Statistique et analyse des données

J. A. Nelder

Les qualités souhaitables dans des systèmes statistiques en se servant de GENSTAT comme référence

Statistique et analyse des données, tome 5, n° 3 (1980), p. 17-27

<http://www.numdam.org/item?id=SAD_1980__5_3_17_0>

Statistique et Analyse des Données
3 - 1980 pp. 17-27

LES QUALITES SOUHAITABLES DANS DES SYSTEMES STATISTIQUES EN SE SERVANT DE GENSTAT COMME REFERENCE.

J. A. NELDER
Rothamsted Experimental Station
Harpenden (Herts)

INTRODUCTION.

Dans cet article je considère la structure des systèmes statistiques sous quatre rubriques, à savoir :

(i) la manipulation des données,
(ii) les algorithmes,
(iii) le langage de contrôle, et
(iv) la communication avec d'autres programmes.

Les illustrations proviennent essentiellement de GENSTAT et de GLIM, deux systèmes dans lesquels j'ai été personnellement impliqué, mais la discussion des points essentiels est entièrement générale. En particulier les systèmes statistiques sont reliés aux langages à usages généraux qui existent, aux bibliothèques de sous-programmes et aux systèmes d'exploitation d'usage courant.

LA MANIPULATION DES DONNEES.

Il est clair que les problèmes qu'a à affronter l'utilisateur statisticien d'un ordinateur sont pour le moins autant ceux de la manipulation de ses données que ceux de leur analyse à l'aide d'algorithmes. Ainsi, par exemple, avant d'attaquer une analyse de régression, il faut rassembler les données, les vérifier et faire des éditions. Enfin après avoir fait la régression, il peut être souhaitable de conserver la sortie comme exemples de structures de données, et avec les données initiales de les écrire sur un fichier disque pour une utilisation ultérieure.

Les langages usuels des années 60, comme Fortran, ne facilitaient même pas la manipulation de structures aussi simples que les structures vectorielles. Ainsi, même si la signification de

$$A = B + C$$

est évidente quand, A, B, C sont tous des vecteurs de type réel et de même longueur, il est

néanmoins nécessaire d'écrire en Fortran

```
      DO 1 I=1,N
      A(I) = B(I) + C(I)
    1 CONTINUE
```

de façon à obtenir le résultat voulu. Notez l'introduction explicite d'un indice I qui n'a d'autre usage que de compter les éléments des vecteurs. Les logiciels statistiques introduisent l'arithmétique vectorielle pour faciliter le calcul de variables qui en découlent, et quelques uns, comme GENSTAT, vont plus loin en introduisant une arithmétique matricielle et une arithmétique de tableau. Ce faisant, ils étendent la définition d'opérateurs, tels que + ou *, à une classe plus large d'opérandes. Une autre caractéristique introduite par les systèmes statistiques a été la représentation de données manquantes ou sans valeur, et l'extension des opérateurs arithmétiques et logiques pour les accepter comme opérandes. A l'exception des opérateurs de relation égal ou non égal, toutes les opérations sur une ou plusieurs opérandes manquantes fournissent une valeur manquante.

Manipulation de données et structures de données.

Une aide puissante pour simplifier les problèmes de manipulation de données existe dans la possibilité de disposer de structures de données adéquates dans un langage de calcul. Dans des langages du niveau de Fortran ou d'Algol 60 les structures de données sont limitées aux scalaires et aux tableaux multidimensionnels de types variés (entier, réel, logique, etc...). Un grand pas en avant a été fait avec les langages définissant des structures comme Algol 68, où le programmeur peut définir ses propres structures, grâce à des composantes de base prédéfinies, et où il peut déclarer (par nom) un nombre arbitraire de cas. Ainsi, en Algol 68, il peut définir la structure complexe par

```
struct compl = (real re, im)
```

où re et im sont des sélecteurs de champ, permettant au programmeur de se référer aux éléments de la structure.
Il faut alors définir des structures de nombres complexes par exemple

```
compl z
```

et se référer à ses éléments, par exemple

```
x:= re de z
```

De plus, il peut alors définir des opérations sur les nombres complexes en définissant, par exemple + par

```
op+ = (compl a, b)
compl : (re de a + re de b, im de a + im de b)
```

Après ces définitions, il peut alors écrire l'ensemble de toutes les expressions arithmétiques sur les nombres complexes, parce que suivant les définitions, le compilateur "comprend" maintenant les nombres complexes.

Etant donné la puissance extraordinaire des langages définissant des structures comme Algol 68, on peut se demander pourquoi ils n'ont pas rendus caducs tous les systèmes statistiques. Une raison tient probablement à ce que leur très grande généralité n'a pas rendu claire leur puissance à un utilisateur occasionnel, qui se sent plus heureux avec un langage adapté à ses problèmes, et ceci même s'il peut être (et généralement il l'est) moins général et moins puissant. Une autre raison est qu'aucun de ces nouveaux langages n'a réussi à être avec succès quelque chose d'aussi uniformément disponible que Fortran et Cobol. Le plus courant, Pascal, a, dans sa forme standard, des limitations importantes comme langage de programmation utilisable pour un travail numérique complexe.
Probablement, l'exigence la plus importante pour manipuler facilement des données dans des systèmes statistiques, est que l'ensemble des structures de données admis par un système puisse être assez étendue pour permettre la conservation de la sortie des algorithmes aussi bien que son entrée. Quand ceci est permis la voie est ouverte à une approche progressive d'une analyse dans laquelle la sortie d'une étape devient l'entrée de la suivante. Sans cette souplesse le système n'est guère plus qu'un ensemble de "boîtes noires" prenant les données sous forme d'une matrice de données, réalisant quelques opérations sur elles et imprimant une sortie. Une réelle analyse progressive est difficile ou le plus souvent impossible. La capacité d'un système à accepter sa propre sortie accroît de façon importante la généralité des applications.

LES ALGORITHMES.

La seconde composante essentielle des systèmes statistiques, les facilités pour manipuler les données étant la première, est de disposer d'algorithmes pour réaliser des procédures variées nécessaires aux analyses statistiques.

Statistique et analyse numérique.

Les formes de ces algorithmes ont été grandement influencées par les résultats d'analyse numérique qui, ces 25 dernières années, ont mis au point de nouvelles techniques pour réduire les erreurs d'arrondi dans les opérations de base comme l'inversion de matrice ou le calcul des valeurs propres et des vecteurs propres. Néanmoins, on doit admettre que les objectifs des statisticiens en sélectionnant des algorithmes n'ont pas de raison de coïncider obligatoirement avec ceux des praticiens de l'analyse numérique. La solution des équations normales des moindres carrés illustre parfaitement cette remarque. Etant donné la matrice du dispositif $\underset{\sim}{X}$ et un vecteur d'observations $\underset{\sim}{y}$ on recherche le $\underset{\sim}{b}$ qui rend minimum

$$\| \underset{\sim}{y} - \underset{\sim}{X}\,\underset{\sim}{b} \|$$

Une méthode traditionnelle conduit à résoudre les équations normales

$$\underset{\sim}{X}'\underset{\sim}{X}\,\underset{\sim}{b} = \underset{\sim}{X}'\,\underset{\sim}{y}$$

en inversant la matrice d'information $\underset{\sim}{X}'\underset{\sim}{X}$ à l'aide de l'algorithme de Gauss-Jordan. Du point de vue de l'analyse numérique cette méthode est moins bonne que celle basée sur la

décomposition

$$\underset{\sim}{X} = \underset{\sim}{Q}\,\underset{\sim}{R}$$

où $\underset{\sim}{Q}$ est orthogonale et $\underset{\sim}{R}$ triangulaire supérieure. Néanmoins l'algorithme de Gauss-Jordan est extrêmement compact, facilement "mis à jour" pour ajouter ou supprimer des termes, et efficace numériquement. De plus quand la matrice est presque singulière (et le problème de précision est alors essentiel) la quantité d'information statistique sur le paramètre responsable du voisinage de la collinéarité est si faible qu'elle en devient inutile. Statistiquement, néanmoins, il est plus sensé de supprimer le terme correspondant du modèle ; ainsi, la cause de l'instabilité numérique est supprimée, et alors les considérations statistiques prennent le pas sur celles de l'analyse numérique. Parallèlement il est souvent préférable, pour augmenter la précision numérique de sélectionner les éléments servant de pivot durant une élimination gaussienne ; mais, alors, il se peut que la signification statistique de l'ajustement de certains termes soit sans intérêt dans l'ordre choisi par la stratégie du pivot, parce qu'un ordre initial des termes peut exister et qu'il est déterminé par la formulation statistique du problème. Ainsi il y a encore conflit entre les considérations statistiques et celles de l'analyse numérique .

Sortie.

Pour être vraiment utile dans un système statistique un algorithme ne peut pas se contenter de ne faire que les calculs. Il faut qu'il puisse :

(i) former des structures auxiliaires (par exemple le vecteur des valeurs ajustées dans une régression),

(ii) imprimer les résultats (y compris des structures auxiliaires si nécessaires) sous une forme convenable avec des en-têtes, des étiquettes etc...,et

(iii) permettre à l'utilisateur de conserver les résultats d'une manière interne dans des positions repérées par nom de structures appropriées de données. Fréquemment, la quantité d'instructions nécessaires pour ces trois caractéristiques est bien supérieure à celle nécessaire à l'algorithme ; ce fait est responsable, à mon avis, de ce que les bibliothèques de sous-programmes ne sont pas satisfaisantes, si on les compare aux systèmes, pour les analyses statistiques. Car, en ne fournissant que les algorithmes, la bibliothèque de sous-programmes laisse encore beaucoup de travail à faire à l'utilisateur. Dans Genstat l'impression est contrôlée par des ensembles de lettres qui représentent les différentes composantes demandées, et la sauvegarde des résultats l'est par des listes auxiliaires dans la directive.

Algorithmes à plusieurs niveaux.

Le danger, en ne fournissant des algorithmes qu'à un seul niveau, provient de ce que l'utilisateur se trouve limité par un petit nombre de procédures analytiques standardisées ; un tel système est bien trop rigide, il peut rendre impossible l'expérimentation, par un utilisateur entreprenant et aventureux, de variantes d'anciennes procédures ou d'essais de nouvelles idées. Dans un système souple, il est donc nécessaire de fournir des algorithmes (et des sorties qui leur sont associées) à plus d'un seul niveau ; ainsi Genstat possède une

directive pour l'analyse en composantes principales qui, partant d'une matrice de données $\underset{\sim}{X}$, construit $\underset{\sim}{X}'\underset{\sim}{X}$ (avec en option la possibilité de prendre la matrice de corrélation), en extrait un nombre de valeurs propres et de vecteurs propres, les plus petites ou les plus grandes, et conserve les résultats, en particulier (en option) les résidus et les valeurs des composantes dans des structures ad-hoc. Chacune de ces étapes peut être faite séparément dans Genstat, que ce soit sous la forme d'une fonction d'arithmétique matricielle ou grâce à une directive séparée (par exemple, pour les valeurs propres et les vecteurs propres). Des variantes de la procédure peuvent ainsi être définies en combinant les différentes opérations qui la composent. De façon analogue, bien que l'analyse des correspondances de Benzecri ne soit pas nommément fournie dans Genstat, il est excessivement simple de bâtir une macro qui la réalise, en employant comme noyau la directive de plus bas niveau SVD de décomposition en valeur singulière.
Les décisions sur les niveaux d'algorithmes à fournir dans un système sont difficiles à prendre ; elles dépendent des évaluations sur leur valeur, de la fréquence d'emploi des techniques existantes et de l'estimation des possibilités d'utilisation dans d'autres domaines. Eviter le plus possible la rigidité est donc capital.

LES LANGAGES DES SYSTEMES STATISTIQUES.

Les systèmes statistiques sont généralement munis d'un langage d'interprétation dans lequel l'utilisateur écrit ses introductions, et généralement ce langage est sous forme de directive ; c'est-à-dire que sa forme est semblable à celle d'un programme Fortran constitué par une séquence d'appels à des procédures variées. Souvent ces langages n'ont pas de phase de compilation, les instructions sont décodées et exécutées au fur et à mesure qu'on les rencontre. On peut en conserver le code pour plusieurs exécutions en employant des macros qui conservent le code sous forme de source plutôt que sous forme compilée. Il en découle que les langages d'interprétations sont plus lents à l'éxécution que les langages à compilation, et les macros sont plus lentes que les procédures compilées car elles doivent être décodées à chaque nouvel appel. Mais alors qu'est ce qui peut justifier l'emploi de tels langages si on les compare aux langages classiques à usage général ? Deux raisons semblent particulièrement importantes ; l'une et leur caractère autonome, associé à leur portabilité, et l'autre est leur souplesse dans la transmission des paramètres. Nous allons analyser ces deux raisons l'une après l'autre.

Langages d'interprétation et portabilité.

L'interpréteur pour un langage d'interprétation est écrit dans un langage classique à usage général que nous noterons désormais GPL (en pratique Fortran), et le système qu'il assure est alors portable sur n'importe quelle machine et n'importe quel système d'exploitation sous réserve que le GPL le soit. De cette manière Genstat est disponible sur onze machines différentes (y compris l'IRIS 80); cette assertion masque il est vrai de nombreuses

difficultés qui peuvent toujours survenir, elles sont dues aux différences entre Fortran, aux différences entre facilités pour faire des recouvrements dans de gros programmes et aux différences dues aux liaisons avec le système d'exploitation. Ces difficultés sont néanmoins mineures, si on les compare à celles rencontrées dans l'écriture de compilateurs d'un nouveau langage pour plusieurs machines différentes.
Une caractéristique utile des systèmes d'interprétation est qu'ils ne demandent qu'une seule étape de travail à l'éxécution. Ceci est à comparer aux systèmes utilisant un traducteur, comme CSMP par exemple, ces derniers demandent quatre étapes de travail pour chaque passage à savoir :

(i) traduction,
(ii) compilation du programme résultant,
(iii) édition de lien avec le reste du système et
(iv) éxécution.

Pour des travaux courts ces quatre étapes peuvent introduire un coût élevé pour chaque éxécution, elles demandent aussi une connaissance plus approfondie du système d'exploitation.

Passage des paramètres dans des langages d'interprétation.

Puisque les instructions individuelles dans la plupart des langages d'interprétations sont semblables aux appels de procédure dans les langages à usage général, les articles de telles instructions sont semblables aux paramètres dans les appels de procédure. Néanmoins, on peut rendre plus souples les règles de passage des paramètres dans les langages d'interprétation que dans les GPL, et ceci peut aider considérablement l'utilisateur. Les caractéristiques des langages d'interprétation sont :

1. la possibilité de passer une liste de longueur variable ; on ne rencontre cette possibilité en Fortran que pour les instructions READ et WRITE.

2. l'omission des paramètres non nécessaires. Supposons, par exemple, qu'une instruction de régression puisse définir en option un vecteur qui contienne les résidus ; si on ne se sert pas de ces derniers aucun paramètre ne doit être fourni. Dans les GPL un paramètre effectif doit être fourni, même s'il n'est pas employé.

3. la dénomination des paramètres. Si les paramètres peuvent être nommés on peut les disposer dans n'importe quel ordre pourvu que leur nom soit fourni. De nombreux utilisateurs trouvent plus facile de se rappeler des noms que de l'ordre des paramètres. De plus, si un petit sous-ensemble seulement des paramètres doit être utilisé l'emploi des noms rend inutile le décompte des séparateurs. Ainsi dans Genstat pour ajuster une régression linéaire sur X_1 et X_2 et pour conserver les valeurs ajustées dans FV, la forme sans nom est

```
'FIT' X1, X2 ; ; ; ; FV ; ; ; ; ;
```

alors que la forme avec nom est :

```
'FIT' X1, X2 ; FVAL = FV
```

4. le changement d'un sous ensemble de paramètres. Dans GLIM les paramètres effectifs d'une macro sont fournis par une instruction ARGUMENT, et ceci ne provoque pas l'appel de la macro. Des instructions ARGUMENT ultérieures peuvent modifier n'importe quel sous ensemble

de paramètres en ne changeant pas les autres.

Il est intéressant de relever que quelques uns des nouveaux GPL ont tendance à fournir ce type de souplesse. Néanmoins il semble que les langages d'interprétation adaptés à un type de problèmes (POL) continueront à exister à côté des GPL ; et un bon moyen pour tester la puissance d'un GPL pourrait être la facilité avec laquelle un programmeur peut définir et mettre en oeuvre un POL d'interprétation en utilisant un GPL comme langage de base. Les techniques "compilateur-compilateur" (Meta-traducteur) pourront être d'un grand secours, et elles pourront aider la mise en place de langages avec une syntaxe claire et non ambiguë.

LA COMMUNICATION AVEC D'AUTRES LANGAGES.

Il peut être important pour un utilisateur d'être capable de transmettre des données (et peut être les instructions qui leur sont associées) d'un système à un autre. S'il peut réaliser facilement cette opération, il n'a plus besoin de se reposer sur un seul système pour répondre à tous ses besoins ; simultanément ceux qui écrivent des systèmes n'ont nul besoin d'essayer de leur faire tout faire, ils peuvent concentrer leurs efforts sur certaines caractéristiques et essayer de les mettre au point de la meilleure manière.

Un désavantage de nombreux systèmes provient de la difficulté qu'il y a à prendre la sortie d'un système pour en faire l'entrée d'un autre programme sans une grande quantité d'édition intermédiaire. Par exemple, si un système imprime des tableaux avec en-têtes et étiquettes pour les facteurs qui les indexent, et encore plus s'il entoure les blocs de nombres de lignes horizontales et verticales, toutes ces excroissances doivent être supprimées si les valeurs du tableau doivent être utilisées comme entrée, par exemple pour ajuster des modèles log-linéaire à des tables de contingence. Une caractéristique essentielle d'un système souple devrait donc être sa capacité à extraire toutes les structures de données sous forme uniquement de valeurs, ces dernières n'étant séparées que par des espaces et des passages à la ligne.
Avec une telle caractéristique, il devient facile d'utiliser la sortie comme entrée de n'importe quel programme qui peut lire une suite de nombres en format libre. Par exemple, une sortie de ce type peut être lue directement par la directive READ de Genstat ou la directive d'entrée de données DINPUT de GLIM.

Transmission de données.

La transmission de données signifie seulement que l'information qui lui est associée, comme le nom d'identification des variables d'une matrice de données, ou les facteurs d'indéxation et leurs niveaux d'un tableau à plusieurs entrées, doit être transmise d'une autre manière. Habituellement cette information est contenue dans la tête de l'utilisateur qui la transmet au nouveau programme en l'incorporant dans les instructions du langage de ce programme. Comme exemple, considérons la transmission à GLIM d'une matrice de données de 30 unités et de 3 variables appelées X, Y et Z. Les données sont des sorties mises sur un

fichier disque de 90 nombres sous la forme

$$x_1 \quad y_1 \quad z_1$$
$$x_2 \quad y_2 \quad z_2$$
$$\text{----------}$$
$$x_{30} \quad y_{30} \quad z_{30}$$

L'utilisateur écrit alors les instructions GLIM

```
$UNITS 30
$DATA X Y Z
$DINPUT 16
```

où 16 est le numéro de l'unité logique assignée au fichier qui contient les données, et quand ces instructions sont éxécutées, ses données sont disponibles pour GLIM.

Transmission simultanée des données et des instructions.

Si la sortie du premier programme est suffisamment souple les données et les instructions peuvent être fusionnées pour que dans le dernier exemple le fichier de sortie ait la forme :

```
$UNITS 30
$DATA X Y Z
$READ
```

$$x_1 \quad y_1 \quad z_1$$
$$x_2 \quad y_2 \quad z_2$$
$$\text{----------}$$
$$x_{30} \quad y_{30} \quad z_{30}$$

```
$RETURN
```

Ce fichier peut alors être lu par l'unique instruction GLIM

```
$INPUT 16
```

En Genstat un tel fichier de sortie peut être fourni par des instructions telles que

```
'OUTPUT' 2
'CAPTION' ''
$UNITS 30
$DATA X Y Z
$READ
''
'PRINT' X, Y, Z
'CAPTION' '$RETURN'
```

Dans un langage le dispositif indispensable pour fournir une telle transmission simultanée de données et d'instructions dans un autre langage est relativement simple ; il demande simplement la possibilité de transmettre des chaînes de caractères séparées avec les valeurs de base des

structures de données appropriées. Tout système souple doit possèder une telle caractéristique.

Standardisation des structures de données.

Une simplification supplémentaire peut exister si on peut représenter de façon standard les structures de données statistiques de base. Par exemple, nous pouvons définir une matrice par un ensemble constitué par

identificateur	nom de la matrice
entier	numéro des unités
identificateurs	noms des variables
réels	valeurs des variables en parallèle

Tous les systèmes statistiques pourraient alors contenir la lecture et l'écriture de telles structures comme élément de leurs utilitaires de base. Notons que l'introduction des noms d'identificateur comme éléments pouvant être lus, plutôt que déclarés dans le programme, est un concept plutôt compliqué. Cette souplesse est disponible dans Genstat, mais pas dans GLIM. Je n'ai pas connaissance que ceci le soit dans n'importe quel autre système statistique.
Pour l'instant nous sommes encore loin d'une compatibilité dans la standardisation de la représentation des structures des données de base, aussi les techniques dont nous avons parlé plus haut doivent encore être utilisées.

L'influence du système d'exploitation.

Quand un utilisateur souhaite éxécuter un travail formé par une suite de programmes séparés il doit d'abord maîtriser le système d'exploitation et le langage de commande qui lui est associé. Cette nécessité constitue un puissant moyen dissuasif pour éxécuter de tels travaux ! Trop souvent les systèmes d'exploitation rendent la construction de suites de travaux beaucoup plus compliquée qu'elle ne devrait être. Le système d'exploitation UNIX, mis au point par les Laboratoires de la Bell Telephone, est un exemple particulièrement brillant des commodités de réalisation d'une telle suite. Le langage de commande emploie la barre verticale comme symbole de concaténation de travaux ; elle implique que le fichier de sortie d'une étape constitue le fichier d'entrée de la suivante. Ces fichiers intermédiaires n'ont pas à être nommés de façon explicite. Ainsi le noyau d'une suite de travaux est tout simplement

programme 1 | programme 2 | programme 3 | ...

L'existence d'un langage de contrôle de ce type est la meilleure façon d'inciter à écrire des programmes modulaires où les interfaces communes sont le format des fichiers qui les relient. Je pense, sans grand risque de me tromper, que l'existence actuelle de systèmes importants et autonomes comme Genstat, est le reflet direct du peu d'attrait de nombreux systèmes d'exploitation, qui rendent difficile la tâche de l'utilisateur pour mettre au point des suites de travaux formées par de petits modules.

Une caractéristique nécessaire du système d'exploitation pour utiliser de façon souple des programmes différents réside dans la possibilité d'interrompre un programme pour en éxécuter un autre, puis de retourner au programme original. Ainsi, dans mon utilisation personnelle interactive de GLIM, j'interromps fréquemment son éxécution pour appeler le système éditeur et modifier un fichier d'entrée en cours de lecture par GLIM (par exemple pour modifier une macro, ou modifier quelques valeurs des données). Tous les systèmes d'exploitation ne permettent pas ces opérations, et les utilisateurs de ces systèmes souhaitent que GLIM contienne ces facilités d'édition de façon à pouvoir entièrement travailler sous GLIM. Ceci est un bon exemple du fait qu'un manque de souplesse d'un système d'exploitation décourage d'écrire des programmes modulaires.

S et GLIM, un exemple de coordination de programme.

S est un système mis au point par le Dr. J.M. CHAMBERS des Laboratoires de la Bell Telephone à Murray Hill, New Jersey. Il possède des facilités très élaborées pour manipuler des données et il fonctionne avec un puissant langage de commande. Bien qu'il puisse faire une régression multiple, il n'a pas, comme GLIM, la possibilité de traiter des modèles faisant intervenir des termes qualitatifs (facteurs) et de résoudre des modèles linéaires généralisés, qui font intervenir des distributions non gaussiennes et des transformations du prédicteur linéaire (fonctions de lien). Il semble par conséquent, que donner à l'utilisateur de S un accès à toutes les facilités de GLIM est utile ; avec la possibilité supplémentaire de ne pas être obligé d'apprendre le langage GLIM sauf les conventions nécessaires pour spécifier un modèle.

La méthode est la suivante : l'utilisateur de S n'aurait qu'à appeler une procédure de S comme useglim, dont la liste de paramètres contiendrait l'information nécessaire pour ajuster un modèle linéaire généralisé (NELDER et WEDDERBURN, 1972). Celle-ci contient la matrice de données, le nom de la variable dépendante (y), la distribution de l'erreur, la fonction de lien et la formule pour le prédicteur linéaire. La procédure useglim n'aurait alors qu'à construire le programme GLIM adapté à ce problème, en utilisant ses facilités de manipulation de caractères, et demanderait au système d'exploitation d'interrompre l'éxécution en cours de S pour éxécuter le programme GLIM. Ce programme GLIM contiendrait un appel à une macro, qui devrait alors fournir à la sortie de GLIM une structure telle qu'elle devienne une représentation externe acceptable de la structure des données dans S capable de prendre la sortie provenant de l'ajustement au modèle linéaire généralisé. La procédure de S useglim devrait détecter la fin du travail de GLIM et rendre la sortie sur fichier lisible par S. L'utilisateur aura ainsi accès aux résultats de l'ajustement sans avoir à s'occuper de l'intervention de GLIM.

Les deux caractéristiques essentielles qui rendent possible cet exercice sont :

(i) la possibilité des deux langages, S et GLIM, d'avoir une configuration de leur sortie acceptable comme entrée dans d'autre programme, et

(ii) un système d'exploitation qui permette à un programme d'appeler l'autre de façon dynamique.