R - un exemple du succès des modèles libres

Diego.Kuonen@epfl.ch et Valerie.Chavez@epfl.ch, Département de Mathématiques

R est «GNU S» - c'est à dire un langage et environnement pour les calculs statistiques et 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, NCSA Mosaic, Tcl/Tk, et Apache (voir [1]). Le but de ce document est de fournir un point de départ pour le débutant intéressé par **R**, qui, de plus est un exemple parmi tant d'autres du succès incontestable des modèles libres.

Qu'est-ce qu'un logiciel libre?

C'est un logiciel qui fait référence à quatres types de liberté (voir [2] et [16]) dont l'une précise que le code source doit être publié. Depuis plus de 30 ans certains choisissent de publier les codes sources des logiciels qu'ils développent. Cette manière d'échanger et de partager les connaissances a donné naissance à de nombreuses technologies et applications. Internet est bâti sur des logiciels libres.

Il existe plusieurs types de licences associées aux logiciels libres et plus généralement aux logiciels dont les codes sources sont disponibles. Il est aussi prévu que le code puisse être modifié et une communauté est toujours associée au développement de chaque logiciel. Ainsi ces logiciels libres peuvent être sans *copyright*, avec une licence GPL (comme la licence de **R**), avec un *copyright* limité (par exemple la licence Mozilla), ou rester propriétaire (par exemple la licence SCSL). Le développement peut être organisé sous une forme hiérarchique ou complètement distribuée. Il est en priorité tourné vers l'utilisateur et est le fruit de nombreux échanges entre un très grand nombre de personnes du monde entier extrêmement motivées. Le résultat donne des logiciels adaptés aux besoins, rapidement très stables et très sûrs. Leur force est donc leur réactivité; chaque point est traité par un spécialiste et l'implémentation des technologies nouvelles ou anciennes est très rapide. Le choix est toujours pertinent. Du fait du grand nombre de lecteurs et de développeurs le code est forcément bien commenté et les problèmes ou les mauvais choix sont rapidement identifiés et corrigés. Les logiciels libres sont donc souvent en avance sur les autres en matière de technologie, notamment ceux qui concernent Internet. Ils sont bien documentés, stables et sûrs.

Qu'est-ce que R?

R est un système d'analyse statistique créé par Ross Ihaka et Robert Gentleman [3] du Département de Statistique de l'Université de Auckland. Un grand nombre de personnes ont également contribué à **R** en programmant ou en établissant des *Bug Reports* [11]. **R** est à la fois un langage et un logiciel; parmi ses caractéristiques les plus remarquables citons:

- un système performant de stockage et de manipulation de données;
- la possibilité d'effectuer du calcul matriciel et autres opérations complexes;
- une large collection intégrée et cohérente d'outils d'analyse statistique;
- un large éventail d'outils graphiques particulièrement flexibles;
- un langage de programmation simple et efficace qui inclut de nombreuses facilités.

R est un langage qualifié de dialecte du langage S créé par Lucent Technologies (AT&T Bell Labs). S est disponible sous la forme du logiciel S-Plus commercialisé par la compagnie américaine Insightful Corporation [4], qui se trouve aussi sur les serveurs du SIC: ASIS pour Unix/Linux et Distrilog pour PC/Windows. Par contre, il n'existe pas de version de S-Plus pour MacOS.

Mais pourquoi R s'il existe déjà S?

Au départ, le but du projet **R** était de démontrer qu'il est possible de produire un environnement tel que S, non pénalisé par le manque de mémoire et les problèmes de performance de S. **R** s'est en quelque sorte transformé en système réel pour lequel une grande part de son efficacité a en contrepartie disparu. C'est pourquoi le mécanisme qui gère la mémoire a été récemment revu. La dernière version, **R** 1. 2 contient un nouveau système de gestion de la mémoire. Ceci augmente radicalement sa performance. Ainsi, l'espace de travail n'est plus statiquement dimensionné; voir le fichier *NEWS* qui vient avec **R** pour plus de détails. Il existe quelques différences entre la conception de **R** et celle de S. Dans ce document, nous ne nous attardons pas sur ces différences bien qu'importantes, mais le lecteur intéressé peut se reporter à [3] ou au **R**-FAQ [5] dont une copie est également distribuée avec le logiciel. Redistribué librement sous les termes de la *GNU Public Licence* de la *Free Software Foundation*, son développement et sa distribution sont assurés par plusieurs statisticiens rassemblés dans le ***R** Development Core*

Team, qui existe depuis mi-1997 et qui modifie l'archive CVS du code source de **R**. Un élément essentiel de ce développement étant le *Comprehensive* **R** *Archive Network* [6] dont un exemplaire se trouve à l'ETHZ, ainsi que sur le site officiel de **R** [7]. De plus, notons aussi que **R** fait partie officiellement du projet GNU (*GNU S*). **R** est disponible sous plusieurs formes: code écrit en C (et certaines routines en Fortran) prêt à être compilé, surtout pour les machines Unix/Linux, ou des exécutables prêts à l'emploi (après une installation très simple) pour Windows 95, 98, ME, NT4 et 2000 et comme beta pour MacOS [8]. En revanche, **R** n'est pas encore disponible sur les serveurs du SIC: ASIS et Distrilog ☺on espère l'y voir bientôt☺

Le noyau de **R** est un langage de programmation interprété qui permet les embranchements et les boucles aussi bien que la programmation modulaire utilisant des fonctions. La plupart des fonctions visibles par l'utilisateur dans **R** sont écrites en **R**. L'interface avec des procédures écrites en langage C, C++ ou Fortran est possible pour l'utilisateur qui gagne ainsi en efficacité. La distribution de **R** contient des fonctionnalités pour un grand nombre de procédures statistiques. Parmi elles, on compte: les modèles linéaires et linéaires généralisés, les modèles de régression non-linéaire, l'analyse de séries temporelles, les tests de classification paramétriques et non-paramétriques, les méthodes de groupement et de lissage, etc. Il existe également un grand nombre de fonctions qui fournissent un environnement graphique flexible pour créer divers genres de présentations des données. Les modules supplémentaires *(add-on packages)* sont disponibles dans des buts spécifiques (pour une liste, voir [5]). **R** est un langage qui comporte de nombreuses fonctions pour les analyses statistiques et les graphiques; ceux-ci sont immédiatement visualisés à l'aide d'une fenêtre et peuvent être exportés sous divers formats (par exemple jpg, png, bmp, eps, ou wmf avec Windows, ps, bmp, pictex avec Unix/Linux).

Les résultats des calculs statistiques sont affichés à l'écran, et certains résultats partiels (p-valeurs, coefficients de régression, …) peuvent être sauvés ailleurs, exportés dans un fichier ou utilisés pour des analyses ultérieures. Le langage **R** permet, par exemple, de programmer des boucles qui vont analyser successivement divers jeux de données. Il est aussi possible de combiner dans le même programme différentes fonctions statistiques pour réaliser des analyses plus complexes. Les utilisateurs de **R** peuvent bénéficier des nombreuses routines écrites pour S-Plus, la plupart de ces routines étant directement utilisables avec **R**. De prime abord, **R** peut sembler trop complexe pour une utilisation par un non-spécialiste. Ce n'est pas forcément le cas. En fait, **R** privilégie la flexibilité. Alors qu'un logiciel classique (SAS, SPSS, Statistica,…) affiche directement tous les résultats (ou presque) d'une analyse, avec **R**, ces résultats sont stockés dans un objet, si bien qu'une analyse peut être faite sans qu'aucun résultat ne soit affiché. L'utilisateur peut en être perturbé, mais ceci se révèle extrêmement utile. En effet, l'utilisateur peut alors extraire uniquement la partie des résultats qui l'intéresse. Par exemple, si l'on doit faire une série de 50 régressions et que l'on veut comparer les coefficients des différentes régressions, **R** peut afficher seulement les coefficients estimés: les résultats tiendront donc sur 50 lignes, alors qu'un logiciel plus classique ouvrirait 50 fenêtres de résultats. On pourrait citer bien d'autres exemples illustrant la supériorité d'un système tel que **R** par rapport à certains logiciels classiques.

Quelques astuces à connaître avant de démarrer

Une fois **R** installé sur votre ordinateur, il suffit de lancer l'exécutable correspondant (`RGui.exe` ou `Rterm.exe` sous Windows, `R` sous Unix/Linux)pour accéder au programme. Le *prompt* de la forme du signe plus-grand-que `>` apparaît alors indiquant que **R** est en attente des commandes. **R** est un langage orienté-objet, c'est-à-dire que les variables, les données, les matrices, les fonctions, les résultats, etc. sont stockés dans la mémoire vive de l'ordinateur sous forme d'objets qui ont un nom: ils suffit alors de taper le nom de l'objet pour afficher son contenu.

Par exemple, si un objet **n** a la valeur 20:

```
> n
[1] 20
```

le chiffre 1 entre crochets indique que l'affichage commence au premier élément de n. Pour donner une valeur à un objet, il faut utiliser le symbole *assign* qui s'écrit avec une petite flèche composée d'un signe moins accolé à un plus-grand-que, ce symbole pouvant être orienté dans un sens ou dans l'autre:

```
> n <- 25
> n
[1] 25
> 15 -> n
> n
[1] 15
```

la valeur ainsi donnée peut être le résultat d'une expression arithmétique:

```
> n <- 25+2
> n
[1] 27
```

remarquez que l'on peut simplement taper une expression sans attribuer sa valeur à un objet, le résultat est alors affiché à l'écran mais n'est pas stocké en mémoire:

```
> (8+2)*5
[1] 50
```

la fonction `ls()` permet d'afficher la liste simple des objets en mémoire, c'est-à-dire que seuls les noms des objets sont affichés.

```
> name <- "Tux"; n1 <- 20; n2 <- 200; m <- 0. 2
> ls()
[1] "m" "n1" "n2" "name"
```

notons l'usage du point-virgule `;` pour séparer des commandes distinctes sur la même ligne. Pour effacer des objets de la mémoire, on utilise la fonction `rm()`. L'aide en ligne de **R** donne de très bonnes informations sur l'utilisation des fonctions. On peut appeler l'aide en format html en tapant:

```
> help.start()
```

une recherche par mots-clefs est possible avec cette aide html. L'aide est aussi disponible en format texte pour une fonction donnée, par exemple:

```
> ?lm
```

affichera l'aide pour la fonction `lm()`. La fonction `help("lm")` aura le même effet. Notons aussi que **R** pos-

sède sa propre documentation en format LaTeX, utilisée pour fournir une documentation compréhensible, disponible sur Internet [6] sous différents formats ou copie brute. La distribution de **R** s'accompagne également de 4 manuels:

- *An Introduction to* ***R*** (R-intro) qui donne des informations sur les type de données, éléments de programmation, modélisation statistique et représentations graphiques;
- *Writing* ***R*** *Extensions* (**R**-exts) décrit le processus de création de **R**, des *add-on packages*, documentation sur comment écrire **R**, le système **R** et interfaces avec les autres langages et «**R** API»;
- ***R*** *Data Import/Export* (R-data) est un guide concernant l'importation et l'exportation de données dans et depuis **R**;
- *The* ***R*** *Language Definition* (**R**-lang) explique l'évaluation, l'analyse, la programmation orientée-objets, etc.

En complément au matériel écrit spécialement pour **R**, la documentation de S/S-Plus peut être utilisée en combinaison avec le **R** FAQ [5] (voir sous «What are the differences between **R** and S?»). De plus, vous pouvez aussi télécharger le polycopié «Introduction to S-Plus for Unix» écrit par le premier auteur [9], pratiquement 100% R-compatible.

Aussi, vous trouverez toujours une aide précieuse dans les divers *mailing lists* de **R** [5] dont les archives sont sur [14]. Fin janvier 2001, le premier numéro de «**R** News» est apparu [10], très prometteur. Cette documentation est destinée à combler le décalage entre les *mailing lists* et les journaux scientifiques.

Visualisation à l'aide de R

Plusieurs personnes utiliseront **R** principalement pour ses outils graphiques. Ces derniers sont une composante extrêmement souple de l'environnement **R**. Il est possible de les utiliser pour représenter une grande variété de graphes statistiques mais aussi pour construire entièrement de nouveaux types de graphiques. Les outils graphiques peuvent être utilisés soit en interactif soit en mode *batch*, bien que dans beaucoup de cas, l'utilisation interactive est plus productive. L'utilisation interactive est aussi plus facile car au lancement de **R**, une fenêtre graphique s'ouvre pour afficher les graphes interactifs. Bien qu'automatique, il est utile de connaître la commande `X11()` sous Unix/Linux et `windows()` sous Windows. Une fois le dispositif enclenché, les commandes de représentations graphiques de **R** peuvent être utilisées pour produire une grande variété de graphes ainsi que de nouveaux types de graphes. Les commandes des représentations graphiques se divisent en trois principaux groupes:

- les fonctions de haut niveau de représentation graphique qui créent une nouvelle représentation dans le dispositif de graphiques, avec possibilité de choix des axes, étiquettes, titre, etc.;
- les fonctions de bas niveau de représentation graphique rajoutent des informations au graphe existant, telles que des points, des lignes ou des étiquettes;
- des fonctions graphiques interactives permettent de rajouter ou d'extraire interactivement des informations sur le graphe existant en utilisant un dispositif de pointage tel que la souris.

Une des forces de R est la facilité à produire des graphiques de haute qualité, incluant si nécessaire des formules ou des symboles mathématiques. Ceci est possible avec **R** en spécifiant une expression plutôt qu'une chaîne de caractères dans l'une des commandes «`text`», «`mtext`», «`axis`», ou «`title`». Plus d'information, notamment une liste entière de possibilités, peut s'obtenir directement dans **R** en utilisant les commandes:

```
> help(plotmath)
> example(plotmath)
```

Bien que le choix des détails graphiques par défaut soit satisfaisant, l'utilisateur détient le contrôle total de la conception de son graphe.

Les Figures 1 et 2 sont des graphes produits en utilisant **R**. Pour la figure 1 les commandes sont:

```
> data("iris")
```

Cette commande donne accès aux données «iris». Ces données sont les résultats de quatre mesures effectuées sur

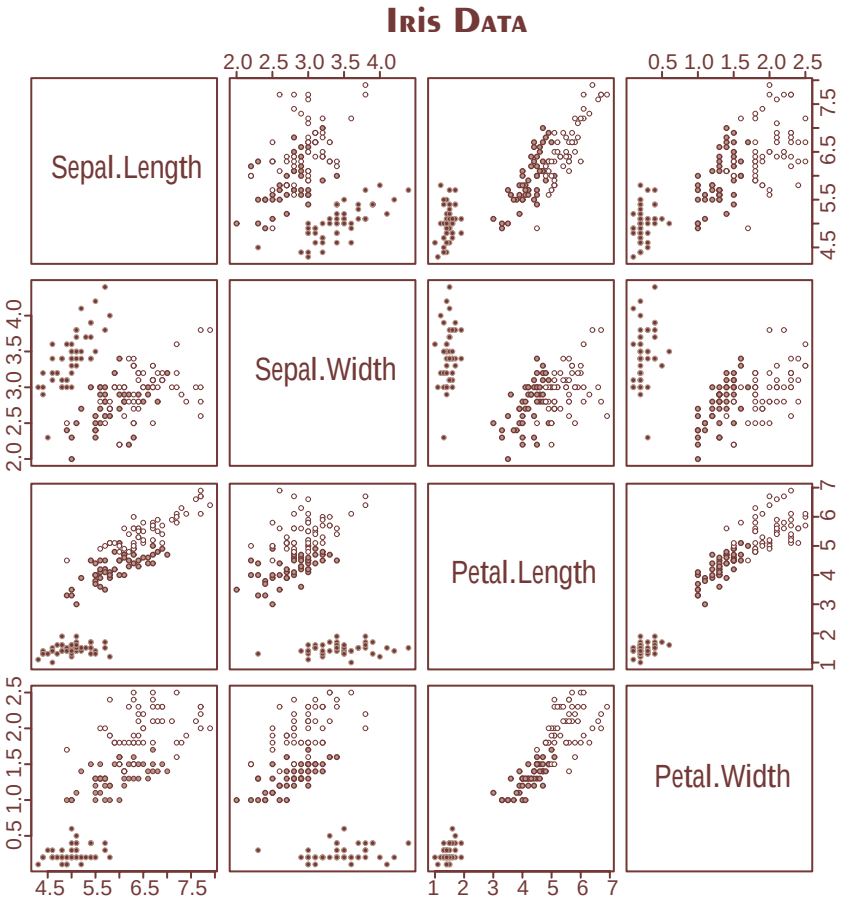

figure 1: «Iris Data» – représentation graphique de toutes les possibilités de variables contre variables

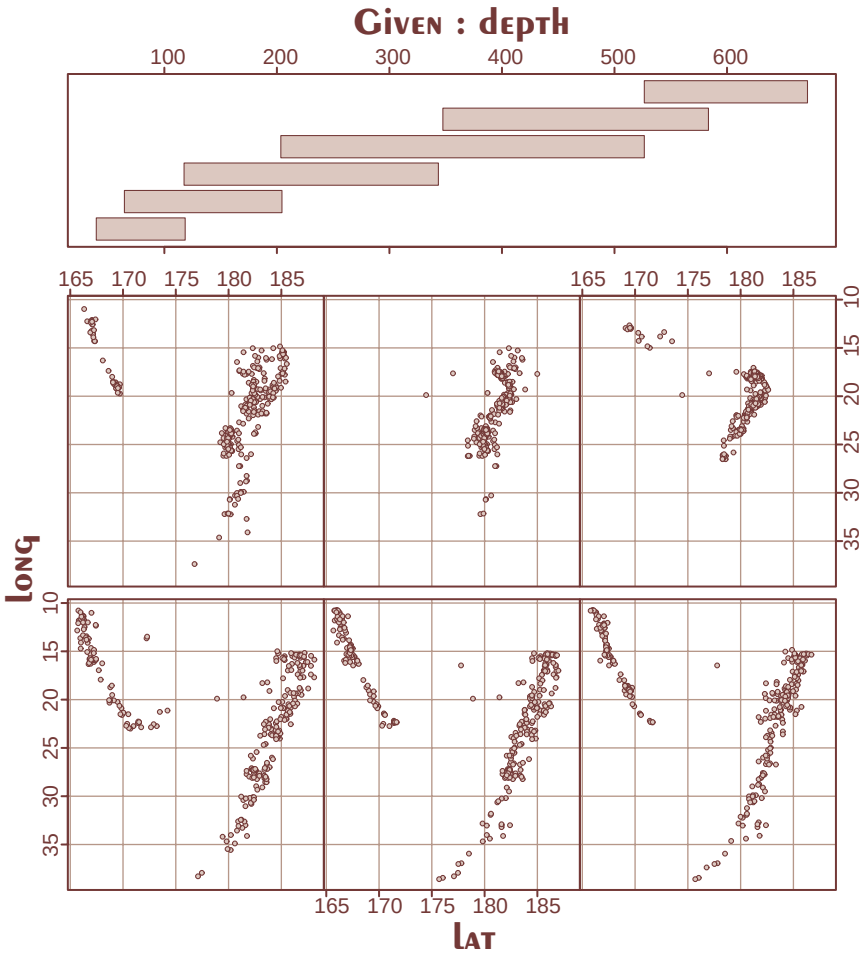

figure 2: «Earthquakes Data» – représentation graphique de la variable longitude en fonction de la latitude, sachant la profondeur des séismes

50 fleurs de trois types différents d'Iris: Setosa, Versicolor et Virginica. Les quatre mesures sont en centimètre: la longueur et largeur du sépale ainsi que celles du pétale.

```
> pairs(iris[1:4], main = "Iris Data", pch=21,
bg=c("red", "green3", "blue")[codes(iris$Species)])
```

La fonction `pairs` reproduit tous les graphiques variables contre variables possibles sur une seule fenêtre graphique, et `main`, `bg` et `pch` sont des fonctions graphiques de bas niveau pour ajouter un titre, définir la couleur et le symbole à utiliser pour représenter les observations.

Pour la figure 2, les commandes sont:

```
> data(quakes)
```

Les données «quakes» donnent la location de 1000 séismes importants qui se sont produits dans un cube proche de Fiji depuis 1964. Les variables mesurées sont «lat» (latitude), «long» (longitude), «depth» (profondeur), «mag» (magnitude sur l'échelle de Richter), «stations» (nombre de stations ayant enregistré le séisme).

```
> coplot(long ~ lat | depth, data = quakes,
pch = 21, bg = "green3")
```

la fonction «coplot» produit une représentation graphique conditionnelle, qui montre comment une réponse (longitude ici) dépend d'une variable explicative (latitude ici) étant donné une autre variable explicative (profondeur ici).

De plus le nouveau module «tcltk» fournit des attaches d'interface et de langage pour le kit d'outils de l'interface graphique Tcl/Tk. Ceci permet de mettre en place des boutons et des zones d'entrée en un nombre restreint de lignes de code. Pour une petite démonstration voir

```
> demo(tkdensity)
```

Autres liens utiles

Voici d'autres liens intéressants en relation avec **R**:

- «ESS: Emacs speaks Statistics» [12], est une interface d'Emacs-Lisp à la programmation statistique interactive et aux langages d'analyse de données, incluant: les dialectes de S (tels que **R**), les dialectes de LispStat et SAS. ESS est distribué aussi sous la GNU GPL.
- XGobi et XGvis [15]: XGobi est un système de visualisation de données permettant des représentations graphiques multi-dimensionnelles, et XGvis est un système de visualisation interactif aussi bien pour données de proximité que pour les graphes ou les réseaux. Il existe une interface entre **R**, XGobi et XGvis. Cette interface est un moyen de visualiser avec qualité les résultats ou données de **R** en plusieurs dimensions, ce qui s'avère efficace pour plusieurs analyses statistiques.
- Le «Omegahat Project for Statistical Computing» [13] est un projet dont le but est de fournir une variété de logiciels pour des applications statistiques. Le projet Omegahat a débuté en juillet 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 utiles 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 comprend la notion d'inclure **R** dans un tel module, ou vice versa.

Il y a actuellement cinq modules qui fournissent différentes manières de communiquer entre **R**/S-Plus et d'autres langages et applications:

XML outils pour documents de lecture et d'écriture XML dans **R**, permettant un échange de données simplifié entre **R** et d'autres applications;

Java une interface pour appeler dans **R** un logiciel écrit en Java, et des fonctions **R** depuis Java, incluant une fenêtre graphique dans **R** utilisant les outils graphiques de Java, ladite fenêtre facilement personnalisable;

RSPerl une interface pour exécuter Perl à l'intérieur de **R**, et **R** à l'intérieur de Perl afin de créer des objets Perl, et appeler leurs méthodes et sous-programmes Perl de **R**, et vice-versa;

Python comme les interfaces de Perl et de Java, ceci permet à des fonctions **R** d'être appelées;

CORBA outils dynamiques pour appeler des méthodes dans d'autres applications, sur d'autres machines et écrites en langage différent et fournissant également ces types de serveurs objets dans **R** lui-même. En d'autres termes, ceci fournit le mécanisme de haut niveau pour l'informatique répartie dans **R**.

Conclusion

Dans ce document nous avons brièvement introduit **R**, qui est un exemple du succès incontestable des modèles libres.

R fournit une grande variété de techniques statistiques (les modèles linéaires et non linéaires, les tests statistiques classiques, l'analyse de séries temporelles, la classification, les méthodes de groupement,…) et graphiques, et est fortement extensible. Le langage de S est souvent le véhicule choisi 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 disponibles.

Dans le prochain Flash informatique spécial été consacré aux logiciels libres, nous montrerons comment il est utile de combiner **R** avec d'autres produits tels que MySQL, GRASS GIS, PostgreSQL et autres.

Nous espérons que **R** sera prochainement disponible sur les serveurs ASIS, Distrilog et Cyclope du SIC.

Bibliographie

[1] *Association for Computing Machinery - Software System Award*: *http://www.acm.org/awards/ssaward.html*

[2] *Qu'est-ce qu'un Logiciel Libre?*: *http://www.gnu.org/philosophy/free-sw.fr.html*

[3] Ross Ihaka et Robert Gentleman (1996), ***R**: A Language for Data Analysis and Graphics*, Journal of Computational and Graphical Statistics, 5:3, 299-314.

[4] Insightful Corporation, *l'éditeur de S-Plus*: *http://www. insightful.com*

[5] Kurt Hornik (2001), *The **R** FAQ*: *http://cran.r-project.org/doc/FAQ/R-FAQ. html*

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

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

[8] ***R** beta pour MacOS*: *http://www.eco-dip. unimi.it/R/*

[9] Diego Kuonen (2001), *Introduction to S-Plus for Unix (with exercises)*: *http://statwww.epfl.ch/splus/*

[10] ***R** News - the newsletter of the **R** project*: *http://cran.r-project.org/doc/Rnews/*

[11] ***R** Bug Tracking System*: *http://bugs.r-project.org*

[12] *ESS: Emacs speaks Statistics*: *http://ess.stat.wisc.edu*

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

[14] ***R** mailing lists archive*: *http://www.ens.gu.edu.au/robertk/R/*

[15] *Data Visualization Systems XGobi and XGvis*: *http://www.research.att.com/areas/stat/xgobi/*

[16] Anne Possoz, *La vie des logiciels libres ou la liberté des logiciels vivants*: *http://sic.epfl.ch/publications/FI00/fi-10-00/10-00-page10.html* ■

Conférences «Un siècle d'informatique !»

Salle DI-IN 200
Mercredi 14 mars à 15h00
Vendredi 16 mars à 15h00

Si les mots *Windows* et *Macintosh* font aujourd'hui partie du vocabulaire de tout le monde, si les termes *C++* et *Pentium* évoquent quelque chose à la plupart d'entre vous, combien de personnes connaissent l'origine de ces concepts ? Ce qu'étaient les débuts de l'informatique ?

A l'aube de ce nouveau millénaire, deux professeurs du Département d'informatique de l'EPFL, le professeur **Eduardo Sanchez** et le professeur **Martin Odersky**, s'exprimeront sur le bond fulgurant de l'informatique au cours du siècle dernier, de l'origine de l'ordinateur ENIAC au fameux langage de programmation Java.

Le professeur Sanchez donnera son exposé sur le thème «**l'évolution du matériel informatique**» mercredi 14 mars à 15h à la salle DI-IN200.

Le professeur Odersky présentera «**l'histoire des langages de programmation**» vendredi 16 mars à 15h en salle DI-IN200. Cette séance se déroulera en anglais.

D'environ une heure chacune, ces conférences sont prévues pour un public non-averti.

Ce sera là une occasion unique d'apprendre et de jeter un regard éclairé sur le passé de cette science avant de se lancer avec enthousiasme sur les différents fronts de recherche actuellement ouverts.

Venez nombreux !

Renseignements complémentaires:
Benjamin.Leroy-Beaulieu@epfl.ch & *Yacine.Saidji@epfl.ch*