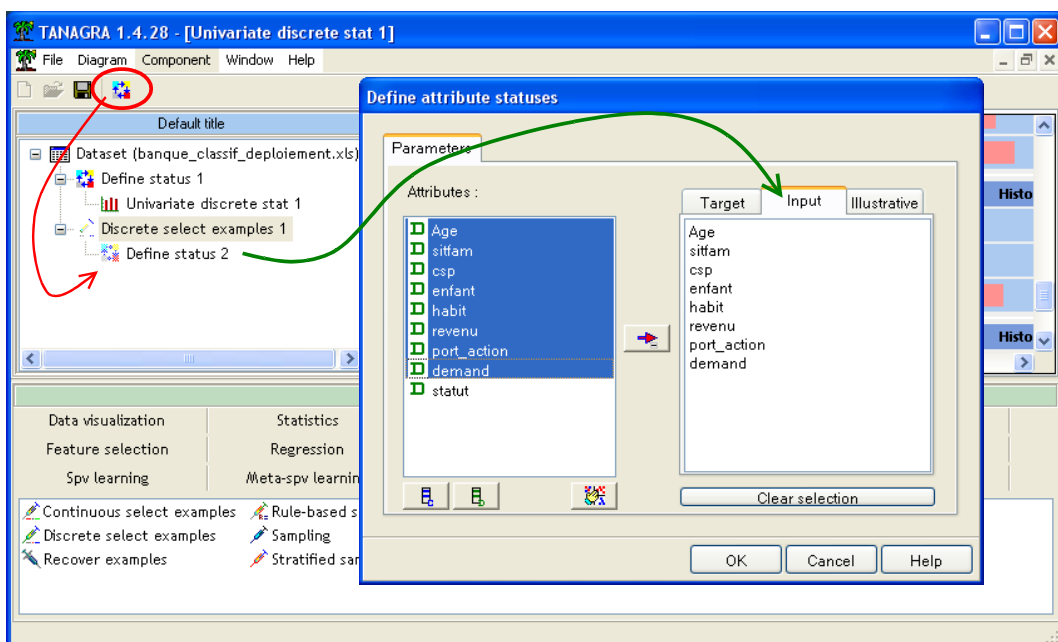
3.3 Selection of active examples

We use the DISCRETE SELECT EXAMPLES (INSTANCE SELECTION tab) in order to subdivide the dataset into train and test set. We click on the PARAMETERS contextual menu. We set the following parameters. We validate: 98 instances are now used for the subsequent treatments.

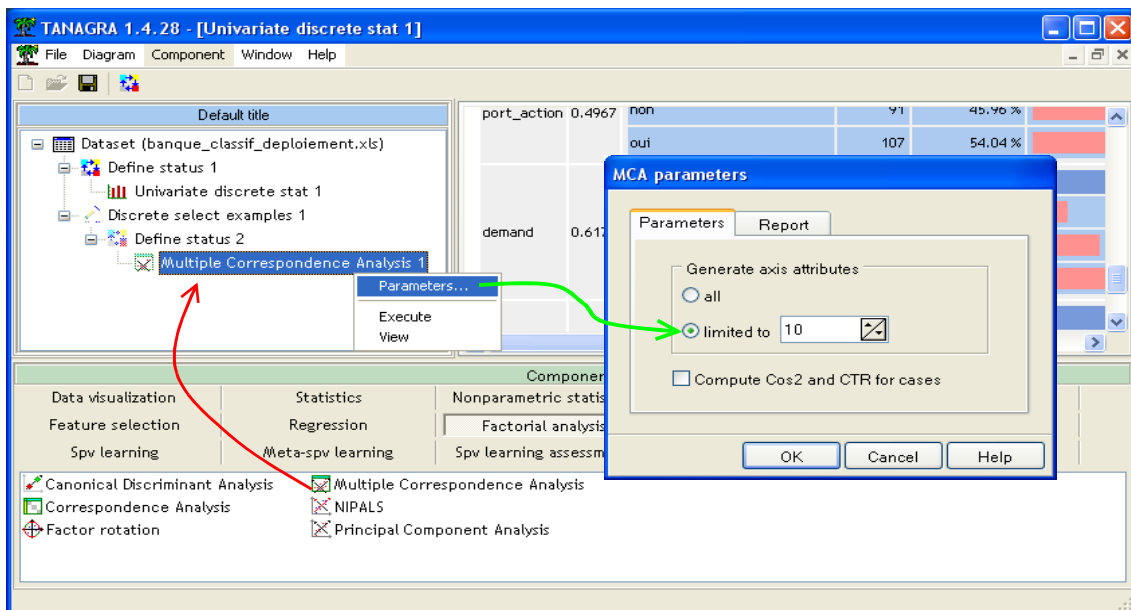


3.4 Multiple correspondent analysis

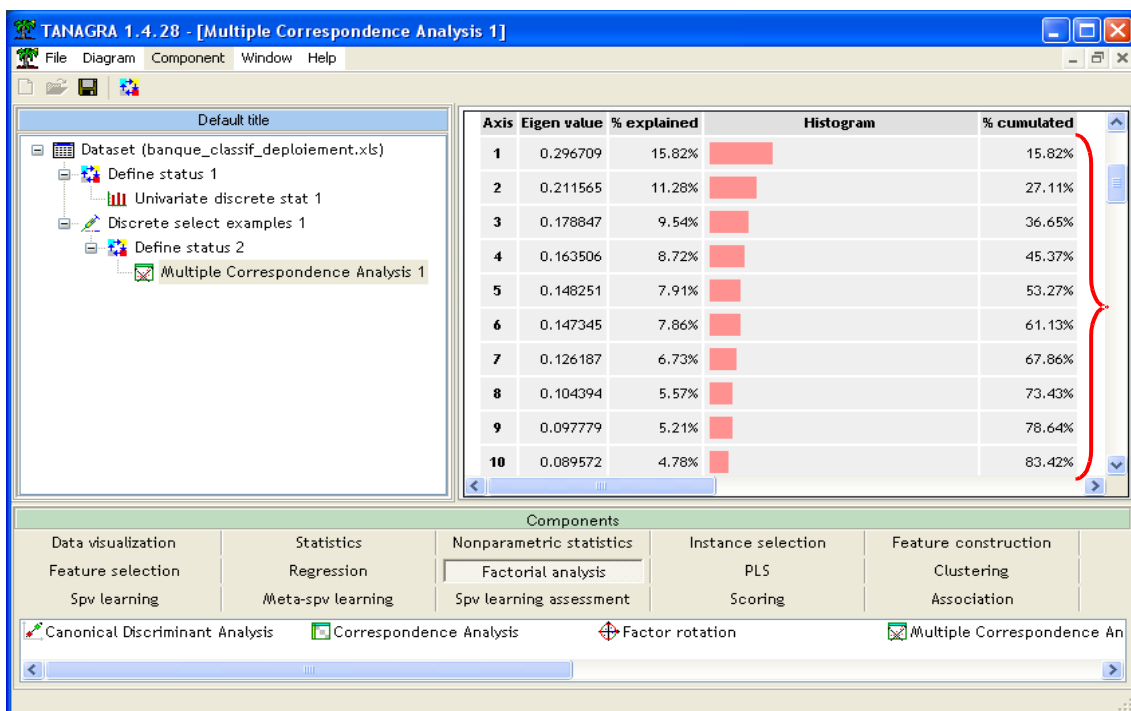
There are not clustering algorithms which can handle discrete attributes into Tanagra. We must transform the variables. We use a multiple correspondence analysis approach. We take mainly two advantages with this kind of pre treatment: the factors, latent variables, are orthogonal; we can retain the first "q" factors which give the useful information, we can neglect the last factors which correspond to noise. The quality of the partitioning will be better. We insert again the DEFINE STATUS component. We set the descriptors as INPUT (AGE ... DEMAND).



Then, we add the MULTIPLE CORRESPONDANCE ANALYSIS component (FACTORIAL ANALYSIS tab). We set the number of factors to 10.



We click on the VIEW menu in order to obtain the results. The 10 first factors correspond to the 83.42% available information.

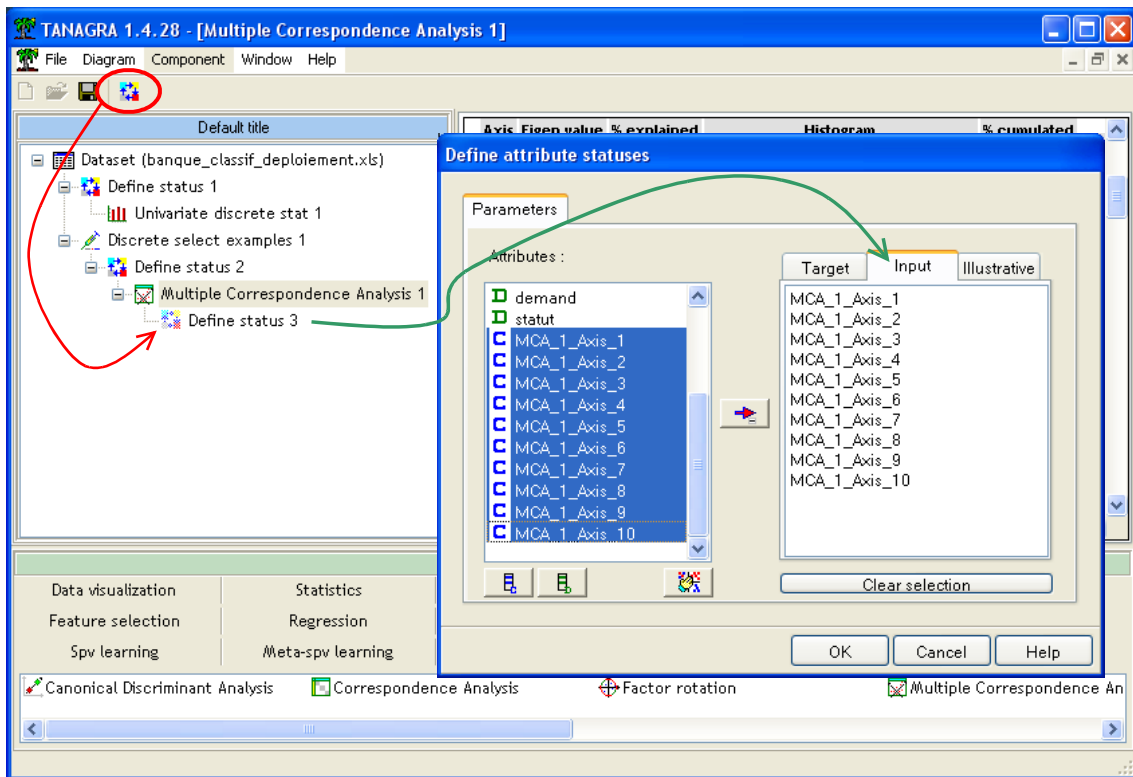


Two important things must retain our attention:

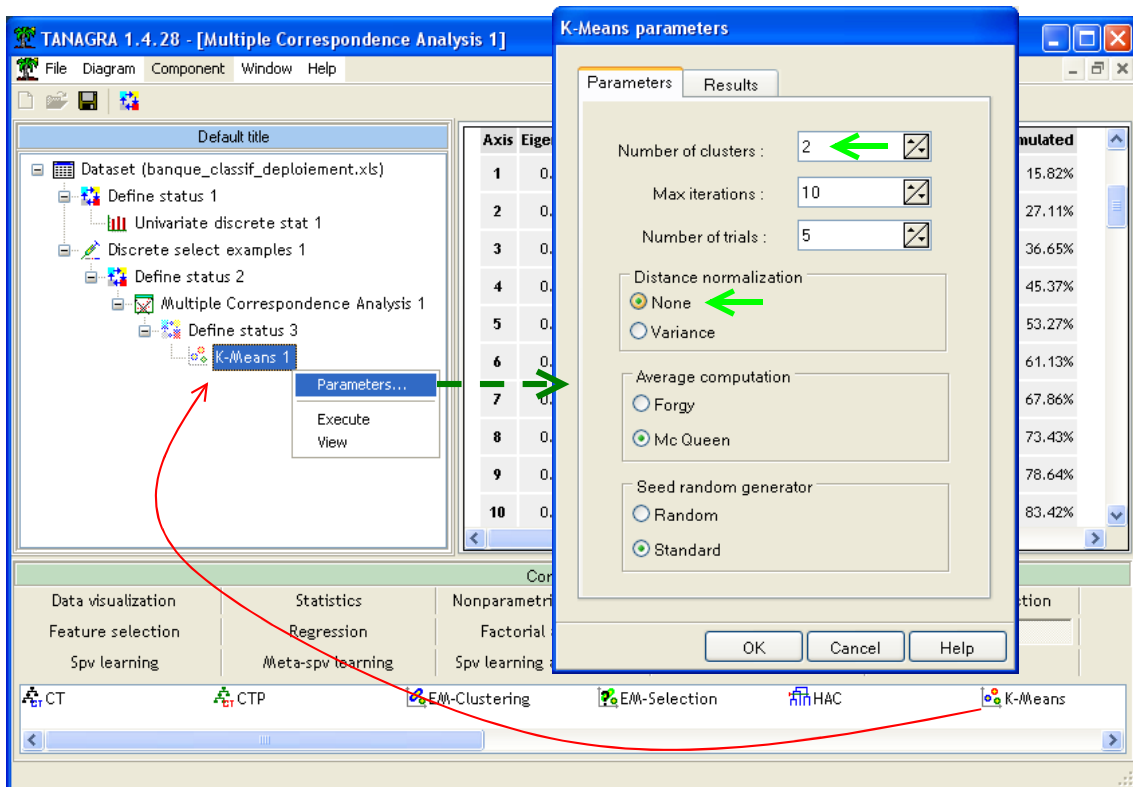
1. **Tanagra computes automatically 10 new variables** which are available on the subsequent part of the node into the diagram;
2. Even if the projection coefficients are computed on the train set, **these new variables are available both the train and the test part of the dataset.**

3.5 K-Means on the latent variables (factors)

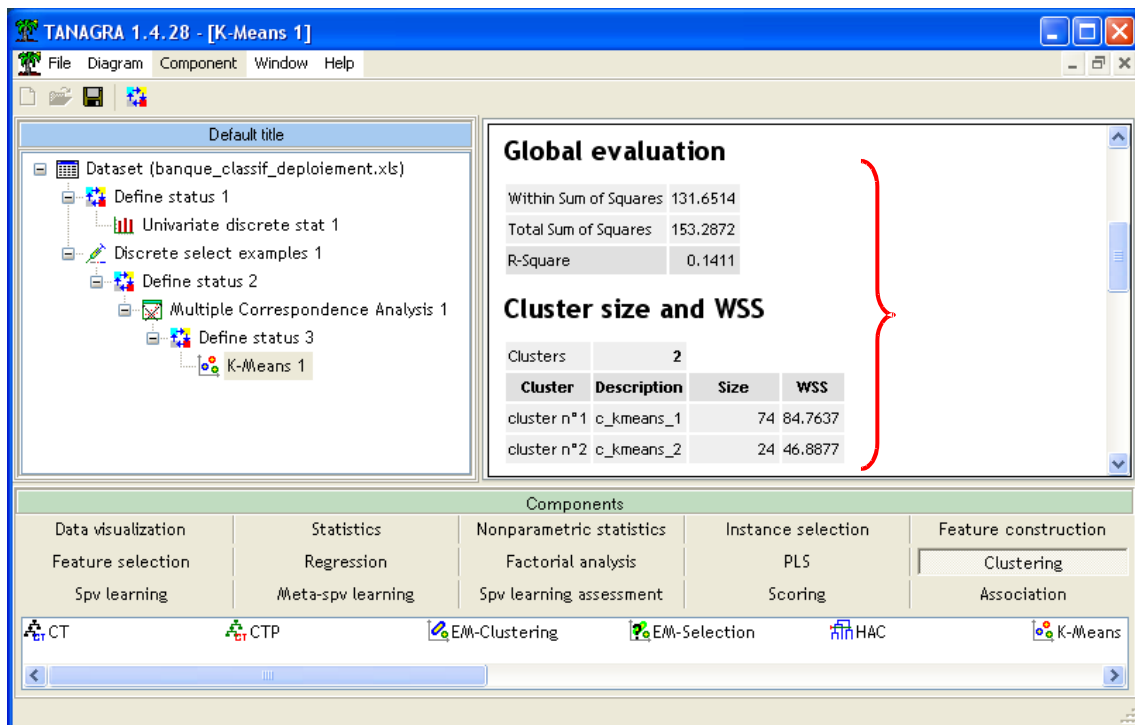
We want to launch the K-means algorithm on the factors. We add a DEFINE STATUS component; we set the new variables MCA_1_AXIS_1 to MCA_1_AXIS_10 as INPUT.



Then we insert the K-Means component (CLUSTERING tab). We set the parameters as the following:

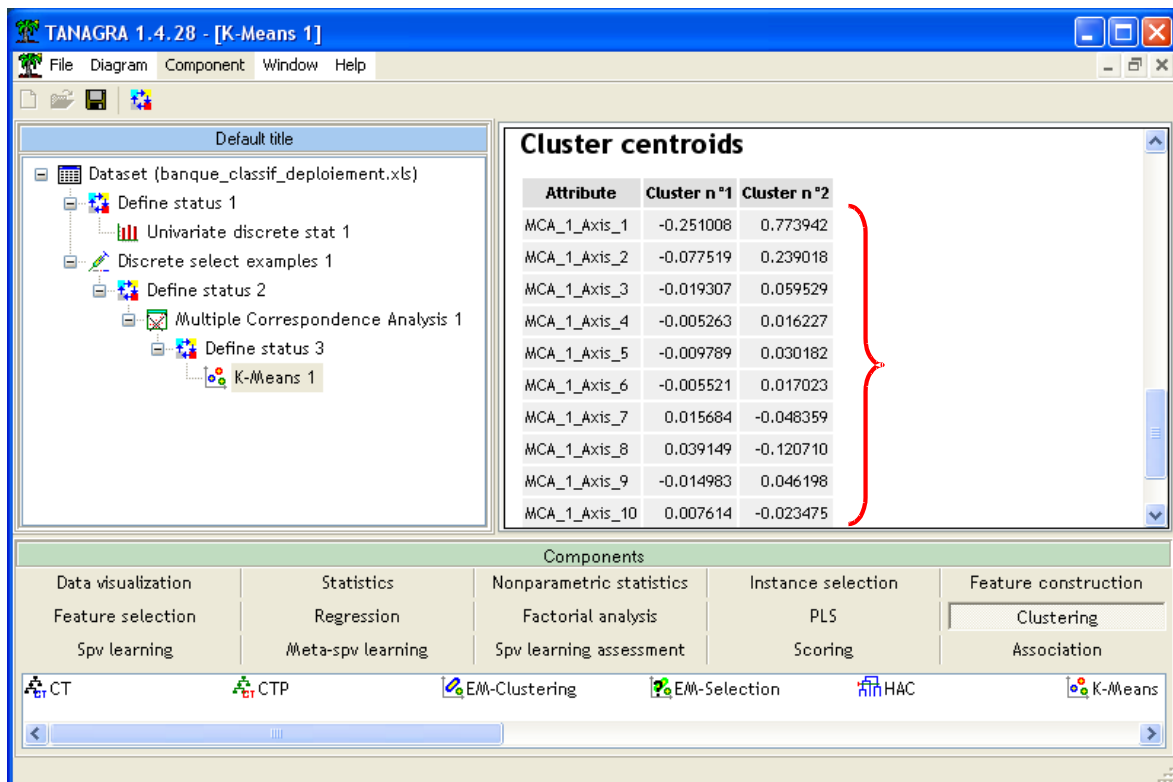


We click on the VIEW menu.



There are 74 examples in the first group, 24 in the second. The within sum of squares is 131.6514 (84.7637 + 46.8877). We will compare these values to the results of R.

In the bottom part of the window, we get the cluster means. It gives an idea about the location of the groups in the representation space. But, because they computed on the latent variables, they are not really useful for the interpretation of the clusters.

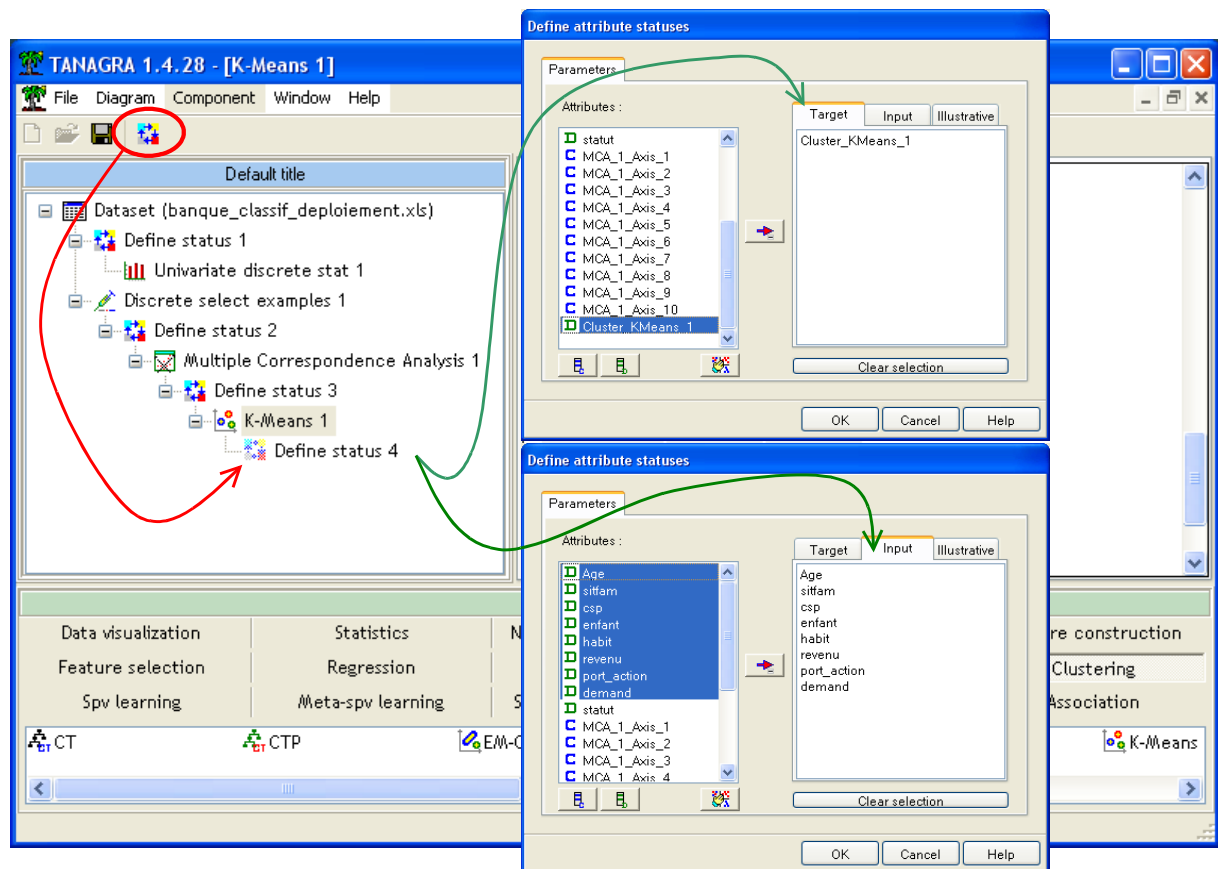


Another important thing here: **Tanagra** computes first the clusters using the learning sample; but, it **determines the group membership of each example using the distance to the centroid**. Thus, a new column is available in the subsequent part of the diagram; it corresponds to the cluster membership to each example. It is computed both on the train and the test set. We utilize this new variable to interpret classes from the clustering process.

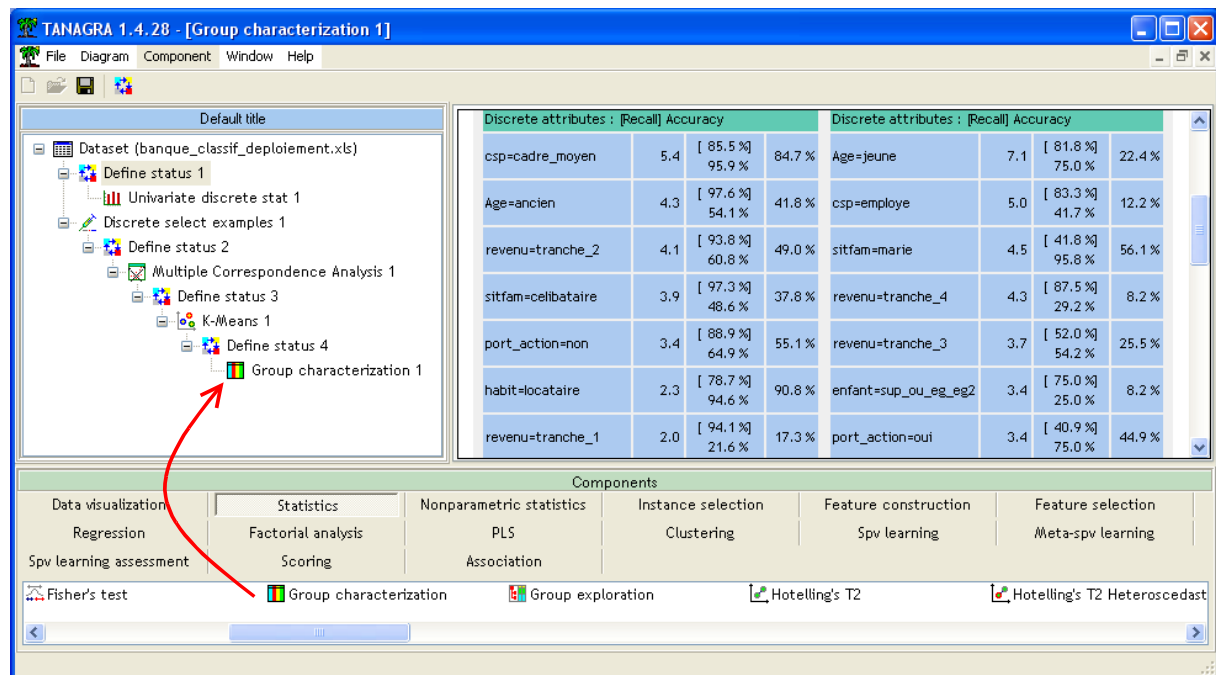
3.6 Interpretation of groups

3.6.1 Descriptive statistics comparison

Even if they are univariate, Conditional descriptive statistics give good indications about the nature of groups. We insert the DEFINE STATUS into the diagram. We set CLUSTER_KMEANS_1, generated by the learning algorithm, as TARGET; we set the descriptors as INPUT (AGE...DEMAND).

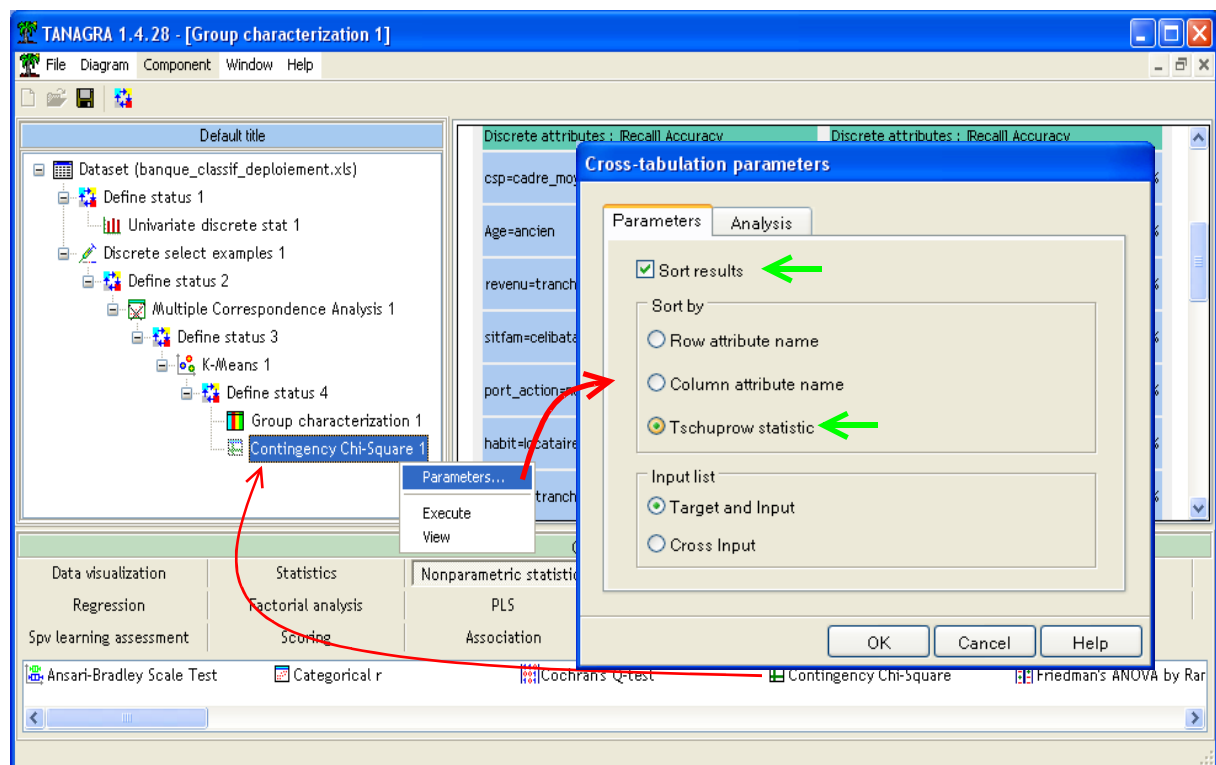


We add now the GROUP CHARACTERIZATION (STATISTICS tab). We click on VIEW contextual menu. Thus, we can characterize the partitioning by comparing the most occurred values into each group.

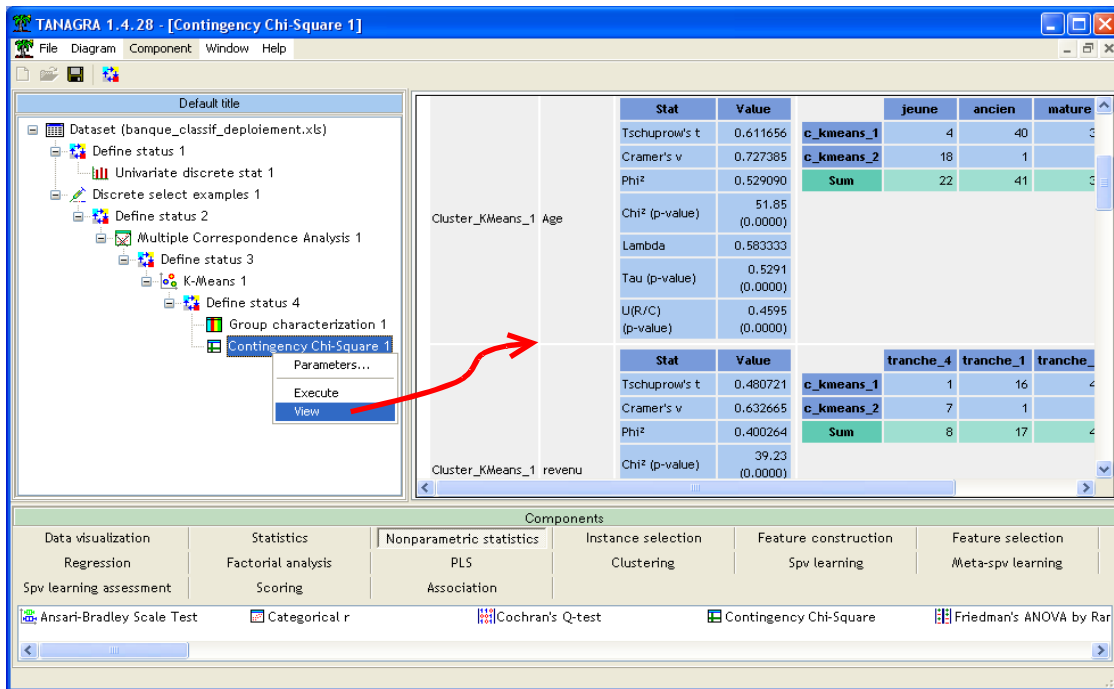


3.6.2 Contingency table between clusters and descriptors

Another way to understand the clusters background is to compute the contingency tables between the cluster variable (which indicates the group membership for each example) and the descriptors. We insert the CONTINGENCY CHI-SQUARE component (NONPARAMETRIC STATISTICS tab). We set the parameters in order to highlight the most important relationship.



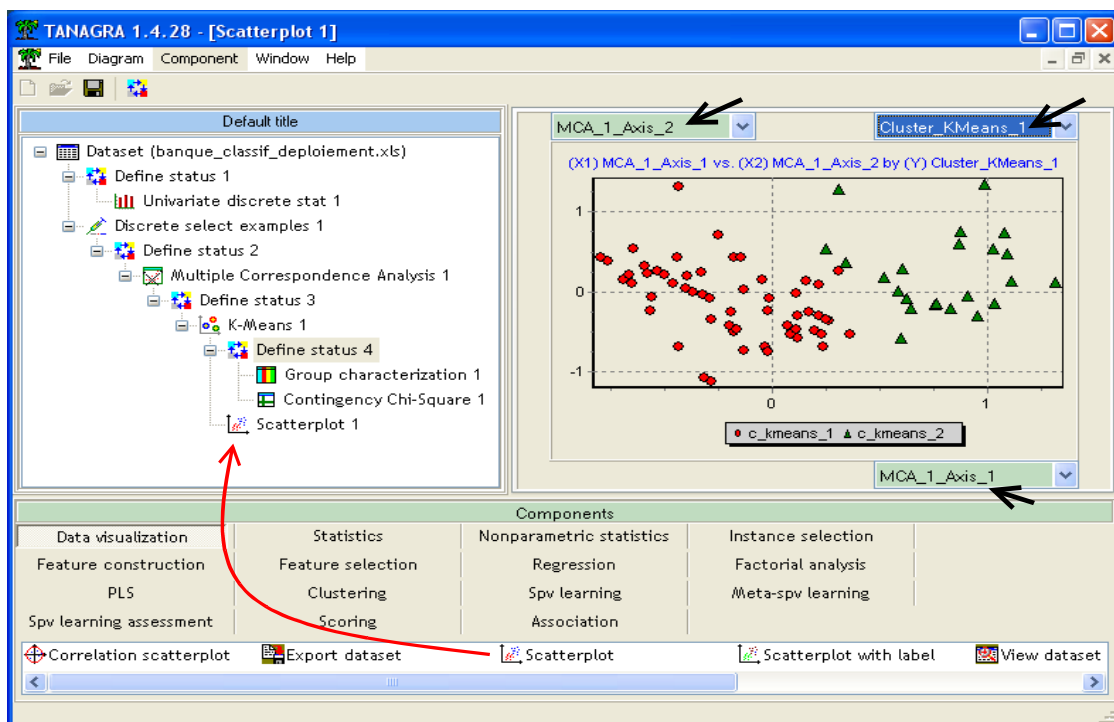
We click on the contextual menu VIEW.



AGE seems the most significant variable to explain the CLUSTERS, then REVENU, etc. Of course, the results highlighted here cannot be in contradiction to the previous analysis (GROUP CHARACTERIZATION). These are univariate point of view.

3.6.3 Scatter plot into the factors space

Finally, last tool for interpreting groups, we use the projection of observations in the space of the latent variables generated by the multiple correspondence analysis. This means that we know how to interpret correctly these factors, this is not always obvious. But when we succeed, the results are very interesting because we have a multivariate point of view.



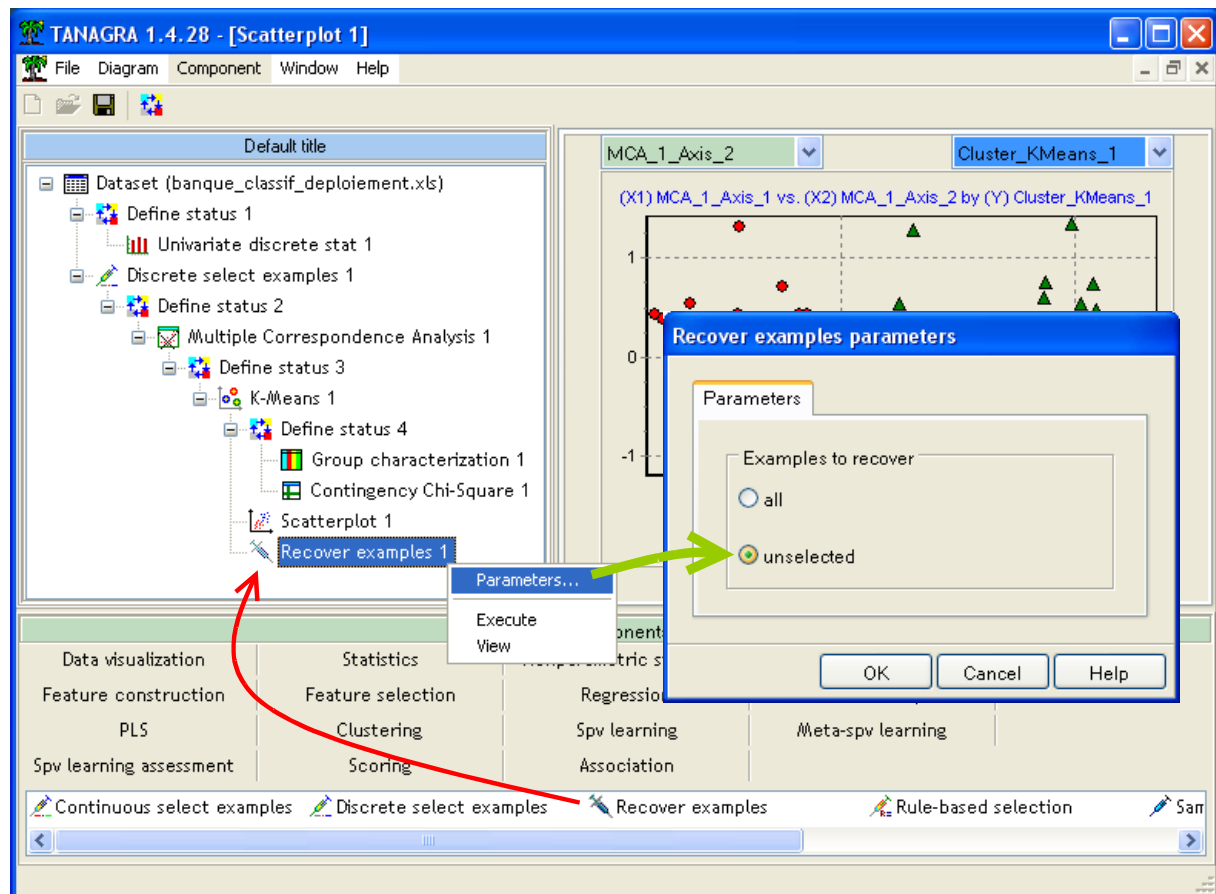
We add the SCATTERPLOT component into the diagram (DATA VISUALIZATION tab). We set MCA_1_AXIS_1 on the horizontal axis, MCA_1_AXIS_2 on the vertical axis. We colorize the points using the CLUSTERS values. We note that we can easily distinguish the groups in this first scatter plot.

In considering the factor analysis results above, we see that the first axis (column contributions) is mainly defined by age, family status and income. We find again the results highlighted in the univariate analysis.

3.7 Recovering the test set

On all of the components of diagram, Tanagra handles the unselected examples. For the factorial analysis, it computes the coordinate of examples of the new axis; for the clustering phase, it assigns each example to a group. We want to consider these results now.

The RECOVER EXAMPLES component (INSTANCE SELECTION tab) allows us to recover the unselected examples. We add it into the diagram. We click on the PARAMETERS menu, we observe that we can recover all the examples or the unselected examples only (test set).

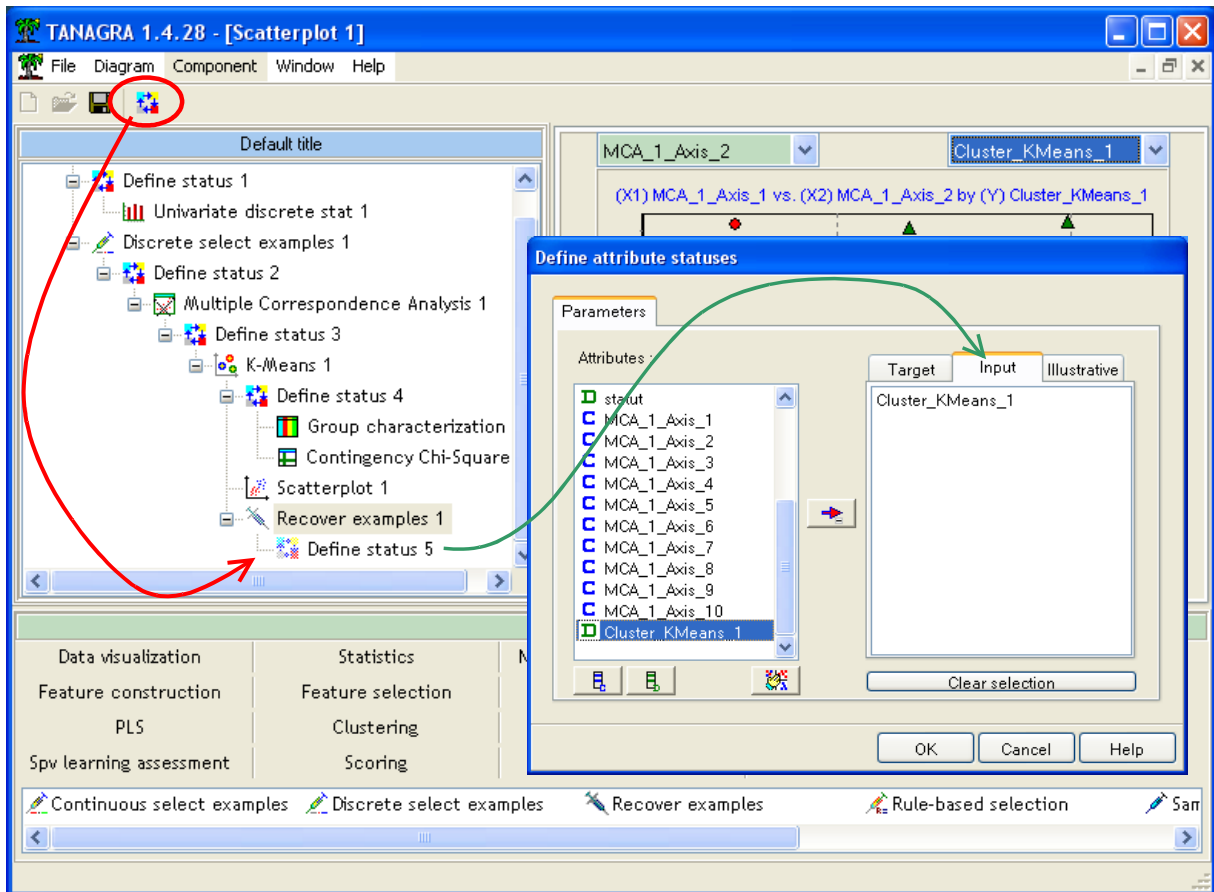


We click on VIEW. The test set (100 examples) is used in the subsequent part of the diagram now.

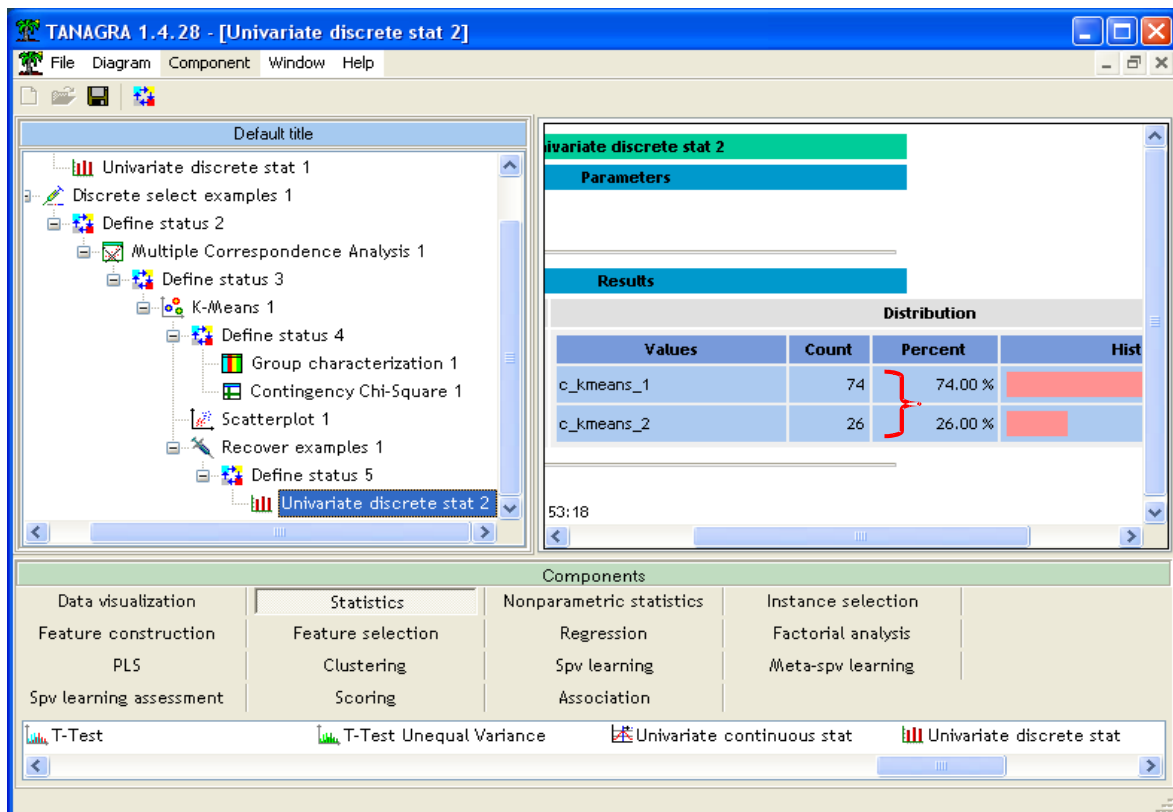
3.8 Group distribution for the test set

A first assessment is the computation of the group distribution on the test set. Because the dataset is randomly partitioned into train and test set, we expect to obtain approximately the same distribution for the two parts of the dataset.

We insert the DEFINE STATUS component into the diagram, we set CLUSTER_KMEANS_1 as INPUT.

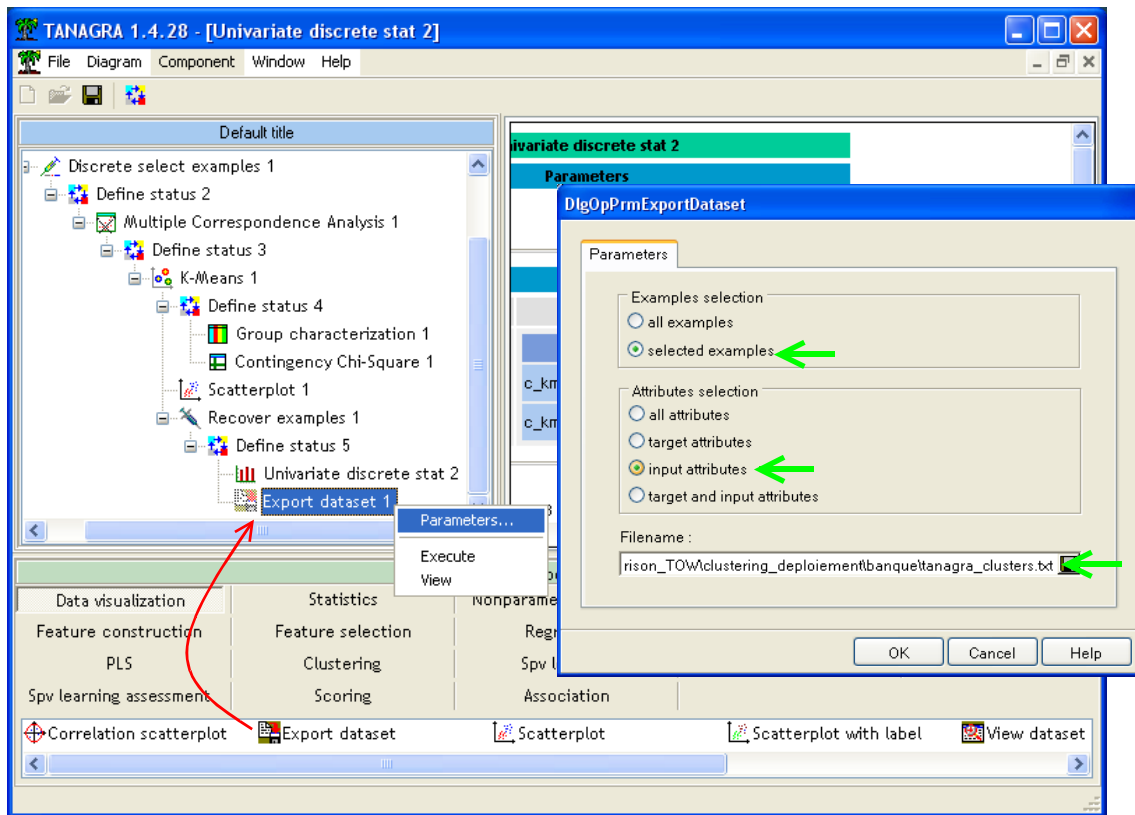


We add now the UNIVARIATE DISCRETE STAT component (STATISTICS tab). We obtain similar proportion, 74 instances belong to the first group, 26 to the second.

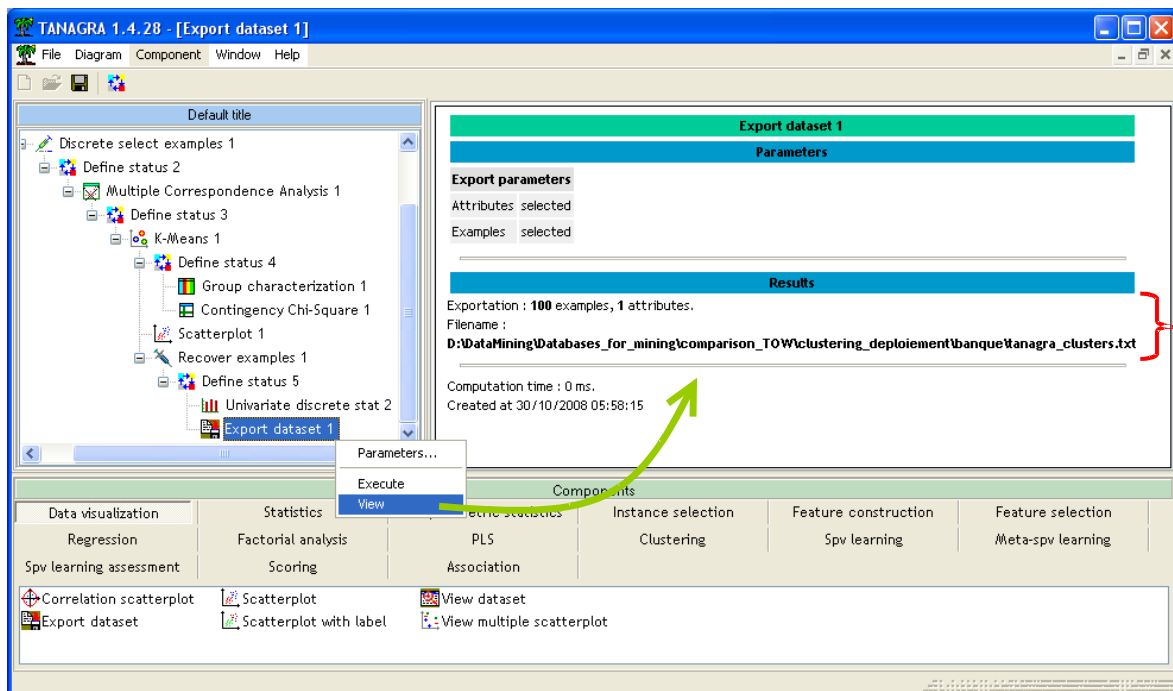


3.9 Data exportation

We want to export the group membership. We want to compare the results of Tanagra with those of R. For this, we use the EXPORT DATASET component (DATA VISUALIZATION tab).



We click on the PARAMETERS menu. We want export only the selected variable (CLUSTER_KMEANS_1 selected on the DEFINE STATUS above) and the selected examples (100 examples of the test set). We set also the file name TANAGRA_CLUSTERS.TXT.



4 K-Means and deployment with R

The goal is to reproduce exactly the previous process with R, then compare the results. We hope also that the description of the detail of operations with R gives better visibility on the calculations made internally by Tanagra.

Data importation. We use the `xlsReadWrite` in order to import the EXCEL file format; the `read.xls(.)` command allows to read the dataset.

```
#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)
```

`summary(.)` computes some descriptive statistics for all the descriptors. We note mainly that there are 98 examples for the training phase (STATUT = ACTIF) and 100 for the test phase (STATUT = SUPPLEMENTAIRE).

```
> 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
```

Partitioning the dataset. We create two indexes (vector of integers) which correspond to examples into the train and the test set.

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

We then obtain

```
> 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
```

Multiple correspondence analysis. We use the MCA function of the FACTOMINER package (<http://factominer.free.fr/>). Its utilization is not common. The `predict(.)` method is not available. The projection on new examples is directly made when we call the MCA function with all the dataset. The two indexes above allow to discern the train and the test set.

The coordinates into the factors space is computed on the train set (`indcoord`) and on the test set (`$ind.sup$coord`). We asked 10 factors.

```
#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)
```

We display below the Eigen values and percentage of explained variance on each factor.

```
> 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
```

K-Means with the factors of MCA. We launch the K-Means method on the 10 first factors. We ask 2 groups.

```
#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)
```

We obtain

```

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  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  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  1  2  1  1  2  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  2  1  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"

```

R obtains 2 groups with respectively 74 and 24 instances. The within sum of squares is the same as Tanagra. The groups are likely the same.

```

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

```

We can also compute the centroids.

```

#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)

```

We note that the group n°1 (resp. n°2) of Tanagra corresponds to clus_2 (resp. clus_1) of 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

```

Deployment i.e. assigning groups to unlabeled examples. We must define various functions for the classification. First, `dist_euclidienne(.)` computes the square of the Euclidian distance between a centroid (reference) and an example (`a_classer`).

```

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

```

Then, we compute this distance for the centroids (reference) of the groups. **le_plus_proche(.)** returns the index of the closest group.

```
#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)
}
```

Finally, we apply this function for all the examples of the test set. A new vector (**resultat**) gives the group membership of each example.

```
#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))
```

The group distribution on the test set is the same as Tanagra.

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

Comparison of the groups. We get the same group distribution but we do not know if Tanagra and R assign the same group to each example. In order to verify this, we want to create a contingency table between the two group assignment vectors.

In the following screenshot, we import into R the group membership exported from Tanagra and we create a contingency table with "resultat".

```
#####
#croiser les résultats avec ceux de Tanagra
#####
#renommer 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)
```

The correspondence is perfect.

```
> print(croisement)

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

5 Conclusion

We describe in this tutorial a possible deployment strategy in unsupervised learning. Curiously, this problem is often neglected in the literature. Yet the practical applications are obvious. The main idea is that the classification strategy must be in adequacy with the learning strategy.

But, we can also adopt a pragmatic approach. In a first step we define the clusters using the unsupervised method. Each instance of the learning set is assigned to their group. Then, in the second step we use the group as a class attribute in a supervised learning process. Of course, the classification can be noised i.e. two examples in the learning set which are the same group during the unsupervised learning phase can be assigned to different groups when we create the classification function from the supervised approach. But at least the strategy has the advantage of being unsophisticated.

We show below a classification function generated with the C4.5 decision tree algorithm.

Error rate			0.0714			
Values prediction			Confusion matrix			
Value	Recall	1-Precision		c_kmeans_1	c_kmeans_2	Sum
c_kmeans_1	0.9595	0.0533	c_kmeans_1	71	3	74
c_kmeans_2	0.8333	0.1304	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)

The model is not accurate. There are 7 false classifications on the training set.