Data mining avec R dans un monde libre

Diego.Kuonen@epfl.ch et Reinhard.Furrer@epfl.ch,
EPFL, Département de Mathématiques

R est un langage et un environnement pour les calculs statistiques et leurs représentations graphiques. **R** est similaire au système **S** qui a été récompensé par le *Software System Award* de l'ACM *(Association for Computing Machinery)* et qui est la plate-forme du logiciel commercial S-Plus. Rappelons quelques technologies reconnues par la récompense ACM à savoir UNIX, TeX, PostScript, TCP/IP, World-Wide Web, Tcl/Tk, et Apache. La citation de l'ACM contenait la phrase suivante: *«… system, which has forever altered how people analyze, visualize, and manipulate data.»*

Le but de l'article précédemment paru dans le FI 2/01 [1] était de fournir un point de départ pour les novices intéressés par **R** [2]. Fin juin 2001, la version 1.3.0 de **R** est sortie, contenant un portage stable pour MacOS/X [3]. **R** est un exemple parmi tant d'autres du succès incontestable des logiciels libres – les éléments essentiels d'un monde libre.

Le présent article se propose d'illustrer les interfaces existantes entre **R** et des bases de données relationnelles, ces interfaces étant un premier pas des logiciels statistiques modernes, comme **R**, vers la reconquête par les statisticiens du domaine du *Knowledge Discovery in Databases* (KDD). Ce propos est illustré à l'aide de pingouins, enfermés derrière des fenêtres d'un zoo et qui aimeraient partir vers un monde libre.

Qu'est-ce que le data mining?

«We are drowning in information, but starving for knowledge.» (John Naisbett dans son livre **Megatrends**)

De manière générale, on peut définir le *Data Mining* (ou *Exploitation des Gisements de Données*) comme l'extraction, à partir de gros volumes de données, d'informations ou de connaissances originales, auparavant inconnues, potentiellement utiles.

Selon *SAS Institute Inc.*, il s'agit du processus de sélection, exploration, modification et modélisation de grandes bases de données afin de découvrir des relations entre les données jusqu'alors inconnues.

Le *data mining* correspond donc à l'ensemble des techniques et des méthodes qui, à partir de gros volumes de données, permettent d'obtenir des connaissances exploitables. Son utilité est grande dès lors qu'une entreprise possède un grand nombre d'informations stockées sous forme de bases de données.

Il existe une distinction précise entre le concept de KDD (*Knowledge Discovery in Databases* ou *Découverte de Connaissances dans les Bases de Données*) et celui de *data mining*. En effet, ce dernier n'est que l'une des étapes du processus de découverte de connaissances correspondant à l'extraction des connaissances à partir des données. Pour pouvoir réaliser des études de *data mining* il faut d'abord disposer d'un *Data Warehouse* (*Entrepôt de Données*).

Les applications du *data mining* sont multiples: la grande distribution, la vente par correspondance, les opérateurs de télécommunications, les banques et les assurances, l'étude des génomes dans la bioinformatique, par exemple, en trouvant des gènes dans des séquences d'ADN, etc. Le domaine majeur où le *data mining* a prouvé son efficacité est la gestion de la relation client, CRM (*Customer Relationship Management*). En effet, dans ce cas le *data mining* permet d'accroître les ventes par une meilleure connaissance de la clientèle. Une bonne référence sur ce dernier point est le livre de Berry et Linoff [4]. De nombreuses adresses Web concernant le *data mining* sont disponibles sur le site Statoo.com [5] et sur celui du premier auteur [6].

Le *data mining* et ses outils, bien qu'utilisant la démarche et des techniques statistiques, sont appelés à être utilisés par des non-statisticiens. Regardons pourquoi.

Le domaine des statistiques, spécialement des statistiques dites mathématiques, a commencé à croître il y a environ 100 ans. A l'époque les jeux de données étaient de petite taille et les calculs s'effectuaient manuellement. L'ère des statistiques ma-

thématiques s'est terminée le jour où, grâce aux ordinateurs, les statistiques dites computationnelles sont apparues. Cependant, le stockage électronique des données restait toujours coûteux et les volumes des données toujours limités. Et, comme les statistiques font traditionnellement partie du domaine des mathématiques et non pas de l'informatique, la statistique trouvait son intérêt premier dans la théorie mathématique sous-jacente, et non pas dans les aspects du calcul et du stockage de données à traiter. Ceci explique pourquoi les logiciels statistiques étaient habituellement bons en lecture, respectivement en importation de données depuis un fichier, mais ont malheureusement ignoré pendant trop longtemps le fait que des données reposaient dans des bases de données et l'intérêt qu'il y aurait de s'interfacer avec ces bases de données. Depuis le début des années 1990 le nombre de très grandes bases de données, souvent distribuées, n'a pas arrêté de croître. Les agences gouvernementales, les grands détaillants ou les commerçants en ligne font face maintenant aux bases de données énormes, difficiles à analyser au delà de simples indicateurs sommaires. Ainsi, les informaticiens ont tiré profit de ce manque de connaissance de base de données des statisticiens et *ont inventé* une discipline appelée KDD. La question reste posée: cette nouvelle discipline est-elle en concurrence avec les statistiques?

Puisque KDD a pour but de recpréveler et de ueillir et données, d'autoriser la conception expérimentale, et l'analyse des données en utilisant les modèles de découverte, de visualisation, de faire des prévisions et des classifications, elle se situe clairement dans le domaine des statistiques. En résumé, on peut dire comme Daryl Pregibon [7]: *KDD = Statistics at Scale and Speed*. Mais, il reste encore à la statistique de relever les défis du *data mining*.

Le processus usuel du *data mining* est représenté dans la Figure 1 et peut se résumer ainsi:

1. Définir une requête pour extraire toutes les données souhaitées de la base de données;
2. Exporter les données sélectionnées dans un fichier;
3. Si nécessaire, les convertir dans un format qui facilite l'importation dans un logiciel statistique tel que **R**;
4. Importer les données dans ce logiciel;
5. Analyser les données dans ce logiciel.

Il faut recommencer le processus tant que l'on n'a pas extrait toutes les données appropriées ou bien si les données fondamentales dans la base de données changent, ce qui nécessite la mise à jour des résultats.

La possibilité de manipuler des données se trouvant dans des bases de données est très importante, mais elle manque dans beaucoup de logiciels statistiques. Travailler directement sur des bases de données avec des outils interactifs semble être beaucoup plus difficile que de travailler sur des données structurées spécialement pour des applications statistiques. Mais c'est la seule possibilité qui s'offre aux statisticiens pour regagner le terrain de la KDD occupé actuellement par les informaticiens.

A ce titre le futur de **R** est prometteur en visant à combler cette lacune. Le premier pas a été fait en mettant à disposition des interfaces avec des bases de données relationnelles.

Les bases de données et SQL

Les bases de données sont devenues une partie intégrante de notre vie. Sans elles certaines tâches deviendraient très pénibles ou même impossibles. Les banques, les universités et les bibliothèques sont trois exemples d'organismes qui dépendent fortement de bases de données. Sur Internet, les moteurs de recherche, les achats en ligne, et même les serveurs de noms (DNS)

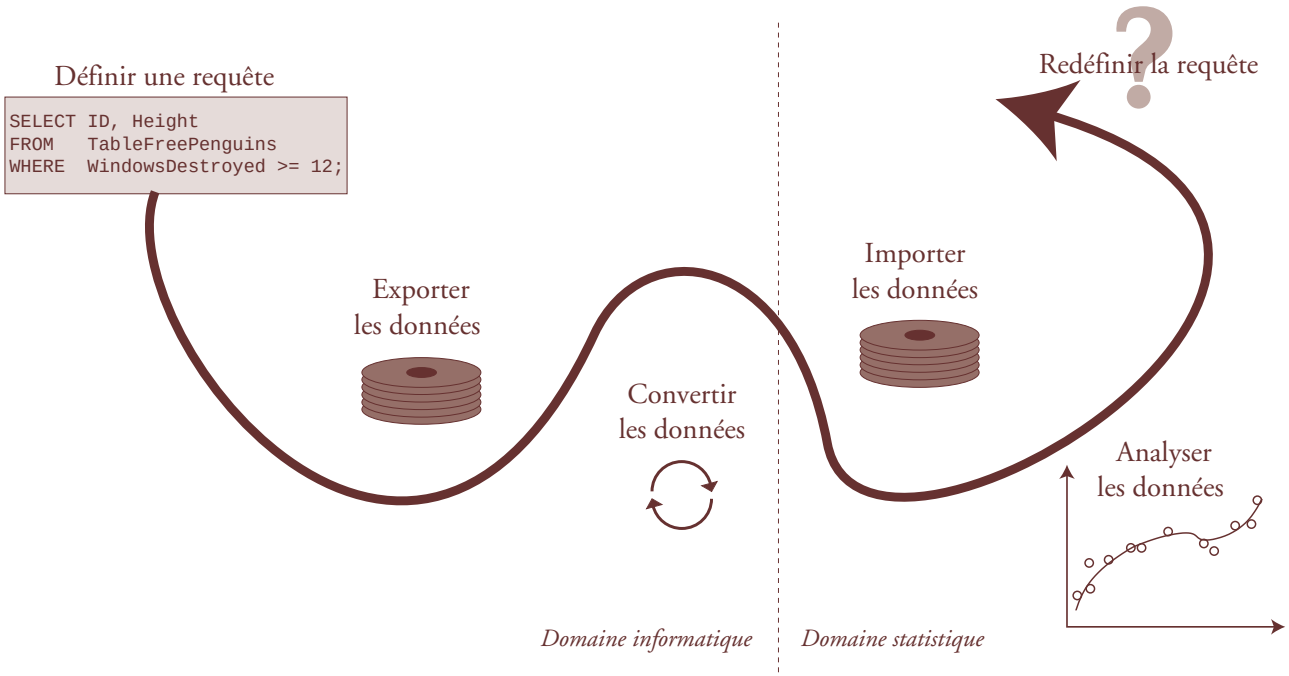

Figure 1: Le processus usuel du data mining.

seraient impossibles à réaliser sans l'utilisation d'une base de données.

On nomme souvent serveur de base de données l'ordinateur sur lequel une ou plusieurs bases de données sont rendues accessibles. La connectivité à ces bases de données est une partie de plus en plus importante du calcul statistique et les interfaces aux bases de données volumineuses deviennent importantes pour les applications d'extraction de données.

Il y a plusieurs types de DBMS ou SGBD (*Data Base Management Systems* ou *Système de Gestion de Base de Données*):

- les fichiers textes et csv;
- les tableurs (programmes et formats);
- les bases de données de fichier *plats* comme DBase;
- les bases de données hiérarchiques (telles que HDF5 [8]);
- les bases de données relationnelles *maigres* comme Access, MySQL [9, 10, 11], MiniSQL (également connu sous le nom de mSQL, [12]); et
- les bases de données relationnelles *lourdes* comme Oracle, DB/2, Microsoft SQL Server, Informix, Sybase, PostgreSQL [13].

Parfois l'utilisateur a le choix du DBMS. Plus souvent il doit utiliser une base de données existante ce qui ne lui laisse pas le choix. La plupart des DBMS relationnels sont des systèmes du type client/serveur, et beaucoup permettent la transmission par TCP/IP. La plupart des DBMS viennent avec un moniteur, un client basé texte, certains ont des clients GUI, et presque tous ont une API C ou C++. Notons que seul le DBMS peut accéder à la base de données par l'intermédiaire de commandes, qui sont normalement introduites avec un dialecte de SQL (*Structured Query Language* ou *Langage de Requête Structuré*).

SQL est le principal langage informatique de création et de manipulation de bases de données relationnelles, permettant tant la définition que la manipulation et le contrôle d'une base de données relationnelles. SQL est défini par des normes et est implanté dans les systèmes de gestion de base de données de la plupart des fournisseurs, parfois avec des modifications mineures dans la syntaxe. SQL possède deux caractéristiques fondamentales: il est ensembliste et non-procédural. La première caractéristique veut dire qu'il porte toujours sur des ensembles (des tables), et qu'il produit toujours des ensembles (des tables). La seconde veut dire que le langage ne décrit pas un algorithme à exécuter afin d'obtenir le résultat désiré, comme les langages de programmation traditionnels (FORTRAN, C), mais qu'il décrit uniquement le résultat désiré, sans se préoccuper de la manière dont le système de gestion de base de données va exécuter la commande. SQL est un outil très puissant et facile à utiliser, mais qui requiert qu'on ait constamment à l'esprit le fait qu'on travaille sur des ensembles. En particulier, il faut se souvenir qu'une jointure fait toujours le produit de deux ou de plusieurs tables et qu'il faut donc, la plupart du temps, restreindre la jointure aux rangées désirées en indiquant une condition à remplir.

Plus généralement, SQL contient trois types de langages:

- un langage de requêtes;
- un langage de manipulation de données;
- un langage de définition de données.

Voici un petit exemple pour illustrer les principes fondamentaux de SQL. Dans une base de données relationnelles, les données sont stockées dans des tables. Le tableau suivant, TableFreePenguins, contient des données sur des pingouins vivant dans un zoo derrière des fenêtres. Ces pingouins aimeraient s'échapper vers un monde libre. Pour ce faire, ils doivent casser les fenêtres imposées par le fabriquant du zoo. Le tableau TableFreePenguins met en relation le numéro d'identification des pingouins (`ID`), leur race (*Species*), leur taille en cm (*Height*), leur poids en kg (*Weight*) et le nombre de fenêtres cassées (*WindowsDestroyed*):

TableFreePenguins

ID	Species	Height	Weight	Windows Destroyed
21	Aptenodytes patagonicus	94	14	98
01	Aptenodytes forsteri	112	30	2000
13	Pygoscelis adeliae	50	4	4
24	Pygoscelis papu	65	6	95
55	Pygoscelis antarctica	47	4	31
76	Eudyptes chrysocome	44	2	1

Maintenant, supposons que nous voulions obtenir la taille et le poids de chaque pingouin. On utilisera l'instruction SELECT, comme ceci:

```
SELECT Height, Weight
FROM TableFreePenguins;
```

Voici le résultat de l'interrogation de notre base de données:

Height	Weight
94	14
112	30
50	4
65	6
47	4
44	2

Pour obtenir toutes les colonnes d'une table sans avoir à taper tous les noms de colonne, on peut utiliser:

```
SELECT * FROM TableFreePenguins;
```

La façon de saisir les instructions SQL et d'accéder à la base de données varie en général d'un DBMS à l'autre; consultez un gourou prêt à vous aider si vous devez utiliser une de ces bases de données.

Il y a sept opérateurs relationnels en SQL: «=», «<>» ou «!=», «<», «>», «<=», «>=». La clause `WHERE` est utilisée pour spécifier que l'on affiche seulement certaines lignes de la table, selon un critère défini par cette clause `WHERE`. De façon plus claire, prenons l'exemple suivant où l'on désire savoir l'`ID` et la taille des pingouins ayant cassé au moins 98 fenêtres:

```
SELECT ID, Height
FROM TableFreePenguins
WHERE WindowsDestroyed >= 98;
```

On aura l'affichage:

```
ID          Height
21          94
01          112
```

La description de `WHERE`, `WindowsDestroyed >= 98`, est appelée une condition. On voit que ce sont les pingouins les plus grands qui ont cassé le plus grand nombre de fenêtres. Le monde libre n'est donc pas uniquement désiré par les petits.

Pour ajouter une difficulté, nous voulons aussi que cette liste soit classée par ordre alphabétique selon la race du pingouin. Pour cela, nous utiliserons la clause `ORDER BY`:

```
SELECT ID, Height
FROM TableFreePenguins
WHERE WindowsDestroyed >= 98
ORDER BY Species;
```

Ce qui donnera

```
ID          Height
01          112
21          94
```

Le but de cet article n'est pas de vous donner une introduction à SQL, que vous trouverez sur de nombreux sites Web comme [14], dans le manuel et les HOWTOs de votre système de gestion de bases de données ou dans des livres comme [11].

R et les bases de données

Actuellement, il y a sur le «Comprehensive **R** Archive Network» (CRAN, [3]) trois modules d'interface de **R** avec des DBMS:

- RPgSQL pour PostgreSQL;
- RMySQL pour MySQL;
- RmSQL pour MiniSQL;

utilisant des interfaces en C. Celles-ci et RODBC sont décrits dans le manuel ***R** Data Import/Export* [15]. L'option la plus portable est de loin RODBC et, à moins que vous puissiez choisir votre système de gestion de base de données, vous n'avez probablement aucun autre choix. La *Open Database Connectivity* (ODBC) fournit une interface client commune à presque tous les DBMS populaires. Pour utiliser ODBC vous avez besoin d'un gestionnaire de pilotes pour votre OS et un pilote pour votre système de gestion de base de données. Heureusement les pilotes nécessaires sont largement disponibles, mais pas toujours conformément aux versions récentes de l'ODBC. Les outils de base de RODBC doivent transférer des tableaux de données vers et depuis un système de gestion de base de données.

Prélever des données d'un DBMS est la tâche commune à tous. La table d'un DBMS relationnel est un concept semblable à un tableau de données (*data frame*) dans **R**. Ainsi, une idée naturelle est de créer une table contenant les données souhaitées, et de la stocker dans un data frame. RmSQL fait ceci ligne par ligne. Pour charger un module dans **R**, il faut utiliser la commande:

```
> library(«Nom du module»)
```

Par exemple, pour charger l'interface RPgSQL taper,

```
> library(RPgSQL)
```

Ainsi, on a pour RPgSQL,

```
> db.connect(dbname=»freeworld»)
> db.read.table(«TableFreePenguins»)
> db.disconnect()
```

pour RODBC,

```
> channel <- odbcConnect(«freeworld»,uid=»linus»)
> sqlFetch(channel, «TableFreePenguins», rownames =
                                                TRUE)
> odbcClose(channel)
```

et pour RMySQL,

```
> con <- dbConnect(MySQL(), dbname = «freeworld»)
> getTable(con, «TableFreePenguins»)
> close(con)
```

Malheureusement, divers problèmes peuvent surgir:

1. Peu de DBMS font par défaut la distinction entre minuscules et majuscules. Ceci dépend aussi de l'OS. MySQL, par exemple est *case-sensitive* sous Linux, mais pas sous Windows.
2. Les DBMS ont en principe les tables non triées, il faut disposer d'un champ d'identification (clé primaire) pour pouvoir le faire.
3. Faire correspondre les types de données dans la base aux types de données de **R** peut être problématique.
4. Les valeurs manquantes sont généralement représentées dans SQL par la valeur NULL. Il faut bien s'assurer de la façon dont elles sont transférées dans **R**.
5. La lecture de la table entière d'un seul coup pourrait être une trop lourde charge pour **R**. Il faudrait donc lire bloc par bloc, et pour ceci il faut probablement utiliser SQL, ce qui peut s'avérer compliqué.

Pour quelques applications, ajouter les données provenant de **R** à une base de données est une tâche importante. Les fonctions nécessaires sont disponibles dans la plupart des interfaces de **R.**

Pour RPgSQL:

```
> data(TableFreePenguins)
> freepenguins <- TableFreePenguins
> db.connect(dbname=»freeworld»)
> db.write.table(freepenguins,write.row.names =
                                                TRUE)
> db.disconnect()
```

pour RODBC,

```
> channel <- odbcConnect(«freeworld»,uid=»linus»)
> sqlSave(channel, freepenguins, rownames = TRUE)
> odbcClose(channel)
```

et pour RMySQL,

```
> con <- dbConnect(MySQL(), dbname = «freeworld»)
> assignTable(con, «freepenguins»,
                  TableFreePenguins, overwrite=TRUE)
> close(con)
```

Nous devons nous assurer que les noms des lignes (`rownames`) sont sauvés. Une alternative est d'écrire les données dans un fichier externe et de charger celui-ci par l'intermédiaire de SQL, de l'interface ou du programme de monitoring.

Toutes les interfaces permettent aux commandes de SQL d'être envoyées au système de gestion de bases de données.
Par exemple, pour RPgSQL,

```
> db.connect(dbname=»freeworld»)
> db.execute(«SELECT ID, Height FROM freepenguins»,
    «WHERE WindowsDestroyed >= 98 ORDER BY Species»,
                                        clear=FALSE)
> db.fetch.result()
> db.disconnect()
```

pour RODBC,

```
> channel <- odbcConnect(«freeworld», uid=»linus»,
                                      case=»tolower»)
> sqlQuery(channel, «select id, heigth from
     freepenguins where windowsdestroyed >= 98 order
                                        by species»)
> odbcClose(channel)
```

et pour RMySQL,

```
> con <- dbConnect(MySQL(), dbname = «freeworld»)
> quickSQL(con, «select ID, Height from
     freepenguins where WindowsDestroyed >= 98 order
                                        by Species»)
> close(con)
```

Le module RPgSQL fournit une interface sophistiquée à PostgreSQL et met à disposition des facilités analogues à celles décrites pour RODBC. En outre, cette interface a la notion puissante d'un *data frame* du type proxy. C'est une classe de **R** qui hérite de la classe des *data frames* usuels, mais qui prend peu d'espace car c'est une référence à une table dans PostgreSQL. Il y aura donc un avantage lorsque l'on veut accéder à de plus petites parties du tableau des données, et ceci chaque fois que les opérations d'indexation dans **R** sont traduites en requêtes SQL. Ainsi le gros de la requête va être fait dans PostgreSQL directement plutôt que dans **R**:

```
> db.connect(dbname=»freeworld»)
> bind.db.proxy(«TableFreePenguins»)
```

`«TableFreePenguins»` est maintenant un proxy et donc tous les accès sont fait dans la base de données directement et pas dans **R**.

```
> TableFreePenguins[, «Height»]
Height
1   94
2   112
…
> db.disconnect()
```

Tous les DBMS offrent la possibilité plus ou moins limitée d'utiliser des fonctions et des opérateurs dans des instructions SQL. Par contre, il y a un bon nombre de différences: par exemple PostgreSQL et MySQL convertissent des types de données convenablement, mais Access ne le fait pas. Tous les noms de fonctions, leur portée exacte et la possibilité de définir des fonctions par l'utilisateur diffèrent largement. Access utilise Visual Basic pour des applications; MySQL et PostgreSQL permettent des fonctions définies par l'utilisateur. (On espère que le lecteur intéressé commence à voir les beautés d'un monde libre.) Pour chacune des trois interfaces ceci ouvre la possibilité d'inclure la fonctionnalité de **R** dans des fonctions définies par l'utilisateur, et ceci par l'intermédiaire d'une *shared library* sur Linux/Unix pour MySQL et PostgreSQL; voir aussi [16].

Dans tous les domaines de la statistique la gestion moderne des données est de nos jours basée sur des systèmes de gestion de base de données, la plupart du temps du type relationnel. Ces systèmes de gestion nous aident à organiser les données d'une manière sécurisée et accessible, même dans les environnements multi-utilisateurs. Par conséquent, le statisticien (et l'utilisateur de **R**) est confronté à de tels systèmes s'il doit extraire ou insérer des données. Il est alors souhaitable d'utiliser des méthodes simples d'accès aux données sans avoir besoin d'une expérience de programmation, l'objectif principal étant de rendre **R** indépendant du moteur de la base de données elle-même.

D'autres langages que **R** fournissent des interfaces de programmation bien définies pour des DBMS, tels que JDBC, Perl DBI et d'autres. **R** n'étant pas seulement un langage de programmation, mais également un outil pour l'analyse de données, il n'a pas seulement besoin d'une interface de programmation mais aussi d'une interface utilisateur avec DBMS. Dans ce cas on peut percevoir le système de gestion de bases de données comme mémoire. Ceci a été fait avec succès pour l'ajustement de modèles linéaires: voir [17] et [18]. Il faut toutefois tenir compte qu'actuellement des ensembles de données trop grands peuvent poser des problèmes avec **R**.

En plus du souhait d'avoir différentes interfaces pour différents moteurs, il faut s'assurer que les utilisateurs expérimentés n'ont pas de problèmes en utilisant des bases de données dans **R** avec les modules actuellement disponibles.

L'utilisation des bases de données relationnelles n'est pas si facile que ça. Compte tenu

de leurs fonctionnalités limitées, MySQL et mSQL sont les deux bases de données parmi les plus simples. D'autres comme Oracle ou Sybase nécessitent une connaissance détaillée des systèmes de gestion de bases de données. Dans un environnement client/serveur, on ne permet habituellement à un utilisateur ni de créer les bases de données lui-même, ni de les effacer. En outre, la puissance des bases de données relationnelles repose sur des relations. Il est difficile de formuler des ensembles joints dans d'autres langages que SQL.

Pour des informations sur les futures directions de **R** et de ses interfaces avec des bases de données, nous renvoyons aux articles [19] et [20] et surtout au *Omegahat Project for Statistical Computing* [16]. C'est un projet collectif dont le but est de fournir une variété de logiciels libres pour des applications statistiques. Ce projet a débuté en 1998, avec des discussions entre les créateurs responsables de trois langages statistiques actuels (S, **R**, et Lisp-Stat). Le projet *Omegahat* développe également une collection de modules pour soutenir les nouvelles directions dans la programmation dans le langage S (comme mis en application dans **R** ou dans S-Plus). Les modules illustrent comment communiquer entre **R** et d'autres langages et applications. Il existe des modules pour traduire en **R** du code écrit dans un autre langage, ou pour faire appel à des fonctions **R** depuis un autre langage ou application. Ceci englobe aussi la possibilité d'inclure **R** dans un tel module, ou vice versa.

Conclusion

Dans ce document nous avons brièvement illustré les interfaces existantes de **R** avec des base de données relationnelles comme étant un premier pas des logiciels statistiques modernes et libres, comme **R**, pour regagner le domaine du KDD. De plus, n'oublions pas que **R** fournit une grande variété de techniques statistiques et graphiques, et est fortement extensible. Le langage de S est souvent le véhicule de choix pour la recherche dans la méthodologie statistique, et **R** fournit un itinéraire libre pour participer à cette activité. Dans un milieu scientifique tel que l'EPFL, c'est à se demander si **R** ne devrait pas littéralement remplacer S-Plus, compte tenu des avantages de **R** et, en particulier, des interfaces qui lui sont possibles. Comme premier pas dans cette direction, nous espérons toujours que **R** sera prochainement disponible sur les serveurs ASIS, Distrilog et Cyclope du SIC.

Pour souligner ce propos, un deuxième article dans ce Flash informatique spécial montre l'utilisation de la géostatistique dans un monde libre en combinant **R** avec GRASS GIS.

Bibliographie

[1] **Diego Kuonen et Valerie Chavez-Demoulin** (2001), **R** - *un exemple du succès des modèles libres*, Flash Informatique 2/01, *http://sic.epfl.ch/publications/FI01/fi-2-1/2-1-page3.html*

[2] *The* **R** *Project for Statistical Computing*, *http://www.r-project.org*

[3] *The Comprehensive* **R** *Archive Network (CRAN)*, *http://cran.r-project.org*

[4] **Michael J. A. Berry et Gordon Linoff** (1997), *Data Mining Techniques: For Marketing, Sales, and Customer Support*, John Wiley & Sons.

[5] Statoo.com - The Portal to Statistics on the Internet, *http://www.statoo.com*

[6] Diego Kuonen's Data Mining Links, *http://bookmarks.kuonen.com/Statistics/Datamining/*

[7] **Daryl Pregibon** (2001), *A statistical odyssey*, In Proceedings of KDD Conference '99.

[8] Information, Support, and Software from the Hierarchical Data Format (HDF) Group of NCSA, *http://hdf.ncsa.uiuc.edu*

[9] MySQL Homepage, *http://www.mysql.com*

[10] **David Axmark, Michael Widenius, Jeremy Cole, and Paul DuBois** (2001), *MySQL Reference Manual*, *http://www.mysql.com/documentation/mysql*

[11] **Paul DuBois** (2000), *MySQL. New Riders*

[12] **Hughes Technologies** - The Home of Mini SQL (mSQL): *http://www.hughes.com.au*

[13] PostgreSQL Homepage: *http://www.postgresql.org*

[14] Interactive/On-line SQL Beginner and Advanced Tutorials: *http://sqlcourse.com* and *http://sqlcourse2.com*

[15] **R Development Core Team** (2001), **R** *Data Import/Export*, *http://www.r-project.org*

[16] The Omegahat Project for Statistical Computing, *http://www.omegahat.org*

[17] **Brian D. Ripley and Ruth M. Ripley** (2001), *Applications of* **R** *clients and servers*, In Proceedings of the 2nd International Workshop on Distributed, Statistical Computing, March 15-17, Vienna, Austria, *http://www.ci.tuwien.ac.at/Conferences/DSC-2001*

[18] **Roger Kœnker and Álvaro A. Novo** (2000), *Statistical Analysis of Large Datasets - An Exploration of* **R** *- MySQL Interface*, *http://www.econ.uiuc.edu/~roger/research/rq/LM.html*

[19] **Brian D. Ripley** (2001), *Using databases with* **R**. **R** *News*, 1(1): 18–20.

[20] **Torsten Hothorn, David A. James and Brian D. Ripley** (2001), **R**/*S Interfaces to Databases*, In Proceedings of the 2nd International Workshop on Distributed, Statistical Computing, March 15-17, Vienna, Austria, *http://www.ci.tuwien.ac.at/Conferences/DSC-2001* ■