1 Objectif

Comparer les performances de Tanagra et R pour une CAH Mixte sur un fichier de grande taille.

La CAH (classification ascendante hiérarchique) est une technique de classification automatique (clustering en anglais). Elle vise à produire un regroupement des individus de manière à ce que les individus du même groupe soient semblables, des individus dans des groupes différents soient dissemblables.

Le succès de la CAH repose sur sa capacité à produire des partitions emboîtées. Au lieu de fournir une solution clé en main, irréversible, elle donne la possibilité de choisir, parmi les regroupements proposés, celui qui correspond au mieux aux contraintes de l'étude et aux objectifs de l'analyste. Cet avantage s'accompagne d'une représentation graphique, le dendrogramme. Il nous suggère, dans le continuum des solutions envisageables, celles qui semblent les plus pertinentes.

Son principal défaut est le temps de calcul. Il devient vite rédhibitoire dès que le nombre d'observations est élevé. Pour dépasser cet écueil, on procède alors à la **CAH Mixte**. Elle consiste à faire précéder la CAH proprement dite par une phase de pré-regroupement, en utilisant un algorithme des nuées dynamiques par exemple, la CAH prend alors comme point de départ ces pré-classes. De fait, avec cette stratégie, il devient possible de traiter de très grands fichiers tout en bénéficiant des avantages de la CAH.

Cette approche a déjà été largement abordée dans un de nos anciens didacticiels (voir CAH Mixte – Le fichier IRIS de Fisher, http://tutoriels-data-mining.blogspot.com/2008/03/cah-mixte-le-fichier-iris-de-fisher.html). La méthode est par ailleurs longuement décrite dans l'ouvrage de Lebart et al. (2000)[1]. Conformément à ce qui est préconisé par les auteurs, nous réalisons la classification sur les axes factoriels de l'ACP (analyse en composante principale). L'idée est de « lisser » les informations exploitées en évacuant les fluctuations aléatoires.

L'enjeu dans ce didacticiel est de **mettre en œuvre cette stratégie sur un fichier de** taille relativement considérable, avec **500.000 observations et 68 variables**. Nous utiliserons **Tanagra 1.4.27** et **R 2.7.2**. Nous nous en tenons à ces deux logiciels. En effet, il n'est pas possible d'implémenter la CAH Mixte avec les autres logiciels libres (Weka, Orange, Knime, Rapidminer). Et lancer directement la CAH standard sur un tel fichier n'est pas raisonnable.

On remarquera au passage la puissance de R à ce niveau. Si Tanagra a été pensé pour mettre en œuvre la CAH Mixte (suite à une lecture assidue de la référence Lebart et al.), je ne suis pas sûr qu'il en ait été de même pour R. Et pourtant, en cherchant un peu, je me suis rendu compte que tout était à disposition pour réaliser les opérations adéquates avec les packages usuels.

2 Données

Nous utilisons les données « 1990 US Census Data » dans ce didacticiel[2]. Il comporte 68 variables. Certaines d'entre elles sont des variables discrétisées, d'autres sont des variables indicatrices. Nous n'allons pas trop nous attarder dessus, considérant que toutes les variables sont numériques. Notre principal objectif est de montrer la faisabilité de la CAH mixte sur de grands fichiers.

[1] L. Lebart, A. Morineau, M. Piron, « Statistique Exploratoire Multidimensionnelle », Dunod, 2000 ; chapitre 2, sections 2.3 et 2.4.

[2] http://archive.ics.uci.edu/ml/databases/census1990/USCensus1990-desc.html

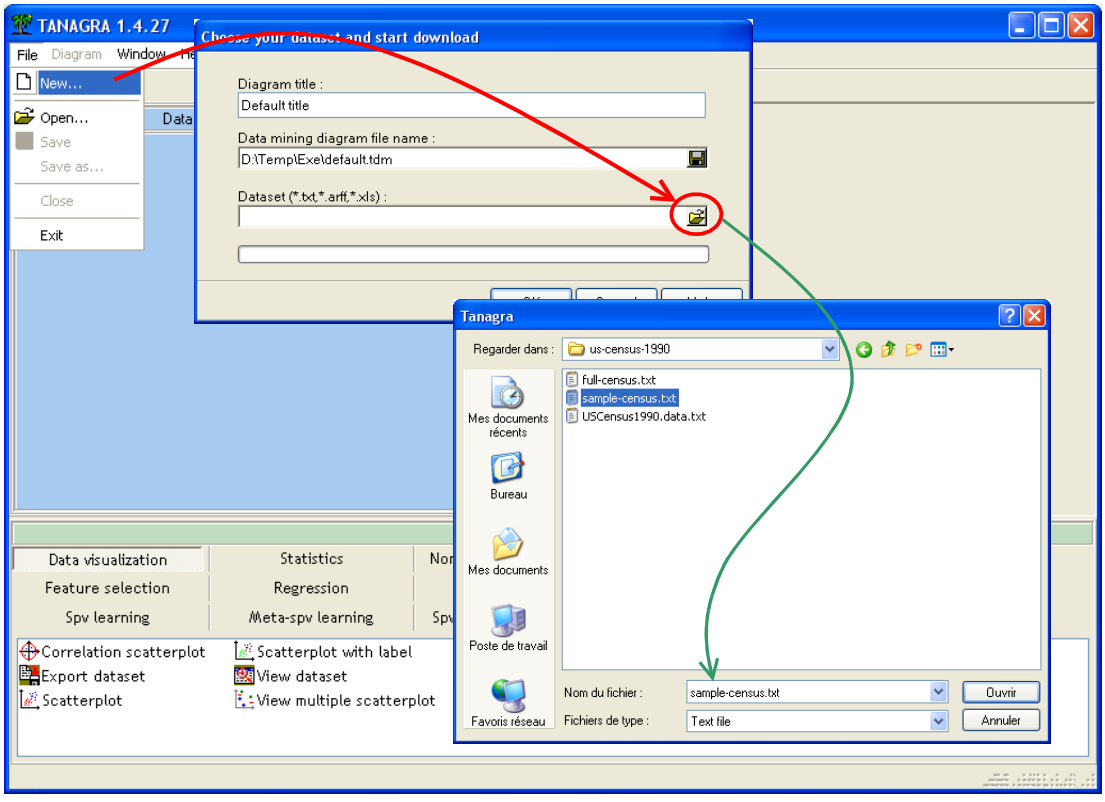
Le fichier originel comporte 2.458.285 observations. Nous en avons extrait aléatoirement en échantillon[3] de 500.000. La raison est que R n'a pas pu mener à bien l'analyse sur le volume initial. Des erreurs lors des allocations mémoires internes sont survenues (ma machine dispose de 2 GB de RAM). J'ai essayé de modifier le paramétrage de la gestion mémoire, mais rien n'y a fait. J'ai également essayé de diminuer graduellement la taille de fichier. A 1.000.000 d'observations, R plante encore. Il faudrait un niveau d'expertise plus élevé que le mien pour trouver la bonne manière de paramétrer R dans ce type de configuration.

Tanagra en revanche a réussi à traiter la totalité du fichier c.-à-d. 2.458.285 observations et 68 variables. Le temps de calcul est un peu plus long bien sûr, avec une occupation mémoire plus importante. Il apparaît que le résultat final, notamment les propriétés des classes (taille relative, variables caractéristiques, etc.), ne diffère guère de celui du traitement d'une fraction du fichier. Ce n'est guère étonnant finalement. L'échantillonnage, quand il est bien réalisé, est une stratégie alternative efficace pour appréhender les gros volumes.

3 CAH Mixte avec Tanagra

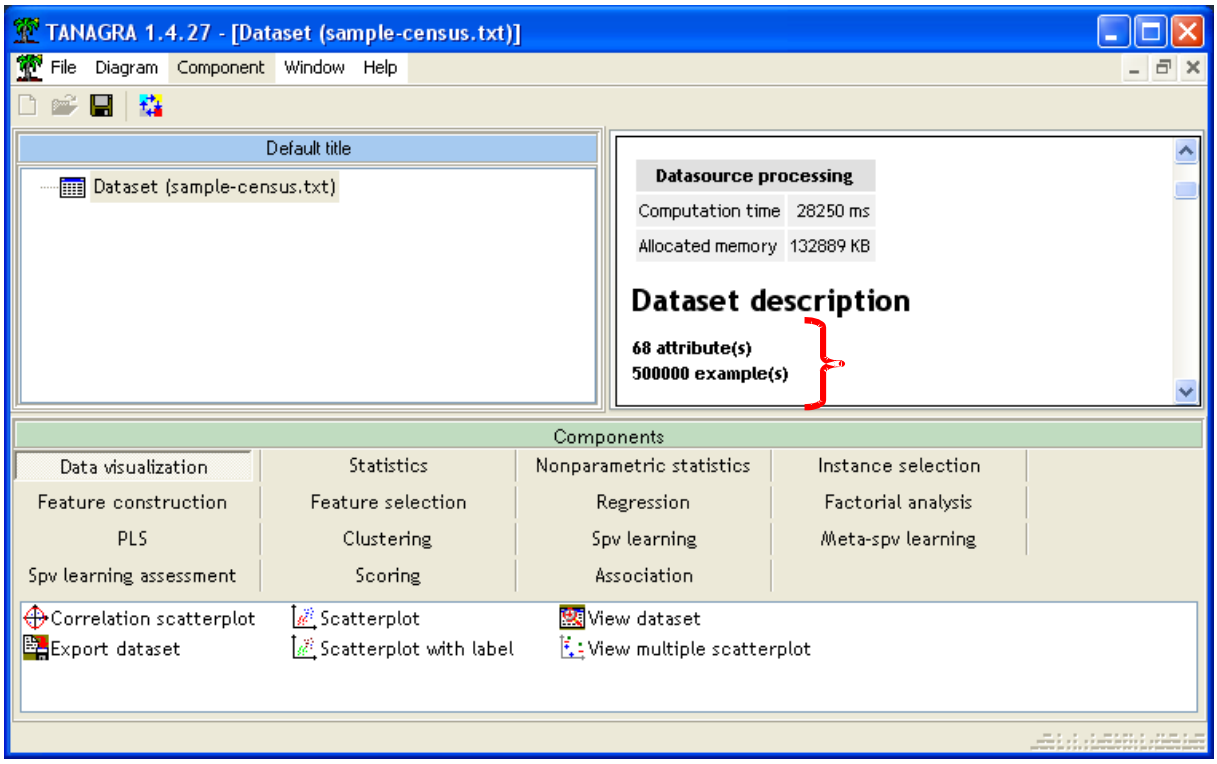
3.1 Importation des données

Première étape, lancer Tanagra. Puis, avec FILE/NEW, créer un nouveau diagramme et importer les données. Nous sélectionnons le fichier « sample-census.txt ».



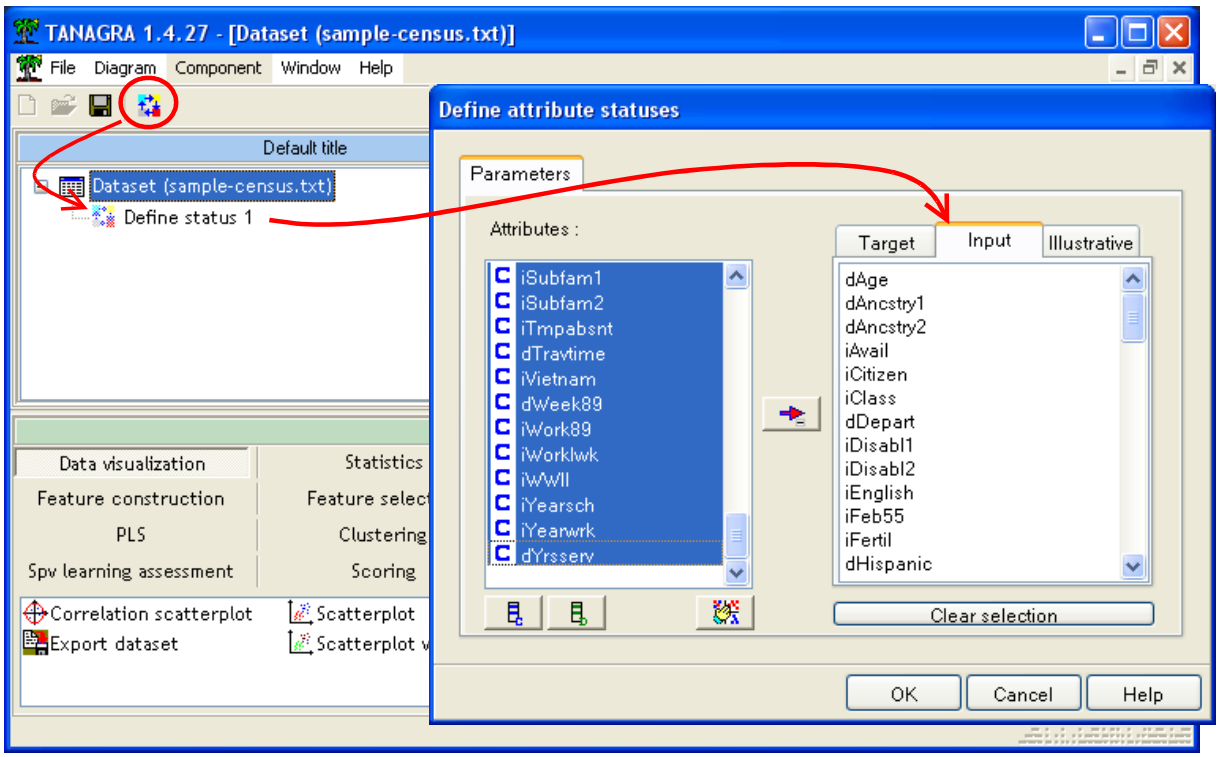
L'importation réalisée, nous vérifions que 500000 observations et 58 variables ont bien été chargés.

[3] http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/sample-census.zip ; l'archive comporte également le code source du script R.



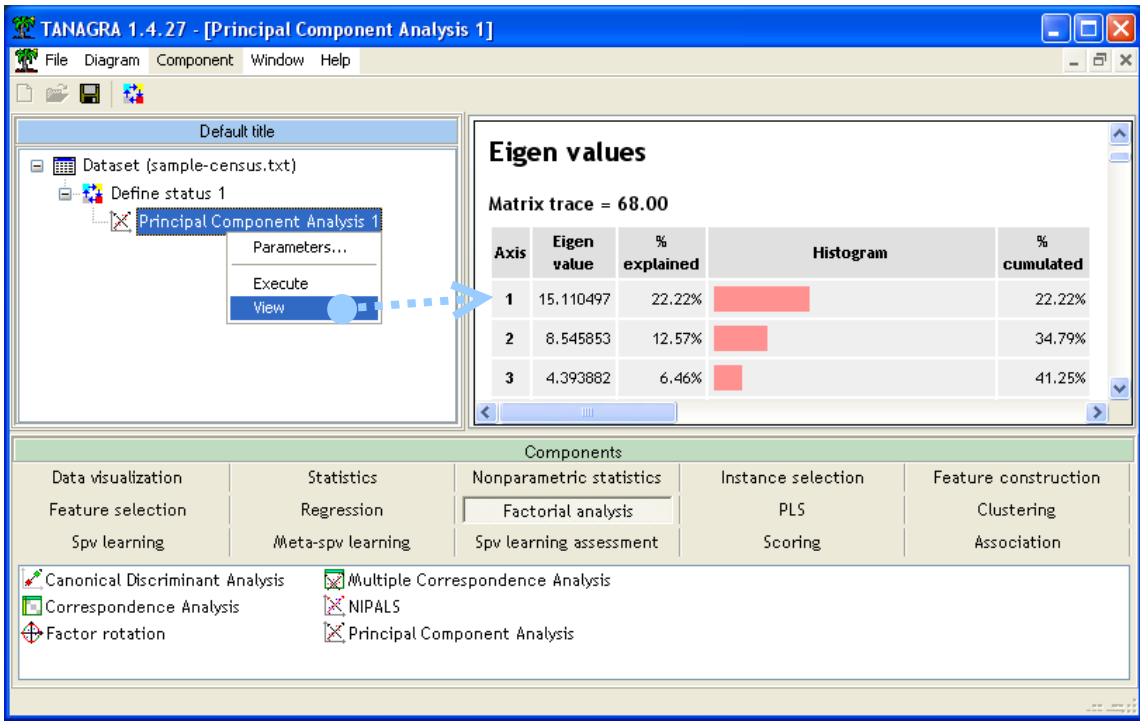
3.2 Analyse en composantes principales (ACP)

Pour effectuer l'ACP, nous devons tout d'abord spécifier le rôle des variables. Nous introduisons le composant DEFINE STATUS via le raccourci dans la barre d'outils. Toutes les variables sont INPUT.



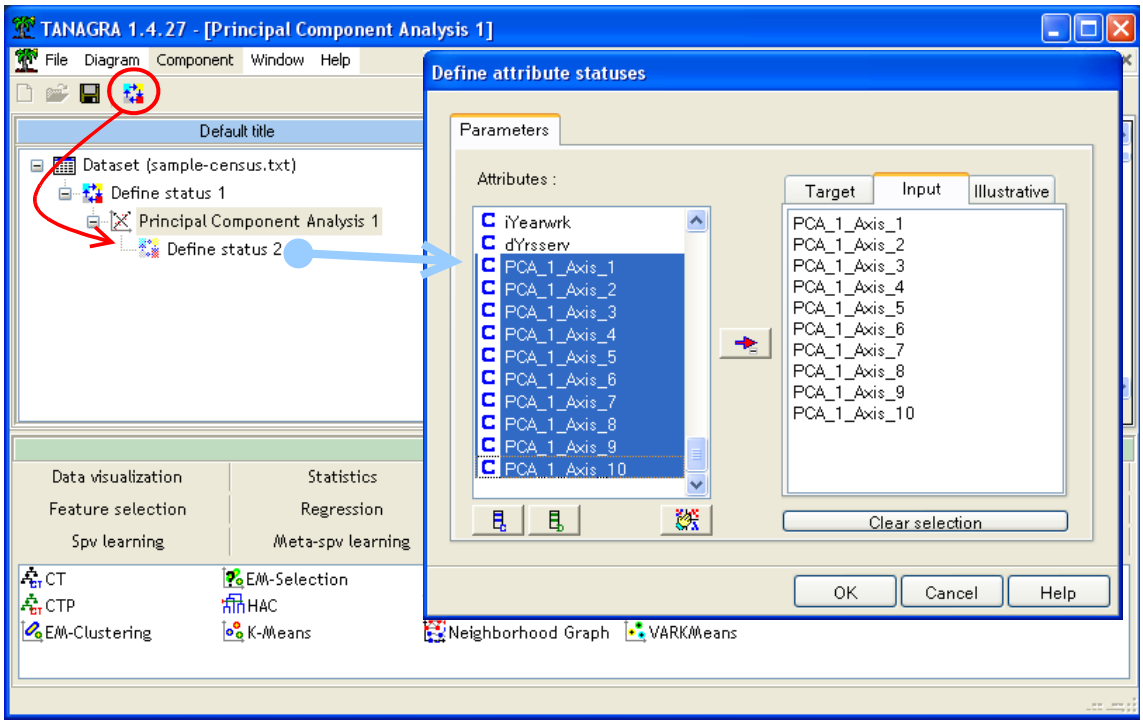
Nous pouvons dès lors insérer le composant PRINCIPAL COMPONENT ANALYSIS (onglet FACTORIAL ANALYSIS) dans le diagramme. Il produit automatiquement les 10 premiers axes factoriels

(paramétrable), utilisables en aval, c'est ce que nous souhaitons. Nous actionnons le menu contextuel VIEW pour accéder aux résultats.

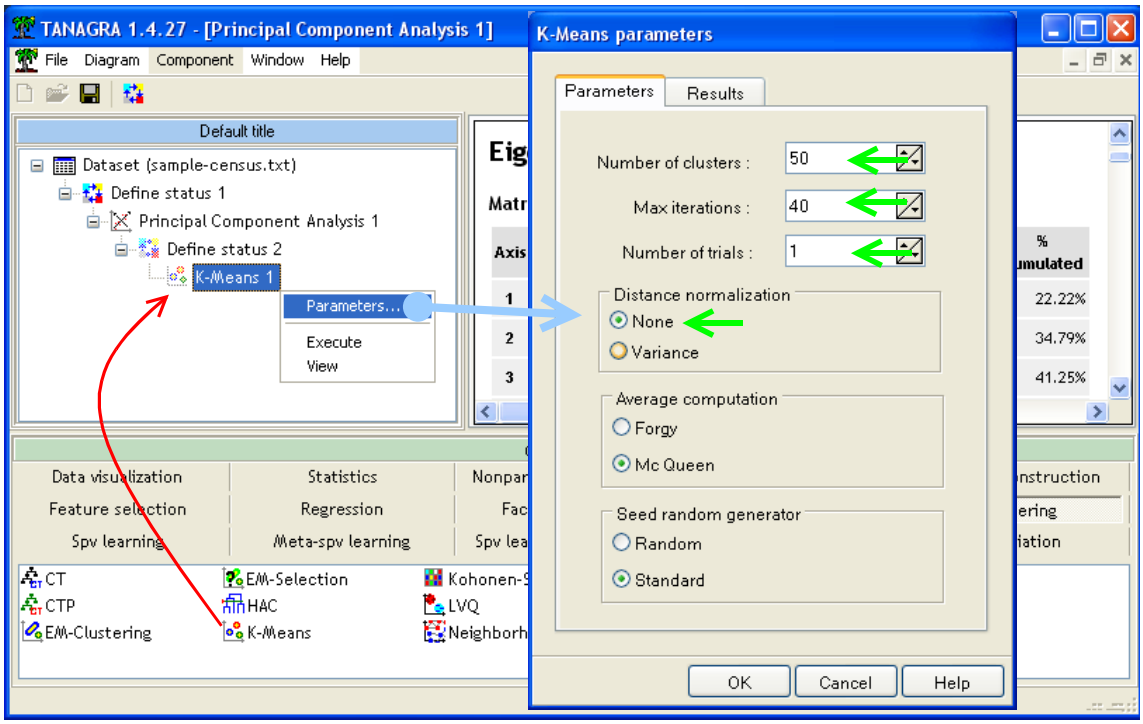


3.3 K-Means (nuées dynamiques) sur facteurs

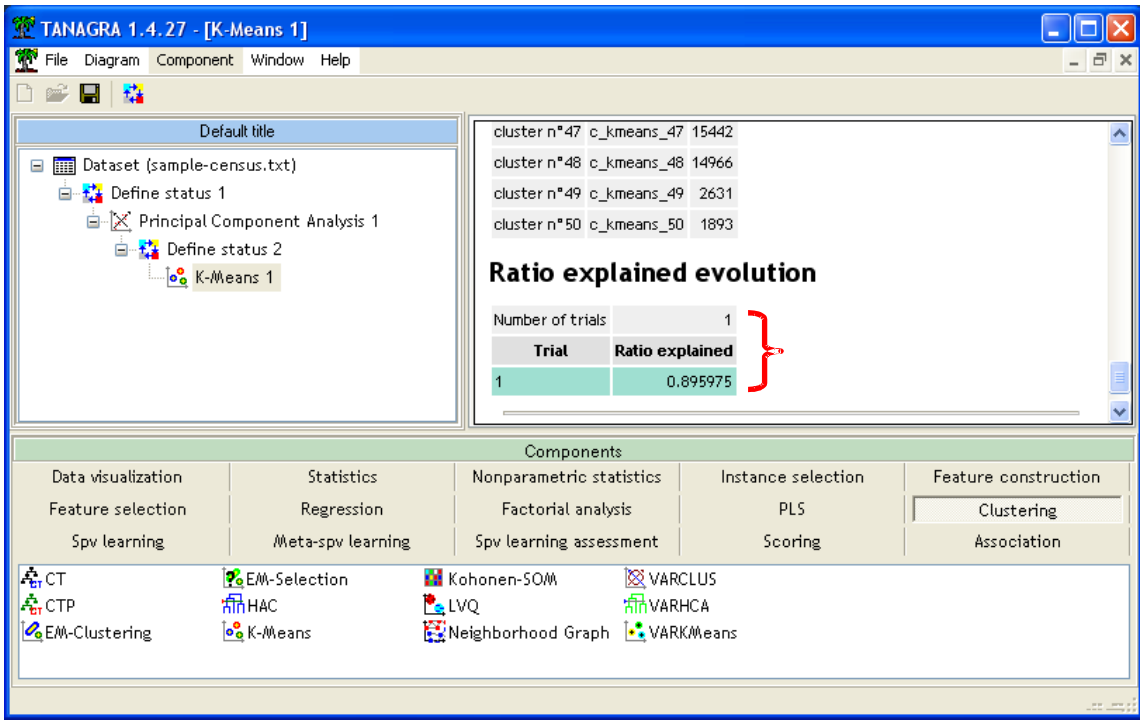
Nous désirons lancer l'algorithme des K-Means sur les facteurs de l'ACP. Nous insérons de nouveau DEFINE STATUS dans le diagramme, nous plaçons en INPUT les variables calculées PCA_1_Axis_1 à PCA_1_AXIS_10.



Nous plaçons le composant K-MEANS (onglet CLUSTERING). Nous le paramétrons en actionnant le menu contextuel PARAMETERS...



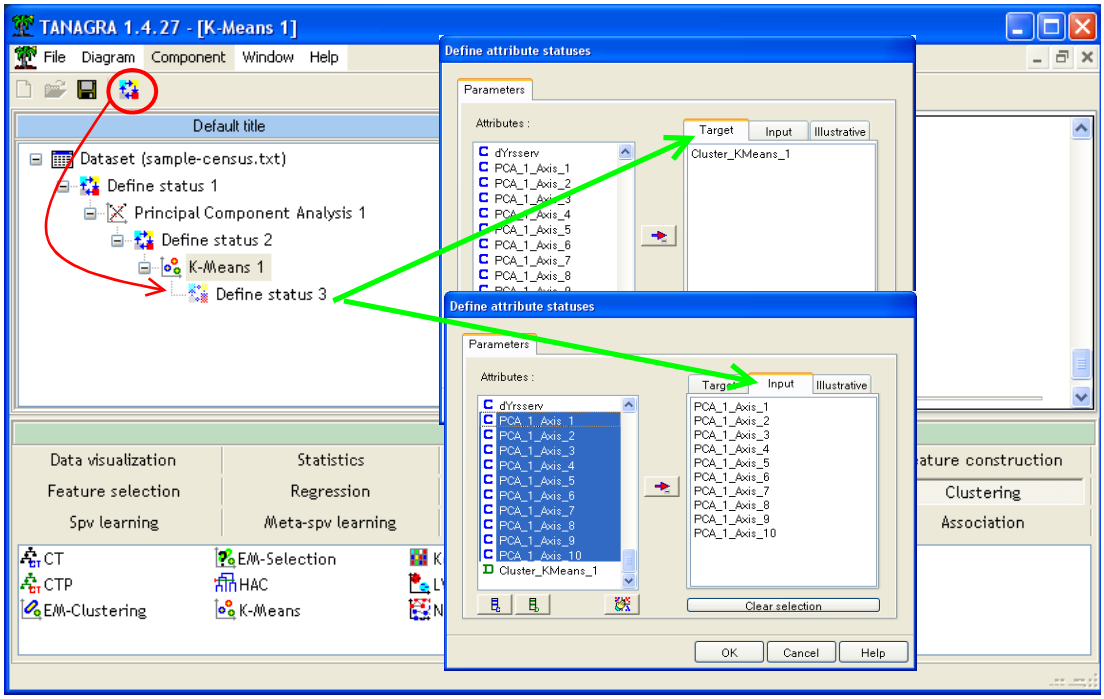
Le nombre de clusters demandé est 50 (Number of Clusters). Nous ne réalisons qu'un seul essai d'optimisation (Number of trials = 1), avec un nombre d'itération maximum à 40 (Max iterations). Attention, les variables (axes factoriels) ne doivent pas être standardisées, nous mettons à NONE le paramètre DISTANCE NORMALIZATION. Nous validons et nous actionnons le menu VIEW.



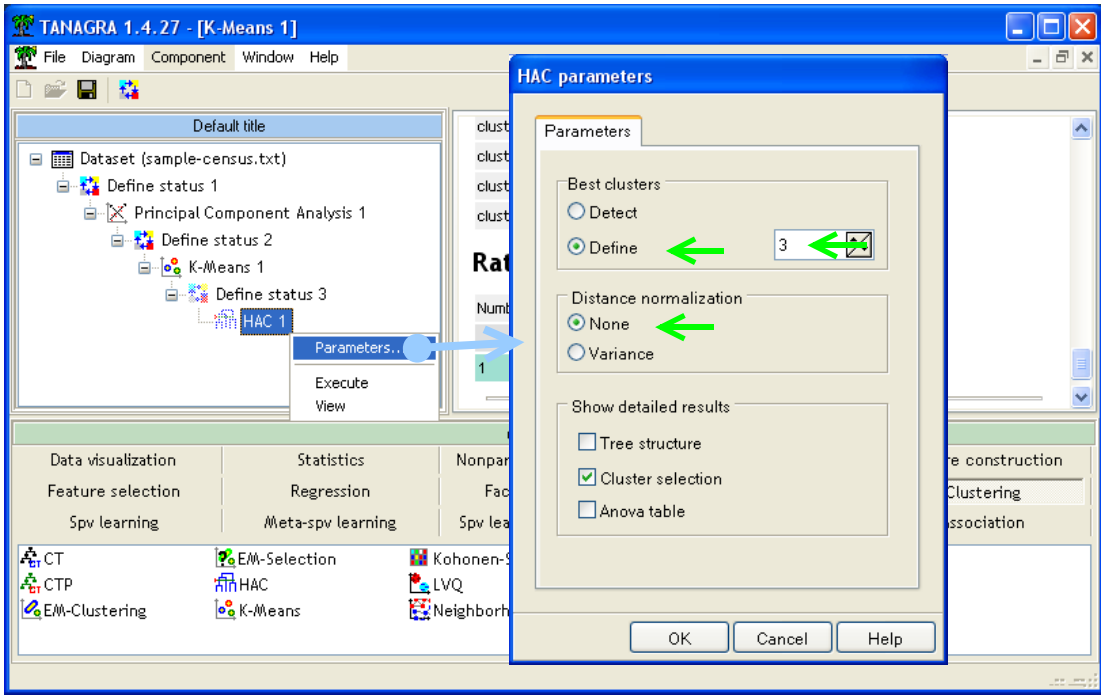
Tanagra énumère les groupes avec les effectifs associés. Nous retiendrons que la part de variance expliquée par le partitionnement est de 89.6%.

3.4 CAH à partir des sous-groupes des K-MEANS

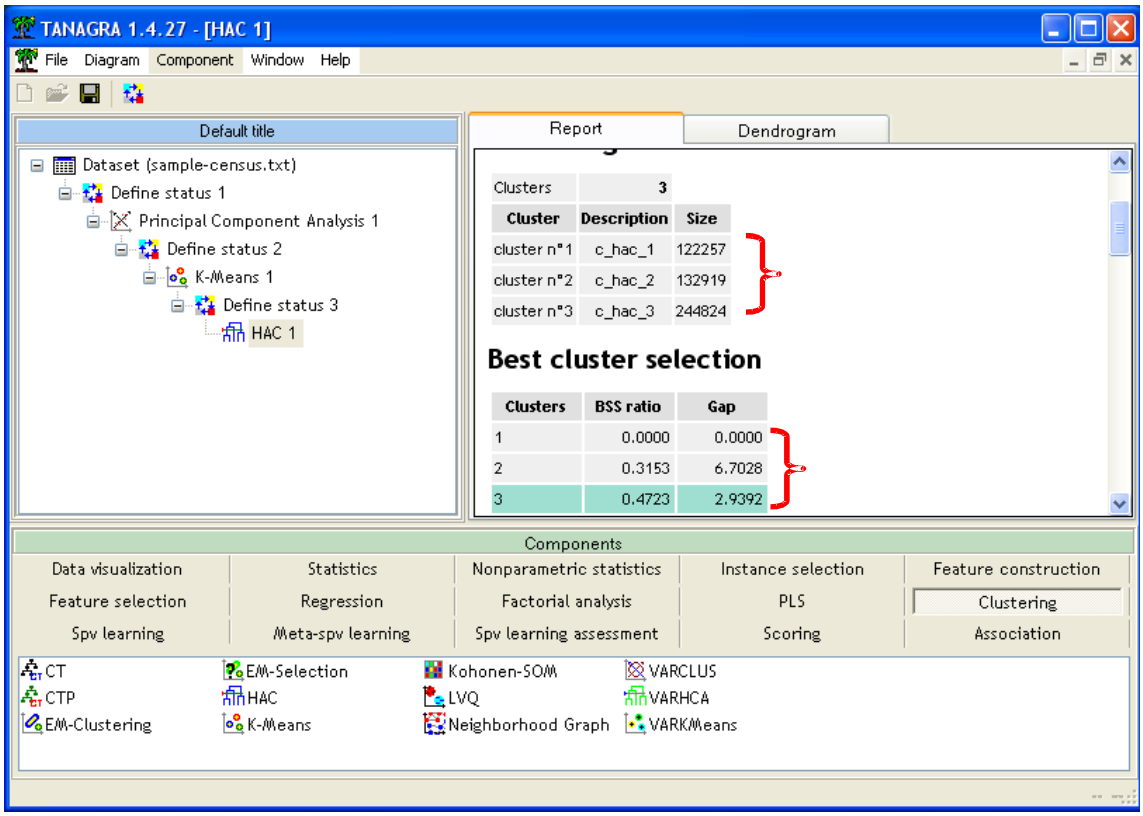
Nous souhaitons maintenant réaliser la CAH en prenant comme groupes de départ, ceux produits par les K-MEANS. Nous insérons encore une fois le composant DEFINE STATUS. Nous plaçons en TARGET la variable indicatrice des sous-groupes CLUSTER_KMEANS_1, produite par le composant K-MEANS. Cette spécification est importante. Si nous l'omettons, Tanagra essaiera de partir des observations individuelles, soit 500.000 individus, c'est suicidaire. En INPUT, nous plaçons les axes factoriels.



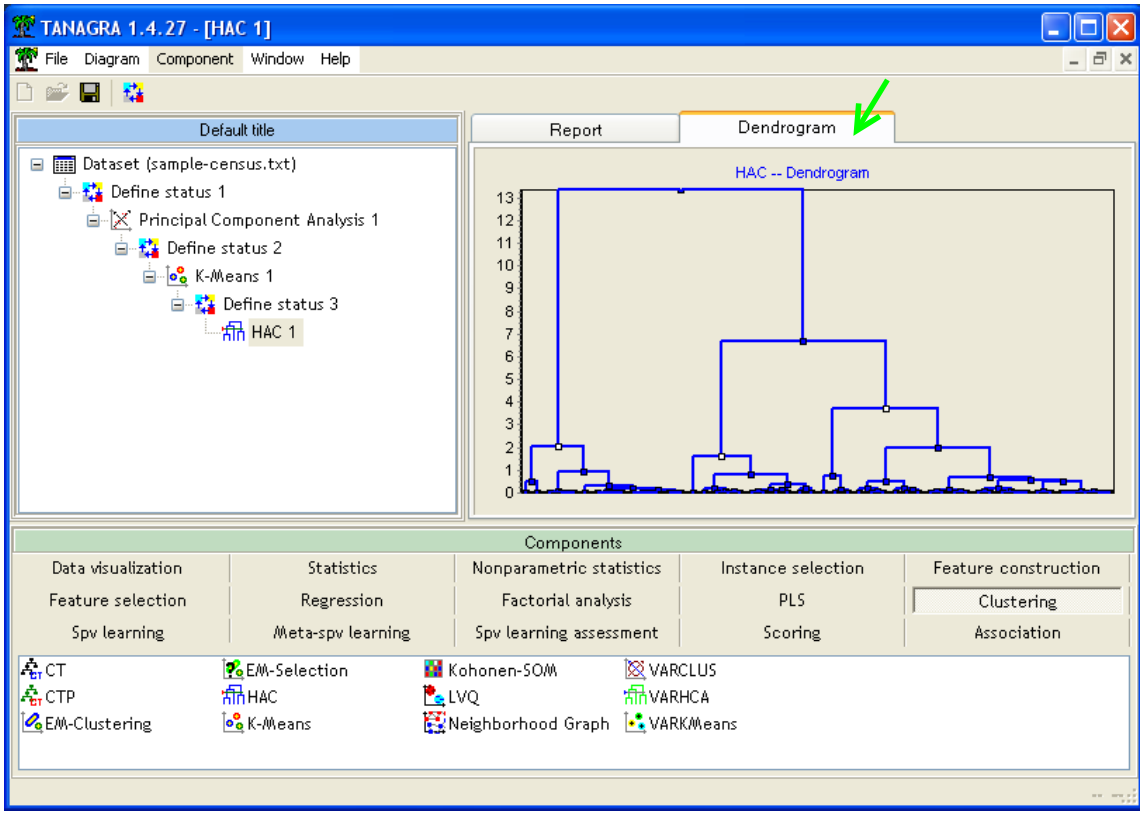
Nous introduisons ensuite le composant CAH. Nous le paramétrons de manière à ce qu'il produise 3 classes, nous verrons pourquoi à la lecture des résultats. Nous n'oublions pas non plus de désactiver la standardisation des données.



Nous cliquons sur VIEW. Le rapport apparaît. Nous avons les effectifs dans chaque cluster. Plus bas, Tanagra nous indique la proportion d'inertie expliquée par la partition.

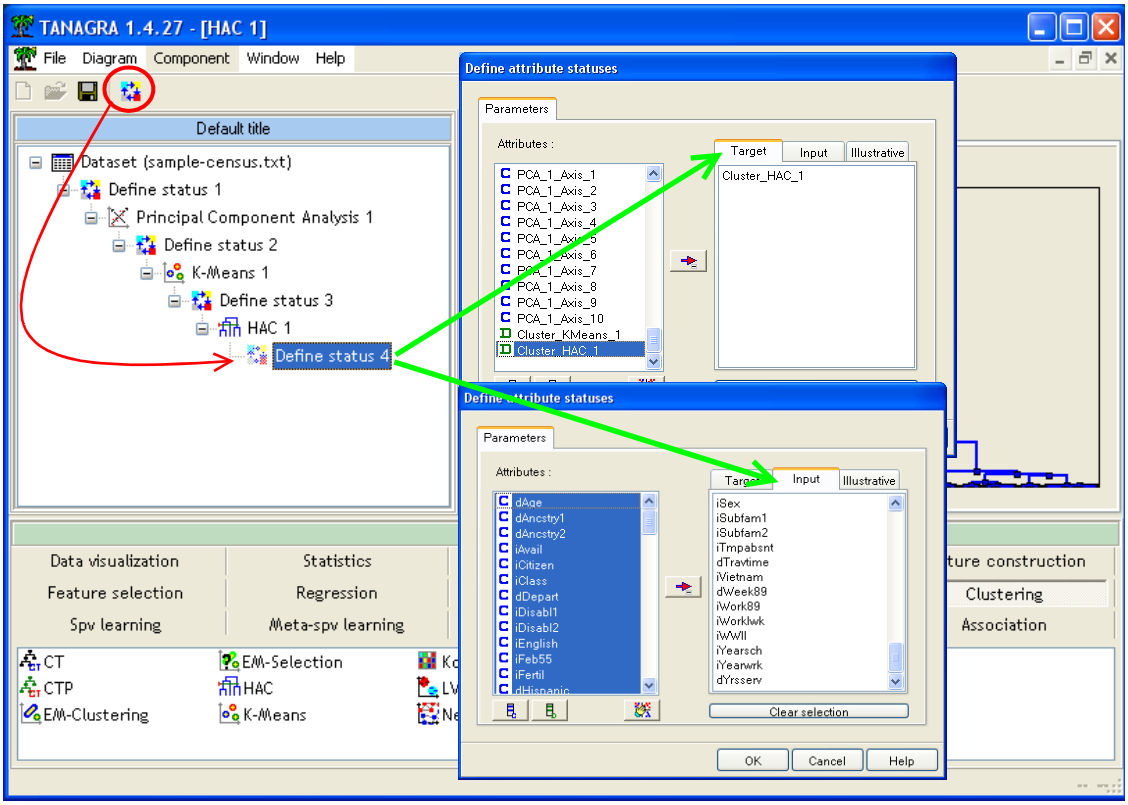


Dans le second onglet de la fenêtre de visualisation (DENDROGRAM), nous avons le dendrogramme. Il apparaît effectivement qu'une partition en 3 classes est crédible.

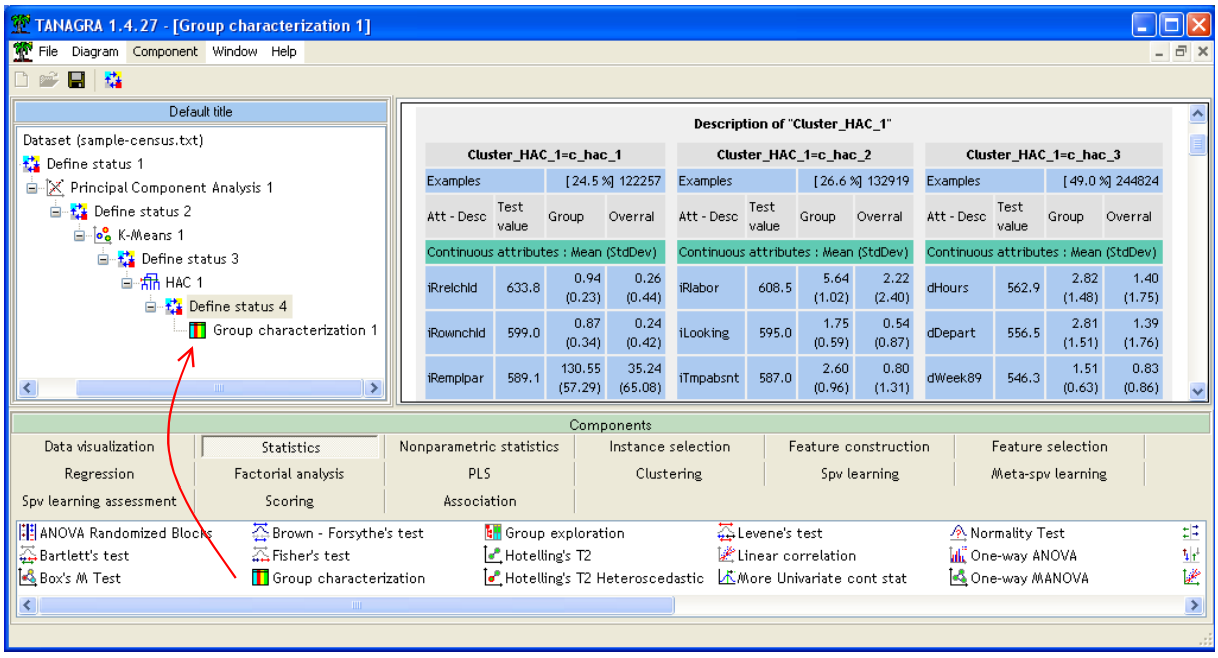


3.5 Interprétation des groupes

Pour comprendre l'appartenance aux groupes, nous pouvons les interpréter à l'aide du composant GROUP CHARACTERIZATION. Nous introduisons tout d'abord le composant DEFINE STATUS. Nous plaçons en TARGET la variable à caractériser CLUSTER_HAC_1, la variable synthétique produite par la CAH ; en INPUT les variables caractérisantes, les variables originelles. Notons qu'il est possible d'introduire d'autres variables illustratives qui n'ont à aucun moment participé aux calculs dans cette étape. Cela peut parfois être utile pour renforcer l'interprétation.



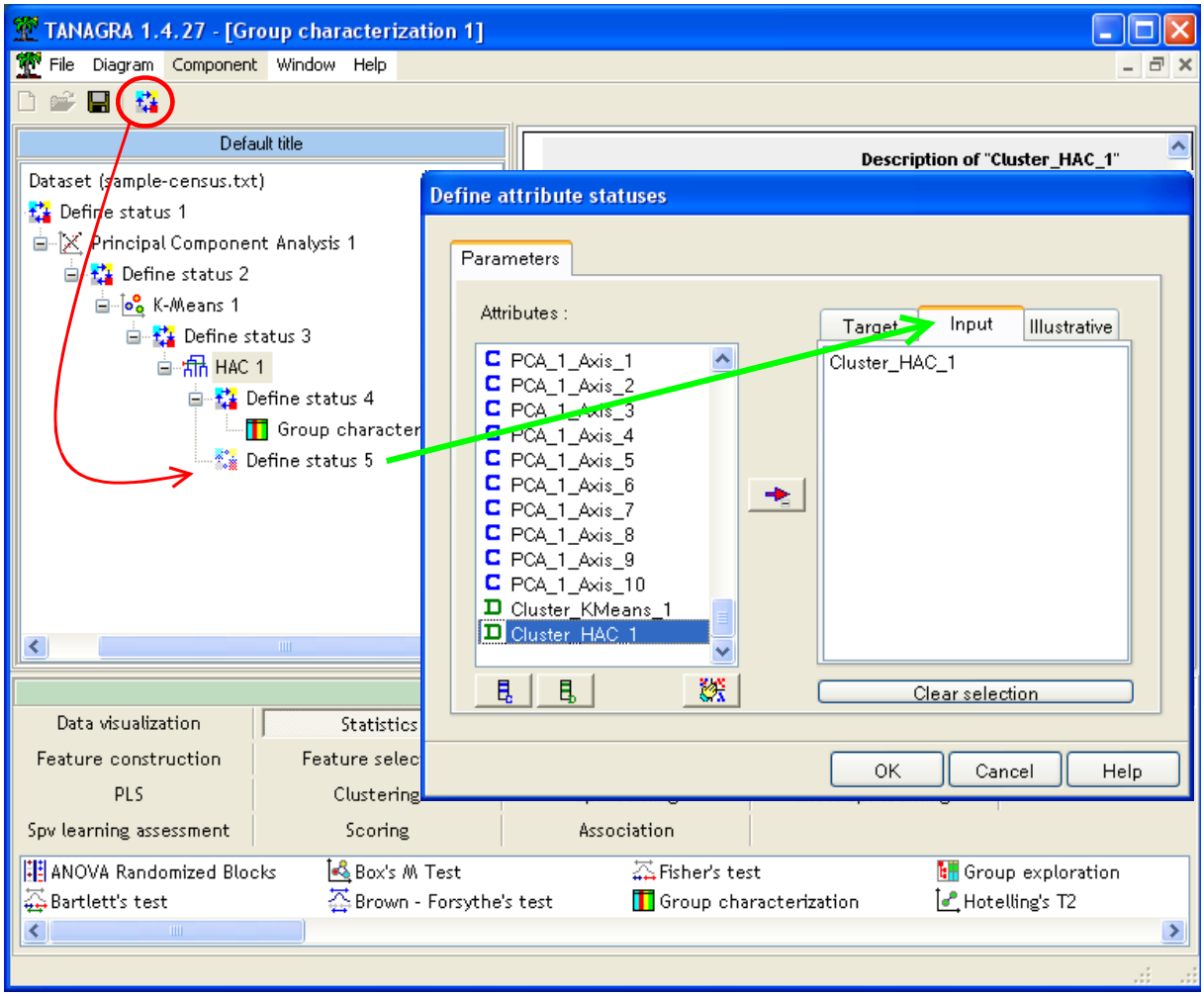
Nous insérons alors le composant GROUP CHARACTERIZATION (onglet STATISTICS).



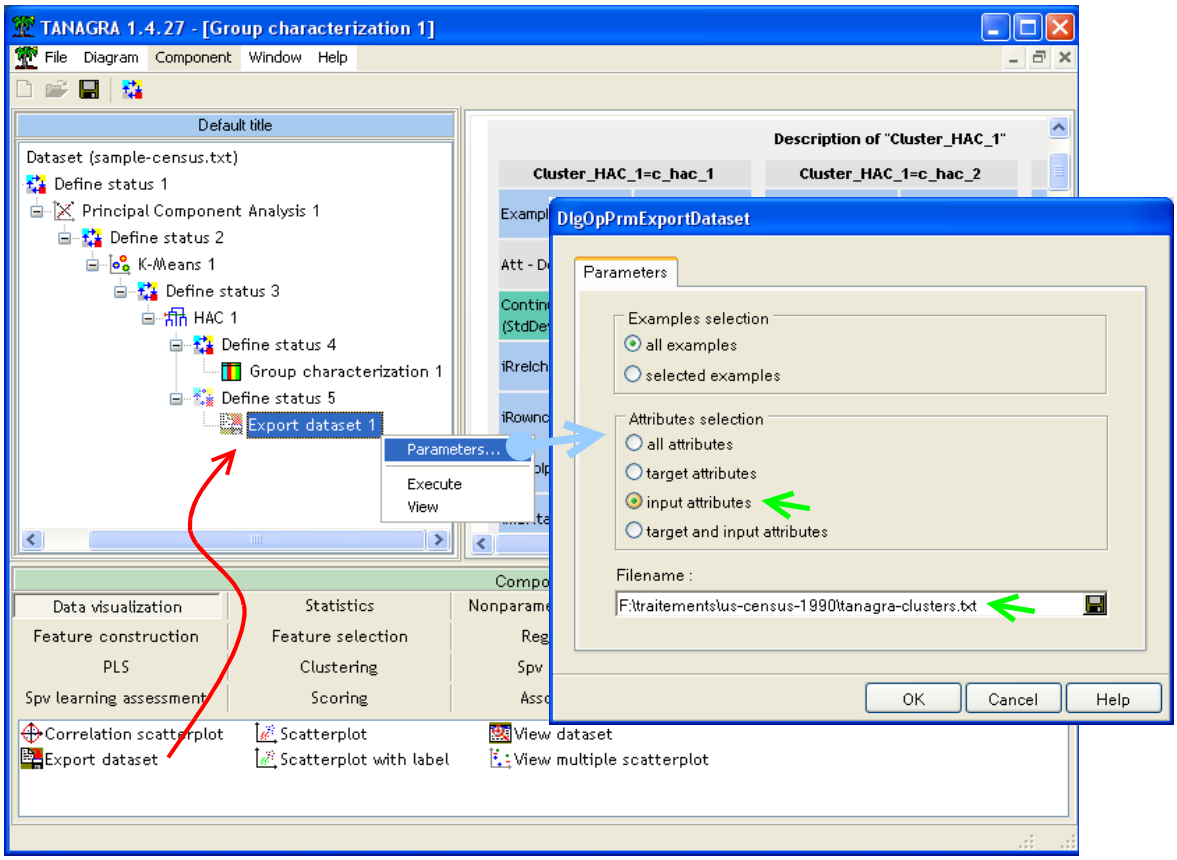
Pour chaque groupe, nous disposons des effectifs et des variables qui le caractérise le mieux. Le critère « valeur test » est utilisé pour les mettre en avant. Pour les variables continues, il s'agit avant tout d'un indicateur qui traduit l'importance de l'écart entre la moyenne calculée dans le fichier global et dans le groupe (voir http://tutoriels-data-mining.blogspot.com/2008/04/interprter-la-valeur-test.html).

3.6 Exportation des observations étiquetées

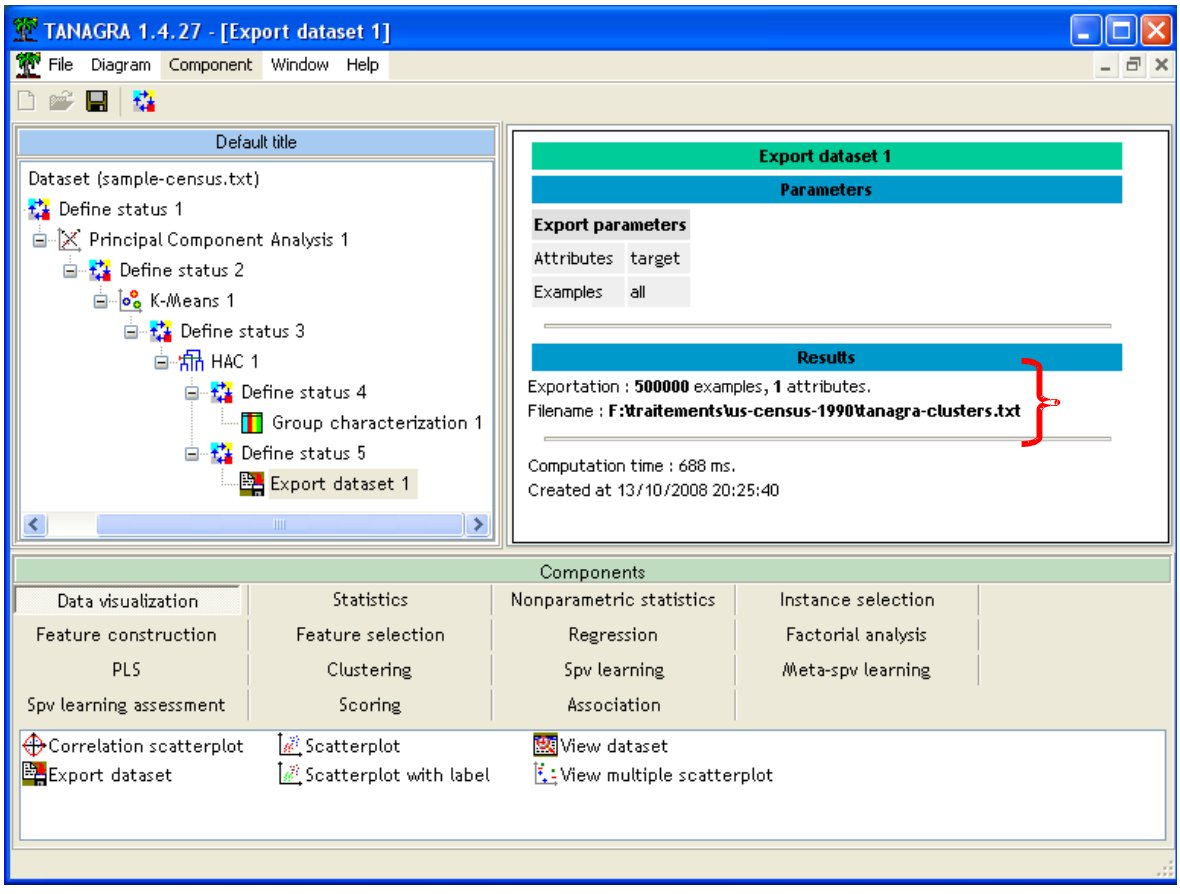
Plus loin, nous souhaitons croiser les partitions induites par Tanagra et R. Pour préparer cela, nous nous proposons d'exporter la colonne identifiant les classes affectées aux individus. Nous insérons un DEFINE STATUS, nous plaçons la variable CLUSTER_HAC_1 en INPUT, afin que seule celle-ci soit exportée.



Nous insérons ensuite le composant EXPORT DATASET (onglet DATA VISUALISATION) dans le diagramme. Nous le paramétrons en actionnant le menu PARAMETERS. Nous voulons exporter la totalité des observations de la variable CLUSTER_HAC_1 que nous placé en INPUT. Nous précisons également le nom du fichier « tanagra-clusters.txt ». Il faudra s'en rappeler, nous l'utiliserons à la fin de ce didacticiel.



En actionnant le menu VIEW, l'exportation est réalisée. Un petit rapport nous indique le nombre de variables et observations effectivement exportées.



4 CAH Mixte avec R

Nous réalisons exactement les mêmes étapes dans R.

4.1 Importation des données

Nous introduisons les commandes suivantes pour importer les données. Bien évidemment, on modifiera de manière adéquate le chemin de recherche. Le fichier est au format texte avec séparateur « tabulation ».

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

4.2 Analyse en composantes principales

Nous demandons une analyse en composantes principales normée. Les 10 premiers facteurs (scores) sont récupérés dans une data.frame intermédiaire.

```
#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 sur les facteurs

Nous réalisons alors les K-Means avec la méthode Mac Queen, en demandant 50 classes. Le nombre d'itération maximum est fixé à 40. Nous récupérons à la sortie une colonne indicatrice, associant chaque individu à son groupe d'appartenance.

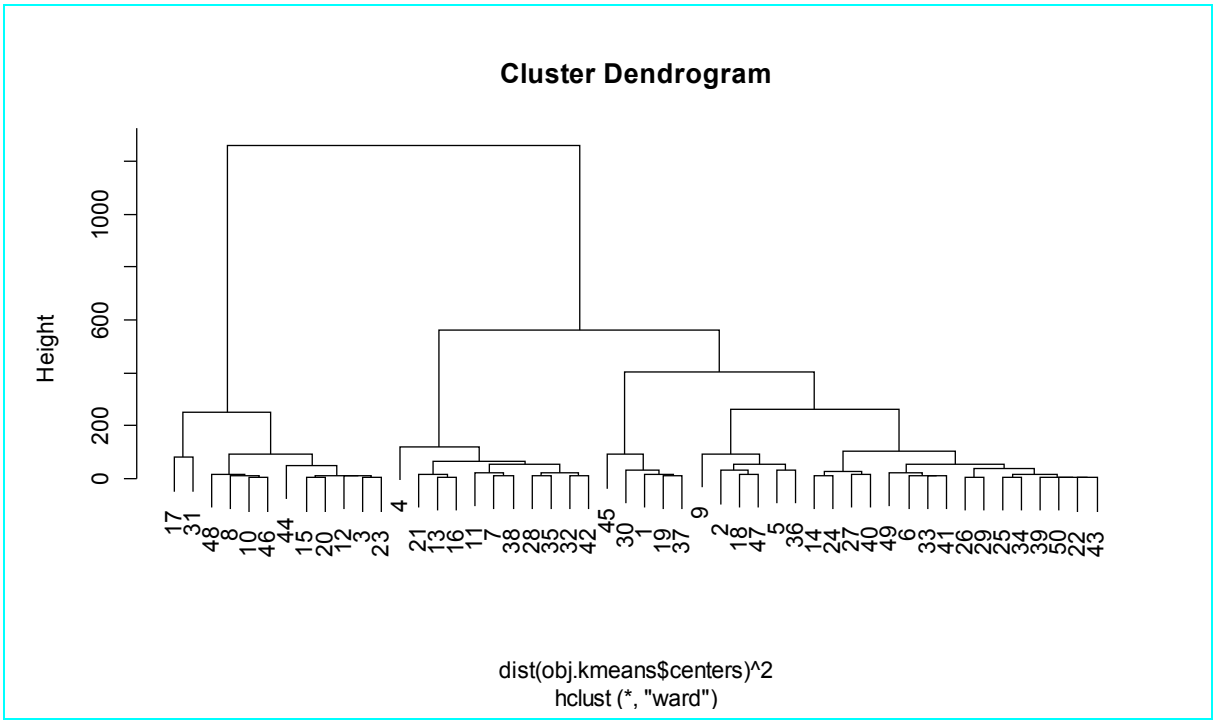
```
#initialiser toujours de la même manière le générateur de nombre aléatoire
#pour avoir le même résultat d'une exécution à l'autre
set.seed(10)

#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 CAH à partir des groupes des K-Means

L'étape suivante consiste à lancer la CAH en prenant comme groupes de départ les classes produites par les K-means. Les distances **dist(.)** sont donc calculés à partir des barycentres des classes associées à l'objet de type K-Means. Nous utilisons la méthode de Ward.

```
#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)
```

Le dendrogramme se présente comme suit.



La partition en 3 classes semble encore une fois pertinente. Nous demandons à R de couper à la hauteur adéquate de l'arbre.

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

4.5 Affectation des classes finales (ceux de la CAH) aux individus

Attention ! A ce stade, R a uniquement affecté les classes de la CAH aux sous-classes des K-Means. Si nous demandons l'affichage du contenu du vecteur **groupe**, nous obtenons

```
> 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
```

Il nous revient de faire la correspondance entre les groupes de la CAH avec les observations en nous appuyant sur le vecteur intermédiaire **id.clus.kmeans** produit par les K-Means. Nous utilisons le script suivant :

```
#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)
```

Nous pouvons maintenant calculer les effectifs pour chaque classe produite par la CAH.

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

Nous obtenons

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

Bien entendu, il nous est facile de calculer les statistiques conditionnelles pour obtenir des sorties similaires à celles du composant GROUP CHARACTERIZATION de Tanagra. Par exemple, si nous nous intéressons au groupe avec l'effectif le plus important (le 1er pour R, il correspond au 3ème groupe pour Tanagra), nous calculons les moyennes conditionnelles avec la commande suivante

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

5 Récapitulatif

5.1 Concordance des résultats

Premier point très important, nous souhaitons savoir dans quelle mesure les partitions proposées par les deux logiciels sont cohérentes. A vrai dire, le K-Means est la seule étape qui peut introduire une variabilité des résultats, notamment par un choix différent des noyaux initiaux pour les 50 sous-groupes. De quelle manière cela pèse-t-il sur la partition finale ?

Un tableau de contingence croisant les classes de Tanagra et R nous rassure quant à la (relative) stabilité des méthodes. Nous avons importé les groupes proposés par Tanagra dans R, puis à l'aide de la procédure **table(.)**,

```
#comparaison des caractéristiques des classes avec ceux de Tanagra
#chargement des classes produites par Tanagar
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)
```

Nous obtenons les concordances suivantes.

```
> #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
```

Les classes obtenues sont fortement cohérentes.

5.2 Performances

Comme nous le signalions plus haut, R n'a pas pu traiter la totalité du fichier original. C'est pour cette raison que nous avons travaillé sur un échantillon de 500.000 observations. C'est une vraie limitation. En revanche, une fois que R peut réaliser les traitements, il le fait à une vitesse qui laisse pantois. Dans le tableau ci-dessous, nous recensons **les temps d'exécution** des principales étapes de l'analyse pour les deux logiciels[4].

Opération	Traitement échantillon (500.000 observations)		Traitement de la totalité du fichier (2.458.285 observations)
	Tanagra (en sec.)	R (en sec.)	Tanagra (en sec.)
Importation des données	28	24	145
ACP (10 axes extraits)	61	37	294
K-Means (40 itérations max.)	56	27	227
CAH à partir des 50 sous-groupes	1	0.1	4

C'est assez ébouriffant. Mis à part le K-Means qui repose sur une heuristique, avec la part de variabilité induite par le tirage aléatoire des données, toutes les autres étapes sont déterministes. Les temps de traitement sont donc directement comparables[5].

Concernant **l'occupation mémoire**, R prend beaucoup plus de place que Tanagra. C'est une constante dans tous les comparatifs que nous avons mis en place. La limite arrive assez vite. Encore une fois, il faudrait s'y connaître pour fixer convenablement les paramètres de gestion mémoire, en manipulant judicieusement la fonction **memory.limit()** par exemple. Je n'ai pas cette expertise.

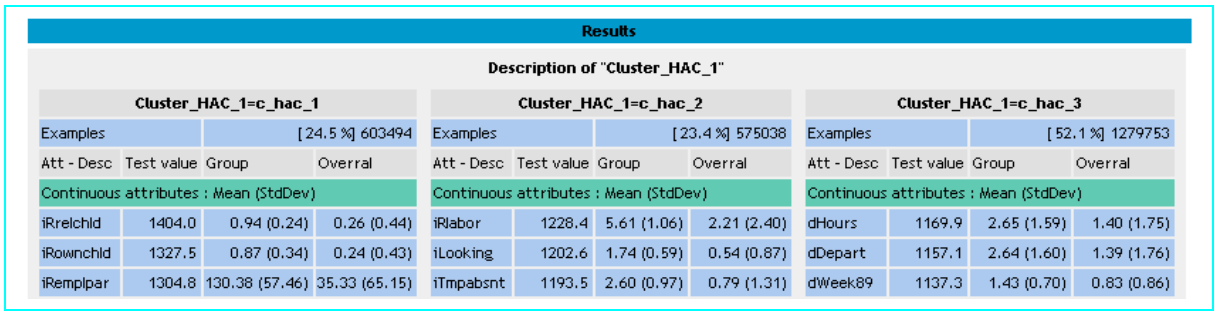
Opération	Traitement échantillon (500.000 observations)		Traitement de la totalité du fichier (2.458.285 observations)
	Tanagra (Mo)	R (Mo)	Tanagra (Mo)
Taille max mémoire occupée	182	488	932

Pour le traitement de la totalité du fichier avec Tanagra (2.458.285 observations). Nous avons rajouté dans le même tableau les temps d'exécution. Ils sont raisonnables. On peut rester devant la machine et surfer en même temps. Autre information importante, l'occupation mémoire de Tanagra n'a jamais dépassé 932 Mo durant l'analyse du fichier complet. On notera enfin les

[4] Tanagra fournit les temps d'exécution en bas de chaque fenêtre de visualisation. Pour R, nous avons utilisé la fonction **system.time(.).** Voir http://tutoriels-data-mining.blogspot.com/2008/09/traitement-de-gros-volumes-comparaison.html pour ce qui est des caractéristiques de la machine utilisée.

[5] NDLA : « Ben voilà, il me reste encore un peu de boulot… »

groupes mis en avant par l'analyse sur la totalité du fichier sont les mêmes que pour le traitement de l'échantillon. A titre de comparaison, nous donnons ici les 3 premières variables qui caractérisent chaque sous-groupe. Par rapport au traitement sur l'échantillon de 500.000 observations, les valeurs test sont logiquement plus élevées puisque nous travaillons sur beaucoup plus d'observations. Les moyennes conditionnelles par contre sont quasiment identiques.



Results

Description of "Cluster_HAC_1"

Cluster_HAC_1=c_hac_1			
Examples			[24.5 %] 603494
Att - Desc	Test value	Group	Overral
Continuous attributes : Mean (StdDev)			
iRrelchld	1404.0	0.94 (0.24)	0.26 (0.44)
iRownchld	1327.5	0.87 (0.34)	0.24 (0.43)
iRemplpar	1304.8	130.38 (57.46)	35.33 (65.15)

Cluster_HAC_1=c_hac_2			
Examples			[23.4 %] 575038
Att - Desc	Test value	Group	Overral
Continuous attributes : Mean (StdDev)			
iRlabor	1228.4	5.61 (1.06)	2.21 (2.40)
iLooking	1202.6	1.74 (0.59)	0.54 (0.87)
iTmpabsnt	1193.5	2.60 (0.97)	0.79 (1.31)

Cluster_HAC_1=c_hac_3			
Examples			[52.1 %] 1279753
Att - Desc	Test value	Group	Overral
Continuous attributes : Mean (StdDev)			
dHours	1169.9	2.65 (1.59)	1.40 (1.75)
dDepart	1157.1	2.64 (1.60)	1.39 (1.76)
dWeek89	1137.3	1.43 (0.70)	0.83 (0.86)