

# 1 Subject

## Two-step clustering approach on large dataset.

The aim of the clustering is to identify homogenous subgroups of instance in a population<sup>1</sup>. In this tutorial, we implement a two-step clustering algorithm which is well-suited when we deal with a large dataset. It combines the ability of the K-Means clustering to handle a very large dataset, and the ability of the Hierarchical clustering (HCA – Hierarchical Cluster Analysis) to give a visual presentation of the results called “dendrogram”. This one describes the clustering process, starting from unrefined clusters, until the whole dataset belong to one cluster. It is especially helpful when we want to detect the appropriate number of clusters.

The two-step clustering strategy relies on the following schema: first, we use the K-means algorithm in order to create pre-clusters (e.g. 50), they contain a few examples; second, we start the HAC from these pre-clusters to create the dendrogram. This combination helps to overcome the disadvantages of these methods taken individually: K-Means requires to define in advance the number of groups and gives no indication on the relevance of this choice; standard HCA, because it computes the distance between all the pairs of individuals, cannot be implemented when the size of the database increases, even not more than a few thousand of observations.

The implementation of the two-step clustering (called also “Hybrid Clustering”) under Tanagra is already described elsewhere<sup>2</sup>. According to the Lebart and al. (2000) recommendation<sup>3</sup>, we perform the clustering algorithm on the latent variables supplied by a PCA (Principal Component Analysis) computed from the original variables. This pre-treatment cleans the dataset by removing the irrelevant information such as noise, etc. In this tutorial, we show the efficiency of the approach on a large dataset with 500,000 observations and 68 variables. We use **Tanagra 1.4.27** and **R 2.7.2** which are the only tools which allow to implement easily the whole process.

## 2 Dataset

We use the « 1990 US Census Data »<sup>4</sup>. There are 68 variables. Some of them are ordinal variables, other are dummies. We consider here that all the variables are continuous. It is not really important in our context. The main subject of this document is to show the feasibility of the treatments on a large dataset (memory occupation, computation time).

The original data file contains 2,458,285 examples. We have randomly drawn a sample of 500,000 examples because R cannot handle the whole dataset on my computer (2 GB RAM under Windows XP). Tanagra on the other hand has been able to handle the whole dataset, but we note however

---

<sup>1</sup> <http://faculty.chass.ncsu.edu/garson/PA765/cluster.htm>

<sup>2</sup> <http://data-mining-tutorials.blogspot.com/2008/11/hac-and-hybrid-clustering.html>

<sup>3</sup> L. Lebart, A. Morineau, M. Piron, « Statistique Exploratoire Multidimensionnelle », Dunod, 2000 ; chapitre 2, sections 2.3 et 2.4.

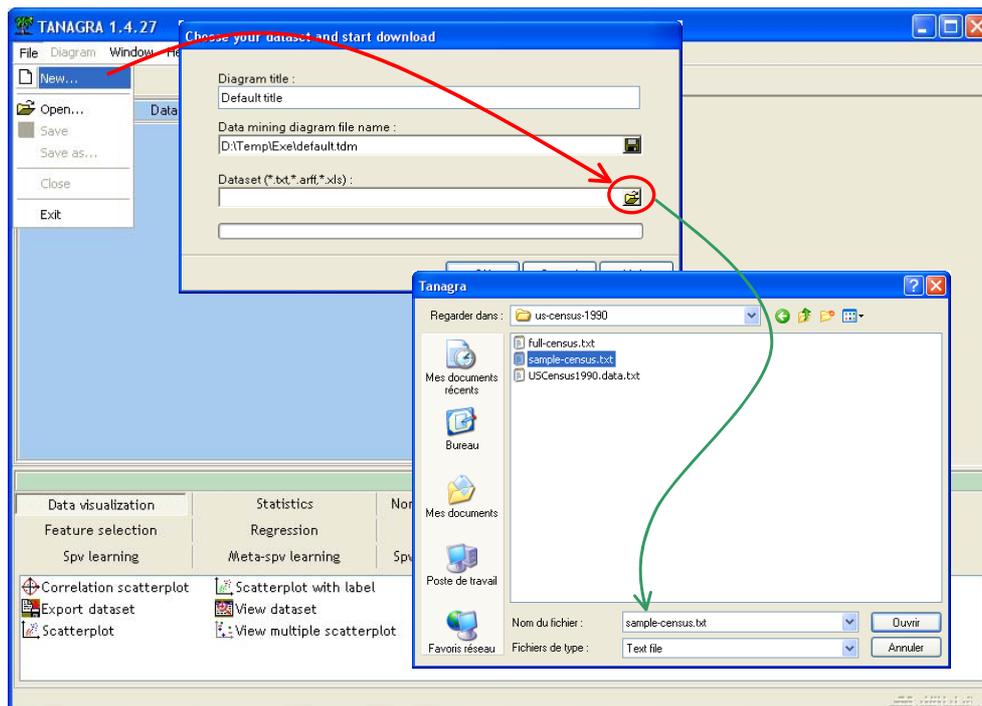
<sup>4</sup> <http://archive.ics.uci.edu/ml/databases/census1990/USCensus1990-desc.html>

that the results on the sample and the whole dataset are very similar. It is not surprising. The sampling, when it is properly done, is an efficient approach when we deal with a large dataset.

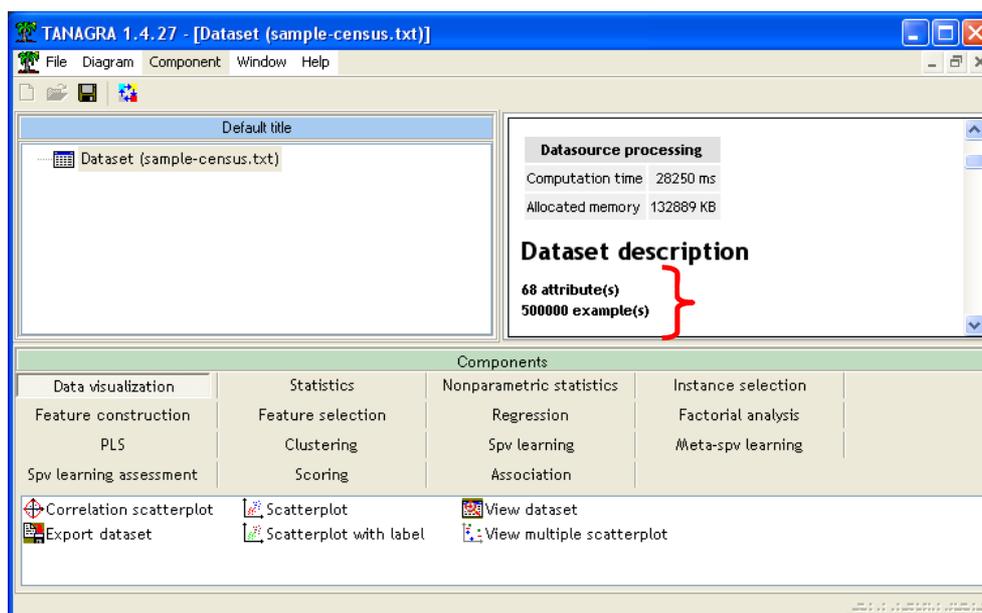
### 3 Two-step clustering with Tanagra

#### 3.1 Importing the dataset

After we launch Tanagra, we click on the FILE / NEW menu. We import the data file « sample-census.txt ». A new diagram is created.

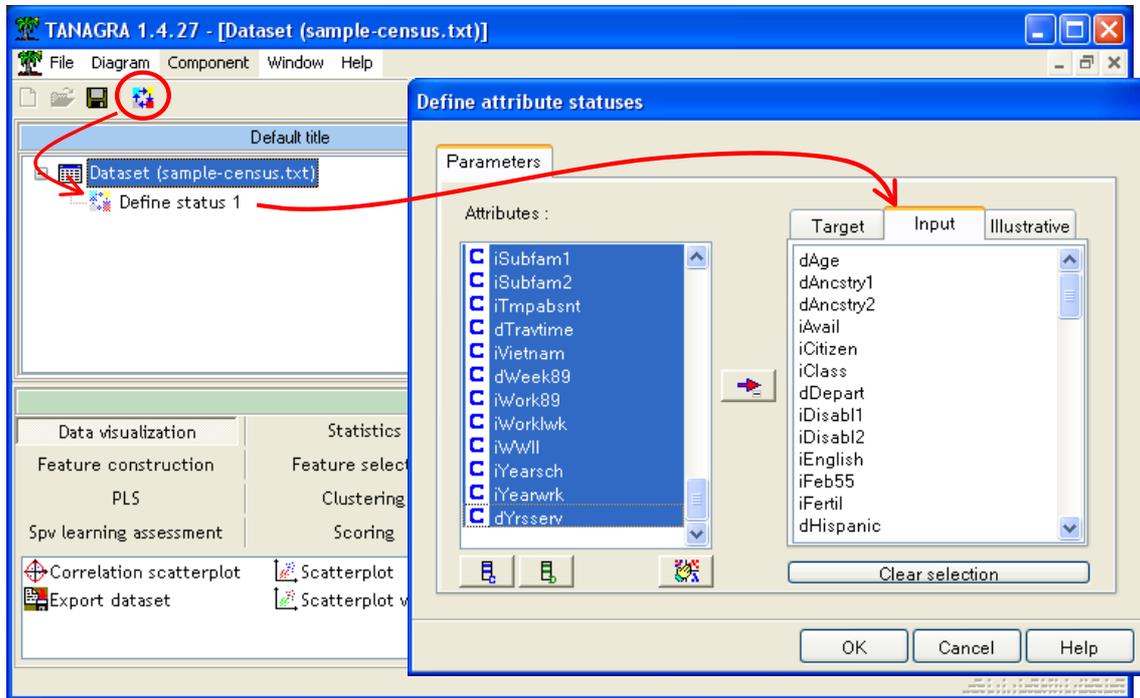


We check that we have 500,000 observations and 58 variables.

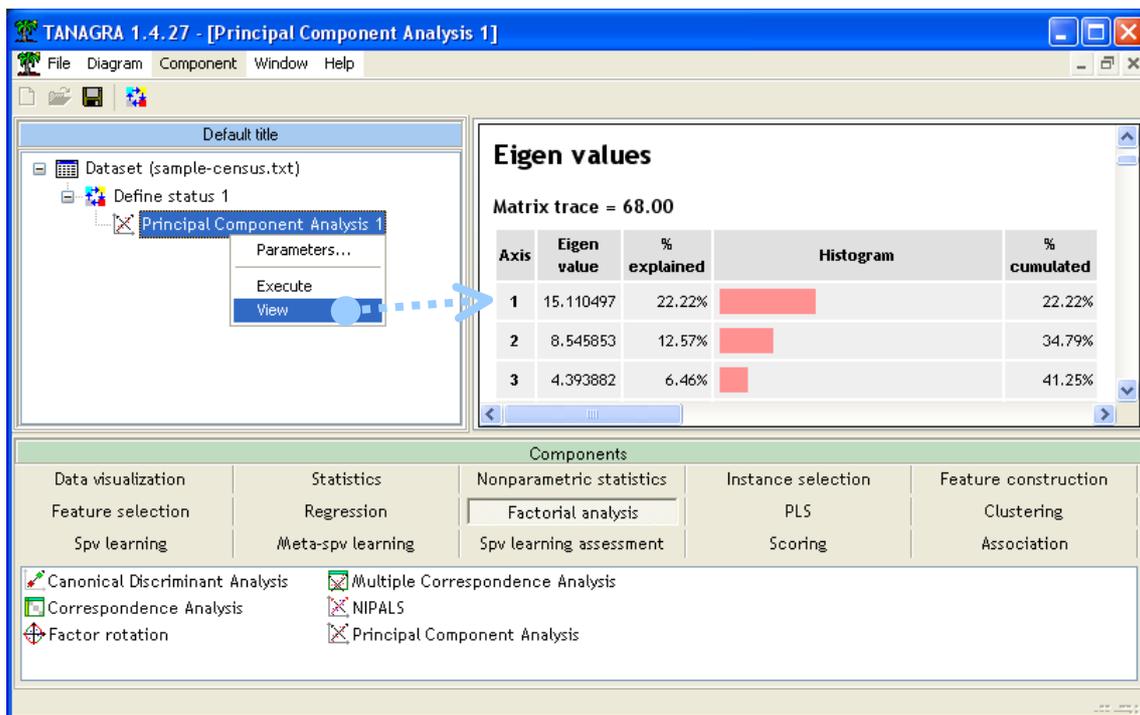


### 3.2 PCA – Principal Component Analysis

Before we launch the PCA, we must define the type of each variable. We insert the DEFINE STATUS component into the diagram. We set all the variables as INPUT.

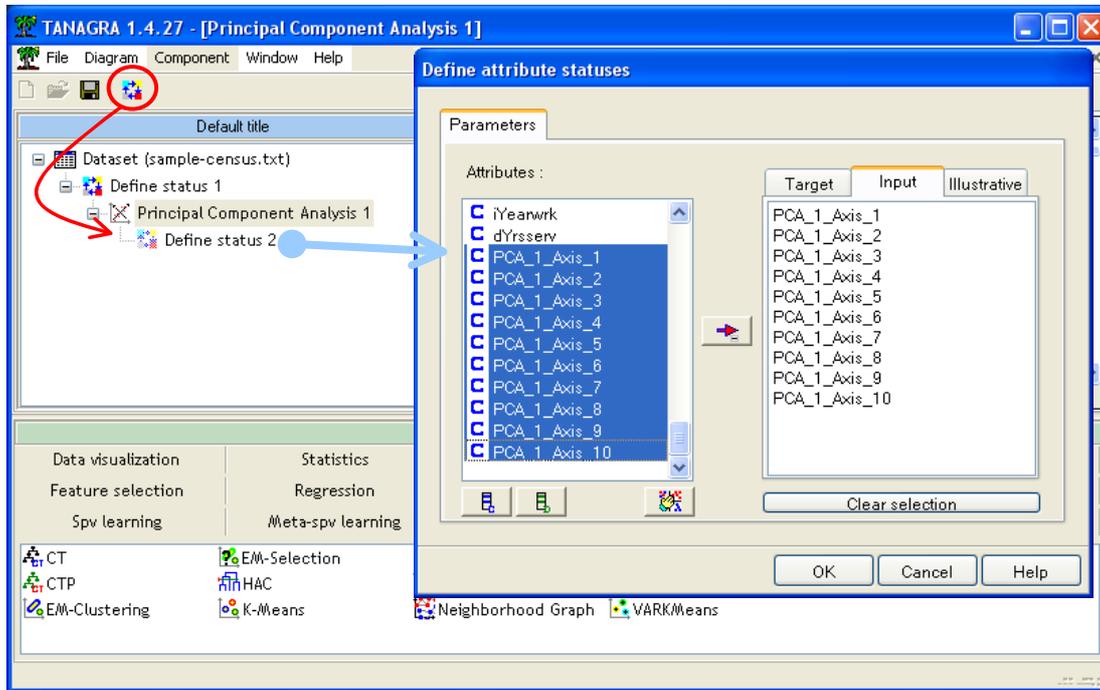


Then we can add the PRINCIPAL COMPONENT ANALYSIS component (FACTORIAL ANALYSIS tab). It computes automatically the 10 first factors which are usable in the subsequent part of the diagram. We click on the VIEW menu to obtain the results.

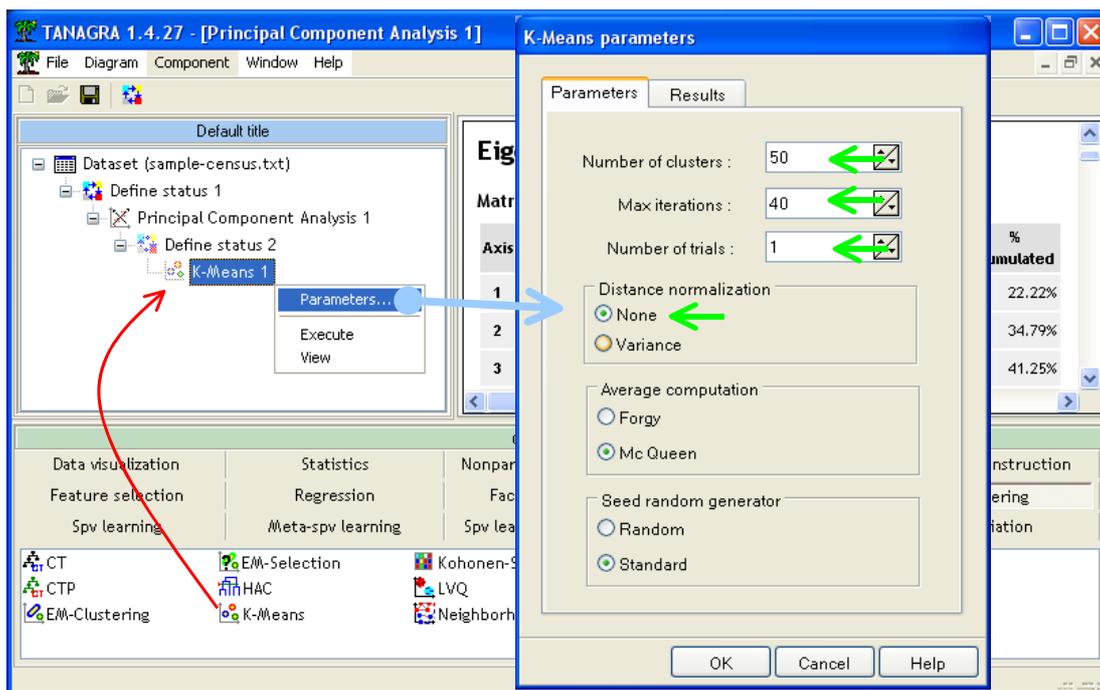


### 3.3 K-Means on the Factors of PCA

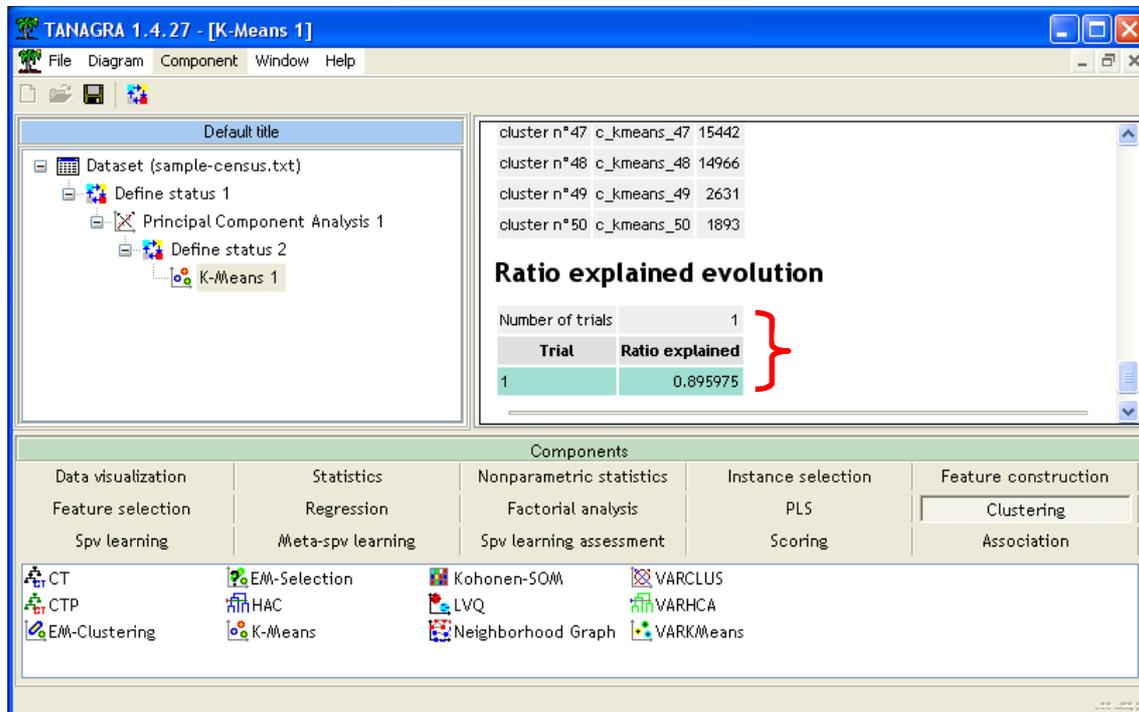
We want to perform the K-Means algorithm on the factors supplied by the PCA. The idea is to smooth the information coming from the dataset, by removing the irrelevant one such as noise. We insert again the DEFINE STATUS component into the diagram. We set as INPUT the factors PCA\_1\_AXIS\_1... PCA\_1\_AXIS\_10.



We add the K-MEANS component (CLUSTERING tab). We click on the PARAMETERS menu in order to specify the settings of the approach.



We want 50 clusters (Number of Clusters). We do only one optimization process (Number of trials = 1). The maximum number of iteration for one process is 40 (Max iterations). We do not normalize the variables; we set NONE to the DISTANCE NORMALIZATION. We validate and we click on the VIEW menu.



Tanagra supplies the number of instances in each cluster. We note that the part of variation explained by the clustering is 89.6%.

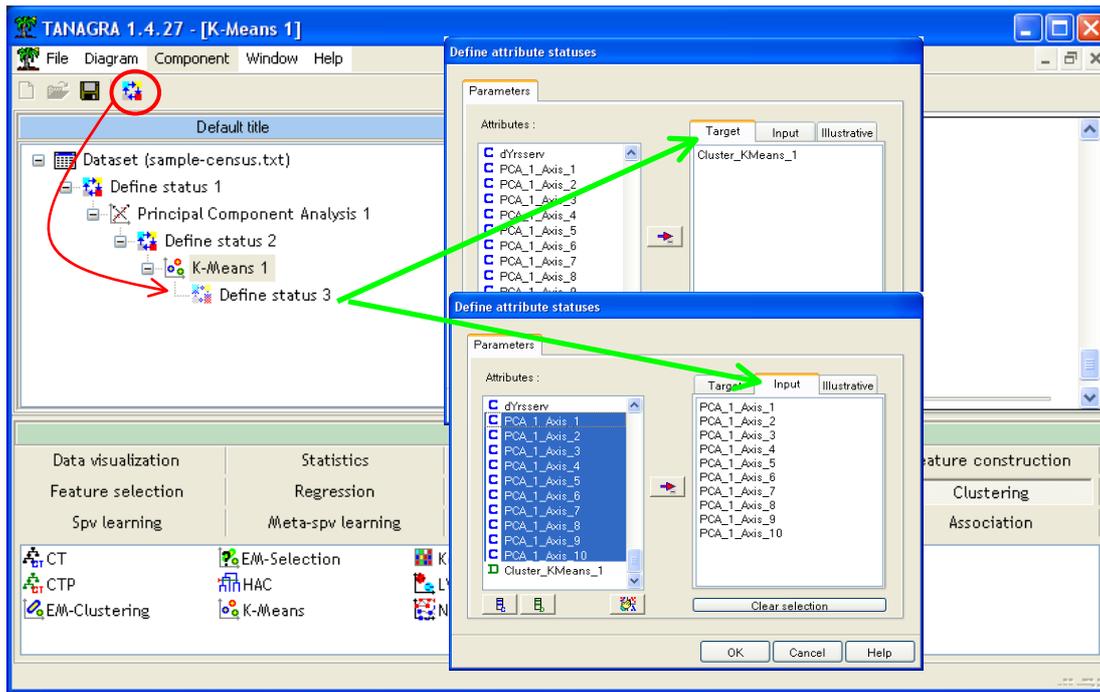
### 3.4 HAC from the pre-clusters supplied by the K-MEANS process

We want now to perform the HCA algorithm starting from the pre-clusters computed with the K-Means process.

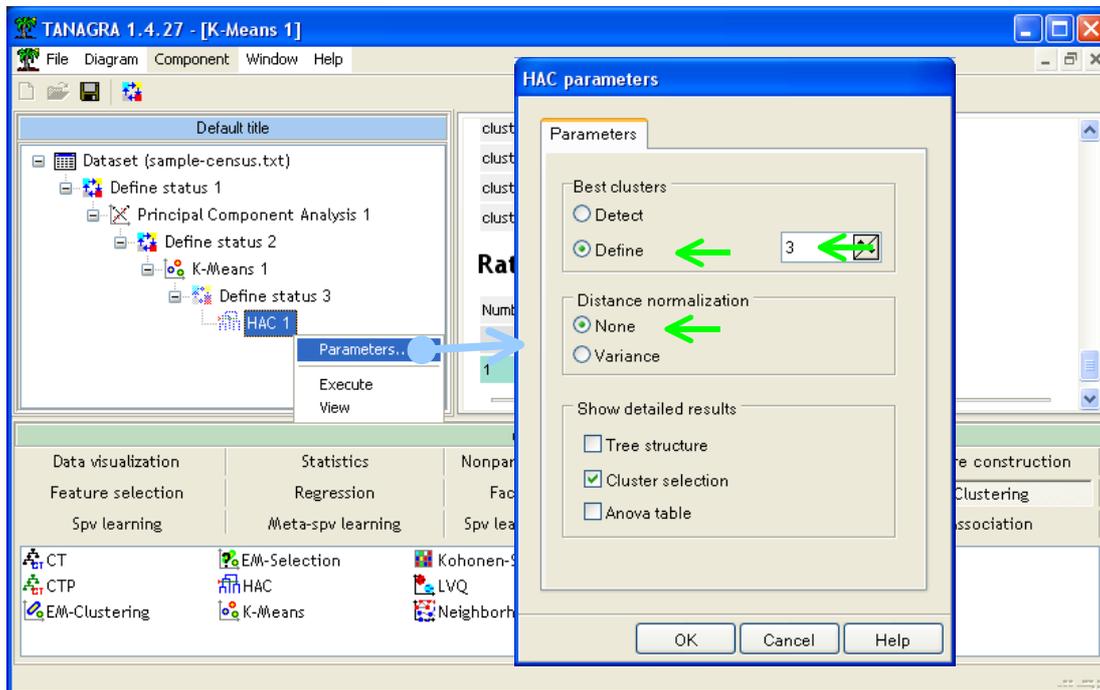
We insert the DEFINE STATUS component into the diagram. We set as TARGET the pre-cluster variable (CLUSTER\_KMEANS\_1). This specification is important. Otherwise, Tanagra tries to create the dendrogram starting from the 500,000 instances.

We set as INPUT the factors supplied by the PCA.

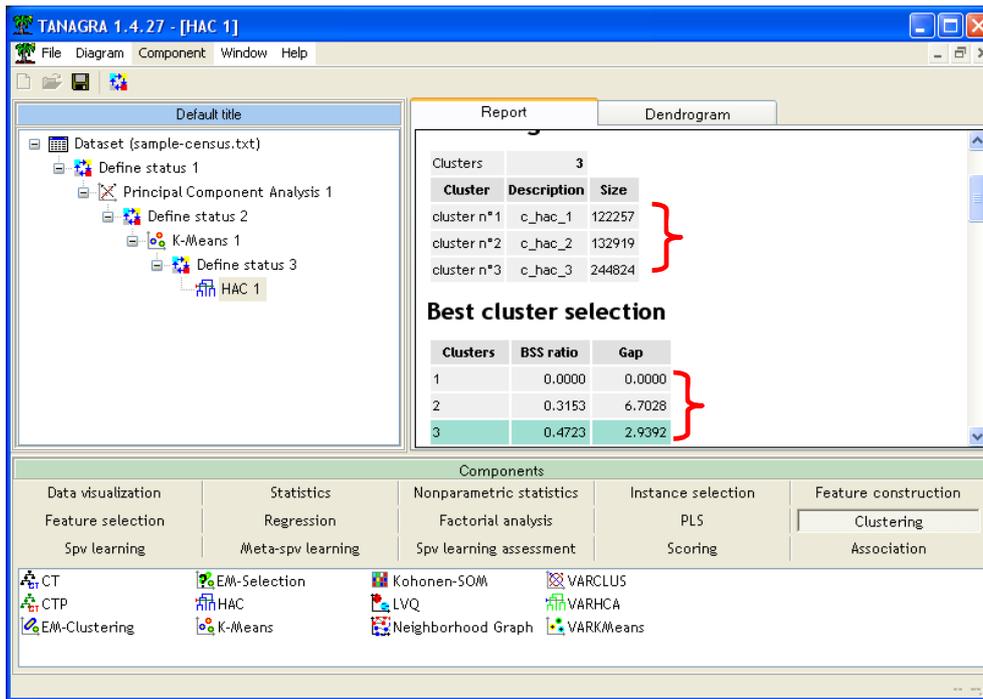
**Note:** We note that any categorical variable can be used as TARGET. It can be computed by other clustering algorithm (SOM; ...). It can be also a natural grouping defined in the dataset (e.g. various districts in a town; according to the job category of the header of the family; ...).



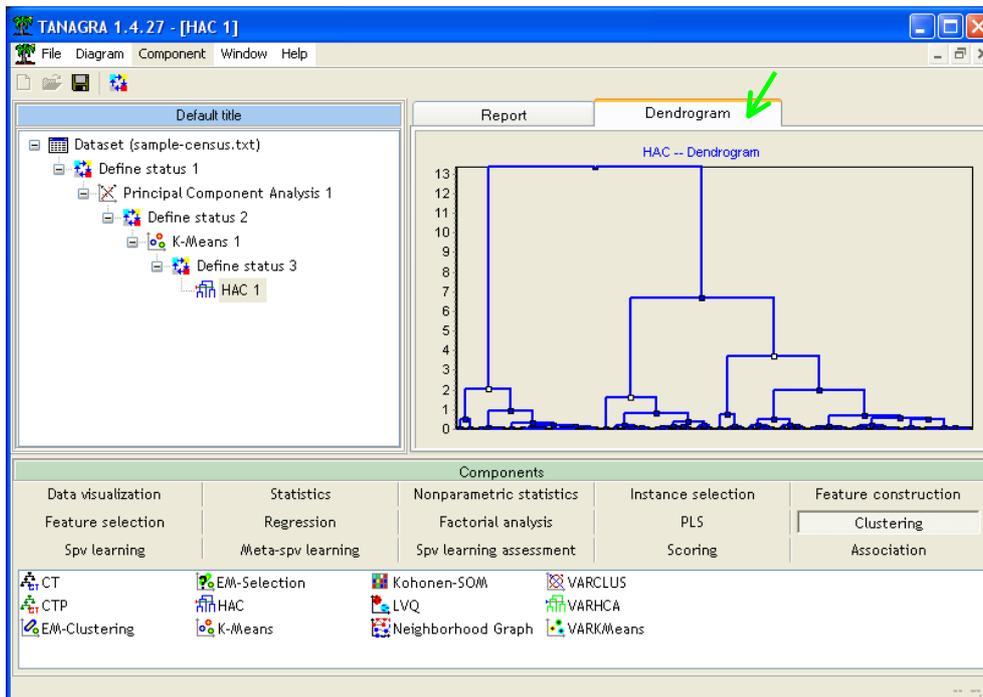
Then we add the HCA component (CLUSTERING tab) into the diagram. We set the parameters in order to obtain 3 groups. We will see why when we will analyze the results. We do not standardize the variables because we use the factors of PCA as INPUT variables.



We click on the VIEW menu. In the report, we observe the number of instances in each cluster. We have also the proportion of variation explained by the partitioning.



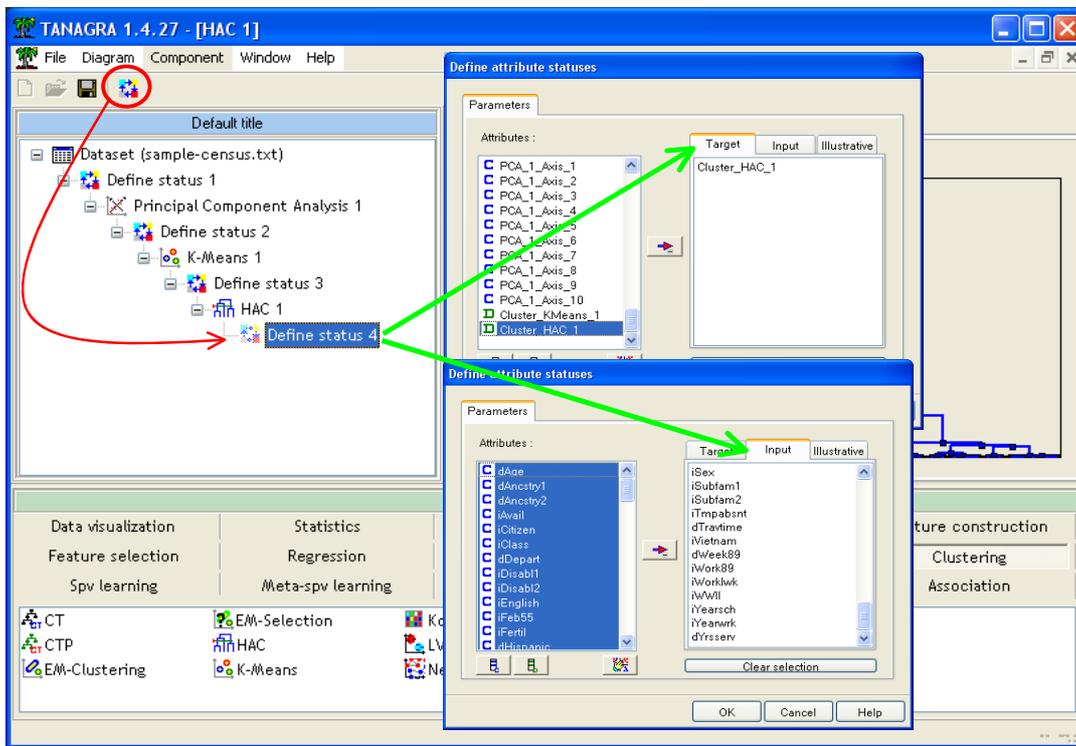
In the second tab (DENDROGRAM) of the visualization window, we observe the dendrogram. Indeed, the segmentation into 3 groups seems the most relevant.



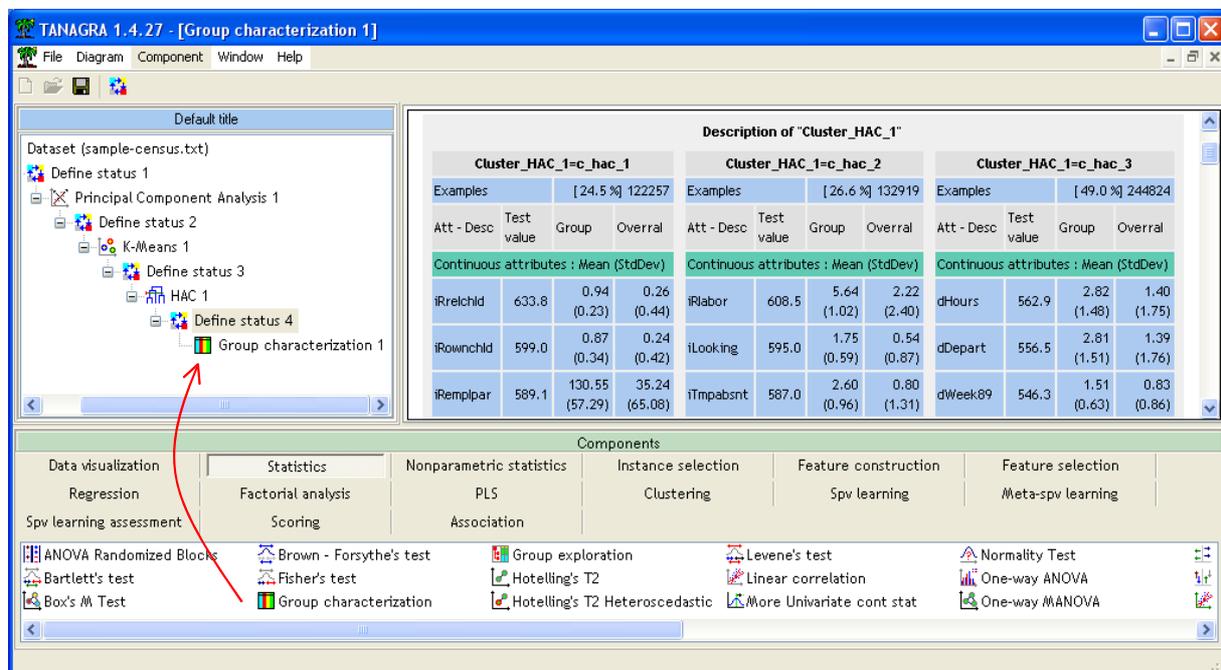
### 3.5 Interpreting the clusters

To characterize the groups, we can use the GROUP CHARACTERIZATION component (see <http://data-mining-tutorials.blogspot.com/2009/06/k-means-comparison-of-free-tools.html> for details about characterization of clusters). We add a DEFINE STATUS component into the diagram. We set CLUSTER\_HAC\_1 as TARGET. This is the cluster membership variable. It associates each

instance to a cluster. We set as INPUT the original variables. We note that we can set as INPUT other variables which are not used during the computation. It allows often to strengthen the interpretation of the groups.



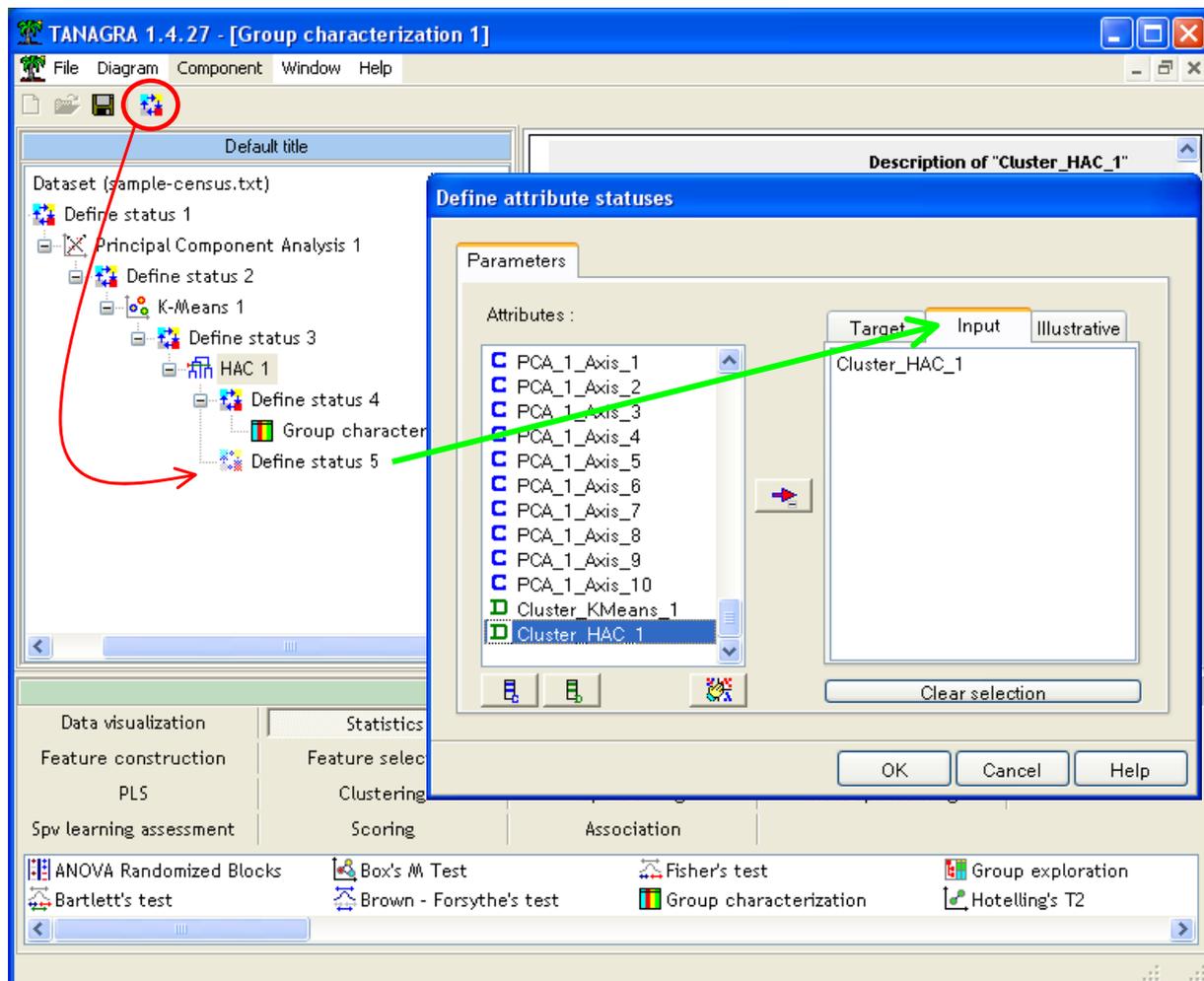
Then we insert the GROUP CHARACTERIZATION component (STATISTICS tab).



We have the number of instances into each cluster. We can compare some descriptive statistics indicators (mean for continuous variables, proportion for discrete one) in order to evaluate the importance of each variable in the segmentation result.

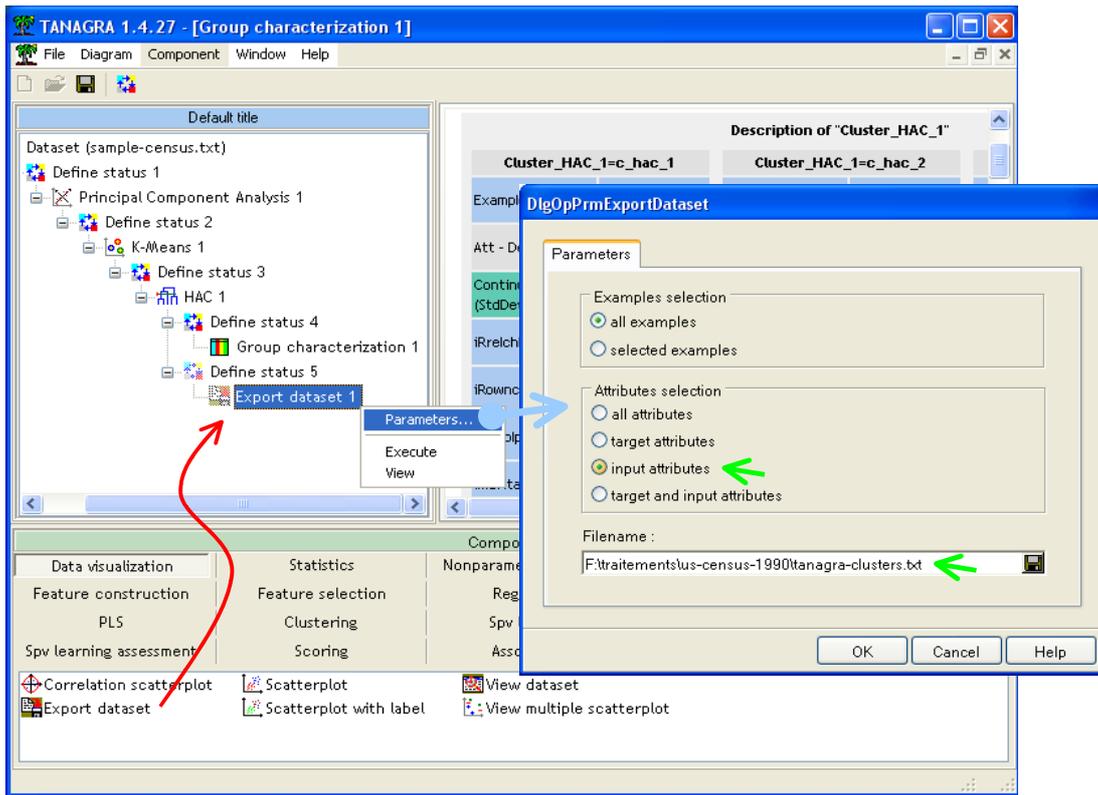
### 3.6 Exporting the dataset including the CLUSTER variable

Below we want to compare the results (cluster membership) of Tanagra to those of R. To prepare this comparison, we want to export the cluster membership column generated by Tanagra. We insert the DEFINE STATUS component. We set the CLUSTER\_HAC\_1 column as INPUT so that this one is the only exported.

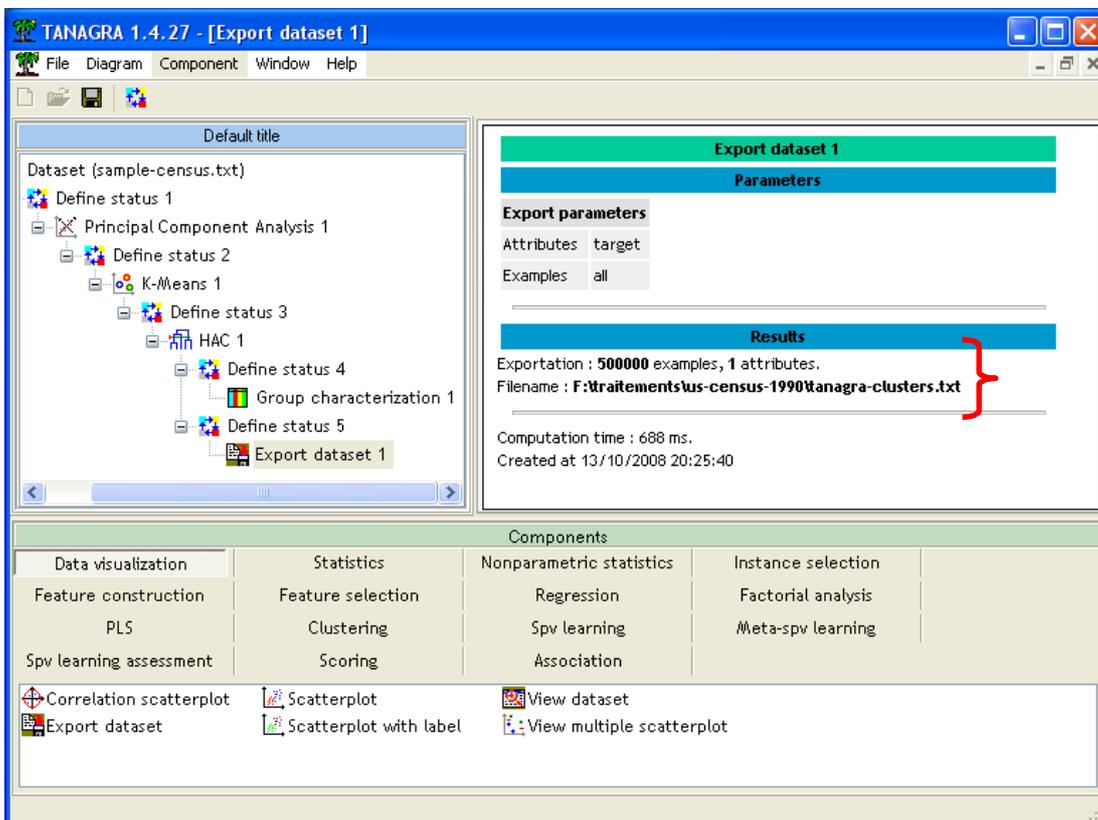


Then we use the EXPORT DATASET component (DATA VISUALISATION tab).

We set the appropriate parameters (PARAMETERS menu). We want to export all the examples but only the CLUSTER (INPUT into the preceding DEFINE STATUS component) column. We specify also the file name "Tanagra-clusters.txt". We will import this data file into R below.



We click on the VIEW menu. The exportation is performed. The number of instances and variables exported is reported.



## 4 Two-step clustering with R

In this section, we perform the same process with R.

### 4.1 Importing the dataset

We set the following instructions in order to import the dataset. Of course, the reader uses a different directory on its computer.

```
#chargement des données
setwd("F:/traitements/us-census-1990")
print("Chargement")
donnees <- read.table(file="sample-census.txt", header=T, dec=".")
```

### 4.2 Principal component analysis

We use the “`princomp(.)`” procedure. The variables are standardized (`cor = T`) i.e. the procedure performs a diagonalization of the correlation matrix. We retrieve the 10 first factors.

```
#acp, en fixant le nombre de facteurs à extraire à 10
nb.facteurs <- 10
print("ACP")
acp <- princomp(donnees, scores=T, cor=T)
facteurs.acp <- acp$scores[, 1:nb.facteurs]
```

### 4.3 K-Means

We perform a K-Means procedure on the factors of the PCA. We ask 50 clusters. The maximum number of iteration is 40. We set here the same settings as for Tanagra. We retrieve the cluster membership column.

```
#initialiser toujours de la même manière le générateur de nombre aléatoire
#pour avoir le même résultat d'une exécution à l'autre
set.seed(10)

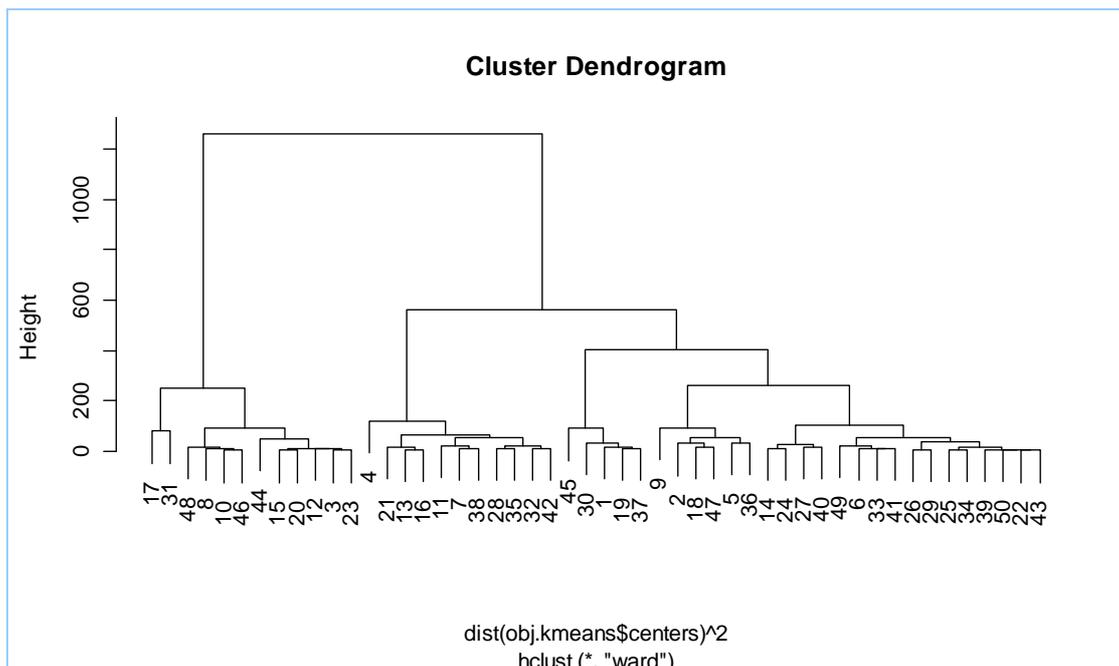
#K-Means, en fixant le nombre de pré-classes
nb.preclusters <- 50
print("K-Means")
obj.kmeans <- kmeans(facteurs.acp, centers=nb.preclusters, algorithm="MacQueen", iter.max=40)
clus.kmeans <- obj.kmeans$cluster
id.clus.kmeans <- factor(clus.kmeans)
```

### 4.4 HAC from the pre-clusters of K-Means

In the next step, we launch the HCA. We set carefully the parameters. The process starts from the center of the pre-clusters. We use the “Ward” strategy.

```
#CAH à partir des pré-classes du K-Means
print("CAH")
obj.cah <- hclust(dist(obj.kmeans$centers)^2, method="ward", members=table(id.clus.kmeans))
plot(obj.cah)
```

We obtain the following dendrogram.



Here also, the segmentation into 3 groups seems relevant. We ask to R to set this partitioning.

```
#couper l'arbre à X classes
nb.finalclusters <- 3
groupe <- cutree(obj.cah,k=nb.finalclusters)
```

#### 4.5 Assigning the cluster id to each instance

At this step, R makes only the correspondence between the group supplied by the HAC and the initial pre-clusters computed by the K-Means algorithm. When we print the “groupe” vector, we obtain the following values.

```
> print(groupe)
 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 27 28 29 30 31 32
 1  1  2  3  1  1  3  2  1  2  3  2  3  1  2  3  2  1  1  2  3  1  2  1  1  1  1  3  1  1  2  3
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
 1  1  3  1  1  3  1  1  1  3  1  2  1  2  1  2  1  1
```

We must detect the cluster membership for each instance. To do that, (1) we make the correspondence between the id of each instance and the id of clusters supplied by the K-means (“id.clus.kmeans” vector), and then (2) we make the correspondence between the k-means clusters and the final clusters supplied by the HAC using the “groupe” vector.

```
#affectation des classes aux observations
#en faisant la correspondance avec les groupes du K-Means
cah.classe <- numeric(nrow(donnees))
for (k in 1:nlevels(id.clus.kmeans)){
  cah.classe[id.clus.kmeans == k] <- groupe[k]
}
cah.classe <- as.factor(cah.classe)
```

Now, we can compute the number of examples assigned to each cluster.

```
#effectifs par classe
print(summary(cah.classe))
```

We obtain

```
> #effectifs par classe
> print(summary(cah.classe))
      1      2      3
247204 122174 130622
```

Of course, we can characterize each cluster by computing conditional descriptive statistics. For instance, we obtain the following values for the 1<sup>st</sup> cluster (which corresponds to the 3<sup>rd</sup> cluster of Tanagra).

```
> colMeans(donnees[cah.classe==1,c("dHours","dDepart","dWeek89")])
 dHours dDepart dWeek89
 2.793029 2.781249 1.500494
```

## 5 Conclusion

### 5.1 Concordance of the results

We want to know if the results provided by the 2 tools are coherent. Actually, K-Means is the only step which can insert a discrepancy between the results, because the tools do not use the same initialization. We do not control this step in the K-Means process. How does that influence the final segmentation?

We compute a cross tabulation between the groups supplied by Tanagra and R.

```
#comparaison des caractéristiques des classes avec ceux de Tanagra
#chargement des classes produites par Tanagra
setwd("D:/DataMining/Databases_for_mining/comparison_TOW/clustering_big_dataset")
tanagra.output <- read.table(file="tanagra-clusters.txt",dec=".",header=T)
tanagra.classe <- tanagra.output$Cluster_HAC_1

#croisement des classes Tanagra et R
croisement <- table(tanagra.classe,cah.classe)
print(croisement)
```

We obtain the following results. The groups are strongly coherent.

```
> #croisement des classes Tanagra et R
> croisement <- table(tanagra.classe,cah.classe)
> print(croisement)
      cah.classe
tanagra.classe  1      2      3
c_hac_1         46 122155     56
c_hac_2       2899      1 130019
c_hac_3     244259     18     547
```

## 5.2 Computation time and memory occupation

As we say above, R cannot handle the whole dataset. For this reason, we take a sample with 500,000 examples. Below, we give the **computation time** for each step of the process. We give also the computation characteristics when Tanagra treats the whole dataset with 2,458,285 instances.

| Operation                    | Handling the sample<br>500.000 observations |          | Handling the whole<br>database<br>2.458.285 observations |
|------------------------------|---|----------|--|
|                              | Tanagra (sec.)                              | R (sec.) | Tanagra (sec.)   |
| Data importation             | 28  | 24       | 145  |
| PCA (10 factors computed)    | 61  | 37       | 294  |
| K-Means (40 iterations max.) | 56  | 27       | 227  |
| HCA for 50 pre-clusters      | 1   | 0.1      | 4  |

When it can handle the dataset, R is really fast. Aside the K-Means step which relies on a heuristic, all the procedures are deterministic. The computation times are comparable. We observe that R is really faster for the PCA and the HCA.

About the **memory occupation**, R is more costly. Perhaps it is an inherent limitation of R; or perhaps, I do not know how to set properly the settings of the tool (by adjusting judiciously the memory limits with the **memory.limitation(.)** command for instance). I do not have this expertise level for R.

| Operation                               | Handling the sample<br>(500.000 observations) |        | Handling the whole<br>database (2.458.285<br>observations) |
|---|---|--------|--|
|   | Tanagra (Mo)                                  | R (Mo) | Tanagra (Mo)   |
| Memory occupation during the<br>process | 182   | 488    | 932  |

**About the treatment of the whole data file with Tanagra (2,458,285 observations)**, we insert in the table above the computation time of each step (last column). The memory occupation is never

exceeded 932 MB during the process. We give below the characteristics of the 3 groups obtained at the end of the analysis. The results are very similar to those obtained on the sample of 500,000 observations. This is not really surprising. Dealing with a sample, when it is well defined, gives consistent results.

| Results                               |            |                  |               |                                       |            |                  |             |                                       |            |                   |             |
|---------------------------------------|------------|------------------|---------------|---------------------------------------|------------|------------------|-------------|---------------------------------------|------------|-------------------|-------------|
| Description of "Cluster_HAC_1"        |            |                  |               |                                       |            |                  |             |                                       |            |                   |             |
| Cluster_HAC_1=c_hac_1                 |            |                  |               | Cluster_HAC_1=c_hac_2                 |            |                  |             | Cluster_HAC_1=c_hac_3                 |            |                   |             |
| Examples                              |            | [ 24.5 %] 603494 |               | Examples                              |            | [ 23.4 %] 575038 |             | Examples                              |            | [ 52.1 %] 1279753 |             |
| Att - Desc                            | Test value | Group            | Overall       | Att - Desc                            | Test value | Group            | Overall     | Att - Desc                            | Test value | Group             | Overall     |
| Continuous attributes : Mean (StdDev) |            |                  |               | Continuous attributes : Mean (StdDev) |            |                  |             | Continuous attributes : Mean (StdDev) |            |                   |             |
| iRrelchld                             | 1404.0     | 0.94 (0.24)      | 0.26 (0.44)   | iRlabor                               | 1228.4     | 5.61 (1.06)      | 2.21 (2.40) | dHours                                | 1169.9     | 2.65 (1.59)       | 1.40 (1.75) |
| iRowchld                              | 1327.5     | 0.87 (0.34)      | 0.24 (0.43)   | iLooking                              | 1202.6     | 1.74 (0.59)      | 0.54 (0.87) | dDepart                               | 1157.1     | 2.64 (1.60)       | 1.39 (1.76) |
| iRemppar                              | 1304.8     | 130.38 (57.46)   | 35.33 (65.15) | iTmpabsnt                             | 1193.5     | 2.60 (0.97)      | 0.79 (1.31) | dWeek89                               | 1137.3     | 1.43 (0.70)       | 0.83 (0.86) |