1 Objectif

Comparer les implémentations de TANAGRA et R (package RPART) de la méthode CART.

CART (Breiman et al, 1984) est une méthode très populaire d'induction d'arbres de décision, peut-être la plus répandue, tout du moins au sein de la communauté francophone du Data Mining (voir Références). A juste titre. CART intègre tous les bons ingrédients d'un apprentissage maîtrisé : arbitrage biais – variance via avec le post-élagage, le mécanisme de coût complexité permet de « lisser » l'exploration de l'espace des solutions, intégration de la préférence à la simplicité avec la règle de l'écart type, préférence que le praticien peut modifier en modulant les paramètres en fonction des objectifs de l'étude et des caractéristiques des données, etc.

La Société Salford Systems détient les droits d'utilisation du nom CART. De fait, même si la méthode est implantée dans de nombreux logiciels commerciaux, elle n'est jamais désignée nommément. Que cela ne nous induise pas en erreur, une lecture rapide de la documentation ne laisse généralement aucun doute quant à l'origine de la technique.

La situation est un peu différente en ce qui concerne les logiciels libres. Ils sont nettement plus rares à proposer CART, une grande majorité d'entre eux lui préférant ID3 ou C4.5, nettement plus faciles à programmer au demeurant.

Dans ce didacticiel, nous comparons deux implémentations libres de CART : le composant C-RT de Tanagra et le package RPART du Logiciel R. A la lecture de la documentation, ils s'appuient tous les deux sur les mêmes schémas génériques (Breiman et al, 1984 ; chapitres 3, 10 et 11). La principale différence apparaît lors du post-élagage. Tanagra ne propose que le post élagage basé sur un échantillon spécifique dit « échantillon d'élagage »[1] (section 11.4). RPART lui s'appuie plus volontiers sur la validation croisée (section 11.5), bien qu'il soit possible d'utiliser également un échantillon d'élagage, au prix de manipulations (un peu beaucoup) compliquées il est vrai[2].

2 Données

Pour mener la comparaison, nous utilisons les données artificielles WAVEFORM de Breiman (section 2.6.2). La variable à prédire (CLASS) comporte 3 modalités, les 21 variables prédictives (V1 à V21) sont toutes continues. Nous essayons de reproduire l'expérimentation décrite dans l'ouvrage de référence (pages 49 et 50), à savoir utiliser 300 individus en apprentissage et 5000 en test. Ainsi, notre fichier WAVE5300.XLS[3] comporte 5300 observations et 23 colonnes, la dernière étant une variable indicatrice spécifiant l'appartenance de chaque observation à la partie apprentissage ou test (Figure 1).

[1] « Pruning set » en anglais. On parle aussi de « échantillon de validation ». Mais j'ai des réticences à le faire. Le terme est ambigu, on le confond souvent avec l'échantillon test qui un rôle tout à fait différent. On note avec amusement d'ailleurs que Breiman et ses collègues utilisent l'appellation « test sample » pour désigner cette fraction des données dédiée au post élagage, ajoutant si besoin était à la confusion. Plus tard, le rôle de chaque portion des données a été clarifié dans plusieurs publications : « growing set » est utilisé pour la construction de l'arbre maximal ; « pruning set » est utilisé pour le post élagage ; « test set » sert exclusivement à l'évaluation des performances et ne doit participer en aucune façon à la définition de l'arbre final. L'addition de « growing » et « pruning » constitue le « learning » set, l'échantillon d'apprentissage.

[2] Voir http://www.math.univ-toulouse.fr/~besse/pub/TP/appr_se/tp7_cancer_tree_R.pdf ; section 2.1.

[3] http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/wave5300.xls

CLASS	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	SAMPLE
A	0.75	0.05	2.60	2.28	2.07	3.42	3.66	2.65	2.30	2.54	1.82	1.69	0.82	1.59	1.02	0.58	-0.37	2.76	0.92	0.89	-1.13	learning
A	1.85	-0.66	0.70	3.69	1.50	2.42	2.57	4.62	2.07	1.76	1.13	3.12	2.01	1.43	2.00	3.06	1.39	2.79	0.34	0.54	0.64	learning
A	-1.37	-0.45	-1.35	2.12	-0.15	1.36	0.27	-0.26	-1.19	1.33	0.46	1.41	2.65	3.10	4.49	3.96	3.65	2.26	1.81	1.04	-0.41	learning
A	-0.29	-0.28	-1.24	1.65	0.94	1.75	4.58	0.29	3.33	1.73	2.83	2.29	2.32	1.82	0.71	2.71	2.61	2.43	0.58	1.66	-0.28	learning
A	-1.16	1.11	1.39	3.09	1.83	4.76	3.01	3.93	2.57	1.56	3.37	1.59	1.21	3.39	1.64	2.84	-0.03	2.11	2.11	0.37	1.02	learning
A	-1.22	2.66	1.26	0.77	2.58	4.56	3.42	4.25	2.92	2.07	2.62	2.03	1.12	0.50	-0.10	1.99	1.16	1.58	1.00	1.51	0.83	learning
A	-0.29	1.35	-0.98	2.63	1.62	1.35	1.60	1.59	0.97	1.32	2.16	3.04	3.96	3.76	5.89	1.34	3.80	3.96	0.95	1.16	1.14	learning
A	0.58	0.28	1.57	0.09	0.77	-0.21	-0.32	0.64	1.36	-0.10	-0.52	3.38	4.49	4.52	6.08	5.84	4.26	4.32	1.37	0.89	-0.38	learning
A	0.78	0.12	0.78	3.20	3.84	2.18	5.04	5.18	4.29	4.06	3.22	1.13	3.01	-0.01	0.93	0.13	-1.21	1.58	0.19	0.05	-0.36	learning
A	1.71	0.77	2.56	2.03	3.33	6.33	6.16	4.52	2.83	3.84	1.20	1.41	0.55	0.55	1.23	-0.46	0.39	-0.98	0.51	2.00	0.99	learning
A	-0.26	1.04	2.52	3.68	3.81	4.70	6.74	4.97	3.01	1.65	1.90	0.23	-0.46	0.59	0.59	1.61	0.82	0.02	0.40	-1.33	0.41	learning
A	0.54	0.57	-1.36	1.68	1.08	3.78	2.41	2.87	3.71	1.25	3.37	0.72	1.81	-0.21	1.44	2.51	0.28	0.48	1.30	-1.80	0.23	learning
A	-0.02	-0.27	0.08	1.14	-0.81	0.20	-0.83	0.83	-0.33	1.24	1.64	2.00	2.90	2.90	6.77	3.04	4.30	0.92	3.23	2.28	0.13	learning
A	1.77	1.59	1.22	-0.16	0.21	0.63	-0.29	-0.08	-1.35	1.58	0.71	4.58	4.42	5.95	7.42	5.43	3.53	2.11	1.64	0.01	0.83	learning
A	-1.07	0.18	1.80	4.21	3.06	4.32	5.52	4.00	4.28	2.70	2.45	1.79	3.03	0.23	2.49	0.68	1.14	-1.42	0.46	0.43	-1.25	learning
A	1.53	-0.84	0.00	1.02	1.70	2.25	0.39	1.28	1.37	1.80	2.39	2.38	3.16	3.32	4.24	2.79	3.18	3.15	2.41	-0.94	0.80	learning
A	-1.24	0.60	1.59	1.54	2.94	4.54	4.30	5.84	2.46	1.90	2.72	0.41	1.66	3.08	1.52	3.41	-0.20	0.57	0.71	-1.07	-0.69	learning
A	0.80	0.96	2.24	2.58	5.07	5.91	4.44	3.94	4.26	1.81	2.21	1.76	-0.53	0.06	1.57	0.69	0.23	-1.35	1.94	-0.71	-0.35	learning
A	0.45	2.22	2.19	1.45	2.42	2.89	4.40	3.21	2.59	2.55	1.39	1.48	0.63	1.87	1.78	3.00	0.99	0.87	1.61	0.16	-1.31	learning
A	1.72	1.94	0.41	2.53	3.36	4.90	4.22	5.40	2.97	3.97	1.74	1.26	-1.37	1.31	1.10	1.26	-0.14	1.40	-0.51	-1.77	-0.01	learning

Figure 1 - Les 20 premières observations du fichier de données

3 La méthode CART avec TANAGRA

3.1 Création d'un diagramme et importation des données

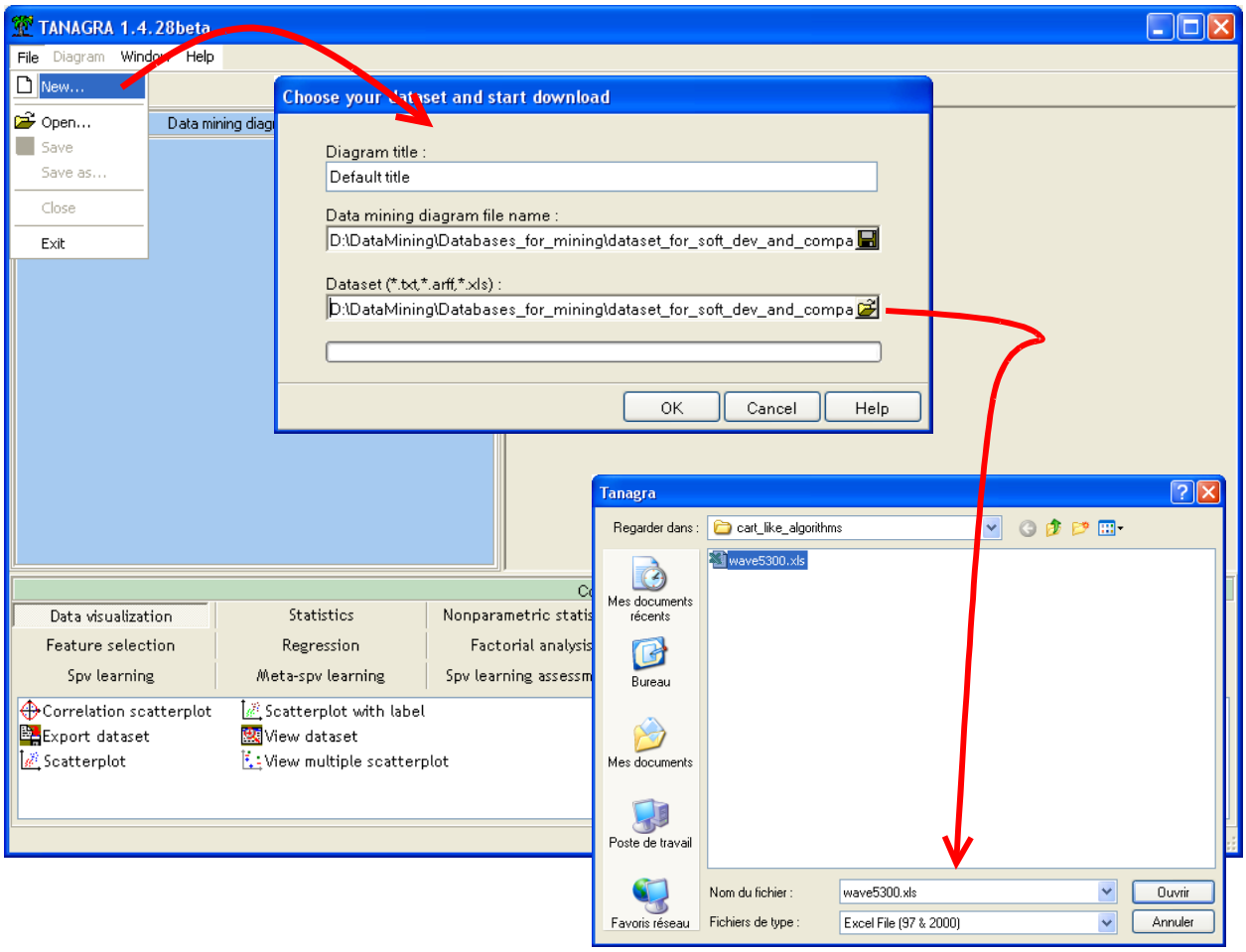
Il y a plusieurs manières d'importer un fichier EXCEL dans TANAGRA. Nous pouvons utiliser une macro complémentaire qui se charge d'envoyer les données à partir du tableur[4], il faut disposer du logiciel EXCEL ; nous pouvons également importer le fichier sans avoir à l'ouvrir au préalable, la présence du tableur n'est pas requise dans ce cas[5].

Nous utilisons cette seconde solution dans ce didacticiel. Elle est préférable dès que le fichier atteint une certaine taille. Pour que la procédure fonctionne correctement, le fichier ne doit pas être en cours d'édition, les observations doivent être situées dans la première feuille, sans colonnes ou lignes vides à gauche ou au dessus de la plage de données. La première ligne correspond aux noms de variables.

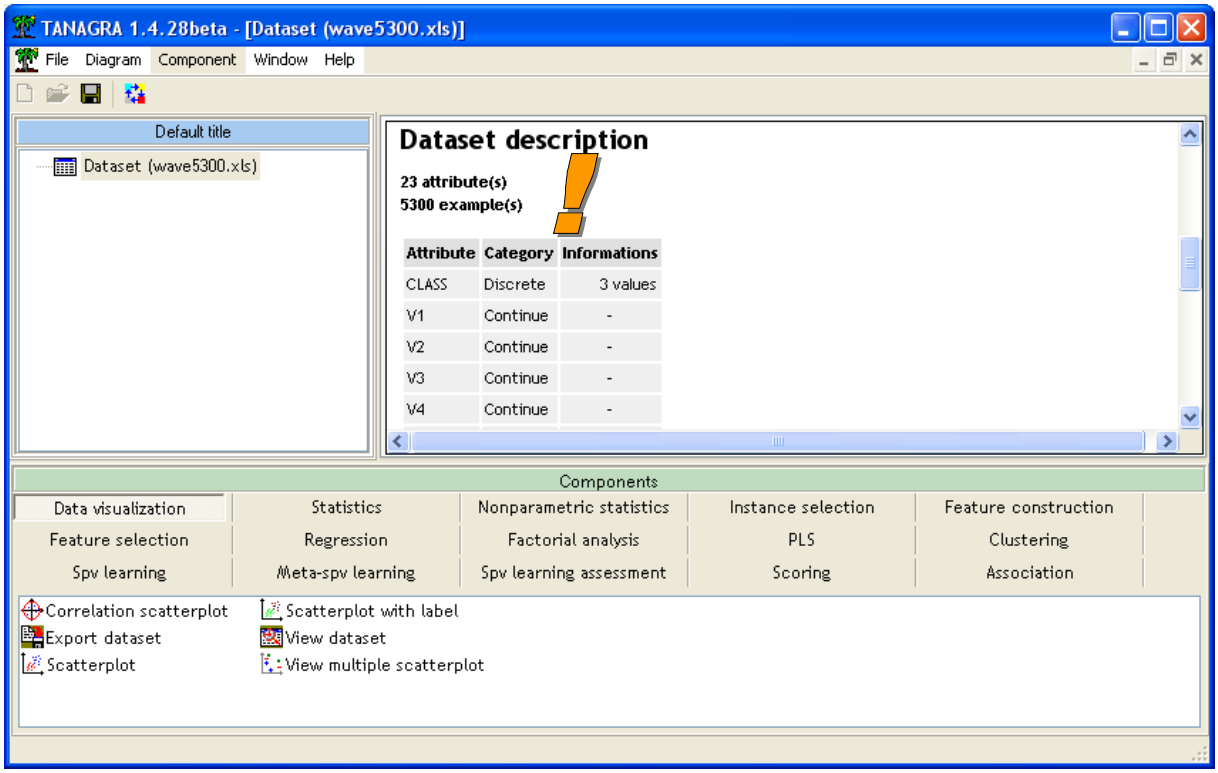
Nous lançons le logiciel TANAGRA. Pour créer un nouveau diagramme et importer les données, nous actionnons le menu FILE / NEW. Une boîte de dialogue apparaît, nous pouvons spécifier le nom du fichier EXCEL et le nom du diagramme.

[4] Voir http://tutoriels-data-mining.blogspot.com/2008/03/importation-fichier-xls-excel-macro.html

[5] http://tutoriels-data-mining.blogspot.com/2008/03/importation-fichier-xls-excel-mode.html

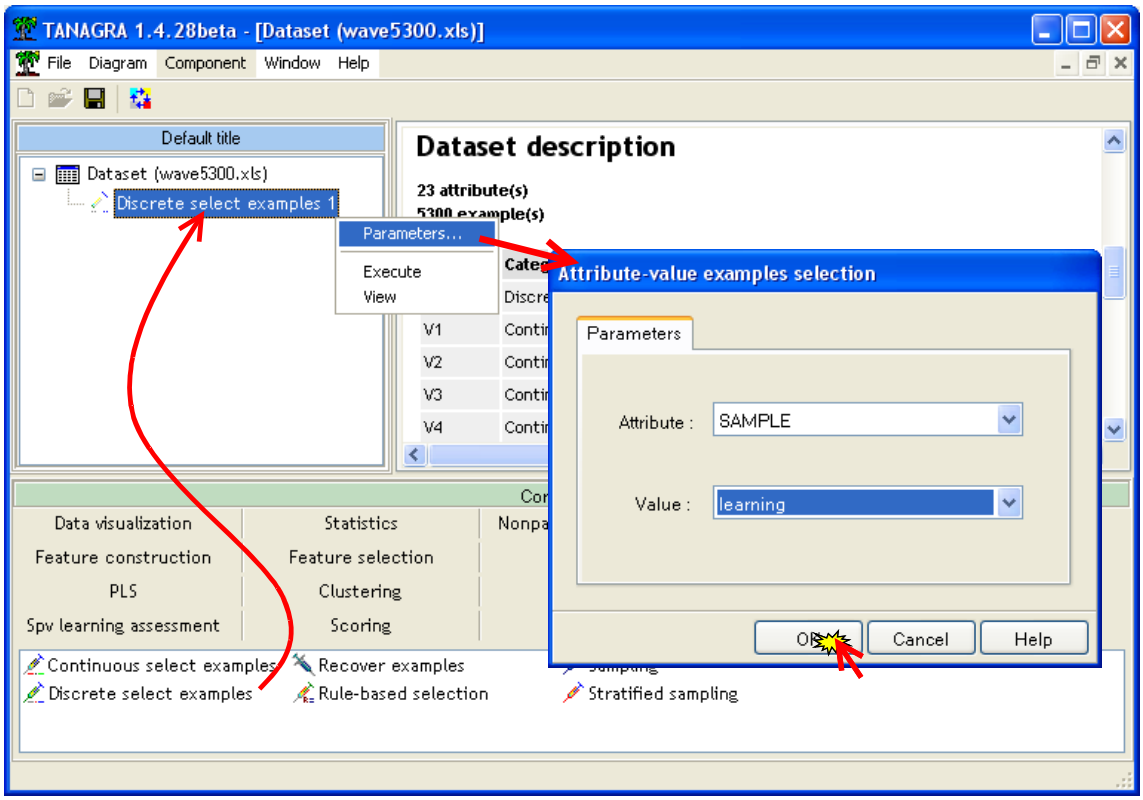


Le diagramme est créé, TANAGRA nous indique que le fichier comporte 5300 observations et 23 variables.

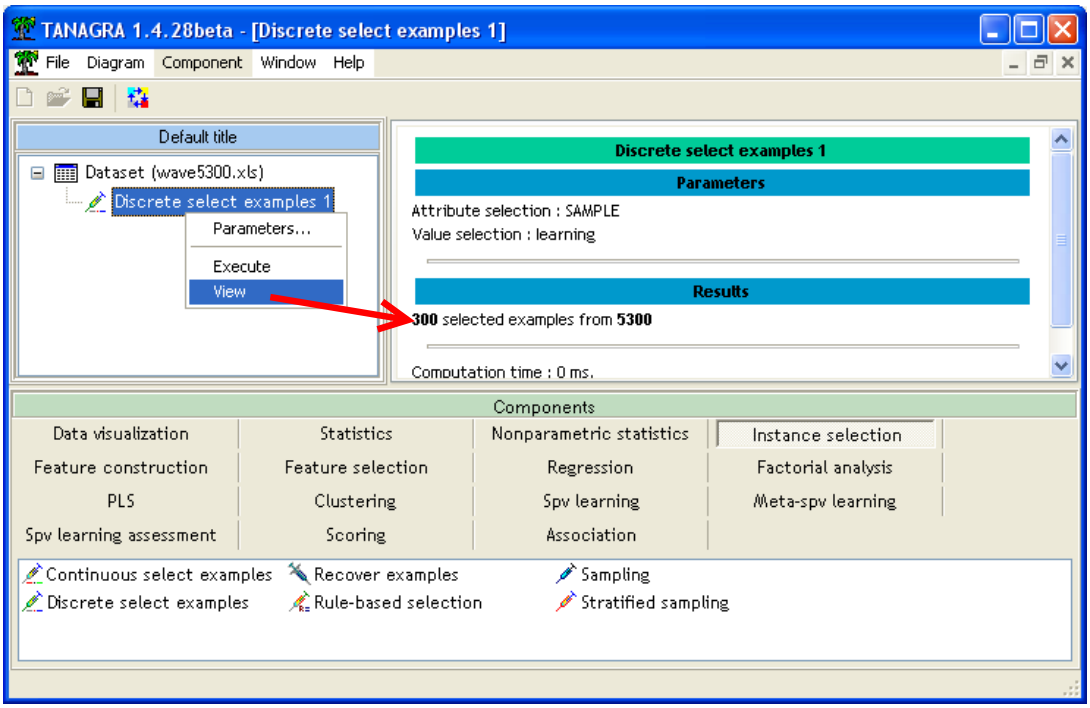


3.2 Partition des données en apprentissage et test

Nous devons indiquer au logiciel les individus que l'on dédie à l'élaboration de l'arbre. Pour ce faire, nous introduisons le composant DISCRETE SELECT EXAMPLES (onglet INSTANCE SELECTION) dans le diagramme. Puis, nous activons le menu contextuel PARAMETERS. L'attribut de sélection est SAMPLE, les individus en apprentissage correspondent à la valeur LEARNING.

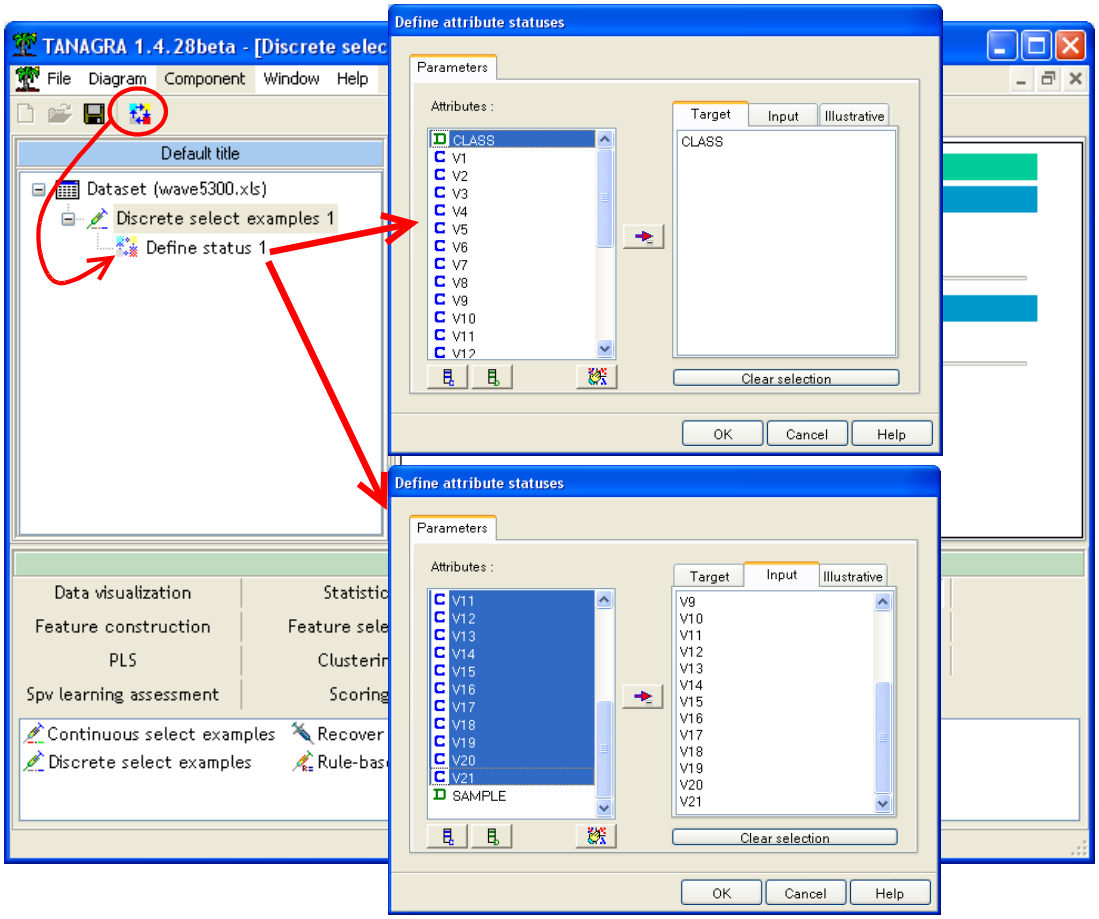


Nous cliquons sur le menu contextuel VIEW. TANAGRA nous indique que 300 observations parmi les 5300 sont maintenant réservées pour la construction des modèles.



3.3 Création de l'arbre de décision

Avant d'insérer le composant de calcul proprement dit, nous devons préciser le rôle des variables à l'aide du composant DEFINE STATUS. Le plus simple est de passer par le raccourci dans la barre d'outils. Nous plaçons CLASS en TARGET et les variables V1 à V21 en INPUT.



Nous pouvons maintenant insérer le composant C-RT (onglet SPV LEARNING) qui implémente la méthode CART dans TANAGRA. Attardons-nous un instant sur ses paramètres. Nous activons pour cela le menu contextuel SUPERVISED PARAMETERS.

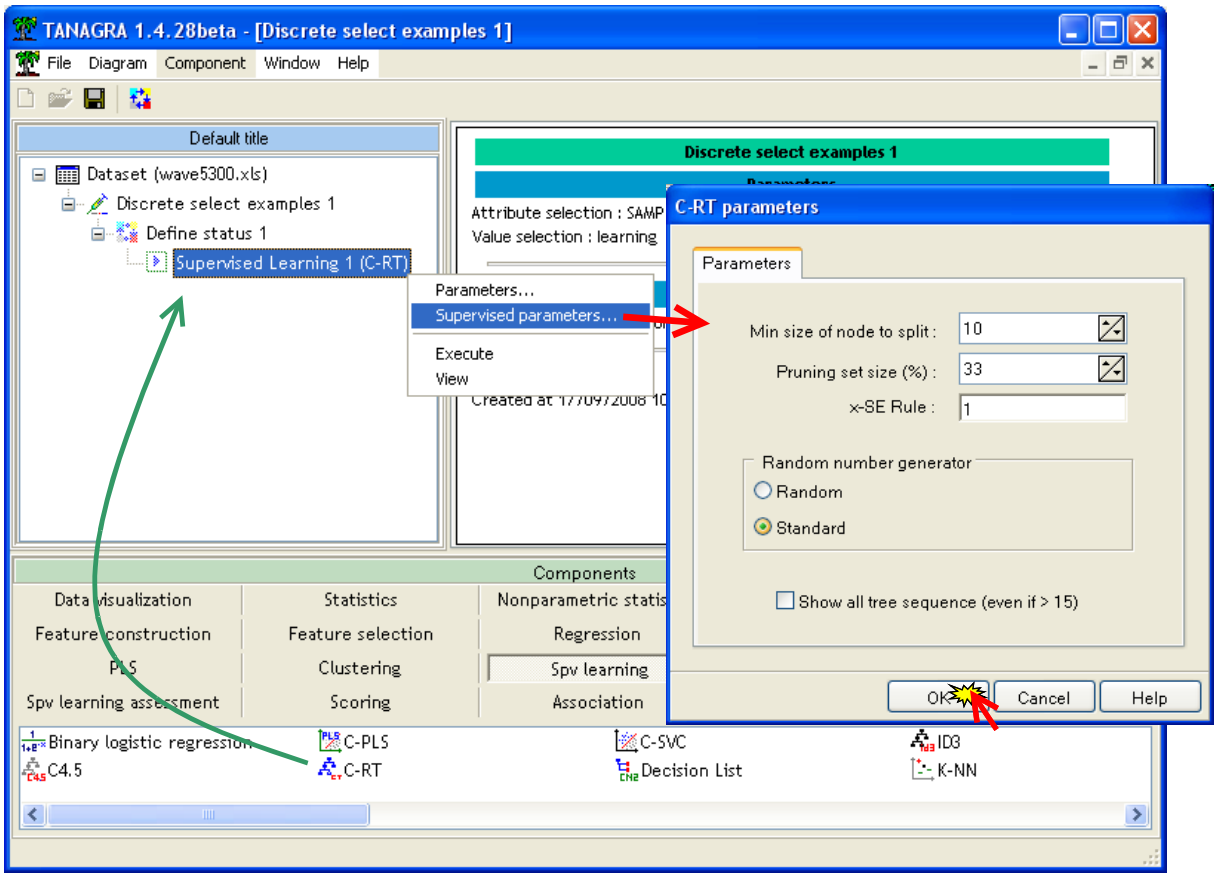
MIN SIZE OF NODE TO SPLIT correspond à l'effectif minimal dans un sommet pour qu'une segmentation soit tentée. Dans notre cas, si un sommet comporte moins de **10** observations, elle devient automatiquement une feuille de l'arbre.

PRUNING SET SIZE indique la proportion de l'échantillon de données qui sera réservée au post élagage. La valeur par défaut est **33%** c.-à-d. 33% de 300 = 99 observations seront utilisés lors de l'élagage de l'arbre et, par conséquent, 300 – 99 = 201 observations seront utilisés lors de la phase d'expansion.

SE RULE est le paramètre qui permet de définir l'arbre qui sera finalement choisi, relativement à l'arbre optimal obtenu sur le fichier d'élagage. La valeur par défaut est **θ=1**. Si nous la fixons à 0, l'arbre obtenu correspond exactement à l'arbre optimal. Plus grande sera la valeur de X-SE RULE, plus petit sera l'arbre finalement sélectionné.

RANDOM NUMBER GENERATOR, lorsqu'il est égal à **STANDARD**, permet de produire toujours la même séquence de nombres aléatoires. L'arbre créé sera alors le même d'une session de travail à l'autre.

Enfin, **SHOW ALL TREE SEQUENCE**, lorsqu'il est activé, force l'affichage de toutes les séquences d'arbres analysés lors du post élagage. Nous n'en avons pas besoin ici.



Nous validons en cliquant sur OK. Puis nous actionnons le menu VIEW pour obtenir les résultats. La matrice de confusion en resubstitution, calculée sur les données en apprentissage (300 observations), est affichée en premier. Le taux d'erreur apparent est 19.67%.

Results

Classifier performances

Error rate	0.1967
Values prediction	**Confusion matrix**

Value	Recall	1-Precision
A	0.8727	0.2195
B	0.7451	0.1915
C	0.7841	0.1687

	A	B	C	Sum
A	96	8	6	110
B	18	76	8	102
C	9	10	69	88
Sum	123	94	83	300

Plus bas, nous avons le tableau des séquences d'arbres, avec : le nombre de feuilles, les taux d'erreur calculés sur le growing set et le pruning set.

Le taux d'erreur sur le growing set diminue à mesure que le nombre de feuilles augmente. Nous ne pouvons pas utiliser cette information pour sélectionner le bon modèle.

Nous observons que l'arbre optimal sur l'échantillon d'élagage comporte 14 feuilles, avec un taux d'erreur de 0.2828 (surlignée en vert).

Data partition

Growing set	201
Pruning set	99

Trees sequence (# 10)

N°	# Leaves	Err (growing set)	Err (pruning set)
10	1	0.6169	0.6667
9	2	0.4129	0.4646
8	3	0.3035	0.3636
7	6	0.1891	0.3434
6	7	0.1692	0.3333
5	9	0.1343	0.3232
4	11	0.1045	0.3232
3	14	0.0746	0.2828
2	21	0.0398	0.3333
1	23	0.0348	0.3333

Figure 2 - Séquences d'arbres emboîtés et taux d'erreur

En appliquant la règle de $\theta = 1$ écart type, nous choisissons alors le plus petit arbre dont le taux d'erreur est inférieur à

$$\varepsilon_{seuil} = 0.2828 + 1 \times \sqrt{\frac{0.2828 \times (1 - 0.2828)}{99}} = 0.3281$$

Il s'agit de l'arbre n° 5, avec 9 feuilles et un taux d'erreur sur l'échantillon d'élagage de 0.3232 (surlignée en rouge). Le taux d'erreur sur le growing set est de 13.43%.

Les courbes du taux d'erreur sur le growing et pruning set en fonction du nombre de feuilles sont accessibles dans l'onglet CHART de la fenêtre de résultats. Nous visualisons mieux les « plateaux », indiquant des solutions équivalentes en termes de performances.

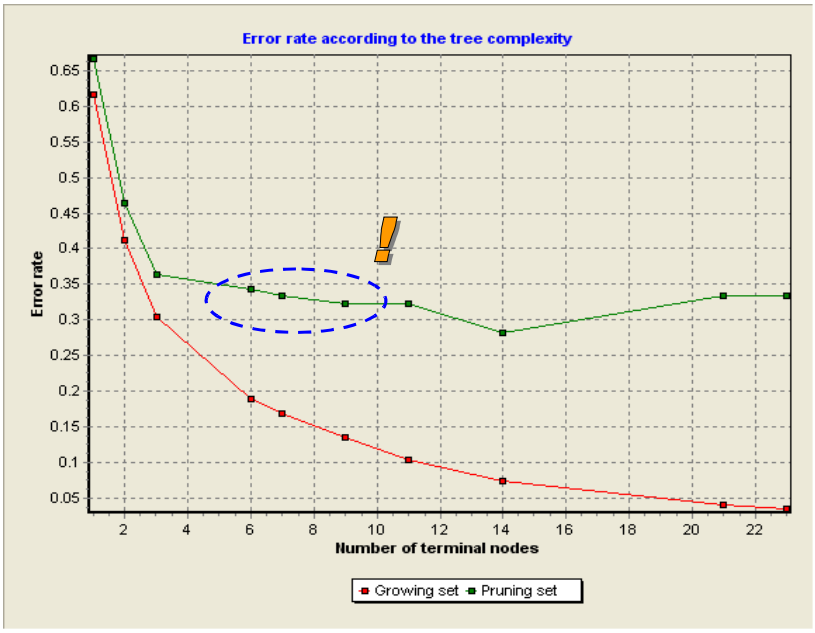
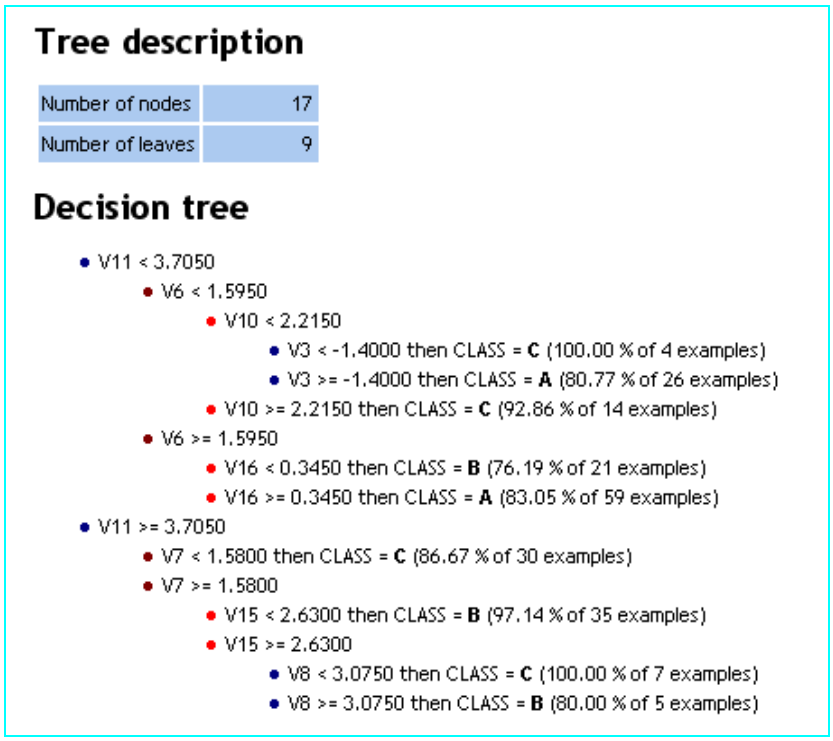


Figure 3 - Courbe "Nombre de feuilles vs. Taux d'erreur"

Nous nous rendons compte que les arbres à 7 ou 6 feuilles sont assez proches de la solution retenue. La procédure à suivre pour moduler la valeur de θ afin d'obtenir un de ces arbres est décrite dans un autre didacticiel[6].

Enfin, dernière information, et non des moindres, si nous revenons dans l'onglet précédent (onglet HTML), nous retrouvons dans la partie basse du rapport une description de l'arbre de décision.



Tree description

Number of nodes	17
Number of leaves	9

Decision tree

- V11 < 3.7050
  - V6 < 1.5950
    - V10 < 2.2150
      - V3 < -1.4000 then CLASS = **C** (100.00 % of 4 examples)
      - V3 >= -1.4000 then CLASS = **A** (80.77 % of 26 examples)
    - V10 >= 2.2150 then CLASS = **C** (92.86 % of 14 examples)
  - V6 >= 1.5950
    - V16 < 0.3450 then CLASS = **B** (76.19 % of 21 examples)
    - V16 >= 0.3450 then CLASS = **A** (83.05 % of 59 examples)
- V11 >= 3.7050
  - V7 < 1.5800 then CLASS = **C** (86.67 % of 30 examples)
  - V7 >= 1.5800
    - V15 < 2.6300 then CLASS = **B** (97.14 % of 35 examples)
    - V15 >= 2.6300
      - V8 < 3.0750 then CLASS = **C** (100.00 % of 7 examples)
      - V8 >= 3.0750 then CLASS = **B** (80.00 % of 5 examples)

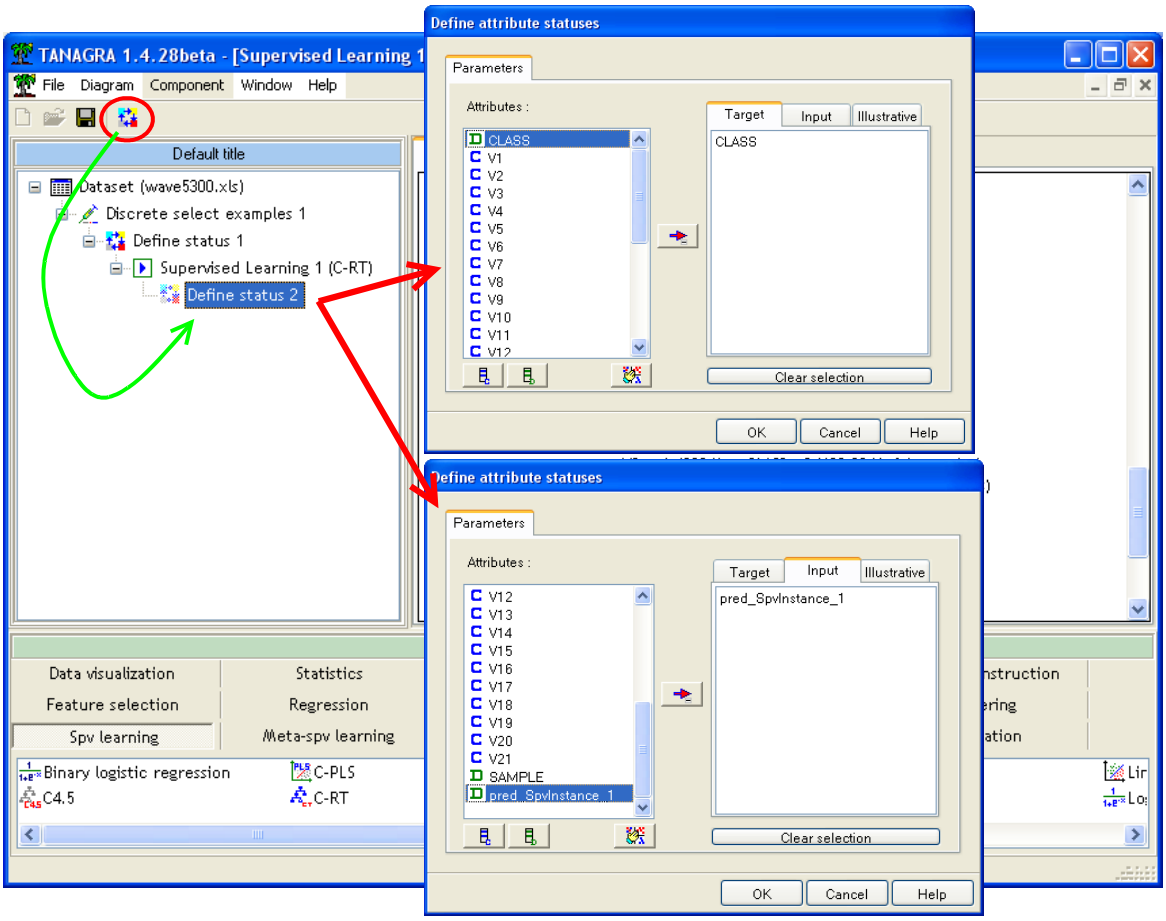
3.4 Evaluation sur l'échantillon « test »

Pour obtenir une évaluation non biaisée des performances de l'arbre en prédiction, le plus simple est de l'appliquer à un fichier à part, l'échantillon test, n'ayant jamais participé à l'élaboration du modèle.

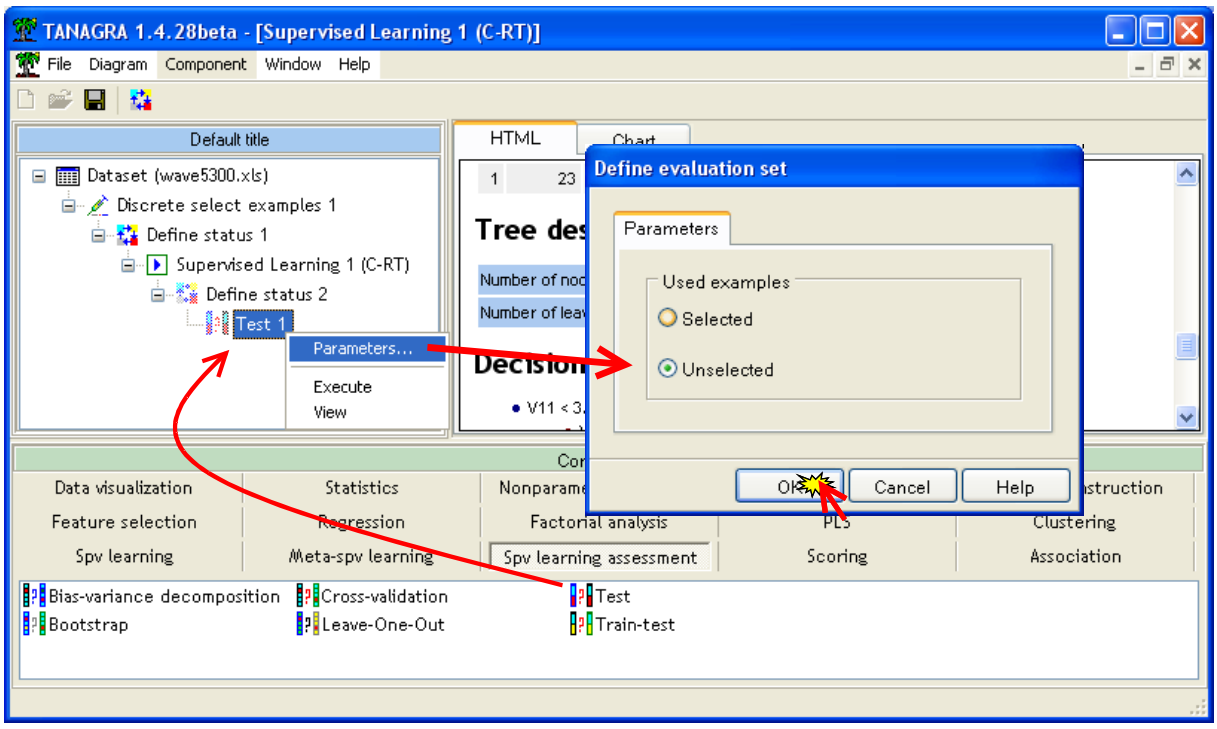
Nous pouvons le faire car les composants « apprentissage supervisé » de TANAGRA génèrent automatiquement une nouvelle de colonne de prédiction, calculée pour les observations sélectionnées (300 individus) et non sélectionnées c.-à-d. les 5000 que nous avions mis de côté. L'objectif est de confronter dans une matrice de confusion cette prédiction avec la vraie classe d'appartenance des individus.

Pour mener à bien cette opération, nous introduisons tout d'abord un nouveau composant DEFINE STATUS dans le diagramme. Nous plaçons CLASS en TARGET, et la nouvelle colonne PRED_SPVINSTANCE_1 en INPUT.

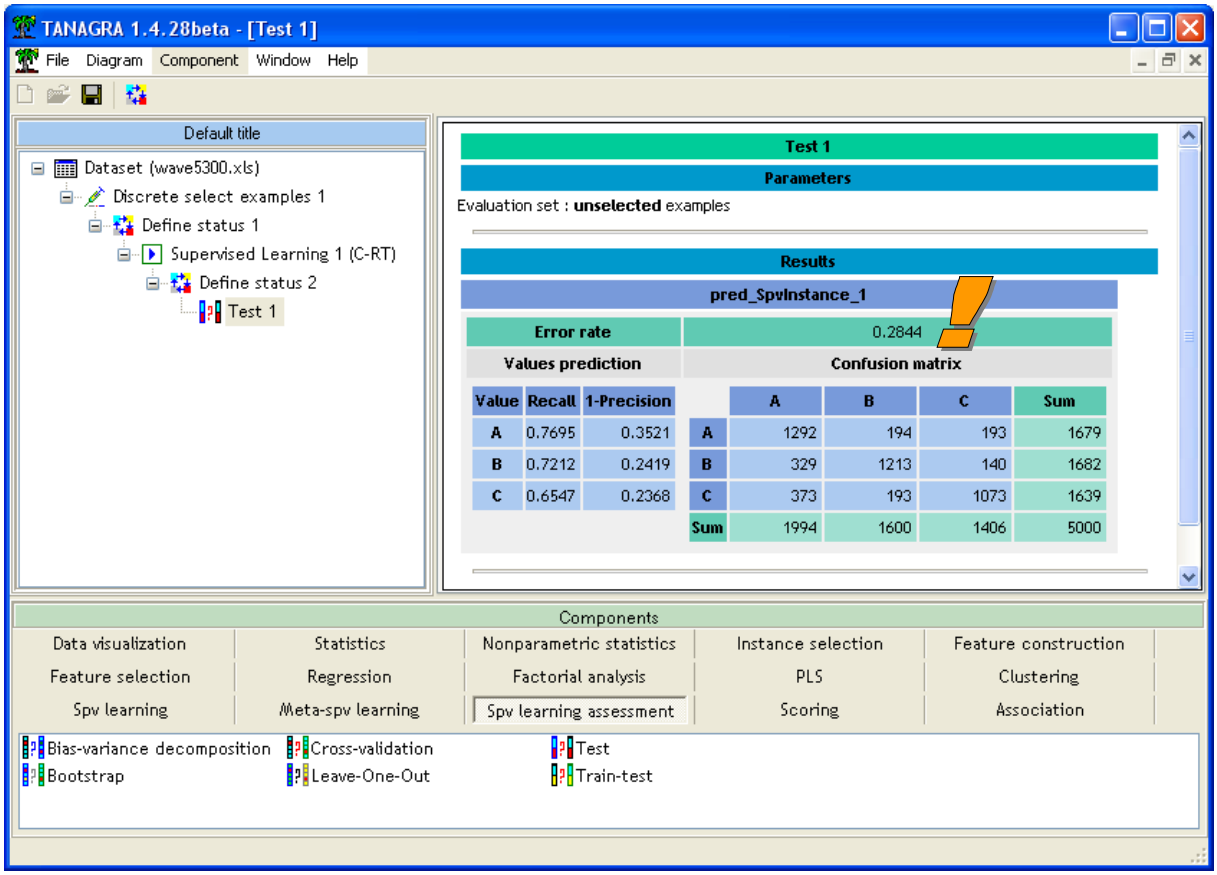
[6] http://tutoriels-data-mining.blogspot.com/2008/03/cart-dtermination-de-la-taille-de.html



Puis, nous insérons le composant TEST (onglet SPV LEARNING ASSESSMENT). Nous actionnons le menu PARAMETERS, nous constatons que nous pouvons réaliser l'évaluation soit sur les données auparavant sélectionnées (300) soit sur celles qui ont été mises de côté (5000). Nous travaillons bien évidemment sur ces dernières.



Nous cliquons sur VIEW, le taux d'erreur en test de l'arbre est 28.44% c.-à-d. si nous utilisons cet arbre pour prédire la classe d'appartenance d'un nouvel individu, nous avons 28.44% de « chances » de réaliser une prédiction erronée.

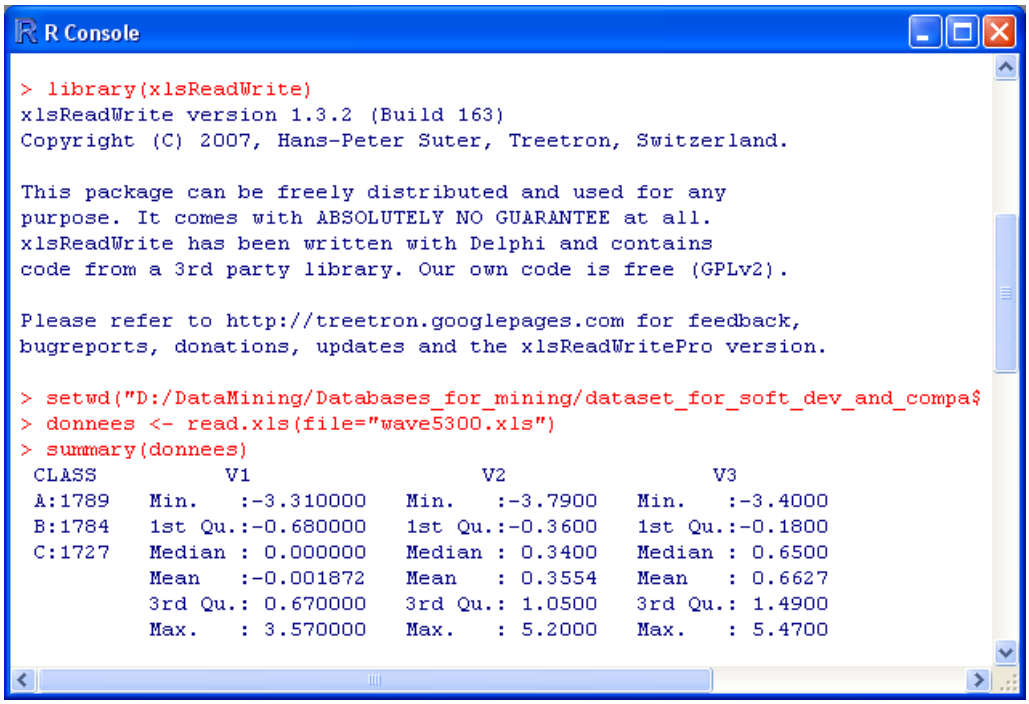


4 La méthode CART avec le package RPART de R

Nous considérons que le lecteur est familiarisé avec le logiciel R. Si ce n'est pas le cas, pas de panique, on trouve très facilement de nombreux supports en ligne, sur le site officiel du logiciel R par exemple http://cran.r-project.org/manuals.html et http://cran.r-project.org/other-docs.html

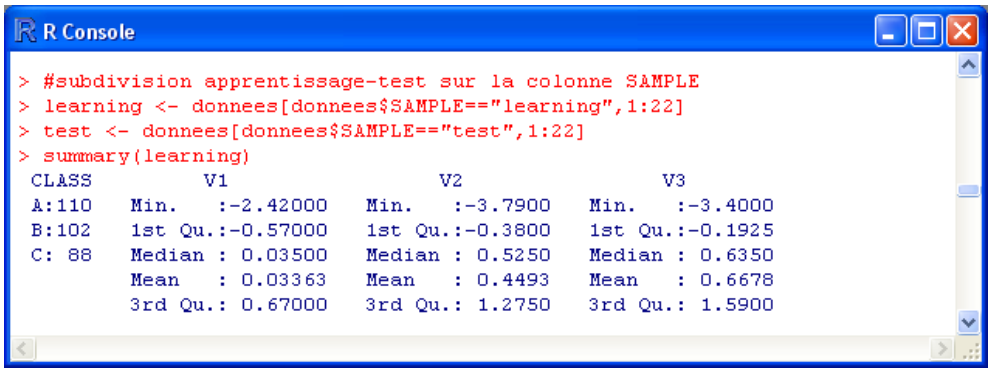
4.1 Importation d'un fichier EXCEL via le package xlsReadWrite

Pour importer un fichier EXCEL, le plus simple est de passer par le package xlsReadWrite. Nous le chargeons à l'aide de la commande **library(.).** Nous pouvons dès lors charger le fichier en spécifiant tout d'abord le répertoire adéquat, puis en invoquant la commande **read.xls(.)**[7]



4.2 Partition des données en apprentissage et test

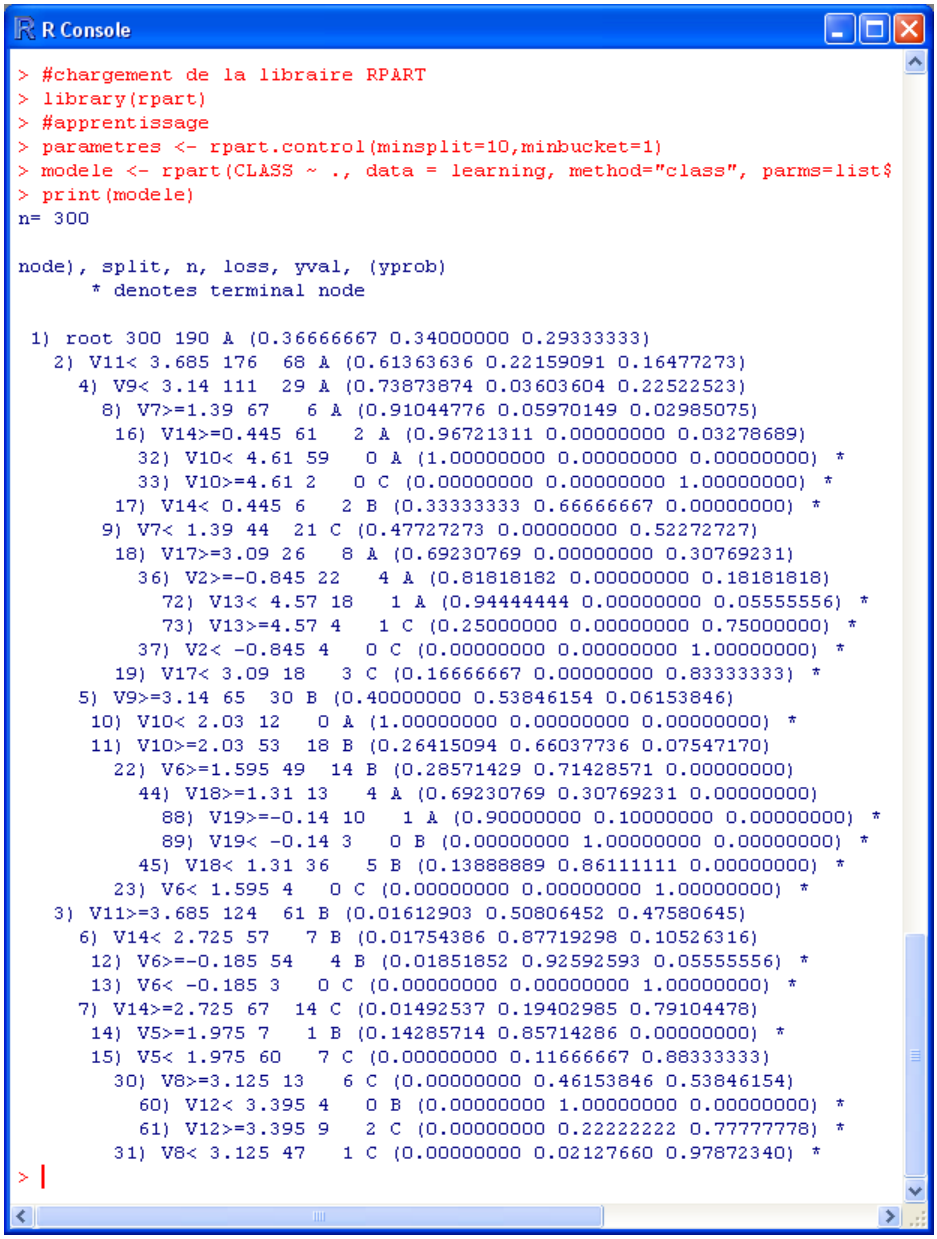
Nous exploitons la colonne SAMPLE pour subdiviser les données en apprentissage et test. Nous déduisons 2 sous-ensembles séparés. La 23ème colonne SAMPLE est par la suite évacuée.



[7] Le résultat complet de la commande **summary(.)** n'est pas affiché dans nos copies d'écran, cela prendrait trop de place. Elle est néanmoins indispensable à chaque étape pour vérifier l'intégrité des données.

4.3 Création de l'arbre maximal

Nous souhaitons appliquer la méthode CART sur les données en apprentissage. Nous faisons appel à la librairie RPART (voir Références, section 6). Nous spécifions quelques paramètres : MINSPLIT = 10 indique qu'on ne doit pas segmenter un nœud avec moins de 10 observations ; MINBUCKET = 1 nous permet de refuser toute segmentation où un des nœuds enfants aurait moins d'une observation. Nous nous rapprochons ainsi des paramètres utilisés dans TANAGRA. Nous remarquerons qu'il n'est pas nécessaire de définir la part de l'échantillon réservée à l'élagage ici. Nous saurons pourquoi plus loin. Pour l'heure, la commande RPART[8] lance l'apprentissage.



Surprise, RPART nous fournit l'arbre maximal avec 18 feuilles (avec des restrictions que l'on précisera plus loin), sans procéder au post élagage.

[8] L'option METHOD = « CLASS » indique que nous traitons un problème de classement, apprentissage supervisé ; SPLIT = GINI indique la mesure utilisée pour le choix des variables de segmentation.

4.4 Choix de l'arbre élagué

En réalité, **nous devons réaliser nous même le post élagage** en nous aidant du tableau « CPTABLE » obtenu avec la commande **printcp(.)**[9].

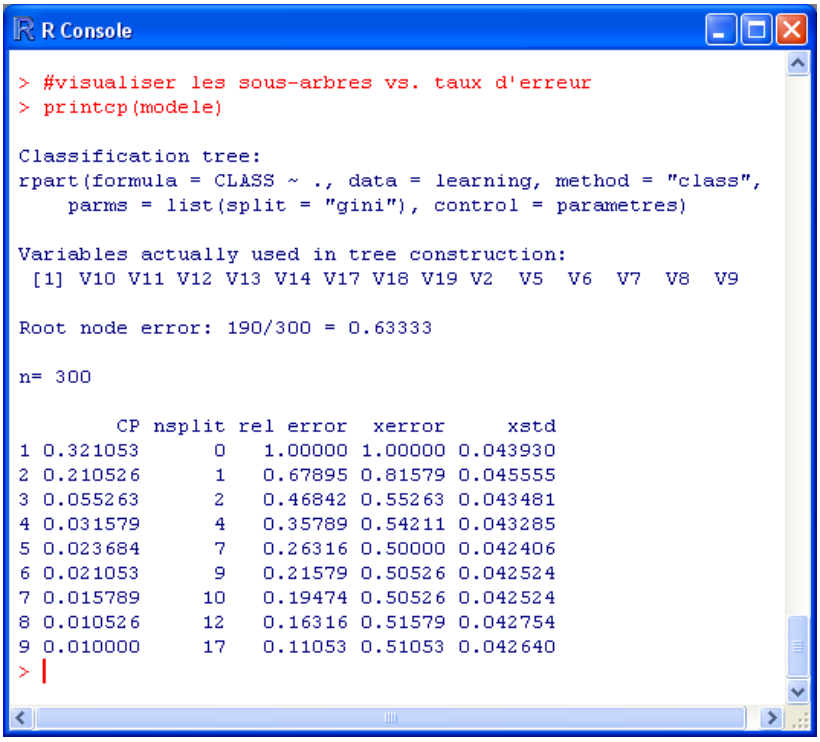


Figure 4 - Le tableau CPTABLE de RPART

Il décrit des séquences d'arbres en mettant en relation le coefficient de pénalité CP[10] avec : le nombre de segmentations (égal au nombre de feuilles – 1) ; l'erreur calculée sur l'échantillon d'apprentissage (normalisée de manière à ce que l'erreur sur la racine soit égale à 1) ; l'erreur calculée en validation croisée (normalisée également) ; l'écart type de l'erreur.

Ce tableau n'est pas sans rappeler celui produit par TANAGRA (Figure 2). A la différence que la valeur de CP est maintenant fournie, l'erreur (normalisée) est calculée en validation croisée. Penchons nous un instant sur la démarche suivie par RPART pour produire ces résultats.

Principe de la validation croisée. RPART construit l'arbre maximal avec la totalité de l'échantillon d'apprentissage, soit 300 observations. Ensuite, il calcule les séquences d'arbres emboîtés en modulant le paramètre de complexité CP. L'arbre élagué sera choisi parmi ces candidats. Pour chaque valeur de CP est calculée : l'erreur en resubstitution, à titre purement indicatif puisqu'on sait qu'elle diminue à mesure que la taille de l'arbre augmente ; l'erreur en validation croisée, de manière pertinente cette fois-ci car nous l'utilisons pour choisir l'arbre final.

L'erreur en validation croisée est substituée à l'erreur sur « pruning set » de TANAGRA. Comment procède RPART pour l'obtenir ?

[9] La subdivision étant réalisée de manière aléatoire lors de la validation croisée, il est normal que nous n'ayons pas exactement les mêmes résultats.

[10] Par défaut, RPART limite les séquences d'arbres à CP = 0.01 pour ne pas avoir à afficher une structure inutilement trop grande. Ce paramètre peut être modifié, CP = 0.0 correspond au véritable arbre maximal.

En interne, RPART partitionne les données en XVAL = 10 portions (paramétrable). Il calcule un arbre sur les 9/10, et utilise la fraction restante 1/10 pour estimer l'erreur. En faisant tourner les sous échantillons, il dispose donc de 10 arbres. Il utilise alors les valeurs de CP de la table précédente pour réduire chaque arbre et calculer l'erreur sur la fraction test qui leur est associée. Ainsi, pour une valeur spécifiée de CP, il dispose en interne de 10 arbres avec leurs taux d'erreur respectifs. La moyenne de ces valeurs constitue le taux d'erreur en validation croisée, RPART calcule également l'écart type.

Ce dispositif est réellement intéressant lorsque nous travaillons sur des fichiers de petite taille. Avoir à partitionner les données en « growing set » et « pruning set » peut pénaliser l'apprentissage. En effet, une partie de l'information disponible n'est pas utilisée pour choisir les variables de segmentation sur les nœuds lors de l'élaboration de l'arbre maximal. A contrario, lorsque la taille du fichier est importante, le temps de calcul de la validation croisée peut s'avérer prohibitif. Nous comparerons avec beaucoup de curiosité les résultats en test de TANAGRA et RPART pour notre fichier de données avec n = 300 observations disponibles. Souvenons nous que TANAGRA n'utilise que 201 individus pour la construction de l'arbre maximal, le reste étant dédié au post élagage.

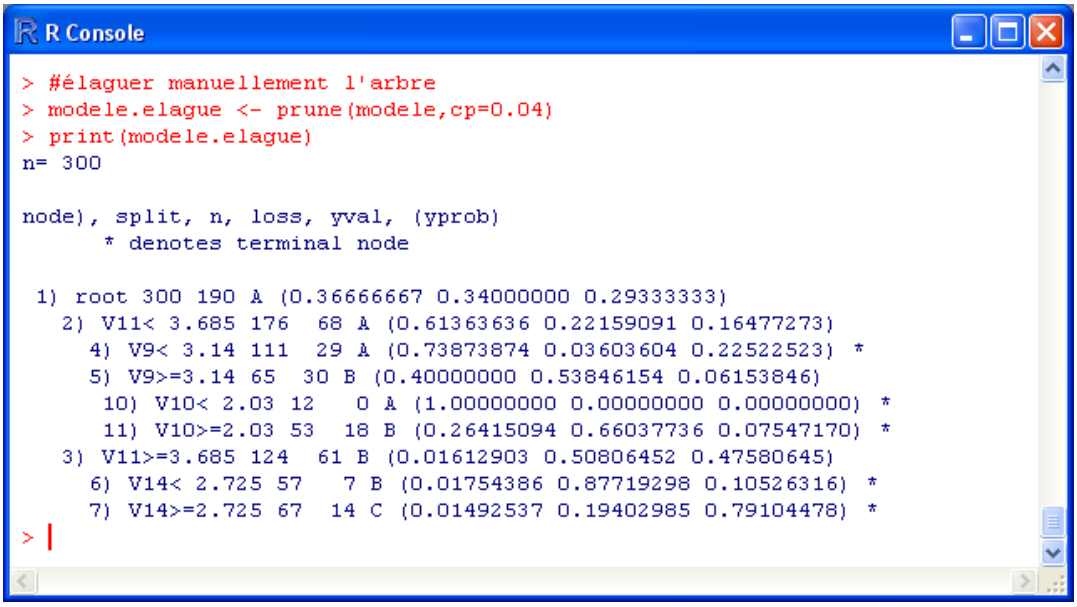
Choix de l'arbre élagué. Pour l'heure, revenons à notre tableau CPTABLE (Figure 4), nous allons essayer d'exploiter les informations proposées pour choisir l'arbre définitif.

L'arbre n°5, avec 7 découpages, soit 8 feuilles, est celle qui minimise l'erreur en validation croisée (ε = 0.50000). Pour obtenir cet arbre, nous devrions faire appel à la commande **prune(.)** en attribuant au paramètre **cp** n'importe quelle valeur comprise entre (>) 0.023684 et (≤) 0.031579.

Mais ce n'est pas un résultat souhaitable. Nous voulons introduire la préférence à la simplicité en appliquant la règle de l'écart type (Breiman et al., 1984 ; section 3.4.3, pages 78 à 81). Calculons l'erreur seuil à l'aide des valeurs fournies par le tableau

$$\varepsilon_{seuil} = 0.5 + 1 \times 0.042406 = 0.542406$$

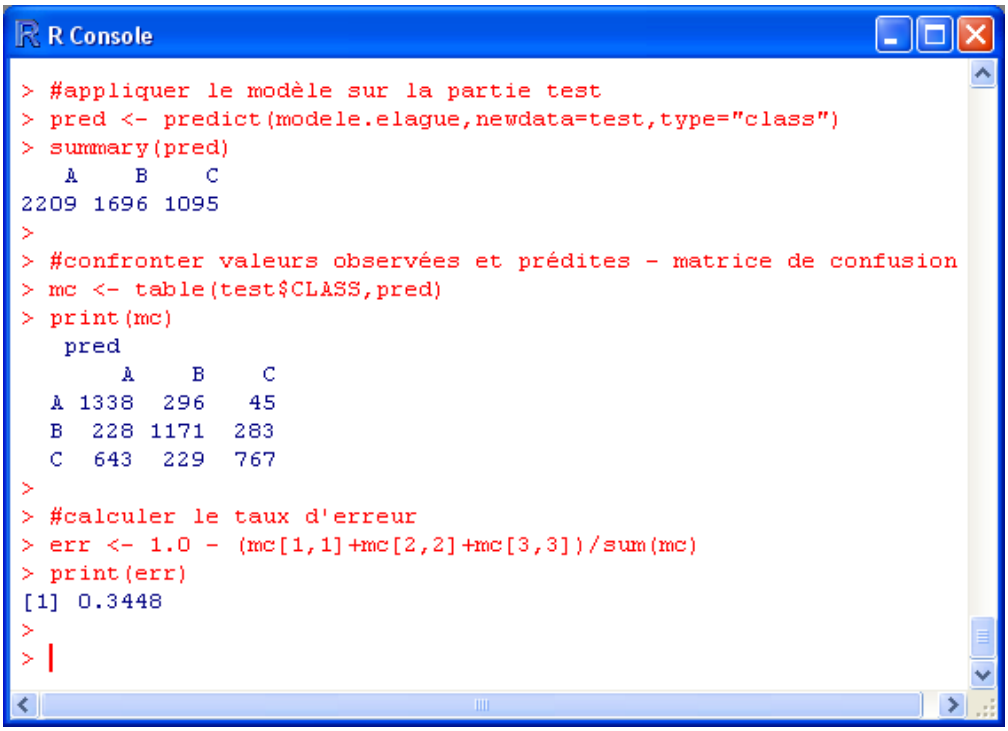
Le plus petit arbre dont l'erreur est inférieure à ce seuil est le n°4 avec 4 segmentations, soit 5 feuilles, son erreur en validation croisée est ε = 0.54211. Pour obtenir cet arbre, nous devons attribuer à cp une valeur comprise entre 0.031579 < cp ≤ 0.055263. Mettons cp = 0.04 pour simplifier. Nous faisons appel à la commande **prune(.)**



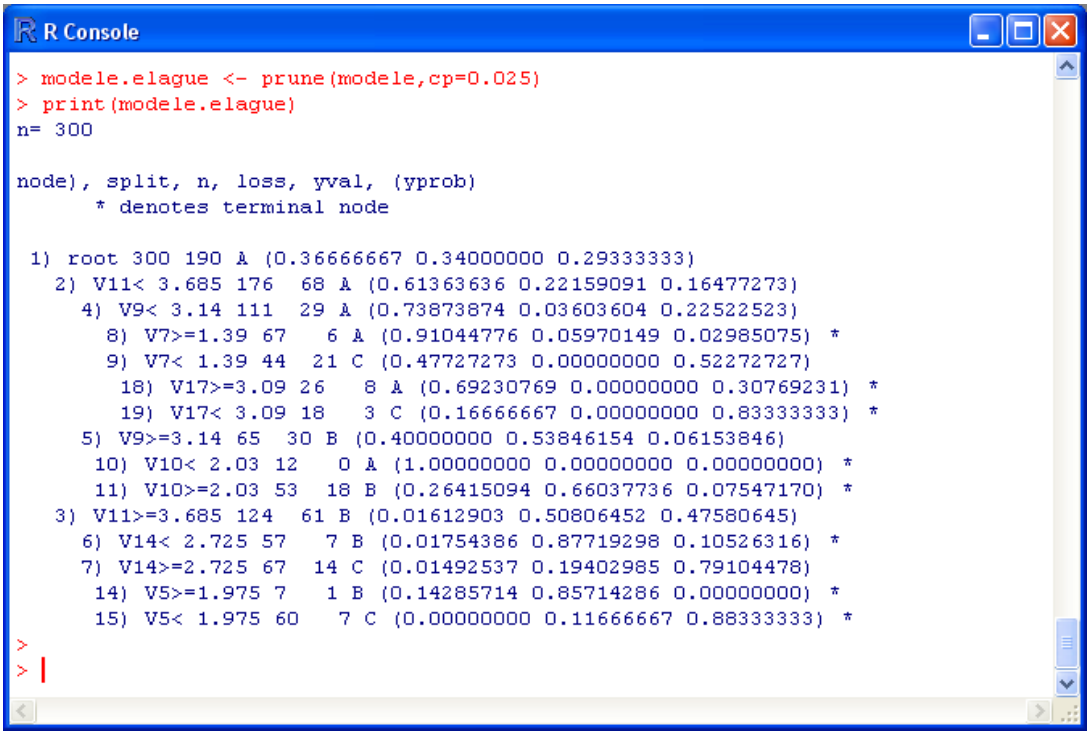
C'est l'arbre que nous choisissons en accord avec la règle de l'écart type de Breiman et al (1984). Nous allons l'évaluer sur l'échantillon test de 5000 observations que nous avons mis de côté.

4.5 Evaluation en test

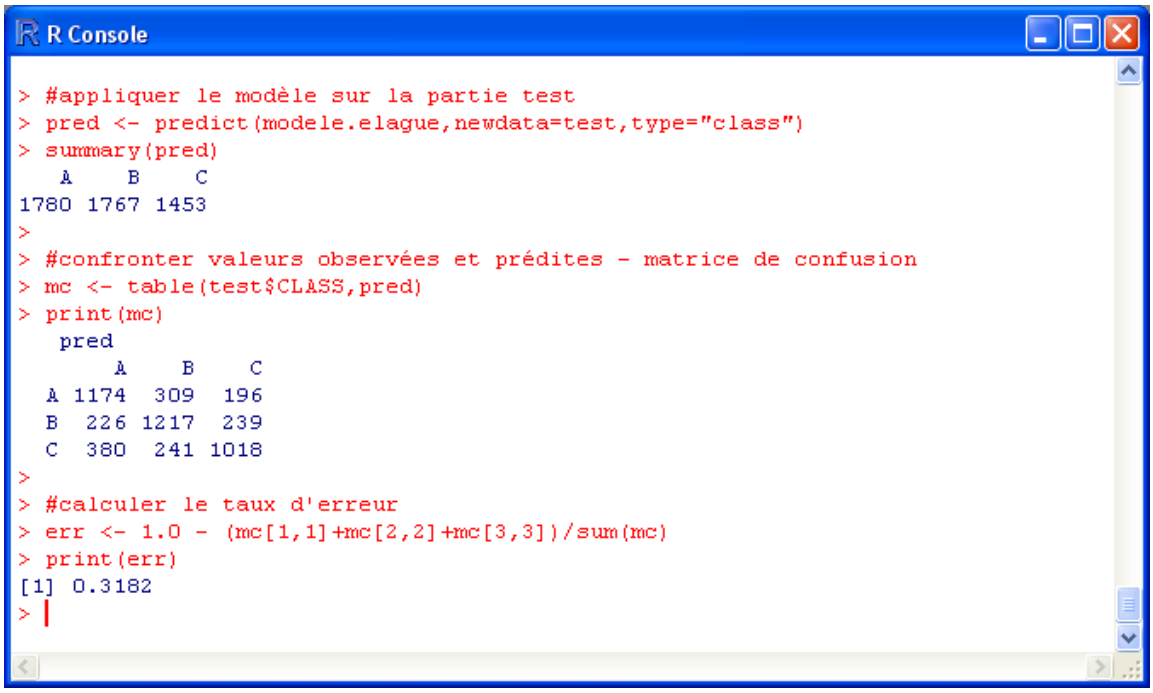
Nous créons une nouvelle colonne de prédiction à l'aide de la commande **predict(.)**. Nous calculons la matrice de confusion et nous en déduisons le taux d'erreur en test.



Dans ce cas précis, le taux d'erreur en test est 34.48%. Finalement, peut être que nous avons trop réduit l'arbre. Si nous choisissons l'arbre à 8 feuilles, avec cp = 0.025, nous obtenons



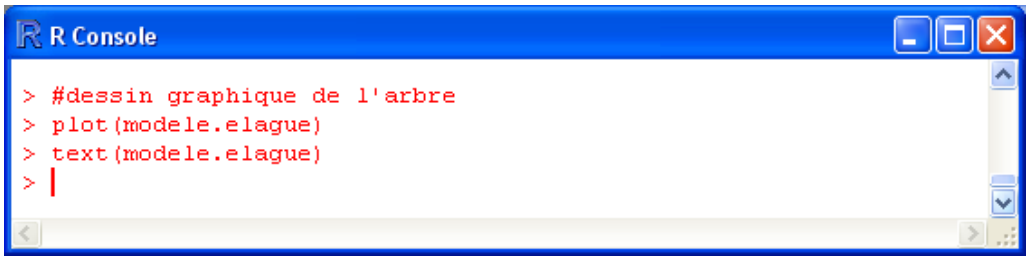
Avec des performances en test



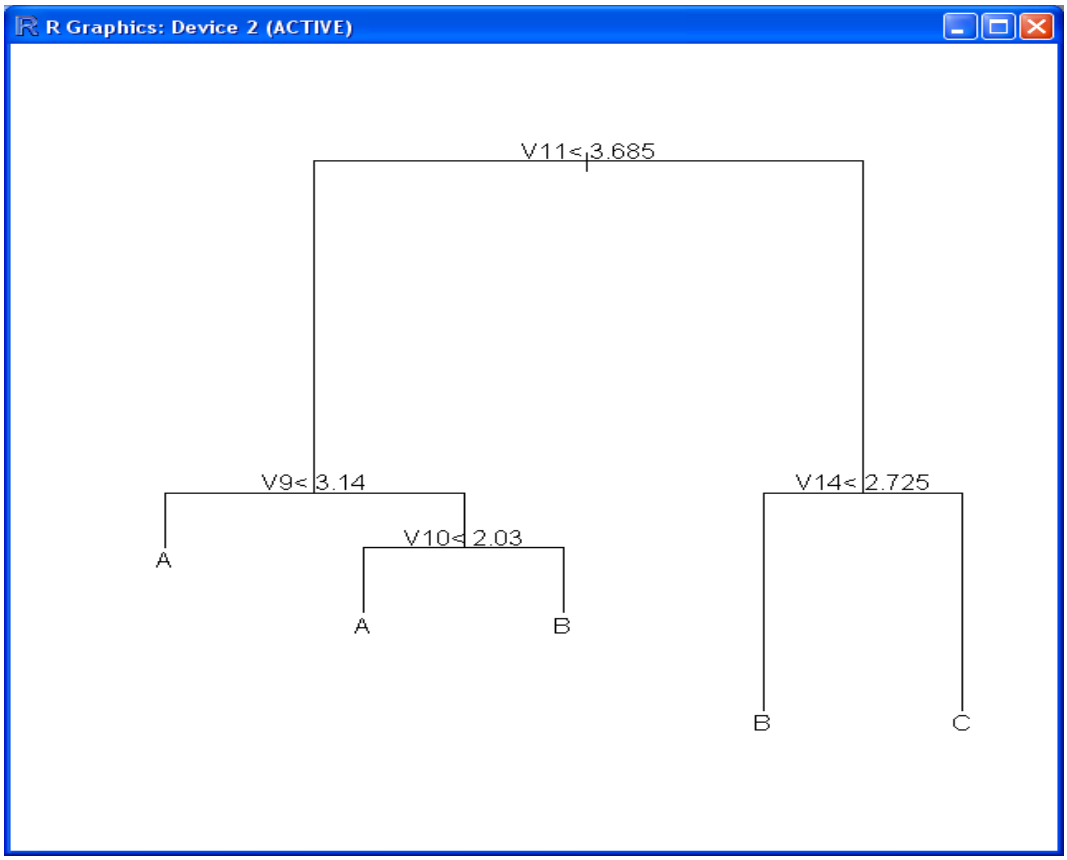
Le taux d'erreur est passé à 31.82%... et nous devons nous arrêter là car **si nous commençons à utiliser le fichier test des 5000 observations pour déterminer l'arbre optimal, il perd son statut de juge impartial** des performances.

4.6 Dessin graphique de l'arbre

RPART propose des fonctions spécifiques pour dessiner graphiquement l'arbre de décision.



Nous obtenons :



Il est également possible d'obtenir des sorties en Postscript (voir la documentation pour plus de détails).

5 Conclusion

La méthode CART est peu présente dans les logiciels libres. Pourtant, elle possède des qualités indéniables qui la positionnent avantageusement par rapport aux autres techniques d'induction d'arbres. Elle est capable de produire des modèles « clés en main », avec des performances au moins aussi bonnes que les autres. Mais, en plus, elle sait nous proposer des scénarios de solutions. Le graphique (Figure 3) ou le tableau (Figure 4) des erreurs sont de précieux outils d'aide à la décision. Ils nous permettent d'intervenir dans le choix du modèle définitif, en fonction des objectifs de l'étude et des caractéristiques des données. C'est un atout essentiel dans le Data Mining.

6 Références

- L. Breiman, J. Friedman, R. Olsen, C. Stone, *Classification and Regression Trees*, Chapman & Hall, 1984.
- A. Guéguen, « Arbres de discrimination binaire », in *Analyse discriminante sur variables qualitatives*, Celeux et Nakache éditeurs, 1994 ; chapitre 7.
- D. Zighed, R. Rakotomalala, *Graphes d'Induction : Apprentissage et Data mining*, Hermès, 2000 ; chapitre 5 « CART ».
- M. Bardos, *Analyse discriminante : Application au risque et scoring financier*, Dunod, 2001 ; chapitre 4 « Arbres de partitionnement ».

- J.P. Nakache, J. Confais, *Statistique Explicative Appliquée*, Technip, 2003 ; chapitre 8 « Méthode de segmentation CART ».
- G. Saporta, *Probabilités, Analyse de données et Statistique*, Technip, 2006 ; chapitre 19, section 19.1 « Arbres de régression et de discrimination ».
- M. Tenenhaus, *Statistique : Méthodes pour Décrire, Expliquer et Prévoir*, Dunod, 2007 ; chapitre 13, section 3 « La méthode CART ».
- Concernant le package RPART
    - http://cran.cict.fr/web/packages/rpart/rpart.pdf pour la description technique du package ;
    - « An introduction to Recursive Partitioning – Using the RPART Routines » (short version) - http://eric.univ-lyon2.fr/~ricco/cours/didacticiels/r/short_doc_rpart.pdf
    - « An introduction to Recursive Partitioning – Using the RPART Routines » (long version) http://eric.univ-lyon2.fr/~ricco/cours/didacticiels/r/long_doc_rpart.pdf