La Revue de Modulad

UNE PRESENTATION DES LANGAGES S ET SPLUS

André Carlier

Laboratoire de Statistiques et Probabilités
Université Paul Sabatier Toulouse
email : carlier@frcict81.bitnet

Table des matières

1 Introduction

Un exposé n'étant jamais objectif, mais relatif à la vision de son auteur, je me permets de me présenter brièvement. Je suis un fan'S, depuis le jour où, perdu au milieu de milliers de lignes Fortran, je l'ai rencontré à l'université de Montréal. Je l'ai utilisé pendant une période d'environ 6 mois pour écrire, parallèlement au développement d'une recherche, un logiciel d'environ 2000 lignes. J'ai alors pu, après quelques efforts, le faire suivre à Toulouse. Je suis plus un chercheur en méthodes statistiques qu'un utilisateur, mais j'effectue cependant de nombreuses applications. Je n'ai pas encore eu l'occasion de le tester dans des applications réelles (je dispose de SPLUS sur 386 depuis moins d'un mois). J'essaie simplement ici de rapporter l'expérience de personnes qui l'utilisent dans l'enseignement ou dans le traitement de données.

L'exposé qui suit est divisé en deux parties. Dans un premier temps, je présente différents aspects de S, sans présenter le langage lui-même. Cette présentation est faite dans une seconde partie, où j'essaie de passer en revue ses principales fonctionalités en faisant ressortir ce qu'elles ont de spécifique. Mais la richesse et la diversité de ce langage font que cet exposé ne décrit que très partiellement ses possibilités.

2 A propos du langage S

Après une brève présentation de S[1], on discute l'intérêt de son utilisation dans les domaines du traitement de données, de la recherche et de l'enseignement, et on précise ce pour quoi il pourrait être moins adapté. On tente ensuite de définir à quels utilisateurs il est plus particulièrement destiné, quel type d'apprentissage il nécessite, et quel aide on peut attendre de la documentation et de l'aide en ligne. On se risque ensuite à donner quelques éléments de comparaison avec certains logiciels existants. Enfin, après avoir précisé comment S s'insère dans l'environnement informatique (interactions avec le système d'exploitation, éditeurs intégrés, drivers graphiques), on présente l'environnement scientifique qui l'accompagne : l'équipe des Bell Laboratories, le forum S, la bibliothèque "STATLIB" de programmes développés par les utilisateurs, et "STATSCI" (Statistical Science), société qui développe SPLUS[2] (un surensemble de S). On parle dans la suite à la fois du langage S (ou plutôt New-S, car une ancienne version, avec une syntaxe différente, avait précédé celle-ci), de SPLUS et de STATLIB, en précisant à chaque fois ce qui est spécifique à l'un ou à l'autre.

2.1 Un environnement statistique et graphique

S est un langage interactif et interprété. Il peut-être considéré comme une boîte à outils alliant les outils les plus simples aux plus évolués. Chaque commande ou fonction S va mettre en oeuvre une méthode, un calcul, ou permettre d'effectuer un graphique. Un assemblage d'outils s'appelle encore une fonction. La définition de fonctions se fait avec une grande simplicité et celles-ci s'insèrent immédiatement sous forme d'objets S dans l'environnement de base, et restent disponibles au même titre que les autres fonctions. Les objets S (voir aussi 3.1.2) ont des structures variées. Un objet peut-être atomique (comme une matrice, un tableau, une série chronologique, ...) et comporter des attributs (par exemple, une matrice est un vecteur avec un attribut dimension). Il peut aussi être récursif, c'est-à-dire avoir une structure de liste. Tous ces objets sont rangés dans différents répertoires sous formes de fichiers binaires séparés.

Le système S est un système ouvert (appelé encore sytème d'accueil [6]) pouvant jouer le rôle de système d'exploitation statistique : il permet alors d'effectuer des traitements divers sur les données, d'intégrer des programmes ou logiciels sous forme de fonctions S, ou de servir

[1] ©AT&T Greensboro, North-Carolina USA

[2] ©Statistical Sciences Inc. Seattle, Washington, USA

seulement d'interface Son graphique interactif, qui sera décrit au paragraphe 3 1 8, a une qualité et une souplesse qui en font un outil apprécié pour le traitement de données, mais aussi pour les graphiques destinés à l'impression (articles, livres, etc)

La nature interprétée du langage S, ses fonctionalités graphiques et le volume de software qu'il met en oeuvre nécessitent un environnement informatique de qualité et suffisamment puissant, mais que l'on trouve de plus en plus fréquemment de nos jours La configuration type est la station de travail UNIX mise en réseau et associée à une imprimante graphique (de préférence laser ou jet d'encre) SPLUS autorise aussi l'utilisation d'un micro-ordinateur 386 sous système DOS, dans une configuration particulièrement solide (vrai 386 avec une mémoire d'au moins 4 megaoctets et un disque dur de 40 megaoctets)

2.2 Quelles utilisations pour quels utilisateurs ?

2.2.1 Le traitement de données

Le langage S est un environnement statistique permettant une manipulation aisée des données Il est particulièrement riche en procédures exploratoires "à la Tukey" [8] et en procédures robustes, souvent associées à des graphiques d'une grande diversité et dont la qualité permet de traiter des ensembles de données importants (voir paragraphe 3 2 2) Son interactivité est particulièrement adaptée à ce type d'analyse, où le résultat d'une procédure permet à l'utilisateur de décider quelle sera la suivante Il comporte aussi (voir paragraphe 3) des procédures dans des domaines aussi divers que la simulation, l'analyse des données multidimensionnelles, les séries chronologiques et le lissage Enfin de nombreuses procédures supplémentaires sont disponibles dans SPLUS ou dans la bibliothèque STATLIB accessible en self-service par courrier électronique (téléchargement) Mais SPLUS et surtout S ne sont pas encyclopédiques comme peuvent l'être SAS ou SYSTAT Ils ont en particulier des faiblesses pour la lecture des données et dans un certains nombres de domaines de statistiques classiques (voir paragraphe 2 4 3)

2.2.2 Le développement de nouvelles méthodes et de nouvelles pratiques statistiques

C'est probablement l'usage actuellement le plus fréquent du langage S

La programmation en S La variété des objets S permet de manipuler des entités complexes : par exemple, une hiérarchie, résultat d'une classification de n éléments, a une structure de liste comprenant : un tableau $n-1 \times 2$ décrivant les aggrégations successives, un vecteur de longueur $n-1$ contenant la suite des niveaux de regroupement, et un vecteur de longueur n donnant un ordre compatible avec la hiérarchie Les noms des objets classés peuvent faire partie de l'ensemble Ainsi, l'organisation d'informations en liste ou en objet avec attributs permet de concevoir simplement des types d'objets variés La programmation en langage S est facilitée par la nature obligatoirement structurée de ce langage fonctionnel, par son caractère interprèté qui évite la phase de compilation, par la richesse des fonctions de base existantes et leurs niveaux très différents de complexité, mais aussi par des outils d'aide à la programmation sophistiqués On donne en 2 4 des éléments de comparaison entre S et d'autres langages matriciels On trouve dans [7] une comparaison entre l'usage de S/SPLUS et du langage IML de SAS dans le domaine du développement qui donne un très net avantage à S

La portabilité des programmes S Cette portabilité est bien sûr subordonnée à la diffusion du langage S Cette diffusion est importante dans les pays ou l'ancien S était bien implanté, aux Etats Unis, Canada, Australie et Nouvelle Zélande La France semble aussi bien placée grâce au potentiel d'utilisation et de développement que représente l'INRA (voir [3]) Le langage New-S

n'en étant qu'à ses débuts (1988), sa diffusion devrait s'intensifier en Europe Aujourd'hui, de nombreuses méthodes faisant l'objet d'articles récents sont disponibles sous forme de fonctions S dans la bibliothèque STATLIB et dans SPLUS Sans cette possibilité, un chercheur qui propose une méthode ou une variante de méthode, est obligé d'écrire son propre logiciel pour la mettre en oeuvre Ceci présente au moins deux inconvénients Le premier est le temps perdu à écrire le logiciel, qui nécessairement doit avoir des fonctionnalités qui existent déjà dans de nombreux autres Le second inconvénient concerne l'utilisateur, qui, pour utiliser une variante d'une méthode, doit implémenter un nouveau logiciel La bibliothèque MODULAD a permis de résoudre en partie le problème, en permettant à un chercheur de n'écrire qu'une étape L'intérêt de S est que le temps de programmation est, relativement à celui d'une programmation en Fortran, divisé par 10 ou 20, et qu'il peut suffire d'écrire une seule fonction Si la méthode s'avère suffisamment utilisée et d'exécution trop lente avec S, il sera alors possible de réécrire le noyau de la fonction ou de l'algorithme en Fortran ou en C, et de l'intégrer dans une fonction S Mais il sera inutile de se préoccuper de la phase gestion des données, toujours très longue à programmer Pour terminer, précisons que la possibilité de mettre sous forme ASCII les fonctions S (par une procédure de dump/restore) permet de les transmettre simplement par courrier électronique

L'insertion dans une communauté scientifique L'utilité de la programmation en S de nouvelles méthodes réside dans la possibilité déjà évoquée de pouvoir faire partager très rapidement sa fonction par une communauté de chercheurs importante Il existe bien sûr d'autres "communautés logicielles" (GLIM letters, clubs SAS, journées SPAD), mais celle-ci grâce au forum S, permet un débat journalier sur différents problèmes, allant de la recherche de jeux de données ou de fonctions particulières, l'échange de cours sur S, à la demande de conseils parfois triviaux (il est fortement recommandé de ne pas s'abonner individuellement à ce service, mais de le faire une seule fois par site)

2.2.3 Le graphique

Les fonctionnalités graphiques de S peuvent justifier à elles seules son utilisation On peut distinguer les graphiques exploratoires qui permettent d'analyser des données et les graphiques de base qui permettent de réaliser des figures avec une grande souplesse Les procédures exploratoires sont souvent interactives et utilisent la souris ; certains graphiques à trois dimensions sont dynamiques (SPLUS) Les principales fonctions graphiques sont passées en revue au paragraphe 3.1.8 La difficulté de réalisation d'un graphique est proportionnelle à la sophistication des résultats souhaités On peut donc passer progressivement des graphiques simples qui demandent très peu d'investissement à des graphiques très évolués qui demanderont une mise au point plus complexe Une des grandes différences entre le graphique de S et celui d'autres logiciels réside une fois de plus dans la flexibilité Le langage S ne propose pas de graphique figé, mais rend possible beaucoup de compositions (matrice de figures, choix des échelles, choix de titres, labels, etc.) Il peut aussi se transformer en logiciel de cartographie statistique, qui ne prétend pas à une qualité de géographe, mais qui, là encore, a une souplesse inégalée Pour terminer, on verra en 2.5 que les graphiques s'adaptent aussi bien pour leur visualisation que pour leur impression sur des écrans et des imprimantes variés

2.2.4 L'enseignement des statistiques

Sans expérience sur ce thème, on se contente ici de résumer très succintement les débats ayant eu lieu dans le cadre du forum S Aux Etats-unis, certains utilisent le langage S pour les étudiants non gradués, donc pour l'apprentissage des statistiques dans un cours d'initiation ou un cours plus avancé, d'autres estiment que ce langage n'a sa place qu'au niveau DEA En France, S devrait être utile au niveau licence pour son langage matriciel, ses procédures

graphiques, et ses facilités de simulation. D'après les commentaires du forum S, dans le cadre d'un enseignement de base de la statistique, l'apprentissage du logiciel lui-même ne demande que quelques heures. L'obstacle majeur reste encore la disposition d'un outil informatique adapté. Mais on peut signaler que S peut très bien s'utiliser sur des consoles ASCII classiques, branchées sur un serveur UNIX. Le seul graphique alors utilisable est de type *printer*, un peu décevant, mais encore très utilisé aujourd'hui. Dans le domaine matriciel, l'utilisation de S est comparable à celle de Gauss, MATLAB, ou de l'APL. Elle permet de faire réécrire rapidement par les étudiants les procédures statistiques, et de les exécuter sur un jeu de données réelles. Mais l'environnement proposé par S le rendra plus attrayant que les logiciels plus classiques. Cela ne signifie pas que son utilisation remplacera l'apprentissage de logiciels spécialisés, qui sont souvent le support d'une pratique particulière dans un domaine particulier. Mais grâce à une personnalisation des sorties, l'utilisation de S doit permettre de mieux coller à une pédagogie donnée. De même, la possibilité de définir des règles de recherche et de partager des répertoires est un atout pour l'enseignement. Enfin S peut faciliter un échange de supports de travaux pratiques à l'échelle internationale. Ainsi la bibliothèque STATLIB a-t-elle déjà reçu ce qu'a réalisé pour l'enseignement de la régression un professeur de l'université de Montréal (voir [5]).

2.2.5 S pour quels utilisateurs ?

A qui n'est pas destiné le langage S Le langage S n'est pas destiné à des personnes connaissant peu de statistiques et voulant effectuer des traitements simples, répétitifs ou occasionnels (calcul de Chi2, comparaisons de moyennes, analyses de variance simples) avec le minimum d'investissement. Il ne remplace pas les logiciels ciblés ou très spécialisés, soit parce qu'il risque d'être moins complet dans un domaine particulier, soit parce que, donnant une grande liberté à l'utilisateur, il peut aussi le laisser plus désemparé.

A qui est destiné le langage S Il peut être employé par des expérimentateurs connaissant peu de statistiques, mais cherchant à "comprendre leurs données" à l'aide de procédures exploratoires et graphiques simples. Ils auront bien sûr à investir un minimum (voir paragraphe 2.3), et il est préférable qu'ils puissent s'appuyer sur un statisticien plus expérimenté. Le langage S est adapté à un traitement totalement interactif, c'est-à-dire où à chaque instant, non seulement un nouveau modèle peut être testé, mais où des outils d'origines diverses peuvent être enchainés, les résultats d'une étape servant de données pour l'étape suivante. Il a sa place dans tout institut, laboratoire ayant à traiter des données variées, et où pour chaque jeu de données se pose le problème de la meilleure méthode à utiliser. Il est alors possible de générer des applications évoluées enchainant des méthodes statistiques et graphiques, et de les mettre à disposition d'un public large grâce à la définition de menus. Enfin il a toute sa place dans les laboratoires de recherche développant des méthodes nouvelles, pour les raisons évoquées plus haut.

2.3 Documentation, Aide en ligne, Apprentissage,

2.3.1 Documentation et aide en ligne

La documentation de S est réunie dans un premier livre [1] comprenant un "tutorial" d'introduction à S, puis une présentation de ses différentes fonctionnalités, et se terminant par une description de l'ensemble des fonctions S. Ce livre en langue anglaise (dont la dernière partie a été traduite en français par l'INRA) est bien construit. Sa partie "tutorial" est adaptée à un public débutant, le reste s'adressant à un public un peu plus averti. Sa forme agréable pour une lecture en continu rend parfois difficile l'apprentissage d'une notion qui peut se trouver référencée à une dizaine d'endroits différents dans l'index du livre. La description des commandes

est claire, comporte les références des méthodes utilisées, mais elle est peut être insuffisante pour comprendre leur but L'aide en ligne reprend la description des commandes, et est donc très pratique pour contrôler une syntaxe. Un second livre [2] (le "red book" par opposition au "blue book" [1]) doit sortir début 1991 et d'autres sont en préparation. Enfin différents cours ont été écrits (dont un est disponible dans la bibliothèque STATLIB).

2.3.2 L'apprentissage

Il peut se faire à des niveaux très différents : la manipulation simple de données, la définition de fonctions élémentaires et la sortie des premiers graphiques peut se faire avec très peu d'investissements Mais la connaissance du langage en profondeur, qui permet de profiter de toute sa puissance et d'éviter parfois les lenteurs d'un langage interprété peut nécessiter plusieurs mois. L'apprentissage peut donc être graduel et se faire sur le tas après une mise initiale qui peut être minime

2.4 Eléments de comparaison avec d'autres logiciels statistiques

2.4.1 Logiciels généralistes et spécialisés

Il n'y a pas lieu de croire que S va rendre rapidement obsolète les logiciels généralistes ou spécialisés. En effet même si de nombreux chercheurs écrivent en langage S des fonctions spécialisées comme les fonctions GLIM déjà citées, ou encore comme les fonctions du New Environment for Statistical Inference (NESI) écrit par Shibata au Japon, de nombreux utilisateurs préfereront sans doute encore longtemps un logiciel qui les guide dans le cadre d'une pratique légèrement corsetée, mais bien établie Ce raisonnement reste valable pour des logiciels généralistes comme SYSTAT, GENSTAT, ou SAS. La raison tient dans le fait que le nombre de méthodes disponibles dans le domaine de la statistique inférentielle est nettement plus grand dans ces logiciels que dans S et SPLUS, mais aussi que, pour des traitements standards et répétitifs ces logiciels font bien l'affaire. On pourra trouver en [7] une comparaison assez détaillée de SAS et S qui conclue à leur complémentarité

On a vu que S peut servir de base dans laquelle les logiciels spécialisés pourront coexister. Si le langage S, grâce à son interactivité et à son ouverture, est utilisé comme environnement de base, on peut penser que l'appel aux logiciels spécialisés ne se fera progressivement que pour des tâches toujours plus pointues, au fur et à mesure de l'extension des fonctionnalités de S et SPLUS

2.4.2 S et les langages matriciels (APL, Gauss, IML de SAS, Matlab,...)

Le manque de pratique directe de ces différents langages par l'auteur ne rend pas possible une véritable comparaison. On essaye de donner dans la suite quelques éléments objectifs APL comme S est un langage interprété. APL est puissant, mais peu lisible et donc difficile à faire partager. Gauss est à la fois interprété et compilé, ce qui lui confère des avantages de rapidité. Il est, de plus, disponible sur des matériels moins sophistiqués que ceux qui supportent S. Par contre, le langage S semble bien supérieur à Gauss d'une part pour la puissance de ses fonctionnalités et l'environnement global qui l'accompagne et qui est décrit dans cet article, et d'autre part pour certains points de détail comme la possibilité de traiter des tableaux d'un nombre quelconque de dimensions. La facilité d'écriture de fonctions en S, la souplesse d'utilisation semblent marquer dans ce domaine une nette supériorité sur IML (voir [7] déjà cité). Enfin MATLAB a beaucoup de qualités mais est un langage mathématique et non pas statistique.

2.4.3 Quelques défauts de S

Les défauts du langage S que l'on rencontre dans les réactions des utilisateurs (le forum S ne connait pas de censeurs) sont les suivants. Le premier est lié au caractère interprété du langage S, qui peut le rendre lent dans certaines procédures réentrantes (le langage est récursif, mais il ne faut pas en abuser!), dans certaines procédures de simulation ou pour certains algorithmes. Ces critiques sont relatives à la vitesse des machines, qui a beaucoup évoluée depuis les premières stations de travail. Dans l'attente de véritable "benchmarks", allons-y d'un peu de subjectivité. Pour 98% des traitements, S a des réponses quasi immédiates. Pour les 2% restant, il est possible avec une station de travail d'ouvrir une autre fenêtre et de travailler à une autre tâche. Le ratio temps gagné à ne pas compiler/temps perdu à l'exécution dépend du type d'utilisation du langage. Il est sûrement très favorable pour les utilisateurs qui ont besoin d'une bonne adaptabilité des procédures statistiques. Cependant, il est important d'en tenir compte dans la programmation. Par exemple, il est possible de faire des boucles dans les programmes S. Ces boucles sont coûteuses en temps calcul, puisque les fonctions de la boucle sont interprétées à chaque pas, mais il y a presque toujours moyen de les éviter par des fonctions plus évoluées du type `apply`, figurant dans la boîte à outils de base, et écrites en C ou Fortran. Par contre la taille des données a une influence relative faible sur la rapidité d'exécution des fonctions de base. Cela signifie que la part du temps d'interprètation ou de chargement des commandes est souvent importante relativement à celle du temps de calcul. Quant aux performances de SPLUS sur (vrai) 386, une courte expérience ne permet pas de porter un jugement définitif. Mais elles paraissent tout à fait honorables. Certaines opérations sont lentes lors d'une première exécution, et certaines autres ont des performances analogues à celles de stations basées sur des 68030.

Un autre défaut qui a été signalé est l'importance de la place mémoire requise par S dans certaines applications : S aurait tendance à encombrer progressivement l'espace mémoire, ce qui rendrait nécessaire de temps à autre une purge, obtenue par exemple en quittant S (est-ce un bug de certaines versions?).

Enfin pour terminer, signalons que les procédures de lecture de données sont élémentaires en comparaison de celles proposées par SAS ou SPSS. Pour y pallier, des procédures d'utilisateurs sont proposées sur le marché et dans la bibliothèque STATLIB.

2.5 S et l'environnement informatique

2.5.1 S dans le monde UNIX

Le langage S ayant été conçu par les Bell Laboratories, ceux-là même qui sont à l'origine du langage C et d'UNIX, on peut s'attendre à une bonne insertion de S dans l'environnement UNIX. Effectivement, de nombreuses commandes S sont les mêmes que celles de UNIX (`ls`, `rm`, `cat`, ...), et il est possible de lancer des commandes UNIX depuis S, y compris de l'intérieur des fonctions. Les éditeurs `vi` et `emacs` du monde UNIX sont disponibles pour éditer les fonctions ou les objets de l'intérieur de S, et de nombreuses fonctions d'aide à la programmation sont disponibles (cf. paragraphe 3.1.6).

2.6 Le chargement des fonctions Fortran ou C

Il est possible de charger dans des fonctions S dynamiquement (pendant la session) ou statiquement (une fois pour toutes) des programmes écrits en C ou Fortran. Une contrainte cependant, S impose de transformer les programmes en sous-programmes dont les paramètres formels servent d'entrées-sorties exclusives: à l'utilisation, les entrées deviennent des arguments de la fonction S, et les sorties sont transformées en un objet S, la "valeur" de la fonction. L'intérêt du chargement de programmes C ou Fortran sous forme de fonctions S tient au fait que cela peut

être un moyen efficace de contourner la lenteur d'une fonction S particulière, sans perdre pour autant les avantages de l'environnement S.

2.6.1 Les drivers graphiques

S possède des drivers graphiques permettant la visualisation des graphiques sur différents types d'écrans (vga, hp, tek, sun, x11, printer), le dernier étant réservé aux écrans ASCII ordinaires. L'impression des graphiques peut être obtenue sur des imprimantes variées (postscript, hp, tty).

2.6.2 Des interfaces écrits en S

De nombreux interfaces ont été écrits entre S et d'autres logiciels. L'INRA (voir [3]) en a écrit ou va en écrire un certain nombre (pour MODLI, NL, Oracle, ...). Dans STATLIB, on trouve aussi un tableur intégré dans S, un interface avec IMSL, un interface partiel avec SAS. Si le chargement de programmes MODULAD entiers sous forme de fonction S peut être rendu difficile par la nécessité de supprimer les entrées/sorties propres des programmes, on peut procéder à une semi-intégration : une fonction S génère à partir des données et de menus éventuels les fichiers paramètres de MODULAD, le programme est exécuté normalement, puis une autre fonction S relit les fichiers résultats pour les intégrer dans l'environnement S. Le tout se réalise très simplement. Enfin, signalons le travail de O. Nicole (encore de l'INRA!) [6] qui permet sous UNIX d'intégrer des graphiques S dans un texte LaTeX.

2.7 S et SPLUS

L'environnement scientifique de S a été largement évoqué dans ce qui précède. Les BELL Labs, le réseau New-S dont les travaux sont réunis dans la bibliothèque STATLIB et les personnes travaillant autour de STATSCI (à Seattle) ou autour de ACE (en Australie) représentent un important potentiel autour du langage S. On revient ici sur les rôles respectifs joués par les Bell Labs, STATLIB et STATSCI. Les Bells sont les concepteurs et réalisateurs du noyau (le langage S). Ils diffusent le source de S, participent largement au débats du forum, et continuent leurs recherches de base : une nouvelle version de S devrait sortir bientôt, dans laquelle des modèles très généraux pourront être définis et ajustés. De même, des structures de données plus variées seront disponibles et la philosophie "orienté objet" sera plus poussée. Parallèlement, un livre de J. Chambers, T. Hastie et collaborateurs [2], qui développe cette approche, devrait sortir dès le début de 91. La bibliothèque STATLIB contient des archives (une quarantaine actuellement) contenant les contributions des uns et des autres, et elle est accessible par téléchargement. Ces contributions sont de qualité, même si elles ne sont pas contrôlées. L'implémentation de certaines nécessite des recompilations partielles de S. D'autres sont écrites en S et s'intégrent immédiatement. Enfin, STATSCI effectue ses propres développements (comme la mise sous DOS), insère dans SPLUS des apports de STATLIB, et joue le rôle d'une société de diffusion : envoi de mise à jour, maintenance, conseil. Malgré son surcoût cette solution procure un certain confort en évitant des manipulations pas toujours aisées.

3 Introduction au langage S

On donne ici, au risque de certaines répétitions avec le paragraphe précédent, une approche concrète du langage. Il ne s'agit pas d'un premier cours d'initiation, mais plutôt d'une présentation de ses spécificités et de ses différentes fonctionnalités, pour mettre en valeur la philosophie générale du langage et son domaine d'application.

3.1 Aspects informatiques

3.1.1 Généralités

S est un langage **fonctionnel** : toute commande est une fonction comportant un nombre quelconque d'arguments et produisant une valeur. Il est partiellement **orienté objet** : différents objets, vecteurs, matrices, tableaux, séries chronologiques ont à la base la structure de vecteur, sur laquelle se greffent certains attributs. Ces différents objets héritent donc des propriétés des vecteurs, et une opération définie pour un vecteur pourra être effectuée sur ces objets. C'est un langage **interprété** : toute commande est exécutée immédiatement, ce qui est très agréable lors de la mise au point de programmes. Mais cela peut être cause d'une certaine lenteur dont nous avons déjà parlé.

3.1.2 Les objets S

La notion d'**objet** est à la base du langage S. Un objet peut être une fonction, un tableau de données, un graphique, une liste de résultats. Ces objets se répartissent en deux grandes classes, les objets dits **atomiques** et les objets dits **récursifs**, que nous identifierons pour simplifier aux **listes**, mais qui contiennent aussi les graphiques et les expressions. Un objet atomique comporte un noyau central, autour duquel "gravitent" des attributs (par exemple, une série chronologique est un vecteur muni d'un attribut `tsp` qui définit les temps ou dates de début et de fin, et le nombre d'observations dans l'unité de temps). Une liste est une collection d'objets dont certains peuvent être des listes. Elle permet donc de structurer des données et fonctions sous forme d'une hiérarchie, dont les feuilles sont des objets atomiques. Tout objet a au moins deux **attributs**, son **mode** et sa **longueur**. Pour connaitre le mode et la longueur de l'objet `m`, il suffit de taper :

```
>mode(m)
```

(nous désignerons par > le prompt de S) et

```
>length(m)
```

Le principaux modes des objets atomiques sont **NULL, logical, numeric, complex, character**. On pourra tester le mode d'un objet par des commandes du type :

```
>is.numeric(m)
```

qui prendra la valeur TRUE ou FALSE. Il sera aussi possible, lorsque cela a un sens, de forcer un objet à un certain mode :

```
>mn <- as.numeric(m)
```

où le signe `<-` est un signe d'affectation. Enfin on distingue différentes **classes** d'objets qui pourront être des classes de **vector, matrix, array, category, and time-series**. Un array est un tableau a plus de deux entrées, un objet de type category décrit une variable catégorielle, et un objet time-series est une série temporelle. Comme pour les modes, il existe pour chaque classe des fonctions qui testent la classe d'un objet (par exemple, la fonction `is.vector`) ou qui le transforment en un élément de cette classe (par exemple `as.vector`). Un troisième ensemble de fonctions est constitué de celles dont le nom est un type d'objet. Elles sont utilisés dans leurs définitions :

```
>m<-matrix(0,nrow=4,ncol=5)
```

Classes et modes ne sont pas de même nature : ainsi, on peut définir des objets de la classe `array` et du mode `character`.

3.1.3 Le langage matriciel

Comme la plupart des langages matriciels, S a une syntaxe assez proche de l'algèbre matricielle. Il est de plus adapté au traitemement de données. Par exemple, les éléments d'un vecteur ou les entrées d'une matrice ou d'un tableau peuvent avoir des noms, qui seront conservés après opérations. Ces noms sont des attributs du vecteur ou du tableau. Si nom est un vecteur contenant les noms des éléments du vecteur x, on pourra effectuer l'affectation :

```
>name(x) <- nom
```

Les opérateurs choisis dans la boite à outils sont très généraux et permettent toutes sortes de manipulations. Un chercheur a lançé un jour sur le réseau S-news la liste de 91 opérations qu'il jugeait être la base d'un bon langage matriciel. Celà allait du calcul de la valeur absolue des éléments d'une matrice au calcul d'une inverse généralisée, en passant par la suppression de lignes ou de colonnes dans une matrice, la transformation d'une matrice symétrique en un vecteur contenant le demi-matrice supérieure, la mise à zéro des éléments négatifs, ou la recherche des maxima par ligne. La réponse est arrivée le lendemain avec la même liste et les solutions en S. Voici les solutions proposées pour ces exemples :

- Valeur absolue élément par élément :

  ```
  >ax <- abs(x)
  ```

- Calcul d'une inverse généralisée ($AGA = A$):

  ```
  g.inverse<- function(x) {lsfit(x,int=F,diag(nrow(x)))$coef}
  ```

  Commentaire: La fonction s'utilise en écrivant `g.inverse(x)`. On utilise ici le programme de multirégression avec pour variable à expliquer les variables de la base canonique de R^n (n = `nrow(x)`, nombre de lignes de x) disposées en colonne dans la matrice `diag(nrow(x))`, et comme variables explicatives les variables situés en colonne dans x. L'option `int=F` effectue une régression sans ordonnée à l'origine. Enfin, le résultat de la régression est une liste et `list$coef` permet d'extraire l'argument `coef` de la liste `list`

- Suppression des lignes 2, 3 et 4 :

  ```
  >xmod <- x[-c(2,3,4), ]
  ```

  Commentaire: l'absence d'indice à droite indique que toutes les colonnes sont sélectionnés; `c()` est un collecteur qui transforme une suite d'éléments ou de vecteurs en un vecteur, et le signe moins signifie qu'on choisit l'ensemble d'indices complémentaire

- Sélection de la partie supérieure d'une matrice symétrique :

  ```
  >xs <- x[row(x)<=col(x)]
  ```

  `row(x)` (resp `col()`) est une matrice de même dimension que `x` dont les éléments sont les numéros des lignes (resp. colonnes), `(row(x)<=col(x))` est une matrice de booléens qui permet la sélection des éléments situés dans la partie diagonale supérieure de `x`.

- Mise à zéro des éléments négatifs de la matrice `x` :

  ```
  >x[x<0]<-0
  ```

- Recherche des maxima par ligne :

```
>maxlin <- apply(x,1,max)
```

Les instructions sont comprises de la manière la plus logique et la plus large qui soit : par exemple `diag(a)` représente soit la diagonale de a si a est carrée, soit la matrice diagonale ayant a pour diagonale si a est un vecteur, soit la matrice identité d'ordre a si a est entier. Enfin, comme nous l'avons vu sur certains exemples, le système d'indiçage est très puissant. On peut par exemple si age, sexe et revenu sont des variables, calculer la médiane des revenus des gens agés de moins de 25 ans et de sexe féminin en tapant :

```
>median(revenu[age<25 & sexe="F"])
```

3.1.4 Les fonctions S

On en a déjà vu une au paragraphe précédent. Pour donner une idée de leur écriture, on présente ici quelques exemples simples, mais qui utilisent des notions déjà évoluées

Pour introduire une notion de pondération dans le calcul d'une moyenne, on peut écrire la fonction :

```
meanp<-
function(x,p)    x %*% diag(p) %*% rep(1,length(x))
```

Commentaires: x et p sont des vecteurs de longueur n, x étant la variable et p le vecteur des pondérations. La fonction `length()` donne le nombre d'éléments d'un vecteur ou d'un tableau. La multiplication matricielle est notée `%*%` et la fonction `rep(a,b)` répète b fois le vecteur a si b est un entier, et répète `b[i]` fois l'élément `a[i]` si b est un vecteur de même longueur que a. ($\forall i = 1; \quad , n$). On calcule une moyenne pondérée en écrivant `meanp(x,p)`.

Pour centrer une matrice x par colonnes, on peut écrire[3] :

```
center<-
function(x,p)
{
   moy <- apply(x, 2, meanp, p)
   x <- t(x) - moy
   t(x)
}
```

La fonction apply réalise le calcul des moyennes sur le deuxième indice (par colonne). On a ici un exemple de l'utilisation d'une fonction (meanp) comme argument d'une autre fonction (apply): les arguments de meanp, autres que l'argument principal, (ici le seul argument p) se mettent à la suite de la fonction. Pour éviter une boucle, on transpose x et le calcul de la différence entre la matrice transposé `t(x)` et le vecteur moy se fait en répétant autant de fois qu'il le faut ce vecteur. Le résultat d'une fonction est donné par sa dernière ligne.

La transmission des arguments peut se faire par rang ou par nom. Ils peuvent être manquants ou avoir des affectations par défaut. On pourra par exemple calculer la matrice de variance covariance ou de corrélation de variables figurant dans les colonnes de x par la fonction :

[3] On peut aussi écrire cela d'une manière plus classique en effectuant les centrages des colonnes dans une boucle.

```
"varcov"<-
function(x, d, cor = 1)
{
 if(!missing(d)) {
  x <- center(x, d)
  v <- t(x) %*% diag(d) %*% x
 }
 else {
  x <- scale(x,center=TRUE,scale=FALSE)
  v <- 1/nrow(x) * t(x) %*% x
 }
 if(cor == 1) {
  unsursig <- diag(sqrt(1/diag(v)))
  unsursig %*% v %*% unsursig
 }
 else v
}
```

La fonction `missing()` permet de tester la présence d'un argument et le signe ! est la négation logique. La fonction S `scale()` permet de "centrer" et/ou de "réduire" une matrice par colonne avec équipondération. On remarquera que la fonction `diag` a ici deux usages différents. Appliquée à la matrice carrée v, elle génère le vecteur de ses éléments diagonaux. Appliquée au vecteur `sqrt(1/diag(v)` elle génère la matrice diagonale ayant ce vecteur pour diagonale. Pour calculer une matrice de corrélation, on écrira `varcov(x)` ou `varcov(x,p)`. Pour calculer une matrice de covariance, on ajoutera l'argument `cor=0`.

3.1.5 La programmation

Elle consiste donc principalement à écrire des enchainements de fonctions. Les arguments sont de types quelconques, vecteurs, tableaux, listes, fonctions. On peut ainsi définir une fonction de bootstrap ayant comme argument les valeurs observées d'un échantillon, une fonction d'estimation, et le nombre d'échantillons à tirer. La structure de liste peut être très utile pour structurer les données, garder la mémoire des traitements utilisés, et stocker les résultats d'une suite de programmes. Son usage comme argument des programmes d'une chaine facilite la tâche de l'utilisateur. Il n'a pas à se soucier du type de données en entrée nécessaires à chaque étape. Chaque programme peut avoir comme seuls arguments un argument de type liste contenant les données et les résultats intermédiaires, et des arguments techniques facultatifs définissant des options et ayant des affectations par défaut (les options importantes peuvent aussi être définies par menu). On conserve le bénéfice de l'interactivité (donc d'un contrôle pas à pas de l'enchainement des opérations) sans alourdir le travail de l'opérateur. De plus, la possibilité de tester la présence d'un composant dans une liste (par `is.null(list$nom)`) permet de le guider : l'absence d'un composant nécessaire à une étape peut provoquer l'envoi d'un message l'invitant à réaliser d'abord cette étape.

L'usage des fonctions facilite une approche modulaire de la programmation. Les variables définies localement ne sont pas connues en dehors de la fonction, mais celles définies à l'extérieur peuvent être utilisées (mais pas modifiées, sauf signe d'affectation spécial) de l'intérieur d'une fonction. Enfin, pour ceux qui préfèrent les visites guidées, il est très facile de générer des menus en langage S. Un bref exemple en donne le principe :

```
 choix <-
function()
{
tex <- c("Voulez-vous ceci ?", "Voulez-vous cela ?", "Ou encore cela ?")
index <- menu(tex)
switch(index,
    cat("Voici\n"),
    cat("Voila\n"),
    cat("Trop, c'est trop!\n"))
}
```

L'utilisation de cette fonction est la suivante :

```
>choix()
1: Voulez-vous ceci ?
2: Voulez-vous cela ?
3: Ou encore cela ?
>3
Trop, c'est trop!
```

Ces menus peuvent être implémentés sous forme de menus déroulants si la console graphique le supporte.

3.1.6 L'aide à la programmation

Sans entrer dans les détails, signalons la présence d'un metteur au point (la fonction `browser()` littéralement feuilleteur), qui permet d'arrêter le programme à des points d'arrêts et de contrôler la valeur des variables, la fonction `trace` qui permet de suivre les appels de certaines fonctions, et d'autres fonctions comme un debugger, une aide en ligne très complète (la fonction `help`) et d'une aide abrégée (la fonction `args`) Un programme `Audit` permet de suivre une session à partir d'une deuxième fenêtre (donc avec une station de travail), de relancer des commandes après modifications. Une commande `history` et `again` donne cette même facilité mais sans possibilité de modifications. Enfin, plusieurs commandes permettent de contrôler l'utilisation de la mémoire par S ou le temps passé pour effectuer une commande,

3.1.7 Le rangement des objets S

Le langage S utilise par défaut, pour ranger les objets, un répertoire `.Data` qu'il crée dans la racine s'il ne le trouve pas dans le répertoire de travail et utilise une liste de recherche qui lui permet d'accéder aux fonctions et données système. Il peut aussi utiliser d'autres répertoires, grâce à une fonction d'attachement qui permet de modifier la liste de recherche, de changer de répertoire de travail. Il est de plus possible de créer des bibliothèques de fonctions (avec la fonction `.Dictionnary(..)`).

3.1.8 Le graphique

Le langage S produisant des graphiques indépendants du matériel, il est nécessaire, avant tout graphique S, de spécifier le type d'appareil sur lequel le graphique doit être effectué. Les fonctions graphiques de S peuvent se classer ainsi :

- Les fonctions de haut niveau, effectuant des graphiques sophistiqués avec des commandes simples, et qui par défaut génèrent un nouveau graphique à chaque appel

- Des fonctions de bas niveau, permettant à l'utilisateur de varier à l'infini ses graphiques, en les contrôlant dans tous leurs détails. Par défaut elles complètent un graphique préexistant.
- Des fonctions interactives qui permettent de lire ou de modifier un graphique en utilisant la souris

Ces différentes fonctions seront souvent combinées : on utilise une commande de haut niveau pour calculer les échelles, effectuer la mise en page, et éventuellement effectuer un graphique de base, puis on ajoute un texte, une courbe, ou un autre graphique en surimpression.

- Les fonctions de haut niveau sont au nombre de 27 (37 pour Splus), depuis la fonction `plot`, `hist`, les faces de Tchernov, le dessin des arbres hiérarchiques, les graphes en étoiles, les qqplots (quantiles contre quantiles) et les qqnorm (quantiles contre quantiles de la loi normale), l'écriture de symboles variés sur un plan (cercles, carrés, rectangles, étoiles, thermomètres, "boxplots" ou boîtes à pattes ?) permettant de visualiser les valeurs de 1 à 5 variables simultanément, des fonctions d'enveloppe convexe, de courbes de niveau, de surfaces en perspectives, de préparation de transparents. Il existe aussi des fonctions permettant de tracer la carte de certaine pays (CEE, Japon, Etats-Unis), et de représenter des symboles sur cette carte, une ou des fonctions permettant de représenter les différents composants obtenus par décomposition d'une série chronologique
- Les fonctions de bas niveau sont au nombre de 19 (dans S), tracé de points, de polygones, de vecteurs situés dans les colonnes d'une matrice, de textes, de flèches, de légendes, de hachures et peintures diverses.
- Les commandes interactives sont principalement la possibilité de récupérer les coordonnées d'un point (pour éventuellement y ajouter un texte ou un graphique), et celle d'identifier un point. Une autre série de fonctions permet d'effectuer des calculs associés aux graphiques : estimation de densité, interpolation, lissage, fonctions spline, arbre de longueur minimum, etc. Enfin SPLUS propose une représentation tournante d'un graphique en 3 dimensions, avec un certain nombre d'outils permettant une analyse dynamique des liaisons entre 3 variables.

3.2 Aspects statistiques

Nous ne faisons ici que citer les principales méthodes implémentées.

3.2.1 Manipulations de données

Toutes les transformations de données sont possibles en utilisant S : transformations des variables par des fonctions, recodification en fournissant une table de valeurs possibles, rangement par ordre croissant, suppression des doublons, concaténations, discrétisation suivant des intervalles prédéfinis, suivant des intervalles ayant de "belles" bornes ou donnant des effectifs de classes égales, croisement de deux variables, etc.

3.2.2 Statistique exploratoire

Grâce à son graphique et aux nombreuses fonctions développées dans ce domaine, S est sans doute, avec le logiciel EDA ([4]), le plus important logiciel pour la statistique exploratoire. On a déjà passé en revue les différents graphiques possibles : citons encore la possibilité de tracer des camemberts et histogrammes variés, la fonction `pairs` qui donne une matrice de diagramme de dispersion, les boites à pattes parallèles, la possibilité d'identifier les données aberrantes dans un diagramme de dispersion, la fonction `star` utile pour des diagrammes d'évolution et les fonctions robustes citées plus loin. Le "livre bleu" [1] contient un bon "tutorial" d'analyse exploratoire

3.2.3 Analyse des Données Multidimensionnelles

Régression et méthodes robustes Les langages S et SPLUS proposent de nombreuses méthodes de régression : deux méthodes linéaires (aux moindres carrés et régression L1), une méthode robuste par repondération itérative, des méthodes non paramétriques, une méthode de régression multiple "à pas de géant" (by leaps and bounds). STATLIB et SPLUS en ajoutent de nombreuses autres dont une méthode par directions révélatrices, le modèle linéaire généralisé par maximum de vraisemblance, etc. Rappelons que l'aspect modélisation sera fortement développé dans la prochaine version de S, ainsi que dans le "red book" [2] à paraître prochainement. Enfin de nombreuses autres méthodes robustes (lissages, estimations) sont proposées par S et par SPLUS.

Analyse multivariée Le langage S propose comme outils factoriels de base l'analyse en composantes principales, l'analyse canonique, l'analyse dicriminante, et une méthode de positionnement multidimensionnel (les coordonnées principales). D'autres méthodes développées par l'INRA (correspondances simples ou multiples en particulier) devraient être disponibles prochainement. Signalons que l'environnement S est très utile pour les aides à l'interprétation. Il est ainsi possible en deux commandes de dessiner les courbes de niveau d'une variable sur un plan factoriel, ou d'en faire une représentation spatiale. En classification automatique, S propose une fonction de classification hiérarchique (lien minimum, maximum et moyen), SPLUS ajoute une fonction kmeans) De nombreux outils d'accompagnement sont proposés pour les méthodes hiérarchiques (coupure d'arbre, dessin d'arbre hiérarchique), et l'aide à l'interprètation que représente par exemple les boites à pattes parallèles est supérieure à la représentation usuelle des moyennes et écart-types par classe. Enfin une fonction d'analyse loglinéaire est disponible pour les tables de contingence à entrées multiples, dans laquelle les zéros structurels sont prévus

3.2.4 La simulation et les générateurs de nombres aléatoires

S propose des générateurs de nombres aléatoires pour une douzaine de lois de probabilités et un générateur aléatoire de permutations de données ou de sous-échantillons avec ou sans remise. Le langage S est très pratique pour la simulation à condition que la rapidité de la machine soit suffisante.

3.2.5 Séries chronologiques

Il existe des méthodes robustes de décomposition saisonnière, avec des fonctions graphiques associées. S propose aussi une commande d'aggrégation, un opérateur retard, un autre de différenciation, une fonction de transformée rapide de Fourrier, la représentation de séries multiples, lissage par aggrégation, etc. SPLUS en propose une vingtaine d'autres, incluant le modèle ARIMA, différents algorithmes d'estimation de modèles autorégressifs, etc

3.2.6 Modèles de survie

Des fonctions sont disponibles dans STATLIB et dans SPLUS. On trouve dans SPLUS la méthode de régression de COX, le calcul et la représentation de courbes de survie pour des données censurées, et le calcul de la différence entre courbes.

4 En guise de conclusion

Il est difficile de ne pas laisser percer un certain enthousiasme quand on présente un tel langage. Lorsque l'on voit l'énergie dépensée dans les universités, mais aussi dans l'industrie,

pour réécrire des procédures déjà mille fois écrites, dans le but de mieux les maîtriser, de les personnaliser, ou tout simplement de les utiliser, on peut penser que S, en tant que *système d'accueil* (cf. [6]) a l'avenir devant lui. Les utilisateurs jugeront.

Références

[1] R.A. Becker, J.M. Chambers, A.R. Wilks *1988 The New S Language, A Programming Environment for Data Analysis and Graphics, Wadsworth & Brooks/Cole, Computer Science Series*

[2] John C. Chambers, Trevor Hastie, *à paraître début 1991 Statistical Models in S, John Wiley*

[3] G.Fayet *1991 dans ce même numéro de la revue de MODULAD*

[4] Eugène Horber *Exploratory Data Analysis, Université de Genève*

[5] Christian Leger *1990 Bibliothèque STATLIB, LINREG : 7 fonctions usefull for diagnostics, residual plotting, Box and Cox transformations, and an Anova table*

[6] Olivier Nicole *1989 Un pilote graphique entre le logiciel statistique S et PiCTeX, Cahiers GUTemberg, 3, 1989 pp.21-31*

[7] Terry M. Therneau *1990 "SPLUS" The American Statistician, August 1990, Vol 44, No. 3*

[8] John W. Tukey *1977 Exploratory Data Analysis, Addison-Wesley, Reading, Massachusetts*