1 Objectif

(30/04/2012) Description du logiciel Revolution R Community. Comparaison des performances.
(06/07/2012) Mise à jour – Lien vers un benchmark très similaire, dans l'esprit, à celui-ci.

Le logiciel R est en train de bouleverser le panorama des logiciels de statistique et de data mining. Rétrospectivement, en 2003 lorsque je me suis lancé dans le projet Tanagra, si R était arrivé à son niveau de maturité actuel, plutôt qu'un logiciel à part entière, j'aurais vraisemblablement développé un package pour R… Et j'aurais eu certainement eu tort. Ne dit-on pas que « l'ennui naquit un jour de l'uniformité » ?[1] R est un outil formidable, personne n'en doute, moi le premier. Tanagra est tout simplement une alternative qui se positionne sur un créneau différent. Il constitue essentiellement un outil pédagogique à destination des étudiants. Il n'a absolument aucune vocation professionnelle, et c'est très bien ainsi. Tanagra se distingue avant tout par sa simplicité d'utilisation, sa légèreté (son fichier setup fait 2.7 Mo quand d'autres distributions font près de 10 Go, on se demande ce qu'ils mettent dedans d'ailleurs) et les très nombreux didacticiels qui l'accompagnent.

Mais revenons à R. Le système des packages est un de ses principaux atouts. Il peut être enrichi à l'infini[2]. Toute méthode statistique est potentiellement disponible dans R. Mais si les packages sont nombreux, rares sont les projets qui cherchent à améliorer le moteur même de R, l'application principale. J'ai découvert récemment les travaux de la société Revolution Analytics. Elle commercialise Revolution R Enterprise qui : améliore très significativement les performances de calculs de R, est capable de traiter les grandes bases de données, et propose un EDI (environnement de développement) évolué avec un débogueur intégré. Cette version étant payante, je n'ai pas pu la tester. En revanche, la société distribue également une version communautaire qui, elle, est en libre accès. Bien évidemment, je me suis précipité dessus pour voir ce qu'il en était.

Revolution R Community est une variante améliorée de R. Elle n'intègre pas les fonctionnalités additionnelles de la version Enterprise. L'effort porte essentiellement sur les performances. Deux aspects sont mis en avant : elle intègre la bibliothèque de calcul mathématique Intel ; elle est capable de titrer profit des processeurs multi-cœurs. Des comparatifs sont accessibles sur le site web de l'éditeur[3]. Apparemment, le gain est spectaculaire pour les techniques de data mining s'appuyant sur des calculs matriciels. L'analyse en composantes principales serait par exemple douze fois plus rapide par rapport à « R base ».

Dans ce tutoriel, nous étendons le « benchmark » à d'autres méthodes de data mining. Nous étudions les performances de « Revolution R Community 5.0 – 64 bits » : pour la régression logistique (glm) ; l'analyse discriminante (lda du package MASS) ; l'induction des arbres de décision (rpart du package du même nom) ; de l'analyse en composantes principales (ACP) avec deux techniques : celle reposant sur le calcul des valeurs propres (princomp) et celle s'appuyant la décomposition en valeurs singulières (prcomp). Nous utilisons une variante binaire de la base « wave » (Breiman et al., 1984) pour mesurer les temps de calculs.

[1] http://fr.wikipedia.org/wiki/Antoine_Houdar_de_La_Motte

[2] http://cran.univ-lyon1.fr/web/packages/available_packages_by_name.html

[3] http://www.revolutionanalytics.com/why-revolution-r/benchmarks.php

2 Données

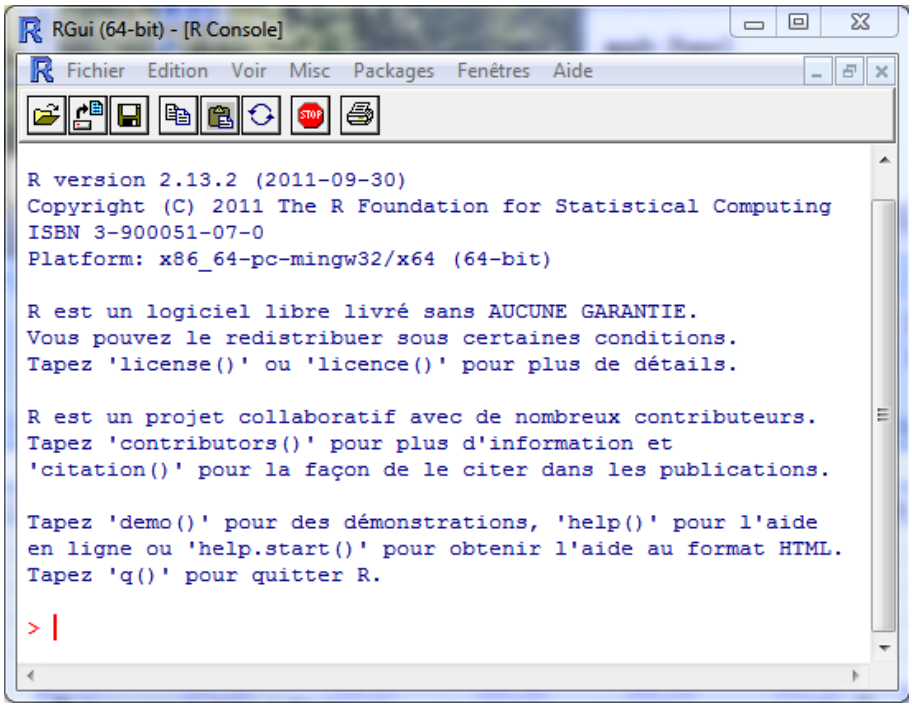
Nous avons mis à contribution la base « **wavebin.txt** » dans deux de nos anciens tutoriels où nous étudions le comportement de différentes implémentations de la régression logistique sur une (relativement) grande base de données (voir « La proc logistique de SAS » et « Régression logistique sur les grandes bases »). Dans cette étude, nous générons n = 500.000 observations et p = 121 variables explicatives. Le code en langage R du programme principal utilisé est le suivant :

```
#generate and save a dataset
set.seed(1)
dataset.size <- 500000 #number of instances
nb.rnd <- 50 #number of random variables
nb.cor <- 50 #number of correlated variables
noise.level <- 1 #noise for correlated variables
data.wave <- generate.binary(dataset.size,nb.rnd,nb.cor,noise.level)
summary(data.wave)
#writting
write.table(data.wave,file="wavebin.txt",quote=F,sep="\t",row.names=F)
```

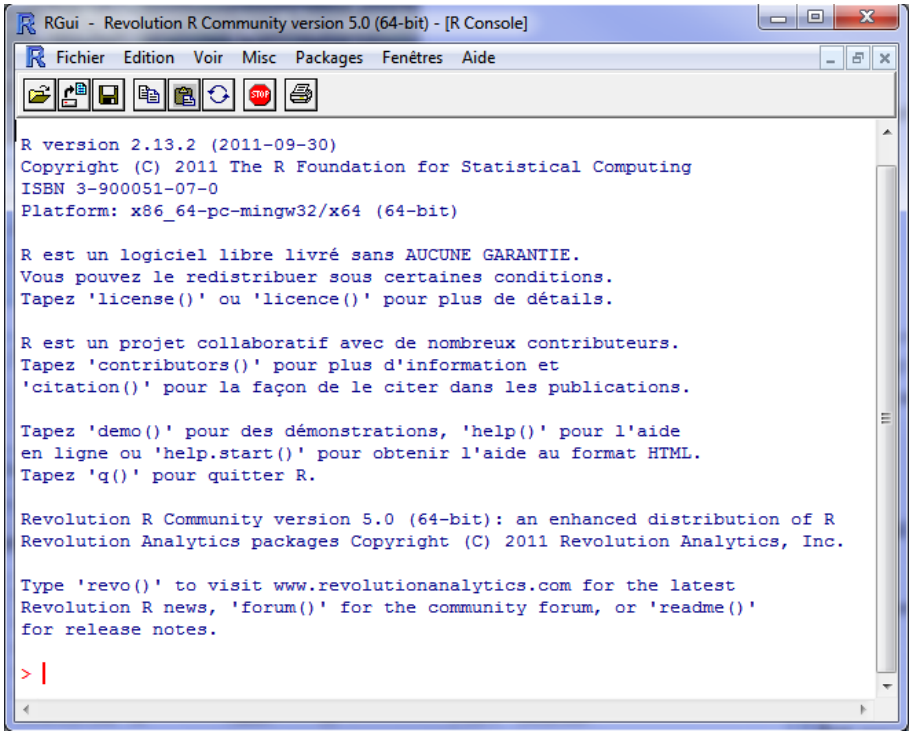
3 Performances comparées de Revolution R Community

3.1 Revolution R Community 5.0

Un résultat n'est intéressant que si l'on a des éléments de comparaison. Dans notre cas, il est tout trouvé, nous prenons en référence **R 2.13.2 64 bits**. Celle qui correspond précisément à la version 5.0 de Revolution R Community. En effet, au démarrage de la version de base de R, nous obtenons l'affichage suivant :



Et pour la Revolution R, nous avons :



Les performances de Revolution R Community sont optimisées dixit [la page web de l'éditeur](). Nous reproduisons ici le tableau confrontant les deux versions :

	Base R	Revolution R Community
Target Use	Open Source	Product Evaluation & Simple Prototyping
Software		
100% Compatible with R language	X	X
Certified for Stability		
Command-Line Programming	X	X
Getting Started Guide		X
Performance & Scalability		
Analyze larger data sets with 64-bit RAM	X	X
Optimized for Multi-processor workstations		**X**
Multi-threaded Math libraries		**X**
Parallel Programming (Single Workstation)		**X**
Out-of-the-Box Cluster-Ready		
"Big Data" Analysis with RevoScaleR		
Terabyte-Class File Structure		
Specialized "Big Data" Algorithms		
Integrated Web Services		
Scalable Web Services Platform		
User Interface		
IDE with Visual Debugger		
Technical Support		
Discussion Forums	X	X
Online Support	**Mailing List**	**Forum**
Email Support		
Phone Support		
Support for Base & Recommended R Packages	X	X
Authorized Training & Consulting		
Platforms		
Single User	X	X
Multi-User Server	X	
32-bit Windows	X	X
64-bit Windows	X	X
64-bit Red Hat Enterprise Linux	X	X

L'effort porte sur la rapidité d'exécution. C'est ce que nous évaluerons dans ce tutoriel.

3.2 Protocole de comparaison

Pour situer Revolution R, nous utilisons plusieurs algorithmes de data mining. Nous essaierons de comprendre les gains obtenus selon leurs spécificités. Le programme de benchmark est le suivant :

```
rm(list=ls())

#data importation
system.time(wave <- read.table(file="wavebin.txt",sep="\t",dec=".",header=T))

#select the 21 first columns for the pca analyses
wave.subset <- subset(wave, select = 1:21)

#package for lda
library(MASS)

#package for rpart
library(rpart)

#the benchmark function
benchmark <- function(){

  tps <- numeric(8)

  #logistic regression
  a <- system.time(model.glm <- glm(y ~ ., data = wave, family=binomial()))
  tps[1] <- a["elapsed"]
  print(model.glm)
  a <- system.time(predict(model.glm, newdata = wave, type = "response"))
  tps[2] <- a["elapsed"]

  #linear discriminant analysis
  a <- system.time(model.lda <- lda(y ~ ., data = wave))
  tps[3] <- a["elapsed"]
  print(model.lda)
  a <- system.time(predict(model.lda, newdata = wave))
  tps[4] <- a["elapsed"]

  #decision tree induction
  a <- system.time(model.rpart <- rpart(y ~ ., data = wave))
  tps[5] <- a["elapsed"]
  print(model.rpart)
  a <- system.time(predict(model.rpart, newdata = wave, type = "class"))
  tps[6] <- a["elapsed"]

  #principal component analysis (using eigen)
  a <- system.time(pca.princomp <- princomp(wave.subset, cor = T))
  tps[7] <- a["elapsed"]
```

```
  print(pca.princomp)

  #principal component analysis (using singular value decomposition)
  a <- system.time(pca.prcomp <- prcomp(wave.subset, center = T, scale = T))
  tps[8] <- a["elapsed"]
  print(pca.prcomp)

  #return the vector
  return(tps)
}

#repeat 10 times the computation time measurement
m <- replicate(10,benchmark())
print(m)
#mean and median
print(apply(m,1,mean))
print(apply(m,1,median))
```

Chaque technique statistique a été lancée 10 fois pour obtenir une mesure fiable du temps d'exécution. Moyennes et médianes ont été très proches pour chaque mesure. Il n'y a donc pas eu des valeurs aberrantes durant le processus d'évaluation.

Nous obtenons les durées d'exécution suivantes :

n = 500.000, p = 121

Computation time (sec.)	R 2.13.2	Rev-R CE 5.0	Speedup
Data importation	75.0	75.0	
glm	288.6	279.4	0.03
predict.glm	6.6	6.5	0.02
lda	256.3	176.0	0.46
predict.lda	16.0	15.9	0.01
rpart	1395.4	1383.1	0.01
predict.rpart	6.9	6.8	0.01
princomp (21 variables)	4.9	3.8	0.29
prcomp (21 variables)	7.5	5.2	0.42

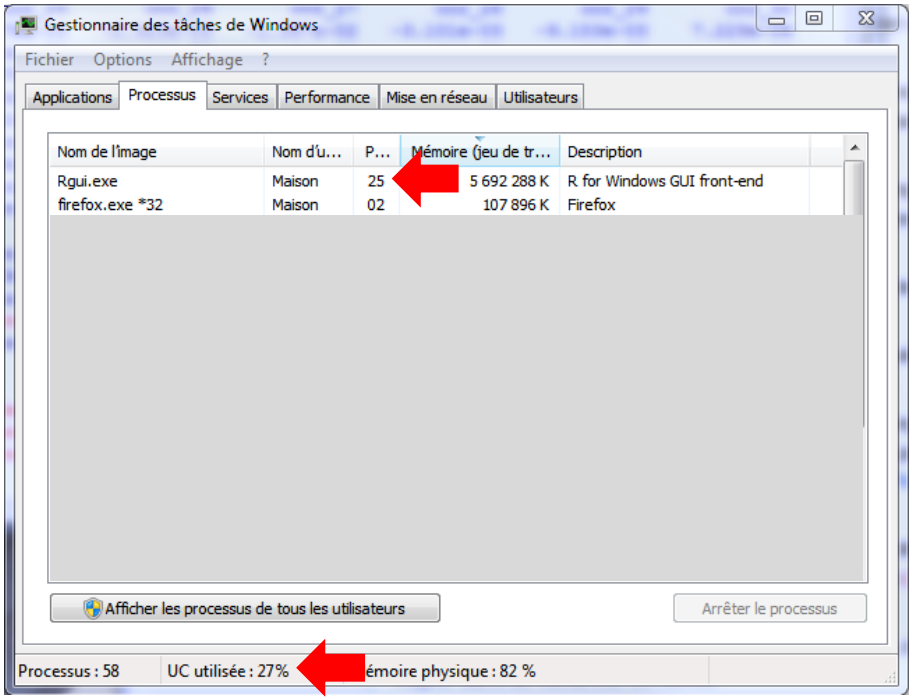
Speedup = Lower Time / Faster Time - 1

Analyse discriminante et analyse en composantes principales. Les gains sont spectaculaires, mais ils sont moins extraordinaires qu'indiqués sur le site web de Revolution Analytics. Visiblement, **les calculs matriciels ont été optimisés**, plus la décomposition en valeurs singulières utilisées par LDA (analyse discriminante linéaire) et PRCOMP (analyse en composantes principales s'appuyant sur la décomposition en valeurs singulières) que le calcul des valeurs et vecteurs propres (PRINCOMP).

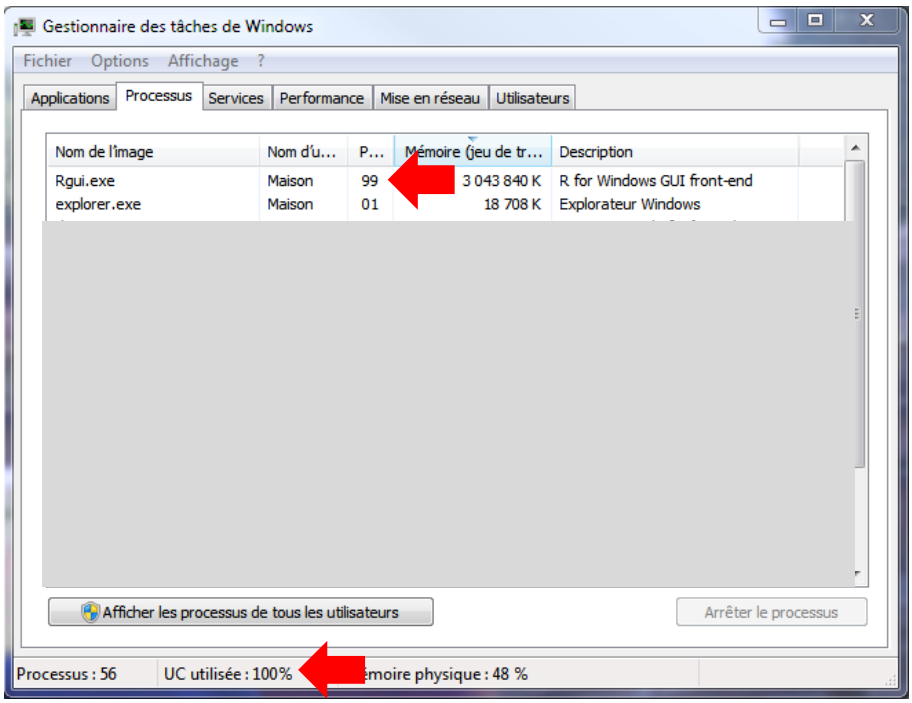
Régression logistique. Pour la régression logistique (GLM), le gain est réel mais reste faible. Peut être parce que, compte tenu des caractéristiques de notre base (beaucoup d'observations pour relativement peu de variables), le remplissage de la matrice d'information[4] prend plus de temps que son inversion (qu'on peut considérer sous l'angle d'une résolution d'équation) durant le processus d'optimisation de la log-vraisemblance.

[4] **glm( )** utilise la technique « Fisher scoring » - http://data.princeton.edu/wws509/notes/a1s1.html

Autre point très intéressant, ma machine étant un Quad Core (Quad Core Q9400 à 2.66 Ghz fonctionnant sous Windows 7 - 64 bits), nous avons pu évaluer les capacités multi-thread des outils. Si « R base » n'exploite systématiquement qu'un seul cœur,



Revoluton R Community, selon la nature de l'opération effectuée en interne, tire profit ***par intermittence*** de la totalité des cœurs disponibles.



Arbres de décision. Enfin, l'amélioration est négligeable pour l'induction des arbres de décision. Très vraisemblablement parce que nous ne bénéficions pas des atouts matriciels de Revolution R. Et le multi-threading n'est pas exploité parce qu'elle n'est pas prévue nativement dans la librairie rpart().

On peut penser que ce comportement peut être généralisé à toutes les techniques d'extraction de règles implémentées sous R.

4 Conclusions et perspectives

Revolution R est très particulier dans le sens où il essaie d'améliorer R même. Il se démarque en cela des autres acteurs qui cherchent avant tout à compléter le logiciel via le système des packages. Nous avons évalué le comportement de la version « Community » dans ce tutoriel. Dès lors que des opérations matricielles complexes sont en jeu, les gains en rapidité d'exécution sont réels. Ils sont moins flagrants en revanche pour les autres techniques de data mining.

Pour asseoir les résultats entrevus dans ce didacticiel, il faudrait approfondir l'analyse en faisant varier les caractéristiques des bases étudiées (nombre de lignes, nombre de colonnes, types de variables) et étendre l'analyse à un plus grand nombre de techniques d'apprentissage, en les situant toujours selon le type de calcul effectué en interne.

Une idée très simple déjà serait de reproduire à l'identique le benchmark décrit sur le site de Revolution Analytics. Pour rappel, ils indiquaient[5] :

	Base R 2.13.2 32	Revolution R (1-core)	Revolution R (4-core)	Speedup (4 core)
Matrix Multiply	174.6 sec	31.0 sec	10.4 sec	15.8x
Cholesky Factorization	25.7 sec	5.0 sec	1.4 sec	17.6x
Singular Value Decomposition	67.6 sec	18.3 sec	7.8 sec	7.6x
Principal Components Analysis	266.2 sec	53.7 sec	20.1 sec	12.2x
Linear Discriminant Analysis	224.4 sec	84.4 sec	61.4 sec	2.7x

Speedup = Slower time / Faster Time - 1

Figure 1 - Source : http://www.revolutionanalytics.com/why-revolution-r/benchmarks.php

Sur notre machine et avec les versions 64 bits de R, nous obtenons :

Calc. time (sec)	Base R 2.13.2 - 64	Revolution R Community 5.0 - 64	Speedup
Matrix Multiply	206.1	7.5	26.5
Cholesky Factorization	25.7	1.6	15.4
Singular Value Decomposition	68.3	11.6	4.9
Principal Components Analysis	283.8	26.7	9.6
Lnear Discriminant Analysis	225.6	56.0	3.0

Par rapport à ceux obtenus avec la base « wave », les résultats sont autrement plus favorables pour Revolution R Community ! Les caractéristiques des données utilisées jouent un rôle fondamental dans les gains de temps de calcul, c'est indéniable. Manifestement, de par la nature des opérations matricielles utilisées, Revolution R se comporte d'autant mieux que le nombre de variables dans la base est élevé comparativement au nombre d'observations.

[5] http://www.revolutionanalytics.com/why-revolution-r/benchmarks.php

Pour confirmer cette idée, nous avons réitéré notre évaluation sur la base wave avec n = 5000 observations et p = 221 variables explicatives. Nous obtenons le tableau suivant :

n = 5000, p = 221

Computation time (sec.)	R 2.13.2	Rev-R CE 5.0	Speedup
Data importation	1.6	1.6	
glm	6.07	4.27	0.42
predict.glm	0.13	0.13	-0.01
lda	4.04	2.59	0.56
predict.lda	0.19	0.19	0.03
rpart	5.45	5.44	0.00
predict.rpart	0.15	0.15	-0.03
princomp (21 variables)	0.03	0.03	0.00
prcomp (21 variables)	0.03	0.03	0.36

Le résultat évolue fortement surtout pour la régression logistique. Principalement parce que le temps de remplissage des matrices prend moins d'importance par rapport à leur traitement (inversion) dans le processus d'apprentissage.

Enfin, **concernant la version Enterprise**, sa capacité à traiter les très grandes bases de données semble être son crédo. C'est d'ailleurs la lecture d'un article sur le web qui a attiré mon attention sur les produits de Revolution Analytics (http://blog.revolutionanalytics.com/2011/07/fast-logistic-regression-big-data.html et http://www.youtube.com/watch?v=KZHioV-DOD8). Une régression logistique sur 1.2 milliards d'observations (!), en 75 secondes (!!), ça laisse rêveur…

5 Note de mise à jour : 06/07/2012

En cherchant un peu sur Internet, je me suis rendu compte que je ne suis pas le seul à m'être posé ces questions. Dans les grandes lignes, les conclusions sont les mêmes. En dehors des techniques de data mining gourmandes en opérations matricielles, Revolution R Community apporte peu.

http://heuristically.wordpress.com/2011/06/27/benchmarking-r-revolution-r-and-hyperthreading-for-data-mining/