1 Objectif

Data Mining avec le logiciel Scilab.

Je connais le nom « Scilab[1] » depuis fort longtemps. Je l'avais catalogué dans la catégorie des logiciels de calcul numérique, au même titre qu'Octave ou, pour parler d'outils commerciaux, de Matlab que j'avais moi-même utilisé du temps où j'étais étudiant. J'y voyais très peu d'intérêt dans le contexte du traitement statistique des données et du data mining.

Récemment un collègue mathématicien m'en a reparlé. Il s'étonnait de la faible visibilité de Scilab au sein de la communauté du data mining, sachant qu'il présente des fonctionnalités tout à fait similaires à celle de R. Ah bon ? Je ne voyais pas les choses ainsi. Curieux comme je suis, je me suis bien évidemment documenté sur la question en fixant un objectif simple : est-ce qu'il est possible de réaliser - simplement, sans contorsions extravagantes – un schéma type d'analyse prédictive avec Scilab ? A savoir : charger un fichier de données (échantillon d'apprentissage), construire un modèle prédictif, en détailler les caractéristiques, charger un échantillon test, appliquer le modèle sur ce second ensemble de données, élaborer la matrice de confusion et calculer le taux d'erreur en test.

Nous verrons dans ce tutoriel que la tâche a été réalisée avec succès, relativement facilement. Scilab est tout à fait armé pour réaliser des traitements statistiques. D'emblée, deux petits bémols me sont clairement apparus lors de la prise en main de Scilab : les librairies de fonctions statistiques existent mais ne sont pas aussi fournies que celles présentes dans R ; leur documentation laisse vraiment à désirer, j'ai du pas mal batailler – et encore, je savais exactement ce que je cherchais – avant de trouver les indications nécessaires sur le web.

Il reste qu'au final, je suis très satisfait de cette première expérience. J'ai découvert un excellent outil gratuit, souple et performant, très facile à prendre en main, qui s'avère être une alternative tout à fait crédible à R dans le domaine du data mining.

2 Le logiciel Scilab

2.1 Présentation du logiciel

Scilab est présenté comme suit sur le site de Wikipédia : « *Scilab (prononciation : [sajlab] contraction de Scientific Laboratory en anglais) est un logiciel libre de calcul numérique multi-plate-forme fournissant un environnement de calcul pour des applications scientifiques. Il possède un langage de programmation orienté calcul numérique de haut niveau. Il peut être utilisé pour le traitement du signal, l'analyse statistique, le traitement d'images, les simulations de dynamique des fluides, l'optimisation numérique, et la modélisation et simulation de systèmes dynamiques explicites et implicites.* » (Wikipédia, page **Scilab** du 02/11/2013).

Dans le contexte du data mining, j'ai noté plusieurs caractéristiques très intéressantes en décortiquant la documentation :

1. Scilab propose un mécanisme de gestion de données. Il dispose notamment de tous les outils pour manipuler les vecteurs et les matrices.

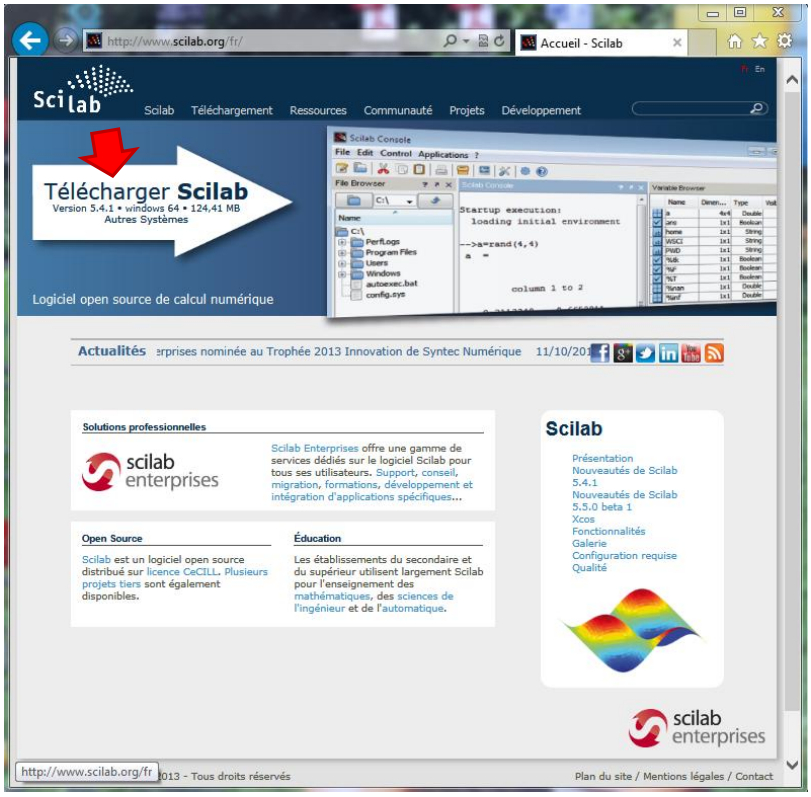
[1] https://www.scilab.org/fr

2. Il peut charger les fichiers de données « individus x variables » usuels du data mining, entres autres ceux au format tableur CSV (« comma separated value », avec en option le caractère tabulation comme séparateur)[2].
3. Il dispose d'un langage de programmation interprété, avec tous les attributs associés : types de données, structures algorithmiques (branchements conditionnels, boucles), programmation modulaire (découpage des programmes en procédures et fonctions).
4. Les objets issus des procédures statistiques possèdent des propriétés que nous pouvons exploiter dans les calculs ultérieurs.
5. Il propose des fonctionnalités graphiques très puissantes.
6. Il est possible d'enrichir – à l'infini – la bibliothèque des fonctions à l'aide de modules externes (les « toolbox ») que nous pouvons nous même créer et distribuer[3]. Un dépôt recense et rend disponible les librairies reconnues à ce jour[4]. Un mécanisme d'importation automatique facilite grandement leur installation et leur utilisation (ATOMS : AuTomatic mOdules Management for Scilab - http://atoms.scilab.org/).

J'ai l'impression de décrire les spécifications de R ! A ce stade, je me suis dit qu'à part l'apprentissage d'un nouveau langage, l'identification et la compréhension des commandes nécessaires à mon analyse, il ne devrait pas être trop difficile de réaliser notre étude.

2.2 Téléchargement et installation

Le logiciel peut être téléchargé sur le site web de l'éditeur.



[2] http://help.scilab.org/docs/5.4.1/fr_FR/csvRead.html

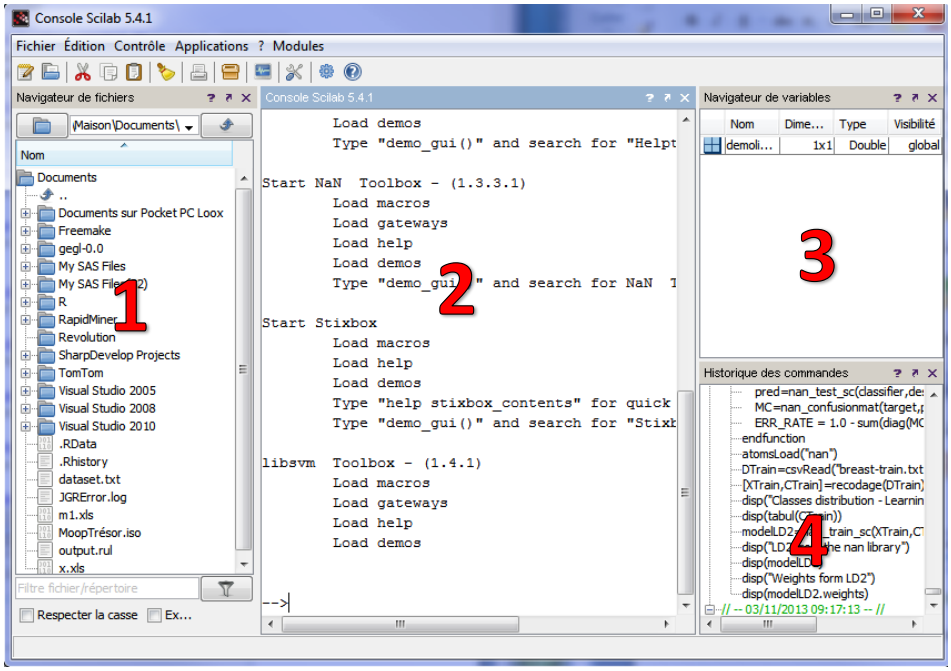
[3] http://wiki.scilab.org/howto/Create%20a%20toolbox (**How to create a module**).

[4] http://forge.scilab.org/index.php/projects/

Sous Windows, nous disposons d'un fichier SETUP qui se charge entièrement de l'installation. J'ai récupéré la **5.4.1** pour Windows 64 bits. Des versions pour d'autres plateformes sont disponibles.

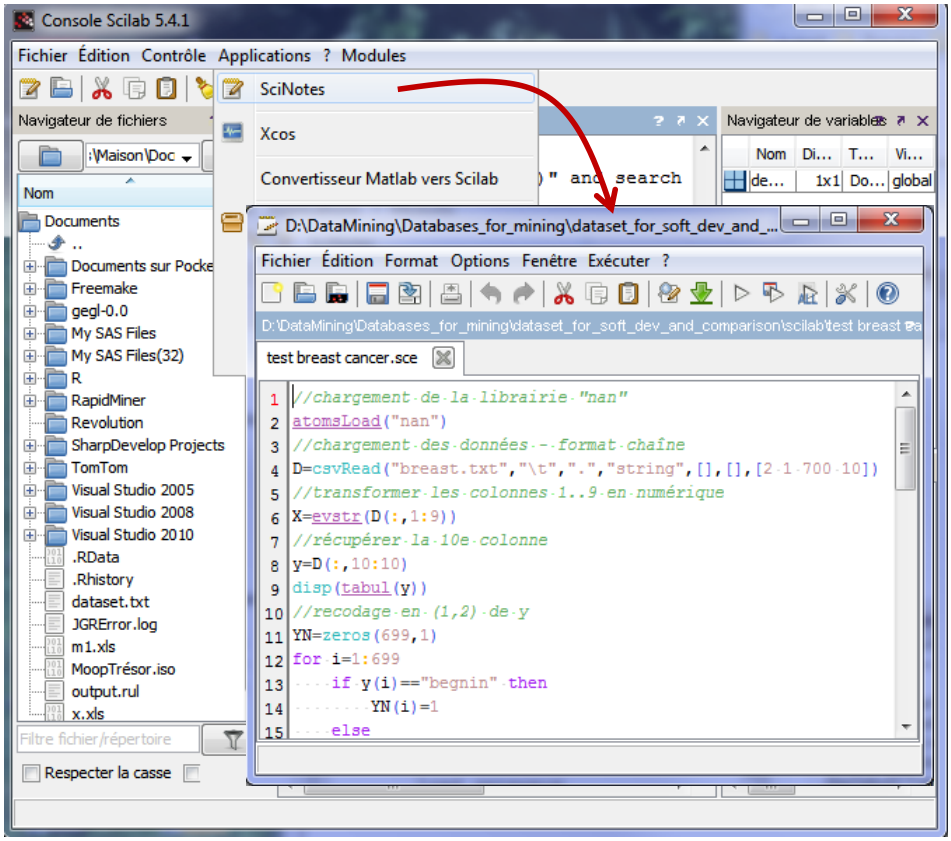
2.3 Démarrage de l'application

Au lancement de Scilab, nous observons un espace de travail subdivisé en plusieurs parties.

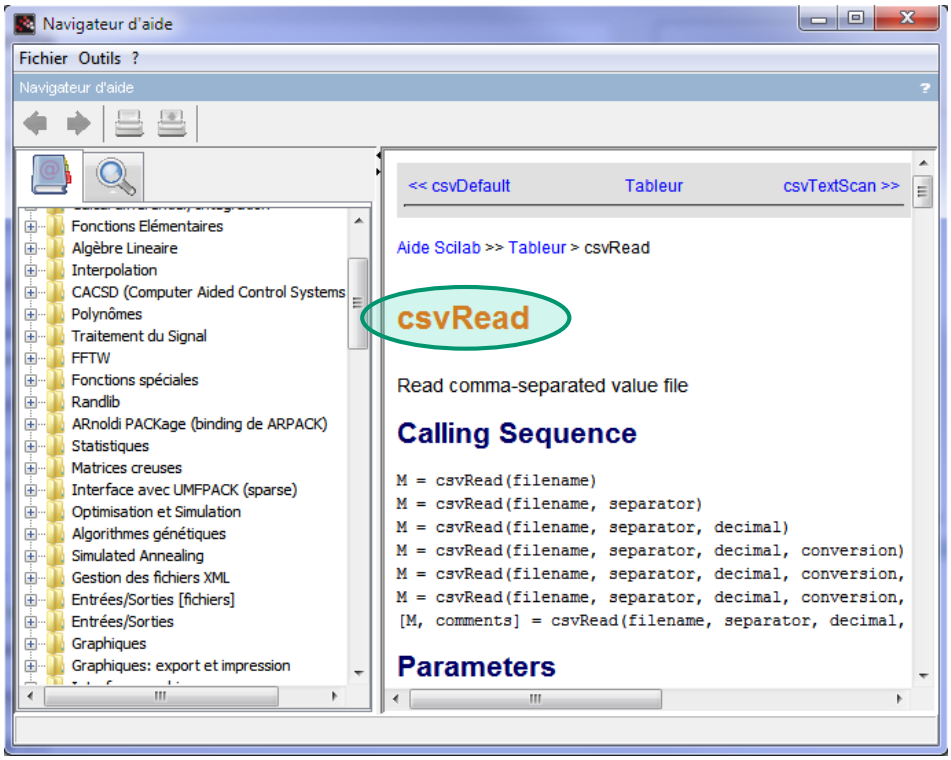


Avec : (1) un navigateur de fichiers ; (2) la console, nous permettant de saisir les commandes et visualiser les résultats ; (3) un navigateur de variables ; (4) un historique des commandes.

Pour le traitement par lots, l'écriture de programmes, il est préférable d'utiliser l'éditeur de code SCINOTES. Nous actionnons le menu APPLICATIONS / SCINOTES, une nouvelle fenêtre apparaît. Il permet d'éditer les fichiers Scilab (*.sce).



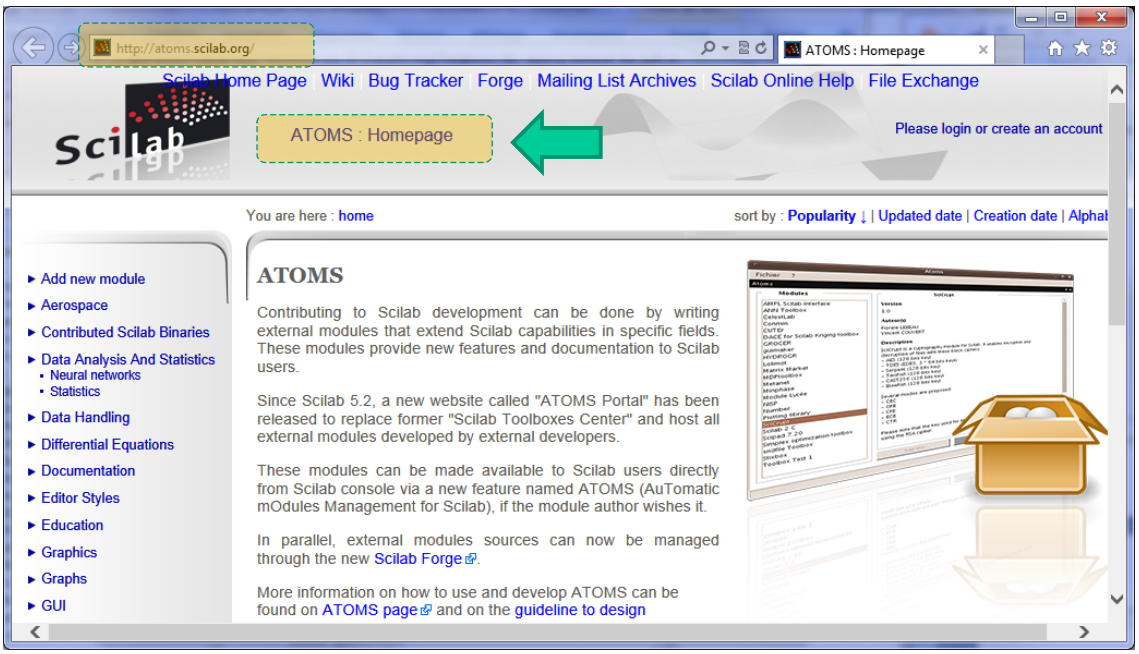
Nous utilisons la commande « **help** » pour obtenir de l'aide. Par ex : **help('csvRead')** affiche dans une fenêtre dédiée l'aide relative à la lecture des fichiers CSV.



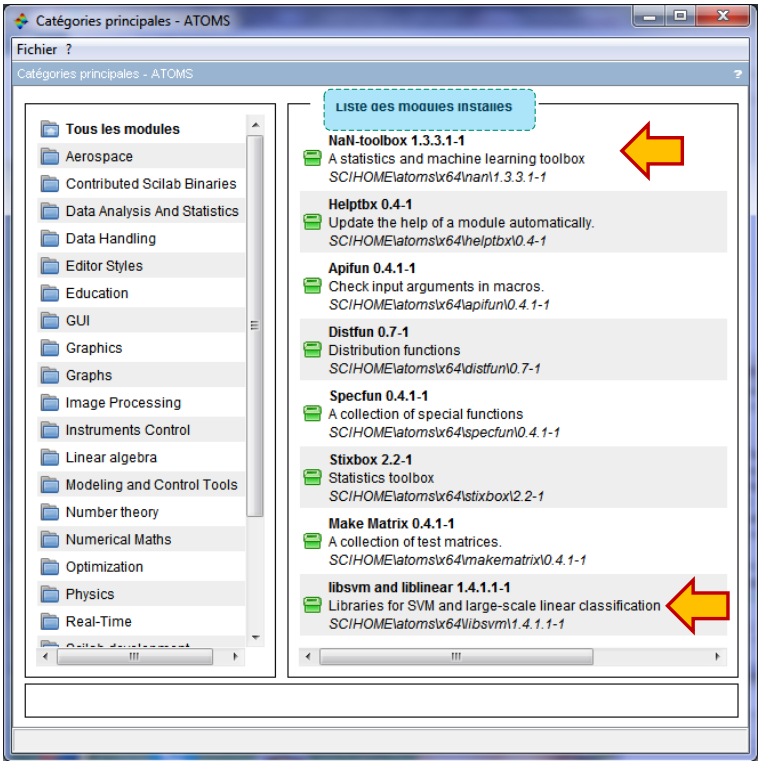
2.4 Les librairies externes (les « toolbox »)

Scilab propose un mécanisme de gestion de packages (« Toolbox »). Il permet d'enrichir sa bibliothèque de méthodes numériques. Cette caractéristique est, entres autres, à l'origine du succès fulgurant de R ces dernières années. Scilab peut prendre le même chemin. On notera néanmoins la relative rareté des modules de data mining et de statistique disponibles à ce jour et, plus ennuyeux à mon sens, la quasi absence de documentation un tant soit peu approfondie.

La liste des Toolbox est accessible sur la page d'ATOMS.



Nous pouvons également la consulter avec la commande **atomsGui()**.

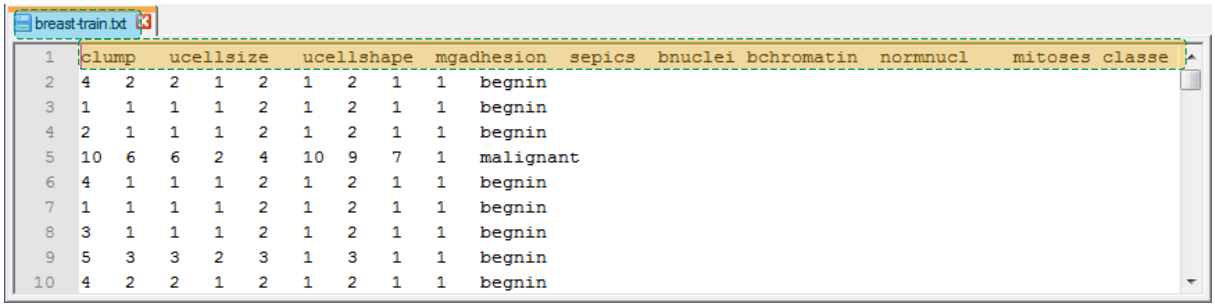


Une fenêtre de gestion des Toolbox apparaît. Nous constatons dans la copie d'écran ci-dessus que les Toolbox « **Nan** » et « **libsvm** » dédiées à l'apprentissage automatique sont présentes sur ma machine. Nous pouvons utiliser cette interface graphique pour les installer. Nous pouvons également utiliser directement les commandes **atomsInstall('nom de la toolbox')** et **atomsLoad('nom de la toolbox')** pour les installer et les charger. Scilab récupère la librairie sur le web. Une fois installée, une Toolbox est automatiquement chargée au prochain démarrage de Scilab[5].

3 Analyse prédictive avec Scilab

3.1 Données à traiter

Nous manipulons la base BREAST-CANCER-WISCONSIN[6] dans ce tutoriel. Elle comporte 9 descripteurs numériques et une variable cible CLASSE à deux modalités (BEGNIN et MALINGNANT). Nous l'avons subdivisée aléatoirement en 2 fichiers : « breast-train.txt » représente l'échantillon d'apprentissage avec 399 observations ; « breast-test.txt », l'échantillon test avec 300 observations. Dans les deux cas, la première ligne est constituée des noms de variables. Les colonnes sont séparées par le caractère tabulation. Nous montrons ci-dessous les premières lignes de « breast-train.txt ».


breast-train.txt

	clump	ucellsize	ucellshape	mgadhesion	sepics	bnuclei	bchromatin	normnucl	mitoses	classe
2	4	2	2	1	2	1	2	1	1	begnin
3	1	1	1	1	2	1	2	1	1	begnin
4	2	1	1	1	2	1	2	1	1	begnin
5	10	6	6	2	4	10	9	7	1	malignant
6	4	1	1	1	2	1	2	1	1	begnin
7	1	1	1	1	2	1	2	1	1	begnin
8	3	1	1	1	2	1	2	1	1	begnin
9	5	3	3	2	3	1	3	1	1	begnin
10	4	2	2	1	2	1	2	1	1	begnin

3.2 Importation des données d'apprentissage et recodage

Nous chargeons le fichier dans la variable (matrice) DTRain à l'aide de la procédure csvRead().

```
//chargement des données - format chaîne
DTrain=csvRead("breast-train.txt","\t",".","string",[],[],[2 1 400 10])
```

La dernière option est importante. Elle spécifie les coordonnées des données dans le fichier. La première ligne contenant les noms de variables est délibérément ignorée, nous chargeons à partir de la 2nde [**2**] ; à partir de la première colonne [**1**] ; jusqu'à la ligne [**400**], qui représente la 399ème observation ; et [**10**] la 10ème colonne.

Autre précision très importante, pour l'instant toutes les valeurs sont considérées comme des chaînes de caractères, c'est le sens de l'option « **string** » dans la commande ci-dessus.

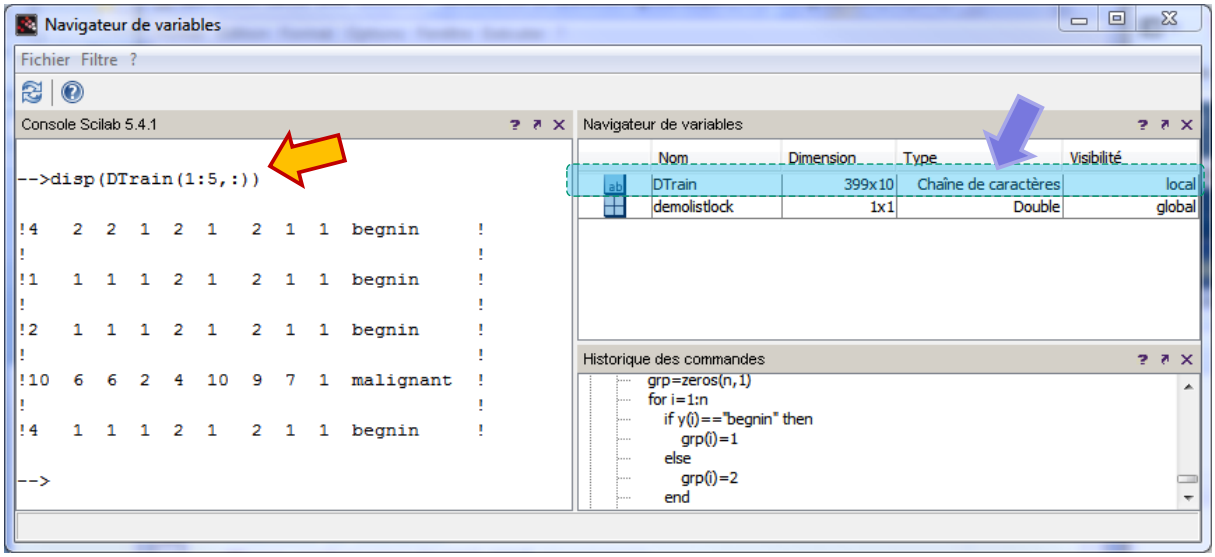
Nous affichons les 5 premières lignes des données l'aide de la commande disp(). Notez les spécifications des plages d'indices en ligne et en colonne.

```
//affichage des 5 premières observations
disp(DTrain(1:5,:)) //nous aurions pu écrire disp(DTrain(1:5,1:$))
```

Nous obtenons :

[5] http://wiki.scilab.org/ATOMS

[6] http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29



Les données ne sont pas exploitables en l'état. Il faut indiquer à Scilab que les descripteurs sont en réalité numériques, et que la cible (10ème colonne) est une variable qualitative binaire à 2 modalités (« begnin » et « malignant ») que nous codons en 1 et 2.

Nous créons la fonction **recodage()** pour ce faire. Nous pourrons la réutiliser pour traiter l'échantillon test. Elle renvoie les valeurs des descripteurs dans une matrice, et celles de la cible dans un vecteur.

```
//fonction pour le recodage de la base en numérique
function [descriptors, target]=recodage(D)
    //transformer les colonnes 1 à 9 en numérique
    descriptors = evstr(D(:,1:9))
    //récupération de la dernière colonne classe
    y=D(:,10:10)
    //recodage en (1,2) de la cible
    n=size(y,1) //n taille d'un vecteur = nombre d'observations
    grp=zeros(n,1) //créer un vecteur de zéros
    for i=1:n
        if y(i)=="begnin" then
            grp(i)=1
        else
            grp(i)=2
        end
    end
    target=grp
endfunction
```

Plusieurs remarques viennent à l'écriture de cette fonction :

1. Il est très aisé d'écrire une fonction renvoyant plusieurs valeurs avec le langage Scilab.
2. La fonction **evstr()** permet d'effectuer une modification de type (chaîne vers numérique) sur l'ensemble des lignes des colonnes de 1 à 9.
3. **size**() renvoie les dimensions des matrices. Telle que nous l'utilisons ici, il indique la taille du vecteur **y** c.-à-d. le nombre d'observations de notre échantillon.
4. **zeros**() permet d'initialiser le vecteur **grp** avec la valeur 0. Cette instruction est surtout une astuce pour dimensionner dès le départ le vecteur et éviter son redimensionnement incrémental durant les affectations dans la boucle. Les modifications successives (c.-à-d. allocations successives de mémoire et copies des données) sont très gourmandes en temps de traitement.

Sur un petit ensemble de données, ce n'est peut être pas trop préjudiciable. Sur un grand fichier avec plusieurs dizaines de milliers d'observations, le temps d'attente est rédhibitoire.

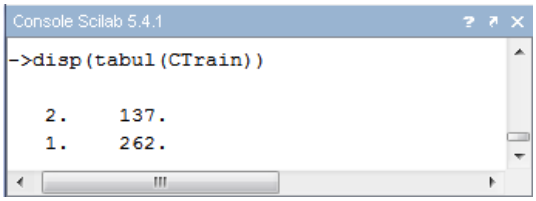
5. Nous utilisons une boucle **for** pour lire le contenu de **y** et écrire dans **grp** la valeur adéquate.
6. Nous observons dans la boucle for l'utilisation du branchement conditionnel **if**.

La fonction est rédigée de manière très scolaire. Les aficionados de Scilab doivent connaître moult astuces pour réduire le nombre de lignes de code et, surtout, pour la rendre plus efficace. Je n'en suis pas à ce stade. Mon objectif était de montrer de manière schématique ce que l'on peut faire. Cette fonction était aussi l'occasion de montrer les grandes lignes du langage de programmation Scilab. Il se révèle très intuitif et conforme de ce que l'on peut attendre d'un langage évolué.

Nous appelons la fonction comme suit dans le programme principal

```
//on peut utiliser un outil interne
[XTrain,CTrain]=recodage(DTrain)
disp(tabul(CTrain))
```

XTrain représente la matrice des descripteurs, CTrain le vecteur associé à la variable cible. La fonction **tabul**() fournit la distribution de fréquence des classes.



3.3 Construction du modèle prédictif avec la Toolbox « Nan » - Analyse discriminante

Il faut avoir installé et chargé la Toolbox « Nan » pour pouvoir poursuivre. Elle sera par la suite automatiquement chargée à chaque démarrage de Scilab.

Nous souhaitons utiliser l'analyse discriminante linéaire (LD2) dans un premier temps. La librairie propose plusieurs approches pour l'analyse discriminante prédictive. J'avoue ne pas avoir très bien saisi les subtilités qui les différencient. Malheureusement, l'aide de Scilab ne décrit que l'usage des procédures, pas les méthodes sous-jacentes [cf. **help**('nan_train_sc')].

Nous introduisons les commandes suivantes.

```
//apprentissage - méthode LD2
modelLD2=nan_train_sc(XTrain,CTrain,'LD2')
disp(modelLD2)
```

Scilab nous affiche.

```
-->disp(modelLD2)

  datatype: "classifier:statistical:ld2"
  Labels: [1,2]
  MD: hypermat
  NN: hypermat
  weights: [10x2 constant]
```

modelLD2 est un objet produit par la procédure **nan_train_sc**(), plusieurs champs sont disponibles. Pour accéder aux coefficients des fonctions de classement, nous faisons :

```
disp(modelLD2.weights)
```

L'opérateur « . » permet d'accéder aux champs d'un objet. Scilab affiche alors :

```
-->disp(modelLD2.weights)

    1.4421052  - 1.4421052
  - 0.0717352    0.0717352
  - 0.0435822    0.0435822
  - 0.0258666    0.0258666
  - 0.0091593    0.0091593
  - 0.0131619    0.0131619
  - 0.0841811    0.0841811
  - 0.0499893    0.0499893
  - 0.0291415    0.0291415
  - 0.0153958    0.0153958
```

L'aide n'est pas très disserte concernant les sorties de la méthode. Nous pouvons imaginer que les coefficients sont dans l'ordre des variables. Nous ne savons pas en revanche si la constante est en première ou en dernière position. J'étais curieux d'en savoir plus car ces coefficients sont différents de ceux produits par des logiciels tels que Tanagra ou SAS[7].

3.4 Evaluation du modèle sur un échantillon test

Nous souhaitons maintenant évaluer le modèle sur un échantillon test à part, n'ayant pas participé à sa construction. Plusieurs étapes sont nécessaires : (1) charger l'échantillon test, (2) le coder en respectant le schéma utilisé pour le fichier d'apprentissage, (3) produire la prédiction en appliquant le modèle sur l'échantillon test, (4) créer la matrice de confusion en confrontant les valeurs prédites par le modèle avec celles observées dans l'échantillon test, (5) déduire de la matrice de confusion le taux d'erreur.

Pour les étapes (1) et (2), il s'agit de reproduire les commandes utilisées pour l'échantillon d'apprentissage, mais sur le second fichier.

```
//chargement des données - format chaîne
DTest=csvRead("breast-test.txt","\t",".","string",[],[],[2 1 301 10])

//on peut utiliser un outil interne
[XTest,CTest]=recodage(DTest)
disp(tabul(CTest))
```

La distribution de la variable cible est très similaire à celle observée pour l'échantillon d'apprentissage. C'est, entres autres, une manière simple de vérifier que la partition apprentissage-test a bien été réalisée aléatoirement.

```
->disp(tabul(CTest))

   2.    104.
   1.    196.
```

[7] Voir http://tutoriels-data-mining.blogspot.fr/2012/07/analyse-discriminante-lineaire.html pour la comparaison des sorties des différents logiciels concernant l'analyse discriminante prédictive.

Une fonction que l'on pourra réutiliser pour l'évaluation d'autres modèles a été définie pour les étapes (3), (4) et (5).

```
//fonction pour l'évaluation d'un classifieur
function [MC, ERR_RATE]=test_classifier(classifier, descriptors, target)
    //prédiction
    pred=nan_test_sc(classifier,descriptors)
    //matrice de confusion : cible observée vs. prédiction
    MC=nan_confusionmat(target,pred.classlabel)
    //taux d'erreur
    ERR_RATE = 1.0 - sum(diag(MC))/sum(MC)
endfunction
```

Elle prend en entrée le classifieur à évaluer, la matrice des descripteurs, le vecteur des valeurs de la variable cible. Elle renvoie en sortie la matrice de confusion et le taux d'erreur.

Nous l'appliquons sur l'échantillon test

```
//évaluation
[mc,err_rate] = test_classifier(modelLD2,XTest,CTest)
disp(mc)
disp(err_rate)
```

Nous obtenons :

```
->[mc,err_rate] = test_classifier(modelLD2,XTest,CTest)
err_rate  =

   0.0233333
mc  =

   193.    3.
   4.      100.
```

7 individus sur 300 sont mal classés, soit un taux d'erreur de 2.33%.

3.5 Apprentissage et évaluation du « Naive Bayes Classifier »

Nous avons mis en place tous les éléments pour rendre reproductible notre démarche. Il nous est facile dans cette section, toujours en respectant le même schéma, de construire et évaluer le modèle d'indépendance conditionnelle '**NBC**' (le « bayésien naïf » selon la terminologie de l'apprentissage automatique) proposée par la Toolbox « NaN ».

Voici le code source utilisé[8] :

```
//apprentissage - méthode Naive Bayes Classifier
modelNBC=nan_train_sc(XTrain,CTrain,'NBC')
//évaluation
[mc,err_rate] = test_classifier(modelNBC,XTest,CTest)
```

Scilab fournit les moyennes et variances conditionnelles pour chaque descripteur. Nous obtenons un taux d'erreur de 2.66% sur l'échantillon test.

[8] Il suffit maintenant de deux lignes de codes pour évaluer toute méthode prédictive proposée par la librairie « Nan ». Voilà tout l'intérêt d'organiser en fonctions nos programmes. Il reste cependant que la fonction « recodage » est spécifique au fichier « breast-cancer-wisconsin ». Il faut un peu plus de travail pour la rendre totalement générique.

```
->[mc,err_rate] = test_classifier(modelNBC,XTest,CTest)
err_rate  =

   0.0266667
mc  =

   190.    6.
   2.      102.
```

3.6 Utilisation de la Toolbox « libsvm »

La très célèbre bibliothèque LIBSVM[9] est également disponible sous forme de Toolbox dans Scilab. Essayons voir s'il est possible de reproduire notre schéma d'analyse prédictive en utilisant les outils qu'elle propose. Bien évidemment, il faudra installer et charger la Toolbox dans un premier temps.

Voici les instructions exécutées dans Scilab :

```
//Librairie 'libsvm'
//SVM - Noyau linéaire '-t 0' - Echantillon d'apprentissage
modelLINSVM=libsvm_svmtrain(CTrain,XTrain,'-t 0')
//Prédiction sur l'échantillon test
[predLINSVM]=libsvm_svmpredict(CTest,XTest,modelLINSVM)
//Matrice de confusion : classe observée x classe prédite
mc=libsvm_confmat(CTest,predLINSVM)
```

En sortie de l'apprentissage[**libsvm_svmtrain**], Scilab fournit la matrice des points supports (**Svs**) et les pondérations associées (**sv_coef**).

```
-->modelSVM=libsvm_svmtrain(CTrain,XTrain)
*
optimization finished, #iter = 302
nu = 0.168887
obj = -40.457819, rho = 0.742082
nSV = 180, nBSV = 27
Total nSV = 180
 modelSVM  =

   Parameters: [5x1 constant]
   nr_class: 2
   totalSV: 180
   rho: 0.7420821
   Label: [2x1 constant]
   ProbA: [0x0 constant]
   ProbB: [0x0 constant]
   nSV: [2x1 constant]
   sv_coef: [180x1 constant]
   SVs: [180x9 constant]
```

A partir de la prédiction sur l'échantillon test [**libsvm_svmpredict**], nous construisons la matrice de confusion avec une fonction dédiée [**libsvm_confmat**]. De fait, la toolbox nous fournit tout ce qu'il faut pour mener à bien notre tâche. Voici la matrice de confusion.

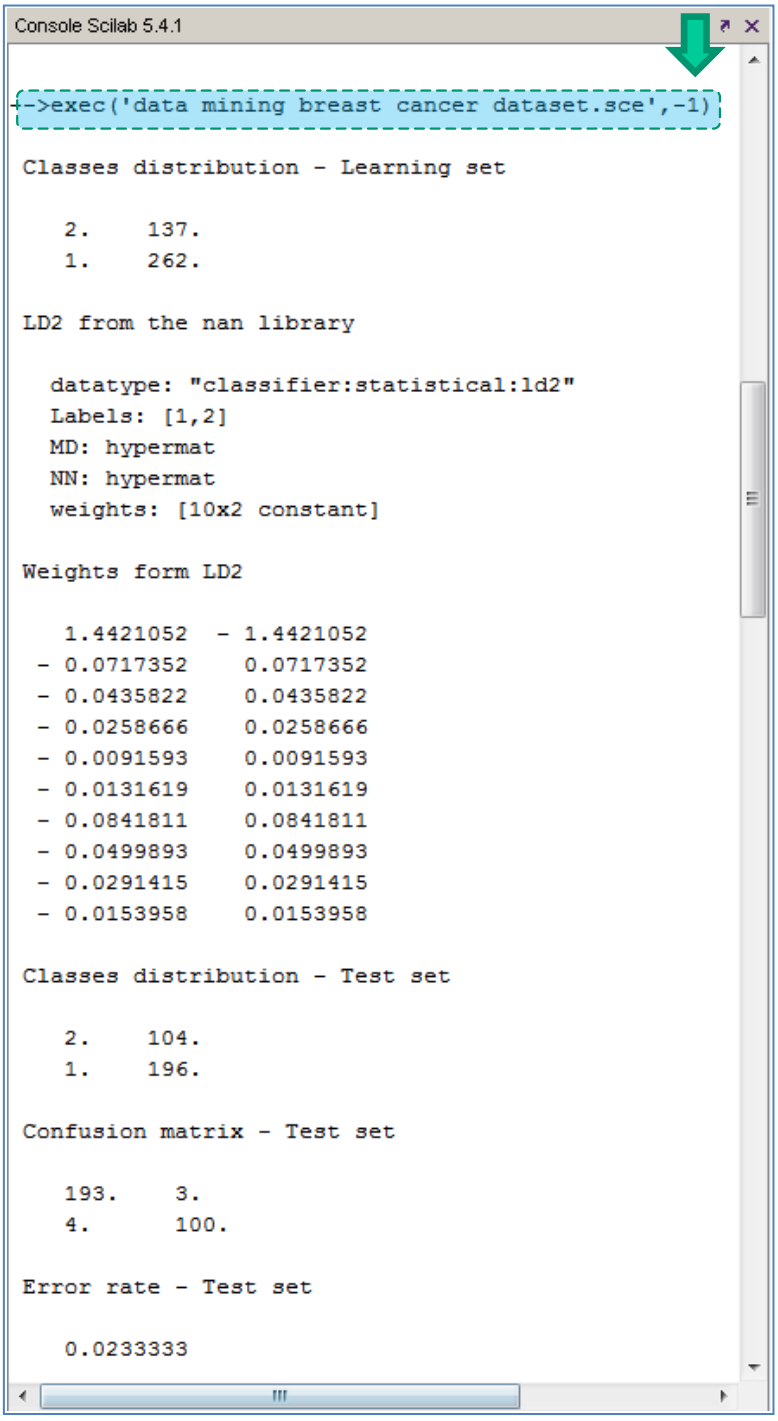
```
-->mc=libsvm_confmat(CTest,predLINSVM)
 mc  =

   193.    3.
   3.      101.
```

[9] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

4 Traitements par lots – Exécution d'un programme

Nous avons réuni le code programme de ce tutoriel dans le fichier « **data mining breast cancer dataset.sce** » en y rajoutant des éléments supplémentaires (affichages intermédiaires, SVM avec noyau RBF, etc.). Nous pouvons lancer directement la totalité des traitements en utilisant la fonction **exec()**[10]. Avec une exécution sans échos c.-à-d. nous visualisons uniquement les informations que nous avons explicitement demandés avec la fonction **disp(),** nous allégeons grandement les sorties. Cela facilite l'analyse des résultats.



[10] Il faut changer le répertoire courant ou spécifier explicitement le chemin du fichier « .sce » dans la commande.

5 Conclusion

Force est de constater que Scilab est tout à fait adapté au data mining. Concrètement, j'ai mis une demi-journée à élaborer le processus décrit dans ce document : en partant du chargement du logiciel sur le site de l'éditeur, son installation sur ma machine, la compréhension des mécanismes de gestion de données (chargement des fichiers, manipulation des matrices et des vecteurs), la compréhension du fonctionnement des Toolbox et la recherche des librairies adéquates sur le net, la mise au point du programme d'analyse prédictive. Si on a l'habitude de R (ou de tout autre langage similaire), l'apprentissage de la syntaxe de Scilab ne pose aucune difficulté.

Aujourd'hui, j'émettrais peut être quelques réserves pour une utilisation courante dans notre domaine :

1. Les toolbox de traitements statistiques des données et de data mining restent peu nombreuses, surtout si on positionne Scilab par rapport à R (à ce jour [04/11/2013], **4986** packages « officiels » sont disponibles sous **R** - http://cran.univ-lyon1.fr/web/packages/index.html).
2. Trouver de la documentation sur ces toolbox est un vrai sacerdoce. J'ai pourtant fait chauffer Google. Mais au final, j'ai surtout appris en scrutant le code des macros que l'on peut consulter en ligne (ex. pour nan_train_sc.sci).
3. Les sorties (actuelles) des librairies de data mining - du moins celles que j'ai testées (« NaN » et « libsvm ») - sont très rustiques, difficilement opérationnelles pour un chargé d'études. On aurait aimé par exemple disposer d'informations sur la pertinence des modèles (les tests de significativité individuelle des coefficients et test de significativité globale du modèle pour l'analyse discriminante linéaire, etc.).
4. La gestion des données peut être également un obstacle. Si on manipule un fichier comportant un mélange de variables quantitatives et qualitatives, la nécessité de procéder nous-mêmes au codage avant de pouvoir lancer les méthodes statistiques peut rebuter les réfractaires à la programmation.

Rien d'insurmontable donc. Le nombre des bibliothèques de méthodes ne peut qu'évoluer positivement. Les toolbox existantes s'amélioreront au fil des années. Quant à la documentation, il ne tient qu'à nous d'y remédier. Pour ma part, je vois déjà poindre à l'horizon plusieurs tutoriels très intéressants.