数 値 解 析 24－1
（1988. 3. 11）

Sの概観

渋谷政昭　柴田里程
慶應義塾大学　理工学部

R. A. Becker および J. M. Chambers により開発された，データ解析とグラフィックスのための環境である"S"を利用者の立場より概観し，他の計算言語と比較して興味ある点を紹介する．

特に，たすさかに汎用計算言語を利用して問題解決に挑む，研究者・技術者・管理者にとって有用な，高水準の対話型演算式型言語であることを評価する．グラフィックスのための諸水準の関数，統計解析用の諸関数が豊富であるだけでなく，システムとして便利な環境を作り出している．

現システムの不満足な点も明記する．

S: An Overview

Masaaki SIBUYA and Ritei SHIBATA
Department of Mathematics, Keio University
Hiyoshi, Kohoku-ku, Yokohama 223

"S", an interactive environment for data analysis and graphics, developed by R. A. Becker and J. M. Chambers, Bell Laboratories, is overviewed from the user's viewpoint. Some interesting features are introduced by examples.

This is a nice high-level interactive expressional language useful for casual use by research workers. It offers a lot of graphics functions of high and low levels, and functions for statistical analysis. It provides users also with a comfortable computing environment.

Some unsatisfactory points of the present system are pointed out.

1. たまさかの計算機利用者

汎用計算言語の評価は，当然利用者の型によって異なるはずである．日常的な仕事として，3次元非定常非線形問題をスーパー・コンピュータにより計算しているとか，機械翻訳のための言語学研究を行っているとか，機械設計のために図形処理中心の計算を行っているとか，それぞれの立場から汎用言語を選び，その上に応用ソフトウェア・システムを購入，開発していることであろう．

他方，日常はあまり"計算"をしない．せいぜいパソコンで，ワープロ，簡易言語，ゲームを走らせるだけである．しかしたまさかに新しい課題に直面し，適当なソフトウェアも身近に見当たらず，手許の汎用言語で分析，解析，を試みる人も多いであろう．上記の日常専門利用者も，このような状況に面する機会は多いであろう．不特定の問題，あるいは問題を限定するためにも試行錯誤しなければならない状況，に面して悪戦苦闘することを"問題解決"と呼ぶ人たちもいるようである．

"たまさかの計算機利用者（casual users）"の問題解決（problem solving）"にとって必要な汎用計算言語は，何よりもプログラムしやすいものでなければならない．できたプログラムを何年も使うわけではない，プログラムし，出を取り，必要な最初の結果が出るまでの手間が短いことが重要である．また，結果を検討し人に伝えるためのグラフィックスも重要である．BASICが急速に普及した理由は，単純な問題解決に手軽であったからである．"S"はワークステーション／パーソナル・システムによる問題解決用の道具である．

J. W. Tukey（Princeton Univ. and Bell Labs.）は1970年代に探索的データ解析（Exploratory Data Analysis）の概念と方法を提唱した．極端に要約するならば，実験，観察，調査により，かなりのバラツキをもつ多量のデータを得たとき，これを種々の面から要約し，特にコンピューター・グラフィックスを活用してデータをいくつかの角度から眺め，データの特徴を浮き出させ，現象の法則性を発見しようという技法である．Tukey (1977)，Hoaglin et al. (1983) 参照．"S"は，探索的データ解析のための"環境"として，R. A. Becker and J. M. Chambers (1984) によって開発されたものである．

2. S環境の特徴

Sの特徴を要約すると次の通りである．

(1) 高水準の対話型汎用言語が主体である．
(2) UNIX環境を十分に活用している．
(3) グラフィックスの諸機能がS関数として充実している．
(4) データ解析，統計解析の諸方法がS関数として充実している．

順次，やや詳しく述べる．

(1.1) 自然とK. E. IversonのAPLと類似している点が多い．(IBM, 1984, 1985)
解釈型，関数型言語である．Sの式を入力すれば，付値式でない限り結果が表示される．1つの式の中で，中間値をいくつでも付値，表示できる．

配列を単位とした演算式である．変数についての宣言は一切ない．　四則演算，関数値の計算など，できるだけ配列を単位に記述することにより，反復計算の指定を少くする．変数の型，構造は代値のとき定まり，したがって変数，すなわちデータセット，は自己記述的（self describing）である．

リスト構造（Lisp などの）のようなデータ構造体を作れる．ただしAPL2のように一般的なものではない．典型的なデータ構造体は，S関数で作り出されS関数の引数となるが，ユーザが自由に新しい型のデータ構造体を作るための関数も用意されている．

データと式をまとめて保存できる．　APLのworkspace に相当することは，UNIX のディレクトリとしてSが管理する．

（1.2）　APLとは違う点も多い．

特殊文字は使用せず，左から右に実行する．　SはUNIXに基き，したがってASCII 文字だけを使用する．関数は普通の慣用のように文字を用いる．APLの特殊記号は式を短くするのに役立つが，初心者を近づけず，使用機器を制限する働きが強かった．右から左への実行も同様に働いた．

演算子は優先順位をもつ．　式の表現を慣用通りとすると，自然 FORTRAN のように，すべての演算子に優先順位（precedence）を与えることになる．代値は最低順位の演算である．面白いことに右向きの代値を許しており，結構役に立つ．

関数型であるが制御文を許す．　if then else, for, while, repeat などの制御文を許し，"演算式型言語 expressional language" と称している．ただし厳密型であるため実行は遅い．

文字単位でなく文字列単位である．　見易い出力を作るためには語を扱いたくなるが，文字単位のAPLではスペースをやたらに挿入した文字行列などを扱うことになる．それが不便でAPL IIとなった．文字列単位とすると逆に，文字処理（string manipulation）が扱い難い．

非常に多くの関数がある．　UNIX 環境の常として多くの関数が，あれば便利であるという観点で導入される．そのために全体が雑多となり，重要な関数を理解し記憶することが困難となる．APLではシステムの関数を精選して基本的なものに限り，それを活用することをユーザに強制するので熟達を要する

関数呼び出しが独特である．　多くの引数をもつ関数があるが，補助的なものはdefault 時の標準値ですませられる．また，引数のデータ構造に応じて関数の意味が変わることが多い．一種の polymorphism が採用されている．

ユーザ関数の定義法が多様である．　S式を書き並べたテキスト・ファイルを端末からの入力のように実行するのが，もっとも単純である．より柔軟なのが，メニューの利用，マクロ展開の利用である．効率の良い，システム関数同様のものを作るには，CまたはFORTRAN のサブルーチンを拡張機能で取り込む．

（2）　UNIXの諸機能の利用

SからUNIXコマンドは自由に実行できる．　Sのデータセット（変数）は2進表示であり，直接コマンドの対象とはなりえないが，入出力ファイル，S式のファイルなどテキスト・ファイルはすべてコマンドで処理できる．

エディタの利用．　S式の構文エラー，実行時エラー，マクロ展開の結果の構

文エラー，などのエラー発生時に，原因となった式がダンプされ，それを編集するS関数を呼べば，UNIXエディタが自動的に呼ばれ，編集を終えれば自動的に実行される．予め指定しておけば，複数個のエディターのうちの好みのものを選べる．

ディレクトリの管理．　S式，マクロ定義，データセット，などを1つのディレクトリーの下で作成し，それをS関数により取り込んだり切り離すことにより，適当な作業条件を作り，あるいは並同作業をすることができる．

help 機能．　S関数についてのhelp機能があるだけでなく，ユーザの作成したデータ，関数について，解説を作成し，それを on line documentation とするための道具が用意されている．

テキスト処理機能との結合．　UNIXコマンド tbl, pic によりSの出力をより美しい表や図表にできるだけでなく，nroff／troff と組み合わせて，文書中にS式を挿入しておき，その計算結果を文書中に埋め込むことができる．Becker and Chambers (1984) の本もこのようにして作られている．

（3）　Sのグラフィックス関数

多様な出力機器選定．　市場にある標準的なグラフィックス機器のドライバーが用意されており，その選択，初期値設定もS関数となっている．

高・低水準のグラフィックス関数．　グラフィックスといっても，科学文献における説明用図表の作成に限られている．高水準の関数を呼べば，詳細を指定しなくとも，かなり立派なグラフを作成してくれる．必要に応じて，期待通りの図を作成するには"図表パラメータ"と称する多くのパラメータを，default 以外の値に設定する．あるいは低水準の関数をいくつか組み合わせて，目的の図表を構成する．

副作用としての作図．　作図のためには必要な諸量をまず計算しなければならない．図表の細部を変更するときに，作図用諸量の変更は比較的少ない．そこで作図関数が値をもつようにし，その値を再び引数とすることにより効率をあげることができる．作図用諸量はやや複雑なデータ構造体となることが多い．

（4）　Sのデータ解析・統計解析関数

処理手続きよりは基礎的関数が用意されている．　統計パッケージとして，BMDP, SPSS, GENSTAT, Minitab, GLIM, SAS などが知られている．最近は IBM-PC, Macintosh 上のパッケージが数多く販売されている．Sではむしろ基礎関数だけを用意し，多くの手法はユーザが容易にS式で書けるようになっている．しかし，グラフィックスを利用するもの，多変量解析の関数は豊富である．

3．諸例題

Ex. 1　　ベクトル演算

C( )　は不定個の要素を結合 combine してベクトルを作る．

1：3　は 1, 2, 3　というベクトルを作る．

ベクトルとベクトルとの演算は成分ごとの演算である．横を矢印 ← で x に付値している．

```
>  Ex. 1
>
> log(6-x<-c(.5, 2, 2)* 1:3) ->y
Warning in log: missing values generated from range limits
> x
     0.5  4.0  6.0
> 6-x
     5.5  2.0  0.0
> y
     1.704748  0.693147        NA
> y+1
     2.704748  1.693147        NA
>
```

> $ の prompt
> <- -> 代値
> :

スカラーとベクトルの演算は，スカラーと各成分との演算である．
ベクトルの関数値も，各成分の関数値のベクトルである．
0の対数は欠足値 missing value NA (not available) である．
NA は数，文字列と同様に扱える．これの演算結果は常に NA である．

Ex. 2　データの型の強制変換 (coercing)

```
>
>  # Ex. 2
> TRUE
    T
> T+3
    4
> T&3
    T
> rep("ab", 3.14)  #repeat 3.14 times
     "ab"  "ab"  "ab"
>
```

T, F は TRUE, FALSE と同等である．数に変換すると，それぞれ 1と0になる．
逆に数は，非零はT，零はFとなる．
後述の関数 rep( ) は第2引数に自然数を要求する．

$では可能な限り，データを解釈して，演算を継続する．もちろん，そのために，エラーの生じた位置が分かりにくくなる危険，エラーを見逃がす危険はある．

Exs. 3 and 4　関数の呼び出し

```
>  # Ex. 3
> z <- c("a","bc")
> rep(z,2)  #repeat: rep(x, times, length)
     "a"  "bc"  "a"  "bc"
> rep(times=2, z)
     "a"  "bc"  "a"  "bc"
> rep(z, len=5)  #length is abbreviated
     "a"  "bc"  "a"  "bc"  "a"
> rep(z, c(3,2))   #when times is an integer vector
     "a"  "a"  "a"  "bc"  "bc"
>
>  # Ex. 4
> seq(0, 1, .4)  #sequence: seq(from, to, by, length)
     0.0  0.4  0.8
> seq(0, 1, len=4)
     0.0       0.3333333  0.6666667  1.000000
> seq(8, 5)
     8  7  6  5
> seq(pi, 5)
     3.141593  4.141593
> seq(4)
     1  2  3  4
> seq(z)
     1  2
>
```

関数 rep は3つの引数をもつが，第2，第3の引数はどちらかを指定する．

関数 seq も第3，第4の引数のどちらかを指定する．
両方省略すると，by = 1 または −1 とみなされる．: と同じ．
引数がスカラー1つ，ベクトル1つのときも適当に解釈する．

引数の数を増し，また引数に多様な型を許すことにより，1つの関数が非常に強力となる．それだけ，1つの関数の理解が困難となる．

Ex. 5　行列，データ構造体

```
>
> # Ex. 5
> mar <- matrix( c(NA, 1:31, rep(NA,3)), 5, 7, byrow=T)
> ?comment(mar, March 1988)
>

> print(mar, rowlab=c('1st','2nd','3rd','4th','5th'),  #continues
+   collab=c('M','Tu','W','Th','Sa','Su'))

Array:
5 by 7

      M  Tu   W  Th  Sa  Su   M
1st  NA   1   2   3   4   5   6
2nd   7   8   9  10  11  12  13
3rd  14  15  16  17  18  19  20
4th  21  22  23  24  25  26  27
5th  28  29  30  31  NA  NA  NA

 $comment
      "March 1988"

> mar  #usual matrix print

Array:
5 by 7

     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]   NA    1    2    3    4    5    6
[2,]    7    8    9   10   11   12   13
[3,]   14   15   16   17   18   19   20
[4,]   21   22   23   24   25   26   27
[5,]   28   29   30   31   NA   NA   NA

 $comment
      "March 1988"
>
>
> compname(mar)
      "Dim"       "Data"      "comment"
> mar$Dim
     5     7
> mar$Data
      NA    7   14   21   28    1    8   15   22   29    2   (
[19]  24   31    4   11   18   25   NA    5   12   19   26   NA
> mar$comment
      "March 1988"
>
```

[データ一部省略]

行列の作成．
コメントをつける
マクロの呼出し．

＋ は継続行の
prompt．

ラベルつきの行列
の印刷

通常の行列印刷

関数 matrix (x, nrow, ncol, byrow) で nrow 行, ncol 列 の行列を作る．x は要素となるベクトルで，x の長さが不足であれば反復して用いられる．x の成分が列ごとに行列を作る．行ごとに作るためには byrow = T とする．省略時の値は byrow = F である．

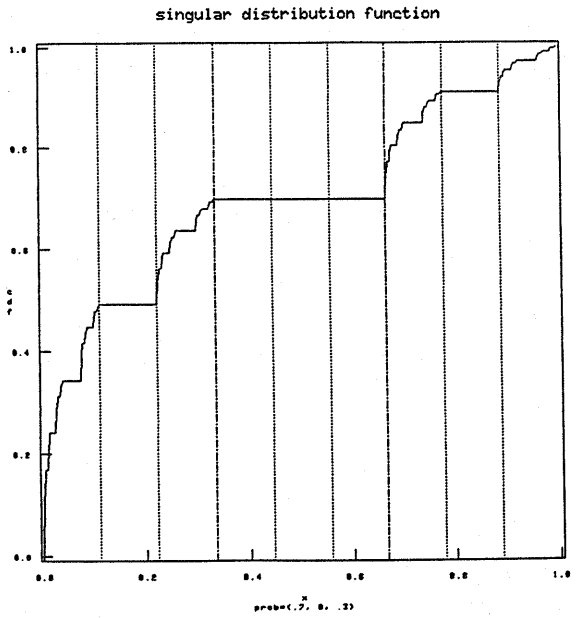
? はマクロを呼び出す．つまり，マクロ展開が行われる．展開結果を表示することもできる．? comment はデータセットに comment という成分を付加する，システム・マクロである．そのデータセットを表示すると comment も表示される．

行列はもっとも簡単なデータ構造体で Dim (dimension) という成分と，Data という成分をもっている．今の場合 comment も追加の成分として入っている．

compname ( ) component-name により データ構造体の成分の名前が分かる．各成分を取り出すには (データセット名)$(成分名) による．

データ構造体の成分にデータ構造体を用いることができ 階層的な構造が作れる．いわゆる リスト構造 (list structure) である．

Ex. 6　制御文，作図関数，マクロ



（フラクタルの例である．）

（特異分布（singular distribution）の分布関数を計算する．$(X_i)_{i=1}^{\infty}$ が独立で同一分布に従う確率変数で $X_i = 0, 1, 2$ となる確率が p_0, p_1, p_2（$p_j \geq 0$, $p_0+p_1+p_2 = 1$）であるとき，確率変数 X を

$$X = \sum_{i=1}^{\infty} X_i 3^{-i}, \quad 0 \leq X \leq 1$$

で定義する．

つまり，X を3進法で表示したとき小数第 i 桁が X_i である．X の分布関数を $F(x) = P[X < x]$ とする．以下3進法を用いる．

1°　$F(.0) = 0$, $F(.1) = p_0$,
$F(.2) = p_0 + p_1$

2°　$F(.jk) = F(.j) + p_j F(.k)$

一般に $d+1$ 桁目を求めるには

$F(.j k_1 \cdots k_d) = F(.j) + p_j F(.k_1 \cdots k_d)$

以上の反復を 3^d が十分に多くなるまで反復して作図する．$(p_0, p_1, p_2) = (.5, 0, .5)$ の場合が Cantor 関数である．上図は $(p_0, p_1, p_2) = (.7, 0, .3)$ の場合である．一般に k 進法表示で $0, 1, \cdots, k-1$ が等確率（X が $(0,1)$ 一様分布に従う）でない限り，$F(x)$ は連続，非減少でありながら，ほとんどすべての点で導関数が0である．

Ex. 6

```
p  <- c(.7, 0, .3)                          # probabilities
ff <- cp <- c(0, cumsum(p)[-len(p)])
lenf <- 300             # at most lenf function values
x  <- while( len(ff)<lenf )       # main computation
  ff <- rep(cp, rep(len(ff), len(cp))) + c(ff %o p)
x  <- c(0, seq(ff)/len(ff))             # add end points
cdf <- c(ff, 1)
par(pty="s")            # graphic parameter: square box
plot(x, cdf, type="l",
    main="singular distribution function")
abline(v= (1: (len(p)^2-1) )/len(p)^2, lty=2)
abline(v= (1: (len(p)-1) )/len(p), lty=3)
```

cumsum( ) はベクトルの累和 cumulative sum で今の場合 (.7, .7, 1) となる．

[ ] はベクトルの成分取り出し．[-3] だと第3成分以外．

ff, cp に (0, .7, .7) ができ，以下 ff に F の値を作る．

ff %o p はベクトル ff とベクトル p の外積 outer product で，ff[i]*p[j] を [i,j] 成分とする len(ff) × len(p) 行列である．c(ff %o p) で $p_j F(.k_1 \cdots k_d)$ ができる．

while (式1) 式2　で，式1 を満足する限り式2 を実行する．式2 が1つでなければ { } で囲って複合式とする．while 式は最後の計算が値になる．付値しなければ値が表示される．

plot (x, cdf, type = "l") はベクトル x の成分を x 座標，ベクトル y の成分を y 座標とする点を打点 plot する．type = "l" の指定で点を線で結ぶ．

abline (a, b) で切片 a，傾き b の直線を描く．v = ベクトル，h = ベクトルで，それぞれ垂直線，水平線を描く．lty で点線，鎖線 などを描く．

・ plot( ) の前の par(pty = "s") で作図領域を正方形 square にする plot type を指定している。これがないと画面いっぱいに横長に描く。

plot( ), abline( ) など作図関数には多くの引数があり、これにより ラベルのつけ方、目盛りの刻み、軸の取り方、凡例の記入、文字の大きさ、傾きなどを詳しく変更できる。より基本的な変更は グラフィック・パラメータを指定する関数 par( ) を用いる。

さて Ex. 6 のように長くなると、タイプミスも生じ、また後から修正もしたくなる。diary(T) と日録をセットしておくと、入力したものがすべて "diary" というテキスト・ファイルに保存される。

S式が完成したところで diary を編集して、きれいなS式とする。このファイルを source("diary") で呼ぶと、diary の中の式を順次に端末から入力することと同じとなる。

さらに使いやすくするためには、これをマクロとする。上のファイルを若干修正し、マクロ定義 define("ファイル名") により定義すると下記のようになる。
? singdf(c(.7, 0, .3)) とこのマクロを呼ぶと、マクロ本体中の第1の部分に c(.7, 0, .3) が文字列として挿入された上で実行される。
? singdf() と引数を省略すると // で囲まれた c(.5, 0, .5) が default 値として用いられる。

```
MACRO singdf(p/c(.5,0,.5)/)
ff <- cp <- c(0, cumsum($1)[-len($1)])
lenf <- 300          # at most lenf function values
x  <- while( len(ff)<lenf )       # main computation
  ff <- rep(cp, rep(len(ff), len(cp))) + c(ff %o $1)
x  <- c(0, seq(ff)/len(ff))          # add end points
cdf <- c(ff, 1)
par(pty="s")          # graphic parameter: square box
plot(x, cdf, type="l",
    main="singular distribution function",
    sub="prob= $1 ")
abline(v= (1: (len($1)^2-1) )/len($1)^2, lty=2)
abline(v= (1: (len($1)-1) )/len($1), lty=3)
END
```

MACRO マクロ名(引数)
　マクロ本体
END

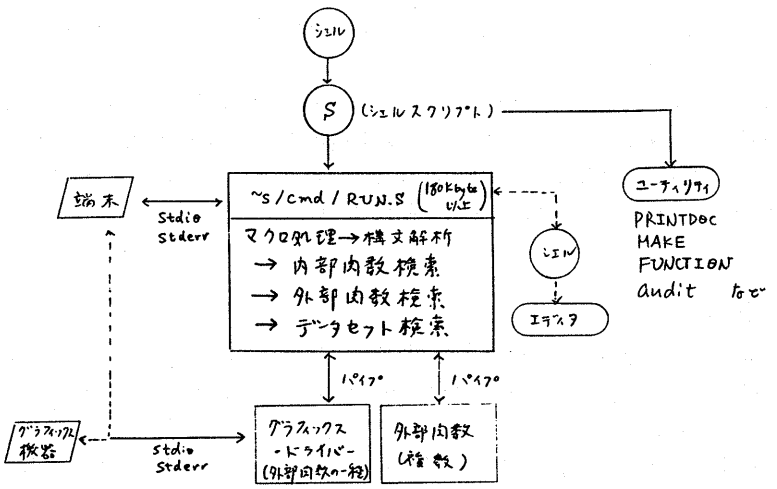
により マクロを定義する。
　p を引数としたので、本体の p が $1 に変わっている。
　lenf の値も、main= のタイトルなども引数にできる。

4. システムとしての S

主として System V 4.2/4.3 BSD, ULTRIX, UTS などの UNIX 上で走る。また VMS 上で走る版も入手可能である。機種は特に選ばないが、コンパイラ、特に FORTRAN コンパイラが問題を起こすことがある。日本でも VAX シリーズ, SUN, NEC - EWS 4800 上では既に稼働しており、Appolo, News 上でも動くものと思われる。

次ページの図は S が走るときに生成されるプロセス間の関係を示している。ただし ~S は S のホーム・ディレクトリである。

データセット、外部関数（別ましたプロセスとして走る関数）それぞれについて以下のような検索リストがあり、これらに従って検索される。これらのリストは S 関数 attach( ), detach( ), Chapter( ) で変更可能である。



データセット：
- ./swork
- ./sdata
- ~s/s/data

外部関数
- ./x
- ~s/s/x
- ~s/slocal/x

また，それぞれのオン・ライン解説も，これらのディレクトリのサブ・ディレクトリ .help を検索する．

データセットは通常バイナリ形式で保存されるが，必要に応じてリスト形式のテキストとして出力できる．グラフィックスの出力イメージは，ドライバーに依存し，文字端末，プロッター，諸種グラフィックス端末に表示できるほか，次の形式のUNIXファイルとしても出力できる．

(i) UNIXのplotコマンドの形式（unixplot.out） (ii) UNIXのpicコマンドの形式（pic.out） (iii) Imagenプリンターの形式（imagen.out） (iv) PostScriptの形式（PostScript.out）．

S使用中に，現ディレクトリに生成されるデータセット以外のファイルは次のように分類できる．

(i) エディタ用 sedit，・mateclit，sdump (ii) 記録用 diary，sdump (iii) 外部形式のデータ dumpdata，data (iv) 表の清書 tbl.out (v) 描き置きグラフィックス sgraph

外部関数だけでなく内部関数も，追加したり変更したりできるが，その構成は

- 外部関数
  - 割り込み，エラー処理（ユーザ・インターフェイス） ～.*.c
  - 内部関数
    - 引数の省略時の値の処理など（ユーザ・インターフェイス） *.～
    - FORTRAN，Cなどのルーチン *.f，*.r，*.C，*.c

となっており，内部と外部とを変更することも容易である．現在約300の関数がある．

Ex.7 ユーザ定義の関数

key.～ （ユーザ・インターフェース）

```
FUNCTION key(x/CHAR,1/)
call grep(TEXT(x))
END
```

grep.C （マクロ使用のサブルーチンCプログラム）

```
F77-SUB(grep)(xp,xn)      /* マクロ */
Char  *xp; long int xn;   /* Cルーチン */
  ...
```

グラフィックス・ドライバーはすべて外部関数で，Tektronix系 Hewlett Packard系のグラフィックス機器ならばすでにドライバーが用意されているが，新たに作成しなければならないものも多い．そのようなときには以下のようなRatfor基本ルーチンに手を入れればよい．

zseekz.r（シーク），zptchz.r（文字を置く），zlinrz.r（線分を引く），zgivxyz.r（座標の読み込み），zeglcz.r（画面消去），zflshz.r（1画面を描いた後のモード切り替え），zparmz.r（グラフィックス・パラメータの解釈），zwrapz.r（ドライバー終了時の後始末）．

5．使用経験と問題点

Sの問題点は次の通りである．

(i) 複雑なマクロを作ると，マクロ展開の優先順位が分かり辛くなる．特に入れ子となっていると，展開結果がどうなるか予測しにくい．一般に，マクロ中でマクロを定義することはできない．

(ii) 付値したデータセットはすべて，1つのUNIXファイルとなるので，途中結果をいつでもチェックできるという点では便利だが，演算速度を低下させる原因の一つとなっている．

(iii) 複素数，倍精度実数が使えない．

(iv) 外部関数のインターフェイスを修正するには，一度Sから抜けて修正し，コンパイルし直さなければならない．

(v) 内部関数のちょっとした修正，インターフェイスの改良などをしたいときは，マクロを作成するか，S自体を作り直さなければならない．

(vi) UNIXコマンドの出力を直接Sから扱えない．

(vii) 文字列の操作が自由にできない．

(viii) 日本語が使えない．

(ix) 3次元の動的なグラフィックス，グラフィックス・データの読み込み・解析・処理，数式・特殊文字のグラフィックス表示への埋め込み，などグラフィックス機能の高度化．

(x) より複雑なデータ構造体の処理．

これらのうち(i)—(vii)は，現在Bell Labs.で開発中のNew Sで解決されるようである．New Sは，流行のオブジェクト指向を取り入れ，マクロを廃止し，データセットと関数定義の区分を原則としてはずす．データ解析言語よりさらに進んで数理的プログラミング環境（QPE／SQPE）を目指している．Chambers (1986)．

これまでの経験では，データ解析用の言語・システムとしてだけではなく，プログラミング環境としても次の点で優れている．

(i) 入力データ，テスト・データの生成，取り扱いが楽である．特に，複数個の行列，配列をまとめた構造を必要とする場合に便利である．

(ii) 出力結果の処理，特にグラフィックスを用いた解析，表示が気楽に行える．

(iii) 記憶管理，型の整合，寸法の整合などはシステムに任せて，プログラミングの主目的に専念できる．

(iv) 作業過程の記録，デバッグ，性能評価の道具がそろっている．
(v) UNIXの環境をうまく利用できる．

現在のところシステム自体が比較的大きいこともあって，いくぶん処理速度が遅い．しかし今後，システムの効率化が計られ，ハードウェアの性能が向上することにより，速度は十分に満足できるようになるであろう．たまさかの利用者としては，ぜいたくなシステムで，プログラミング，出取り，操作，の時間を短縮したい．

参考文献

[1] Becker, R.A. and Chambers, J.M. (1984) S; An Interactive Environment for Data Analysis and Graphics, Wadsworth, Belmont, CA (渋谷，柴田訳 (1987) Sシステム I(概説篇)，II(詳説篇)，共立出版)．
[2] ——— (1984) "Design of the S system for data analysis", Communications ACM 27, 486-495.
[3] ——— (1985) Extending the S System, Wadsworth, Belmont, CA.
[4] Chambers, J.M. (1986) "A computing environment for statisticians", Proceedings of the Anual Amer. Statist. Assn. Meeting.
[5] IBM (1984) APL2 Programming: Language Reference, SH20-9227
[6] —— (1985) 日本語APL，言語解説書 N: SB18-1094
[7] 柴田里程，渋谷政昭 (1985) データ解析言語S，bit 17, 986-994.
[8] ベッカー他 (1987) 座談会；ベル研，UNIX & S，bit 19, 1872-1883.