S 言語による社会調査結果の視覚化データベース

吉 田　光 雄

目　次

S言語による社会調査結果の視覚化データベース

吉田　光雄

1　Sとは何か

SはAT&TのBecker,R.A. & Chembers,J.M. らによって作成されたデータ解析用言語である。1984年にS-Systemとして発表され、1988年に大幅に改良された後、S-Languageとしてリリースされ、今日に至っている。UNIX上で構築され、UNIXの発展とともに拡充されてきた。

その特徴を柴田(18)は一言で「単純で、統一的で、開放的」と述べている。「単純さ」とは諸機能が部品化され、基本的には部品の組み合わせで仕事が進められること、「統一的」は徹底した部品化が統一的な設計方針で行われ、同様の操作で部品を扱えること、「開放的」は、狭義には、ユーザーが自由に作り変えたり、他のソフトウエアと結びつけることができ、広義には、だれでもがソース・コードを入手できること、と説明している。

これをさらに敷衍しよう。「オブジェクト指向」の言語であり、取り扱うデータ、プログラム、処理結果をすべてオブジェクトとして同様に取り扱うことができる。そして処理の単位を関数として部品(モジュール)化してあるため、新たな処理への展開が容易であり、かつ、ユーザーが計算機と対話しながら、プログラムを作成し、データ解析を行うことができる。通常、多くの統計処理パッケージは処理結果を印刷(またはファイルにセーブ)するのみであるが、Sでは結果をオブジェクトとして他の名前で付置し、それを次の処理の入力に用いることができるため、柔軟な解析が容易となる。

また、他の統計パッケージとの相違は、多くのパッケージが処理内容が固定されており、その範囲内であれば目的の処理を行うことができるものの、それを外れた処理は不可能である。本格的なパッケージほど、その傾向が強い。ソースが公開されていないので、修正すら不可能である。たとえば、SASはメニューが完備され、その処理の多様さ、豊富さを誇っているが、マニュアルで目的にかなった特定の処理を行うためのオプションを探すのは一苦労であり、見いだせる場合はまだしも、オプションに用意されていない場合も多い。たとえば、ファジィ統

計学など、最新の統計処理に関しては、当然のことながら無力である。

ところが、S はプログラミング環境であり、基本的な統計処理は公開の関数として提供されているので、修正はもちろん新たな構築も容易である。アプリケーションとして見たときは、ほとんど何もできない不便なものであるが、使い込むにつれ、思いどおりの統計処理を可能とする、便利な、データ解析のための環境が提供されていることに気づく。グラフィクスにしても同様で、S で出力されるグラフは、概ねお世辞にも美しいとはいえないが、S の目的がデータ分析の環境であり、プレゼンテーションを目指したアプリケーションではないことに留意すべきである。

本稿では、SAS[25], *Mathematica*[24,26]に引き続き S を用いてデータベースを構築することを試みる。その開放性によりソースコードをコンピュータ・ネットワークで入手できるし、全国大学共同利用の大型計算機センターならびに主要大学の計算機センターで利用可能と思われる[1]。ユーザー領域のハードディスクにデータを保存し、グラフィックス画面のデバイスとデータ・ディレクトリへの検索パスを設定すれば、統計解析と作図によるデータの一瞥に活用することができる。

まず簡単に S の使用方法について説明する。次いで、主要な統計グラフ作成のための関数ならびに標準関数で不十分な点を補って作成されたユーザー関数について解説し、最後にそれらをデータベースに応用した具体例を述べる。S の使用に際しての細部は他の成書[1~8,19など]に詳しい。

2 システムに関する操作

2.1 S の起動と終了

%S(大文字)<return>	S の起動	^c(control+c)	中断
>q()<return>	終了	^Z, ^D	強制終了

UNIX プロムプト(入力待ち)%(または $)から S<return> で S が起動する。S のプロムプトは > である。S を終了するときは q()<return> であるが、暴走時の強制終了には ^Z(コントロールキー + Z)を用いればよい。S コマンドの入力ミス、実行ジョブの中断などには ^C(コントロールキー + C)が用いられる。ただし、マシン環境により異なる場合がある。

S の基本用語として、オブジェクト(S で扱うデータ、関数、コマンドなどの総称)、関数(一連の処理を関数としてまとめたもの)、ベクトル(1 次元の数字データの集まり)、行列(2 次元の数字データの集まり)、リスト(数字、文字列が混在したデータ)、コマンド(S に対して処理を行う命令)、付置(データ、関数などを文字(列)に置き換え、S 固有の方式で保存)などがある。

付置として本稿では記号 ← を用いるが、入力に際しては <− または _(アンダーバー)を用いればよい。また以下、立体の文字列はSのコマンド、関数としてそのまま(大文字、小文字を区別)入力することを示し、イタリック体はユーザーが自由に変更して用いるディレクトリ名、ファイル名、オブジェクト名などを示す。

attach()	Sオブジェクトの検索パス一覧
attach(" *dir* ")	ユーザーディレクトリ *dir* をSに接続
ls()	ユーザー作成のオブジェクトの一覧
ls(pos= *i*)	システムオブジェクトの一覧
history()	Sへの入力が逆順で示される
help(*obj*)	コマンド、オブジェクト等のオンラインヘルプ

Sの標準オブジェクトが保存されているディレクトリの検索パスを出力するのが attach()であり、その中身(オブジェクト名)を示すのが ls()である。pos＝1 でユーザー領域、pos＝2^2でシステム関数、pos＝3 でシステムデータが示される。ただし、他のユーザー付置のディレクトリがある場合にはシステムに先行する。

詳細についてはオンラインマニュアルが完備しており、help で参照することができる。

2.2 グラフィックス画面

Sでは処理結果を図示する機能が豊富に用意されている。テキスト画面でも可能であるが精緻なグラフは描けない。グラフィックス画面のデバイスとしてポストスクリプトやX画面を選べば高品質のグラフ出力が得られる。printer()で出力先を指定できるが file＝" *file.name* " でファイル名を与えたときはファイルに、ノーオプションのときは標準出力(モニタ出力)となる。

postscript()	標準プリンタへ出力
x11()	X画面へ出力
printer()	キャラクタ画面へ出力
graphics.off()	ラフィックス画面の終了
Ps	ポストスクリプト･ファイルの作成

あらかじめ、ポストスクリプト･ファイルを作成するコマンド

```
Ps ← postscript("gr.ps", height=5.5, width= 6 )
```

が作成されているので、これを用いればグラフを ps(ポストスクリプト)ファイルに保存する

ことができる。デフォルト値は、ファイル名＝"gr.ps"、大きさ:縦＝5.5, 横＝6 (単位inch)であるが、これらを変更したい場合には直接、オプションを設定して入力すればよい。

一例として、乱数を発生[3]させ、そのヒストグラムをファイルに保存し、フロッピィを経由して、他のコンピュータに移す方法を以下に示す。

> Ps	出力をファイル gr.ps に切り替え
> hist(rnorm(100))	例: 正規乱数を 100 個発生させそのヒストグラムを描く(図 1 参照)
> hist(runif(100))	例: 同様に一様乱数(図 2 参照)
> graphics.off()	グラフィクス画面を終了し、ファイルへ書込む
> !mwrite gr.ps gr1.ps	gr.ps ファイルを gr1.ps 名でフロッピィに転記
> !eject	フロッピィ・ディスクの取出し
% open gr1.ps	他のコンピュータで描画 or プリント

ポストスクリプト・ファイルの作成はグラフィクス画面の終了(graphics.off())、X-Windowへの切替え(x11())、S の終了(q())のいずれでもよい。これらのコマンドのいずれかを実行しておかないと書込みは行われない。

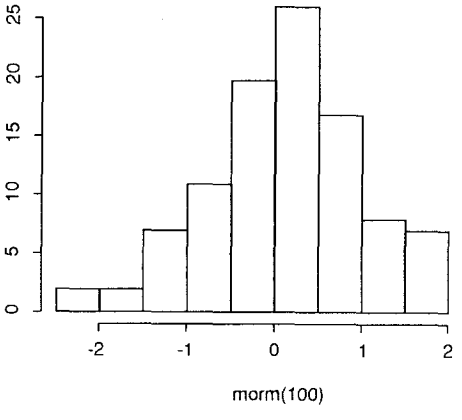

図 1：正規乱数

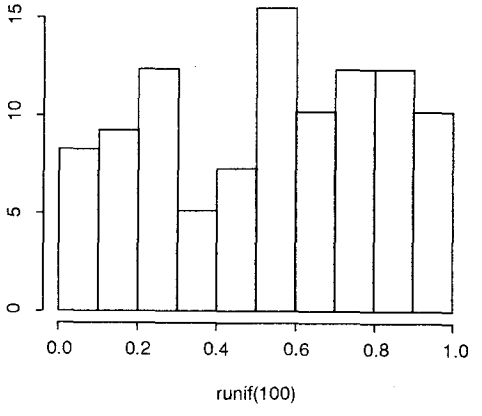

図 2：一様乱数

2.3 ユーザー・カスタマイズ

.First 初期設定	.Last 終了時操作の設定

S を起動した時に初期設定用に読み込まれるファイルが .First であるので、ここに一連の作業を書いておけば自動的に実行される。

たとえば、ユーザー関数のあるディレクトリやデータベースとしてデータの保存されているディレクトリを接続し、x11()でグラフィクス・ドライバを接続する。

同様に終了時にはオブジェクト .Last を実行するので、必要な処理をここに書いておけばよい。ただし .Last は必ず必要で、システムによりすでに書かれており、これを消して変更するのではなく、追記してユーザーが書き加えるのがよい。

3　Sにおける主要な関数

Sは多数の関数の集まりであり、それらの関数を用いて、各種の処理を行う。本稿に関連する主要なものについてその内容と使用方法を要約するが、記述に際しては以下の記号を用いる。

obj: オブジェクト全般

x: 一般的なデータの入力(スカラ、ベクトル、行列)

特にそれらを区別するときは、*s*(スカラーの定数、変数)、*vec*(ベクトルデータ)、*mat*(行列データ)、*list*(数字、文字列の混在したオブジェクト)等を用いることとする。

3.1　データの入力、加工・編集

scan()	KB(キーボード)からの入力
c(*a* , *b*, ···)	ベクトル(数列)の作成
matrix(*vec*, ncol=*i*, byrow=T)	行列(数列)の作成
vec ← *mat* [, *j*]	行列 *mat* の第 *j* 列の抽出
seq(*a* : *b* , by= *c*)	系列データの作成(*a* から *b* まで幅 *c*)
rep(*a* , *n*)	同一データの反復(*a* を *n* 回反復)
cat(*obj*)	*obj* のテキスト表示

小さいデータは直接 KB(キーボード)から入力し S 内にオブジェクトとして保存する。c()はベクトル作成のための関数であり、任意個の数字、文字列の系列を作成する際に用いられる。行列データの作成には matrix()を用い、オプション ncol=*n*, byrow=T を用いて、次々と入力されるベクトルデータを行列形式に変換する。seq(), rep()はオブジェクトの反復作成などに便利である。

3.2 データの入出力(UNIX ファイルとの互換)

scan(" ～/*file.name* ")	UNIXファイルからの入力
source(" *file.name* ")	ファイルから S 式を読み込み、構文解析を行う

大量のデータはファイルから読み込むのが能率的である。あらかじめカレントディレクトリにアスキー形式でデータファイルを作成しておき、それをオブジェクトとして読み込む($x \leftarrow$ scan()で読み込んだデータを x に付置しなければならない)。オプションの指定がない時はファイルの改行コードは無視されてベクトルとして読み込まれるので、データがすべて数字であるときは、必要に応じて行列データに編集しなければならない。たとえば、ファイル *body*1 *.dat* がカレントディレクトリにあり、これを読み込むには

$x \leftarrow$ matrix(scan(" *body*1 *.dat* "), ncol=2, byrow=T)

とする。T または TRUE は真値、逆に偽は F または FALSE である。オプション ncol=2, byrow=T がない場合には 1 次元のベクトルとして読み込まれる。

ファイル *body*1 *.dat* の内容

```
164  58
152  45
160  60
158  58
```

ファイル *body*2 *.dat* の内容

```
1  aa  m  164  58
2  bb  f  152  45
3  cc  m  160  60
4  dd  f  158  58
```

また、文字列や数字･文字列の混在するものであるときは、オプション what=で読み込みのためのフォーマットを指定する必要がある。このときは、行列ではなくリストとして読み込むこととなる。たとえばファイル *body*2 *.dat* がカレントディレクトリにあり、これを読み込むには

body ← scan(" *body*2 *.dat* ", list(0,"","",0,0))

であるとか、あらかじめ読み込み用のフォーマットを作成しておき、それを用いて

fmt ← list(0,"","",0,0)

body ← scan(" *body*2 *.dat* ", what=fmt)

とすればよい。0 は実数、"" は文字列であることを示す。さらにフォーマット

fmt ← list(*number* =0, *name* ="", *sex* ="", *height* =0, *weight* =0)

を用いれば、変数名つきで入力される。

sink()	出力の切り替え
sink(" *file.name* ")	端末出力をファイルに切り替え
write(*x*," *fname* ",ncol=)	UNIXファイルへの出力
cat(*x*,file=" *fname* ")	UNIXファイルへの出力
dump(*x*,fileout=" *fname* ")	オブジェクトをAsciiテキストとして出力
restore(" *fname* ")	ダンプしたオブジェクトを復元する

sink は以後の出力をファイルに切り替えるものであり、もとの標準端末に戻すときはsink()とすればよい。write, cat はデータ、計算結果などをテキスト形式でファイルに書くものであり、ファイル名にディレクトリを省略するとカレントディレクトリとなる。オプションとしてappend＝TRUE を追加すると、すでにあるファイルの最後に追記される。write()では ncol＝で列数を指示する。ないときのデフォルト値は ncol＝5 である。cat()では file＝" *filename* "としなければならない。file＝がない場合には文字列として *x* の後に追記される。dump はオブジェクトを Ascii テキストファイルに変換してファイル出力するものであり、restore はダンプしたオブジェクトを復元するものである。異機種間のデータのやり取りなどに便利である。

3.3 行列処理

行列(ベクトルを含む)に各種の処理を施し、演算を行う関数も多数用意されている。

mat[*i* , *j*]	行列 *mat* の第 *i* , *j* 要素の抽出
mat[, *j*]	行列 *mat* の第 *j* 列(変数)の抽出
mat[*i* ,]	行列 *mat* の第 *i* 行(サンプル)の抽出
t(*mat*)	行列 *mat* の転置
cbind(*a* , *b* , ･･･)	ベクトル(行列)を列結合して行列を作成
rbind(*a* , *b* , ･･･)	ベクトル(行列)を行結合して行列を作成

また、次のような行列演算の関数が用意されているので、多変量解析の多くの手法は、これらを組み合わせることにより、プログラミングすることができる。

変数を選択して検討するとき、もとの行列から変数を抽出しなければならないが、その際必要となるのが小行列の作成，Makemat()，Minor()，Cofactor()である。行列と選択する変数(列)を引数で指定すればよい。

行列の階数や逆行列を求める基本的な行列演算も標準関数として組み込まれていないが、そ

れらを作成したものが、Det()などである。

diag(*a* : *b*)	対角要素(*a* : *b*)の対角行列の作成
solve(*mat*)	行列 *mat* の逆行列
eigen(*mat*)	対称行列の固有値問題
svd(*mat*)	特異値分解
qr(*mat*)	QR 分解
chol(*mat*)	コレスキー分解
Makemat(*mat* , *ivec*)	小行列の作成
Minor(*mat* , *i* , *j*)	小行列の作成
Cofactor(*mat* , *ivec*)	余因子行列の作成
Det(*mat*)	正方行列の行列式(絶対値)
Tr(*mat*)	正方行列のトレース

3.4 統計処理

ベクトル･データを *x* とするとき、昇順に並べ換えるのが sort(*x*)であり、昇順に並べたときそれが何番目のデータであるかを示すのが order(*x*)である。原データの並べかたで、その大きさの順位を与えるのが rank(*x*)である。

連続データを階級の上限値(*class*)に従って分類する関数が cut(*x* 、 *class*)であり、離散量データ(整数値)の集計を行うのが category(*x*)である。また、多変数のデータをクロス集計するのが table(*x*1, *x*2, ･･･)である。2 次元以上、多次元の集計も可能である。

sort(*x*)	並べ換え	sum(*x*)	総和
order(*x*)	順序数	mean(*x*)	平均
rank(*x*)	順序	med(*x*)	メディアン
cut(*x* , *class*)	連続データの分類	var(*x*)	不偏分散
category(*x*)	離散データの分類	var(*x*1, *x*2)	共分散
table(*x*1,･･･)	多変数の集計	cor(*x*1, *x*2)	相関係数

ベクトル *x* をデータとして統計値を求める。var()，cor()に関しては、データが行列のときは分散･共分散行列、相関行列を計算する。不偏分散(共分散)は自由度 n-1 で割ったものである。標準偏差や n で割る標本分散が必要な場合は計算結果を修正するか、プログラミングにより自作しなければならない。以下はその一例である。S における標準関数のオブジェクト名はすべて英小文字であるが，ユーザー関数の場合、それらと区別するため最初の一字を大文字

とする(*Mathematica* の場合と逆[24])。

Range(x)	範囲	RangeQ(x, q)	100 q パーセント範囲
Var(x)	標本分散	Std(x)	標準偏差
Uvar(x)	不偏分散	Ustd(x)	不偏分散の平方根
Cov($x1, x2$)	共分散	Cor($x1, x2$)	相関係数(Std を使用)

これらはディレクトリ ～/ *s* /*myfunc* に保存し、attach(" ～/ *s* /*myfunc* ")で布置しておけば、自動的に計算時に読み込まれる。

3.5 統計的分布

主要な確率分布について、確率密度(d)、分布関数(p)、上側確率(q)を計算し、また乱数(r)を発生させる関数が用意されている。関数名や引数は、すべての分布を通じて統一的に構成されているので分かりやすい。

平均＝ a , 標準偏差＝ b の正規分布について、確率密度、分布関数、上側確率 p に対する確率点を求める。また、同分布に従う正規乱数を n 個発生する。

dnorm(x, a, b)	正規分布 N($\mu = a, \sigma^2 = b^2$)の確率密度
pnorm(x, a, b)	正規分布 (〃)の分布関数(下側確率)
qnorm(p, a, b)	正規分布 (〃)の上側確率点
rnorm(n, a, b)	正規分布 (〃)の n 個の乱数

以下指数分布、一様分布、ガンマ分布、ベータ分布などの確率分布の他、標準正規分布、χ^2－分布、t－分布、F－分布などの標本分布についても同様である。確率密度、分布関数、確率点、乱数を求める関数が用意されている。[ ]は目的に応じて d, p, q, r のいずれかを選択することを示す。

[d,p,q,r]exp(x)	指数分布 $f(x)=e^{-x}$の確率密度
	[分布関数,上側確率点,乱数]
dunif(x)	一様分布 U(0,1)の確率密度他
dgamma(x, s)	ガンマ分布(s:形状パラメタ)の確率密度他
dbeta(x, $s1$, $s2$)	ベータ分布(s_2, s_2:形状パラメタ)の確率密度他
dnorm(x)	標準正規分布の確率密度他
dchisq(x,df)	χ^2－分布(自由度df)の確率密度他
dt(x,df)	t－分布(自由度df)の確率密度他
df(x,df_1,df_2)	F－分布(自由度df_1,df_2)の確率密度他

引数x,pがベクトルのときは、その要素分の値を出力する。従って、それを利用して分布のグラフを描くことができる。次はその一例であるが、他の分布も同様である。

```
x← seq(-3,3, by=0. 01)
plot(x, dnorm(x), type="l")    正規分布の確率密度関数
plot(x, pnorm(x), type="l")    正規分布の分布関数
df← c(5, 10, 20, 25)
plot(dt(x,df), type="l")       t-分布の確率密度関数
```

3.6 グラフによる分布の正規性、類似性の検討

確率プロットを用いて、データの正規性の程度を視覚的に検討することができる。qqnormはデータベクトルxを昇順に並べ換えて確率点を求め、y－軸には正規確率点を用いて、両者をプロットするので、データxが正規分布に近いとき、データ点はほぼ直線上に並ぶが、正規分布を示さない場合にはS字状カーブを描く。

qqnorm(x)　正規プロット	qqplot(x,y)　qq プロット

さらに一般的に qq プロット(確率点プロット)を用いれば、2つのデータの類似性がグラフ的に示される。(x,y)両データの分布が近似しているとき、プロットは直線上に配置されるが、異なる場合には直線からのずれが大となる。

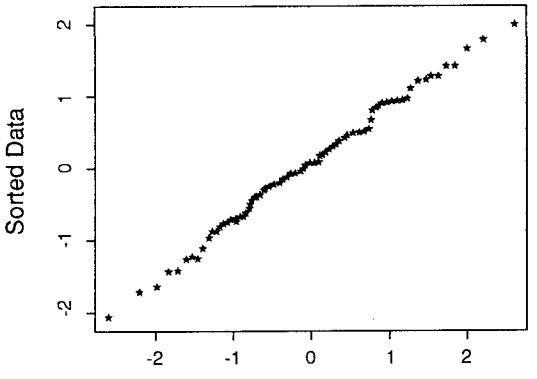

図3：正規乱数の正規プロット
＞ qqnorm(rnorm(100))
正規乱数100個のプロット

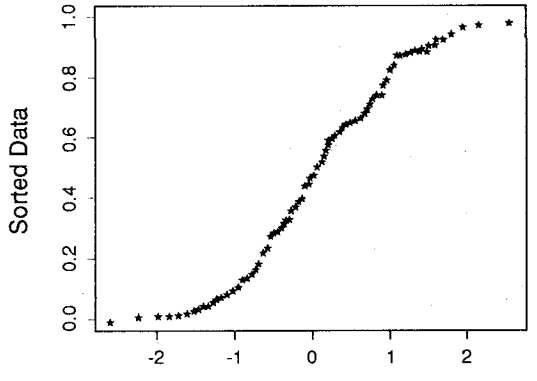

図4：一様乱数の正規プロット
＞ qqnorm(runif(100))
一様乱数100個のプロット

4　視覚化統計処理関数

データ解析としての統計処理とは、従来は各種の統計量を算出してデータのもつ情報を簡単化された指標に要約することであった。データのもつ情報を十二分に抽出するための技法として、Turkey[21]によりグラフィカルな探索的データ解析の方法が提唱されて以来、データの全貌や要約統計量の詳細を検討するためには統計図の描画は不可避となってきた[9,10,11,17,20,27,28]。コンピュータの性能の向上もまたそれに寄与した一要因であるが、統計図のメニュを持たない統計アプリケーションは稀であろう。

Sにおいても統計図の作成は可能であるが、他のアプリケーションに見られるプレゼンテーションのための統計グラフの作成とは趣を異にしている。Sの特徴は「図との対話」であり、探索的なデータ解析の手段として統計図が活用される点である。次の理念に従って作図に関する多くの関数[1,2]が用意されている。

- データの特徴や構造を知るために、データを様々な形の図に描いて、多角的に眺めることができる.
- 加工処理したデータを、それ以前の図に重ねて描画させ、比較·検討することができる.
- 図を見ながらデータをピックアップし、そのデータについて検討することができる.

- 作図結果をオブジェクトとして付置し、オブジェクトを指定するだけで作図することができる.
- 図形本体の作図部分(高水準作図関数)と、凡例、タイトル、座標ラベルなどの記入(低水準作図関数)とが分離されており、必要に応じて使い分けることが可能である[4].
- 作図機能は作図機器に依存せず、同一式で様々の機器において作図することができる.

S に内蔵されている統計グラフを描くための関数は豊富であるが、本データベースに活用可能な関数は次のごときものであろう。関数そのままを使用してもよいが、オプションの入力など複雑であり、そののまま毎回 KB 入力するのは煩雑であるので、使用頻度の高いと思われるものについてプログラミングを行い、ユーザー関数とした。それらは、先の統計関数の場合と同様、標準関数と区別するため関数名の冒頭の 1 字を大文字としてある。

統計図標準関数			
hist(x)	ヒストグラム	pie(x)	円グラフ
barplot(x)	棒グラフ	stars(x)	星型図
tsplot(x)	時系列プロット(折線グラフ)	dotchart(x)	点図
stem(x)	幹葉図	matplot(x)	行列プロット
boxplot(x)	箱ひげ図	contour(x)	等高線プロット

4.1 1 変量処理のための統計図関数

ヒストグラム	
Hist.1(x)	1 桁データの度数分布・平均・SD・ヒストグラム
Hist.cont(x , *class*)	連続量データの度数分布・平均・SD・ヒストグラム

1 桁の整数値のデータについて、度数を集計し、ヒストグラムを描く。また、あわせて平均・標準偏差、データの上限(最大値)、下限(最小値)も表示する。

連続量の場合には集計する階級の上限をベクトル *class* で与える。たとえば

class ← c(-3:4)	ファクタースコアなど正規分布変数を(-3,-2,⋯,4)を上限とする階級で集計する.
class ← c(1:10)	1 から 10 までの整数に分類.
class ← seq(0,1,0.1)	(0,1)乱数を 10 倍して切り捨てる.
class ← seq(−0.05,1.05,0.1)	(0,1)乱数を 10 倍して四捨五入する.

class ← seq(139.5,180,5)　　身長のデータなど 139.5cm から 5 cm 幅の階級で集計する.

など、データに応じて設定しなければならない。あわせて平均·標準偏差、データの上限、下限も算出するのは整数値の場合と同様である。あらかじめ *class* で階級の上限を付置してもよいし、関数内で設定してもよい。

円グラフ	
Pie.1(*x*)	1 桁データの度数分布・平均・標準偏差・円グラフ
Pie.cont(*x*, *class*)	連続量データの度数分布・平均・標準偏差・円グラフ

1 桁の整数値のデータについて、Pie.1()により、度数を集計し、円グラフを描くことが可能である。また、あわせて平均・標準偏差、データの上限(最大値)、下限(最小値)も表示される。ラベルをふって円グラフを描くが、カテゴリは 1 ～ 9 以内とし、数字のラベルとする。カテゴリ名を文字列で与えることも可能であるが、その場合は関数を使用するのではなく個別に描かねばならない。

連続量データの場合は Pie.cont()を用いて円グラフを描く。度数分布表を作成するために階級の上限 *class* が必要であり、カテゴリのラベルとして *cname* が必要である。デフォルトは "1", "2", ～ であるが、特定の文字列をあらかじめ与えておいてもよい。たとえば

myname ← c(" *aa* "," *bb* ", " *cc* ", ···)

などであるが、ラベルの数とデータのカテゴリ数とが一致していなければならない。

棒グラフ	
Bar.1(*x*)	1 変数データの棒グラフ
Bar.mat(*mat*,*fp*=" *f* ")	多変数データの棒グラフ(度数)
Bar.mat(*mat*,*fp*=" *p* ")	多変数データの棒グラフ(百分率)

Bar.1()は、1 変量(数)の整数値データ(カテゴリに 1, 2, ··· の数値を付与したもの)について、度数分布に集計し、棒グラフを描く。データの最大値はいまのところ 30 であるが、この上限は容易に変更可能である。

Bar.mat()は、行列データを与え帯グラフを描く。すなわち、各列を大カテゴリとみなして列毎に棒を立てる。各行の要素を列内のサブカテゴリとして柱を分割する。第 2 パラメタ fp ="f" の場合には行列要素の値そのものを使用し、列和が棒の高さとなる。第 2 パラメタ fp= "p" の場合には行列の列和を分母とする比率を用いて、帯グラフを描く。従ってすべての高さは同じである。行データのラベル表示をしないが、行番号に従って、帯内のカテゴリを下から

積み上げていく。

各行毎にグラフを描くようなデータの場合にはあらかじめ転置した行列を作成しておくか、入力時に、行列転置の関数を t(*mat*)を用いて入力すればよい。

箱ひげ図(標準関数)	
boxplot(x1, x2, ・・・)	データの箱ひげ図

1つ以上、複数のデータベクトルについて、標準関数[5]を用いて箱ひげ図を描く。平均を中心に上下に標準偏差が箱で示され、また標準範囲(四分位範囲の1.5倍)がひげで図示される。データはいくつでもよく、必要な数だけ指定すればよい。ただし、データ数が多いとき箱の幅が狭くなる。names=c(" *a* "," *b* ",・・・)のオプションを与えれば、各データのラベル(*a* , *b* ,・・・)を付与することができる。

幹葉図(標準関数)	
stem(x)	データの幹葉図

1変量のデータベクトルを集計し、幹葉図でその分布の状況を図示する。整数値でも連続量データでも自動的に幹の位や範囲、葉の桁などを判断しグラフを描く。オプションでそれらの値を設定することも可能であるが、自動的に描かれる図が最適であろう。ただし、複数個のデータを相互に比較したいようなときは基準を統一しなければならず、オプションでの設定が必要である。

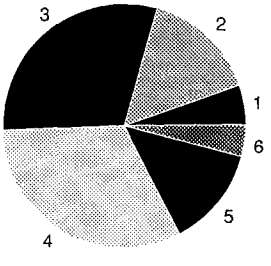

図5：円グラフ

> class ← c(-3:4);

Pie.cont(rnorm(100), class)

正規乱数 100 個を集計して図示

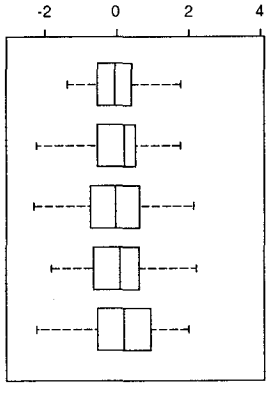

図6：箱ひげ図

> boxplot(rnorm(100),・・・, rnorm(100))

正規乱数(100個)を5回発生させ、分布の範囲を図示

4.2　2変量処理のための統計図関数

2変量のデータベクトル(x, y)を用いて、両データ間の関係を直感するためのグラフを描く。連続量データの場合は散布図、定性的データの場合には鳥瞰図が有用であろう。

散布図	
plot(x, y)	データ(連続量)の散布図(データ点＝1)
plot(x, y, pch＝"*")	散布図(データ点＝*)
plot(x, y, type＝"n");text(x, y)	散布図(データ点＝番号)
plot(x, y,type＝"n");text(x, y, z)	散布図(データ点＝z の内容)
plot(x, y);identify(x, y, z)	散布図(データ点を識別)
matplot(x, y)	データ(連続量)の散布図(データ点＝1)
matplot(x, y, pch＝"*")	散布図(データ点＝*)
matplot(x, y, type＝"n");text(x, y)	散布図(データ点＝番号)
matplot(x, y, type＝"n");text(x, y, z)	散布図(データ点＝z の内容)

最も汎用の2変量散布図は plot(x, y)である。連続量の場合、最大値、最小値から座標軸・目盛りを自動的に判断して、最適の散布図を描いてくれる。オプションにtype＝"p"(点プロット), type＝"l"(線で結ぶ), type＝"n"(プロットなし)などがあり、必要なら対数目盛りのプロットも可能である。各観測値の記号はデフォルト＝1　であるが、pch＝"*" で他の文字(*)を指定できる。type＝"n" と text(x, y)を用いて観測値番号とすることもできるし、第3の変数(z)の文字を用いることもできる。text()はすべてのデータ点を表示するが、identify()は必要なデータ点をカーソルで示せば、そのデータのみの特定がなされる。特異値などの識別に便利である。

plot()と同様の関数に 行列の各列についてプロットする matplot()がある。使い方はほとんど plot()と同様である。

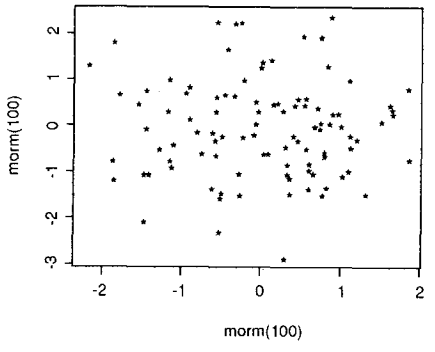

図 7：散布図

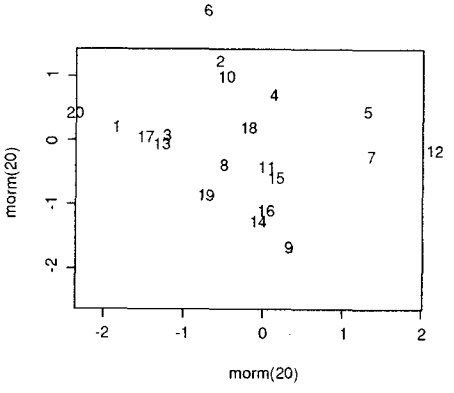

図 8：散布図

> plot(rnorm(100),rnorm(100))
正規乱数 100 個を 2 回発生し，
(x, y)座標でプロット.

> plot(rnorm(100),rnorm(100),
type="n"); text(x, y)
正規乱数のプロット. ただし、
点名は発生番号.

鳥瞰図	
persp(*mat*)	行列データ(mat)の鳥瞰図
persp(*mat*, eye=c(10,0,5))	行列データ(mat)の鳥瞰図
persp(*mat*, eye=c(0,10,5))	行列データ(mat)の鳥瞰図
Cplot(*vec1*,*vec2*,*nc*)	ベクトルデータ(*vec1*, *vec2*)のクロス集計と鳥瞰図

カテゴリデータの場合には plot()では度数が描かれないので不都合であるため、鳥瞰図を描く標準関数 persp()を用いる方がよい。

行列 *mat* は(0～1)または(-1～1)の範囲に標準化しておく必要がある。

eye = c(x, y, z)で視点を変更し、さまざまな角度からデータの構造を鳥瞰することができる。eye = c(10,0,5)は x 方向から、eye = c(0,10,5)は y 方向から、高さ＝5 で眺めることを意味する。

2 つのベクトルデータを入力し、それをクロス集計して、相対度数を鳥瞰図で描くユーザー関数 Cplot()を作成した。鳥瞰図は eye = c(10,0,5)で描くので、プリントする行列データと平行関係にあり、位置を重ねて構造を検討することができる。この鳥瞰図において、グラフの横軸が y(右が高スコア)であり、縦軸が x(手前が高スコア)である。第 3 オプション *nc* は描画のカラー番号であり、*nc* = 1 (黒), *nc* = 2 (赤), *nc* = 3 (緑), *nc* = 4 (青), *nc* = 5 (黄), *nc* =

6 (水色), $nc = 7$ (紫)のいずれかを入力する。

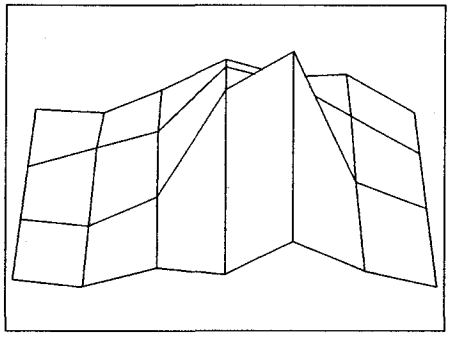

図 9：鳥瞰図(正規乱数のクロス集計)

> plot(rnorm(100),rnorm(100))
正規乱数 100 個を 2 回発生し, クロス集計してプロット.
鳥瞰図は列(x)方向から見たものであるため、行列データの視点と同じ.

x→	−3	−2	−1	−0	1	2	3
−1	0	0	2	6	4	4	0
−0	0	2	4	11	9	5	1
1	0	0	3	14	18	4	1
2	1	0	2	2	5	2	0

表 1：フェイス図変数の割り当て

表情	高スコア	低スコア	表情	高スコア	低スコア
1. 顔の面積	大	小	9. 両目の角度	逆八字	八字
2. 顔の形	縦長	横長	10. 両目の形	上に開	下に開
3. 鼻の長さ	長い	短い	11. 両目の幅	大	小
4. 口の位置	鼻下短い	鼻下長い	12. 瞳の位置	内側	外側
5. 笑いの曲線	上向き	下向き	13. 眉の位置	高い	低い
6. 口の幅	長い	短い	14. 眉の角度	逆八字	八字
7. 両目の位置	高い	低い	15. 眉の幅	長い	短い
8. 両目の位置	大	小			

4.3　多変量処理のための統計図関数

多変量処理という時、多変量を 2 変量ないしは 3 変量の組み合わせから検討したり、多変量の図示を工夫する方法と多変量の情報を圧縮し、より少数の要因でデータの構造を検討する多変量解析の方法とがあろう。前者の方法として S では対散布図や、星形図、フェイス図(Charnoff method)などが用意されている。

多変量図(多変量を同時に描画)	
tsplot(*x1*,*x2*,···)	折線グラフ
pairs(*mat*, full=T)	対散布図
stars(*mat*, byrow=T)	星型図
faces(*mat*, byrow=T)	フェイス図(Chernoff)

tsplot()は多変量ベクトルの折れ線グラフを描き、他は行列データ(*mat*)を与えて、pairs()は変量毎の散布図、stars(), faces()は各列を変量、各行をサンプルとみなして、多変量のグラフを描く。この時、グラフは縦に描くのでサンプルを横に並べたいときはオプションbyrow=T を挿入する。pairs()で full=F のときは対の下半分を描く。フェイス図で顔に割り当てるのは 15 変数である。

これらは、完成度が高く、多変量のデータ構造を直感できる有用なグラフであり、特にユーザー関数を作成して修正する必要も認められないので、標準関数をそのまま用いることとする。

5 Sの拡張

5.1 UNIX コマンドの実行

S はデータ解析のための総合的環境であり、各種の統計処理に関する手続きが関数としてモジュール化されている。それらは、処理されるデータと共に、オブジェクトとして統一的に取り扱うことができる。多様な統計処理の基本的な手法はすでに用意され、その意味では内部完結した一つのアプリケーションであるが、S の特徴はそうしたメニューの豊富さにあるのではなく、ユーザーにプログラミング環境を提供し、自由な問題に対しても、それを処理しうる有効な環境を提供している点であろう[23]。

具体的には、S のコマンド·関数の処理のみならず、ウインドウを移動したり、Sを終了したりすることなく、直接 UNIX コマンドを実行できる点である。データの入出力として、アスキー·ファイルの読み書きが可能なことは前述の通りであるが、それ以外にも unix()や！は処理を一時 S からシェルに移すので、すべての UNIX コマンドを実行することができる。

unix(" *cmd* ")	UNIXコマンド *cmd* の実行
!*cmd*	UNIXコマンド *cmd* の実行
!*Meansd*	例: *Meansd* の実行

従って、一般的に言語の種類を問わず、プログラミングされ、コンパイルされた実行形式のファイルを呼び出し、実行することができる。たとえばMeansdはKBから任意の数のデータを入力し、平均·標準偏差を計算するC−プログラムをコンパイルしたものであるが、!Meansdでそれを直接実行することができる。ただし、出力結果をオブジェクトとして付置することはできない。

5.2 多変量解析のための関数

多変量の持つ情報を圧縮するのは多変量解析の本来の使命であるが、Sで標準として準備されているのは以下のようなものである。ただし、x:独立(説明)変数、y:従属(目的)変数。

prcomp(x)	主成分分析	cmdscale(x)	多次元尺度法
discr(x, y)	判別分析	hclust(x)	クラスター分析
lsfit(x, y)	重回帰分析	loglin(x)	対数線型モデル
cancor(x, y)	正準相関分析		

多変量のグラフ化は十分強力であり、データベースに活用されるのであるが、多変量解析の方法自身は、一般の統計アプリケーションのように完成度の高いものではない。本来Sはプログラミング環境であって、誕生の経過からして、探索的な統計解析に活用されるべき性質のものである。未完の部分はユーザーがカスタマイズしつつ完成させて行くことが期待されている。オプションを選びさえすれば、ほとんどあらゆる統計処理が可能となる、たとえばSASのようなアプリケーションとは一線を画するものであろう。Sはむしろ *Mathematica* に近い対話的環境を提供するものといえる。

たとえば、主成分分析ではデータ行列から重心を中心として軸の回転を行う、すなわち、分散·共分散行列を用いるのが標準であり、他の方法はオプションとしてさえ加えられていない。従って、相関行列から固有値を求めようと思えば、データ行列を事前に標準化しておかねばならない。

また、判別分析法でも判別係数は算出されるが、判別関数が出力されないのでユーザーでその部分を書き加えなければサンプルの判別ができず、また判別の精度に対する出力もなく極めて不親切である。

同様に、対数線型モデルの解き方も簡便にすぎる。モデルの与え型に対する説明が不十分であるため、実際にマニュアルだけでの解法は不可能であろう。

多変量解析に関しては、かなりの部分をユーザーのプログラミングに期待しないければ、活用は困難である。Sは専門家のデータ解析言語である。本稿はデータベースのプラットホームとしてのSの可否の検討であり、データを多角的に分析することを目的としていないため、

こうした点は障害とはならないが、多変量解析が必要な場合は自作すればよいし、他のアプリケーションを使用して、その結果をSにimportすることによっても解決できる。さらに、C, Pascal, Fortran等で書かれコンパイルされた、オブジェクトプログラムをUNIXコマンドとして直接実行してもよい。

以下の多変量相関係数(重相関係数, 偏相関係数, 正準相関係数)はS言語で書かれたものであるが、他の言語によるプログラミングも困難ではないし、ソースプログラム記載した成書を活用することも可能である。これらの相関係数は、端的に多変量間の関連を見るものとして必要であろう。グラフィックス機能をも盛り込んで作成しておけば TDA(探索的データ解析)として有効と思われる。これらはいずれも相関行列から変数を指示して、多変量相関係数を算出するものである。P.cor()は行列の操作から求めるものであり、 p.cor()[18]は粗データから重回帰の残差を用いて計算される。

多変量相関係数	
M.cor(*mat,i,vec*)	重相関係数
P.cor(*mat,i,j,vec*)	偏相関係数
p.cor(*x*)	偏相関係数(柴田による)
Can.cor(*mat,vec1,vec2*)	正準相関係数

5.3 他言語プログラムのダイナミック・ロード

他言語で作成された関数やサブルーチンをロードし、コンパイル、実行することができる。Sのバージョンによっては dyn.load()が使えず、LOAD を用いる場合もあるが、他言語とのインターフェースについても配慮され、Sの拡張性を高めている。

dyn.load(" *prog.o* ")	ダイナミック・ロード
.F(" *prog* ")	Fortran プログラムの呼び出し
.C(" *prog* ")	C プログラムの呼び出し

たとえば、C 言語で書かれた平均・標準偏差を計算するソースプログラム meansd.c をコンパイルし、オブジェクト・プログラムロードしておけば、以後はそれを呼び出し、S のオブジェクトとして実行することができる。S 以外で作成されたプログラムを活用できるので、拡張性は格段に向上する。ただし、グラフィックスに関しては、多くはコンパイラ固有の作図用ライブラリを利用して作図されるため、プログラムの他機種への移植は困難である。多変量解析などの数値計算の部分は成書にも多く報告されており、それらを活用する道が開かれている。

6 具体例: 情報処理教育に関する調査

Sによる視覚化データベースの一例として、情報処理教育センターと共同で行った情報処理教育に関する調査結果を示す[12,16]。

平成6年度より、大阪大学では前期課程の学生に対するカリキュラム改革を行なったが、その一環として、情報処理教育を重要視し、多くの学部では必修の科目として、1年次に「情報活用基礎(2単位)」を課すこととなった[15]。カリキュラム改革に先立って、学生が情報処理教育をどのように考え、またコンピューターをどの程度使用し、コンピュータに対してどのような印象を持っているのかを調査した。調査は質問紙形式と、コンピューターによるオンライン調査の形式の両方で行われたが、質問紙形式の調査結果を取り上げ、データベース化する[6]。調査の項目は表2のごとくである。

調査対象は本学1年次生999名(文科系学部510名、理科系学部489名)、調査時期は平成5年6～7月であり、分析後データベースとして調査結果を体系的に保存した。データベースは情報処理教育センターの人間科学部分散端末上に構築されており、SASおよび *Mathematica* で検索可能である。本データベースは現在のところ講座内のワークステーション上に構築され、全学のネットワークからはoff－lineであるが[7]、前2者とは異なる分析が可能であり、多様なデータベース構築の可能性を探る研究の一環として試みられたものである。

6.1 起動

まず S を起動する。ワークステーションのハードウエア、ソフトウエア環境により方法は異なるが、ここではX－WindowsベースでSを起動し、グラフィックスはx11()のデバイスを用いて描くこととする。従ってその際の初期設定は次のごとくである。

表2:情報処理教育に関する調査: 調査項目

F. フェイスシート項目

ID(1)～F4(5)　　ID番号、学部、学年、性別、NeXT使用の有無

Q1. 大学進学以前に、どのようにコンピュータとかかわっていましたか.[0,1]
Q1_1(6)～Q1_11(16)　例: 授業でコンピュータ実習の時間があった.

Q2. どの程度コンピュータゲームをしましたか.[1－4]
Q2_1(17)～Q2_4(20)　例: 小学生の頃

Q3. 家にパソコンやワープロ専用機があり、使うことがありますか.[0,1]
Q3_1(21)～Q3_5(25)　例: 家にはワープロ専用機もパソコンもない.

Q4. 大学では主にどこのコンピュータを使っていますか.[0,1]
Q4_1(26)～Q4_7(32)　例: 使わない

Q5. 次にあげた内容についてコンピュータをどの程度使っていますか.[1－4]
Q5_1(33)～Q5_9(41)　例: 文書作成(ワープロ)

Q6. 次にあげた言語でプログラムを書くことができますか.[0,1]
Q6_1(42)～Q6_9(50)　例: BASIC

Q7(51). あなたは,文書を作成するときどのようにしますか.[1－4]

Q8. コンピュータについていろいろな意見がありますが、どう思いますか.[1－5]
Q8_1(52)～Q8_30(81)　例: コンピュータはなんとなく親しみにくい.

Q9. 文科系大学生にとって、どのような情報処理教育が望ましいと思いますか.[1－5]
Q9_1(82)～Q9_10(91)　例: 語学のように全学生必修の科目とすべきである.

Q10(92). 文科系学生が教養課程で学習する情報処理科目の単位数は,どのくらいが適当だと思いますか.[1－8]

Q11. 理科系大学生にとって、どのような情報処理教育が望ましいと思いますか.[1－5]
(93)～Q11_10(102)　例: 語学のように全学生必修の科目とすべきである.

Q12(103). 理科系学生が教養課程で学習する情報処理科目の単位数は,どのくらいが適当だと思いますか.[1－8]

( )は通しの項目番号,[ ]は回答の選択肢を示す. 本調査の場合, (0,1)項目の選択回答か, あるいは1桁の順序変数のいずれかである. ただし, フェース項目は1桁のカテゴリ変数.

% xinit	X−Window をオープン
% kterm &	kterm を裏ジョブで開く(wterm, xterm などでも可)
% S	S を起動(プロムトは > に変化)
> init.my	ユーザー統計関数の接続
> init.dbs	dbs に関するディレクトリの接続
> attach()	接続を確認
> scheck	チェックデータを呼び出し、接続を再確認
> x11()	グラフを描くためのデバイスの設定

こうした一連の初期化作業を自動化することは容易である。S のコマンドの部分は .First で実行することもできるが、最初だけの設定であるので、逐一行ってもさほどの手間ではない。

調査結果の保存されているディレクトリを info とする。その下にデータ(テキスト形式ファイル="all.dat", S オブジェクト=all)、調査項目("sheet.txt")、調査対象一覧("sample.txt")が保存されている。

調査項目·項目参照番号("sheet.txt")を参照するウインドウを裏ジョブで開いておけば常時見ることができる。また、調査項目の略称が item.name に保存されているので、結果のプレゼンテーション時などに使用することができる。

データの S への読み込みは

```
all ← matrix(scan("info/all.dat"), ncol=103, byrow=T)
```

を実行すればよい。この実行により、サンプル数(行)=999、調査項目(列)=103 の行列データが作成される。

さらにそれをもとに、文科系学生、理科系学生のデータ行列、および各学部別データを作成しておけば、各種の集計や比較検討を簡単に行うことができる。

	調査対象	
all		全サンプル 999 名
bunka	← all[1 :510,]	文科系 510 名のデー タの付置
rika	← all[511:999,]	理科系 489 名 〃
let	← bunka[1 :111,]	文学部 111 名 〃
hus	← bunka[112:193,]	人間科学部 82 名 〃
jul	← bunka[194:398,]	法学部 205 名 〃
eco	← bunka[399:510,]	経済学部 112 名 〃
sci	← rika[1 :152,]	理学部 152 名 〃
med	← rika[153:219,]	医歯薬学部(サンプル数小のため合併)67 名 〃
sig	← rika[220:371,]	基礎工学部 152 名 〃
tec	← rika[372:489,]	工学部 118 名 〃

ただし、S オブジェクトは info/ .Data に作成されているので、データ読み込みに先立って

> attach("info/ .Data")

を行い、データオブジェクト検索のためのパスを設定しておく。データベース使用終了後、系別·学部別データ等は消去してハードディスク資源を節約しておく。S オブジェクト all のみ書き込み禁止のプロテクトをかけておき、他はテンポラリーとしておく。RemoveData()で一括して系別·学部別データを消すことができる。

2 学部のデータを合併したいようなときは、行結合の関数 rbind()を用いればよい。引数としてはベクトルのみならず、行列も使用できるので、たとえば

lethus ← rbind(let, hus)

で文学部と人間科学部のデータを合併し、サンプル数 111+82=193 のデータオブジェクトを作成することができる。また、引数として、任意の個数が使用可能であり、3 学部以上を合併することもできる。行データは引数に書いた順で結合される。

以下、紙数の都合で全調査項目の一覧は省略するが、ファイル sheet.txt で詳細を見ることができる。項目番号を特定し、調査対象群を指定して、グラフィックス·コマンドを入力すれば、統計グラフが描かれる。

複数のグループについて描画し、両グループを比較したい場合には、グラフィックス画面をメモリーの許容範囲内で複数枚開き、各々にグラフを描けばよい。

6.2 1変量処理

6.2.1 平均値ベクトルの計算

学部別に各項目の平均を計算しておく。(0,1)回答の項目では選択比率、順序カテゴリの項目では、カテゴリ平均値が算出される。そのためのユーザー関数は

```
Mvec.sing( xvec, group )
```

である[8]。

学部別平均値ベクトルを行結合して、全データについての行列としておく。mq34はコンピュータ使用の実態[9]を結合したものであり、mq911は情報処理教育に関する意見[10]を結合したものである。

6.2.2 平均値のプロフィール

項目の平均値を用いてプロフィールを描き、学部間の比較を画面では色別に行う。先に平均値を求めているのでそれを呼んでデータとし、凡例、表題を記入する。比較するグループ数が2と4の場合に分け、psファイルに描くユーザー関数 Tsplot2(), Tsplot4()を作成した。結果の一例が 図10, 図11 である。

mq34に大学入学以後、自宅(項目1-5)と大学(6-11)におけるコンピュータ使用の実態が集計されているので、グラフにより比較する。

```
splot2(mq34[2,],mq34[7,], c("Hus","Sig"), "Q3-4")
```

で文·理科系学部の一例として人間科学部[2]と基礎工学部[7]とが比較され、理科系の方が使用の実態が高いことが示されている。

結果をモニターで眺め、特にファイルに書き込む必要がない場合には、標準関数を用いて比較するグループ·変数のベクトルのみを指定すればよい。グループ数も任意である。グラフが色分けされているので特に凡例の必要もない。

```
tsplot2(x1,x2)
```

により、様々な項目について学部の比較を試み、その特徴を探索すればよい。再度個々のデータを観察することにより、分析で気付かなかった特徴が見出されれば、分析を深化させることができる。全項目について入力するのが大変な場合には、あらかじめ入力分のファイルを作成しておき、ファイルから一括処理することも可能である。S ではこうした機能は十分にサポートされている。

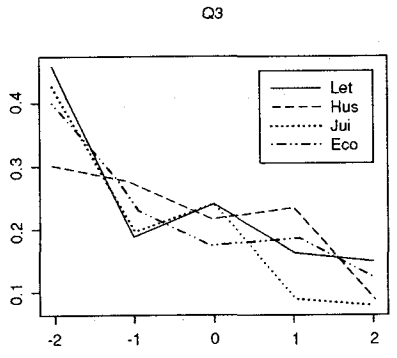

図10: 平均値プロフィール

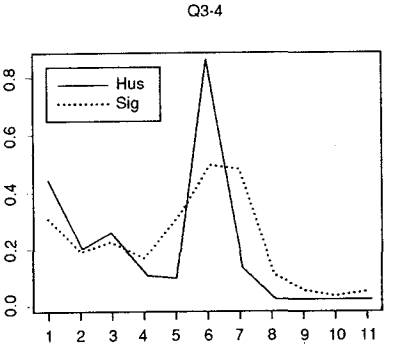

図11: 平均値プロフィール

6.3 多変量処理

6.3.1 対散布図

2つの変数間の関係は散布図・クロス集計図を描き検討すればよい。同一グループの異なる項目間の反応が取り上げられる。連続量では散布図、カテゴリ変数では集計された度数を第3変数として 3D−グラフが用いられる。前述の鳥瞰図(persp, Cplot)などが有用であろう。任意のグループ(学部、系)に対して、各種の項目を組み合わせて検討するのは、煩雑な作業となるので、プログラミングしておくことが望ましい。判定の基準を定め、一瞥の後、有用な情報のみを保存ないしはプリントすればよい。

多変数を2変数づつ組み合わせて、対散布図を描くことができる。連続量の変数が要請されるので、ここでは一例として Q8:コンピュータに対する意見に主成分分析を行い、その結果得られる主成分得点をプロットしてみる。標準関数は prcomp(x)であるが、データの編集、得られた主成分の構造、得点の学部別分割などの一連の作業が必要であり、これらを "Q8 prcomp.s" として作成した。

主成分の内容は、粗データとの相関から、'1.コンピュータに対する不安'、'2.コンピュータの現代社会に対する貢献であろうか[11]。本来主成分得点は相互に独立であるが、学部別ではスコアの高い場合、低い場合が現れる可能性があり、一例として図12のごとく基礎工学部に、第1主成分が低く第2主成分が高い、という偏りが見られた。こうした示唆をもとに、この学部の特徴の検討を始めることができる。他学部ではこれほどの特徴は見られず、データはほぼ均等に散布し、コンピュータに対してポジティブからネガティブまで様々な学生が存在することを示していた。

また対散布図ではなく単独の散布図に戻して描けば、関数 identify()を用いて外れ値(outli-

er)などを特定することができ、個々のサンプルレベルでの分析が可能である。

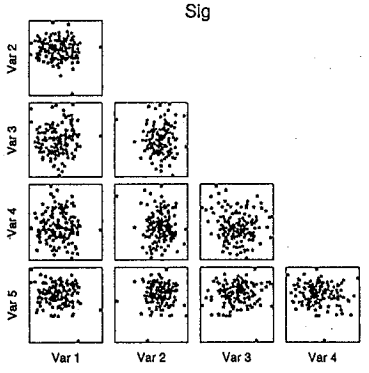

図12: 主成分得点の対散布図

6.3.2 フェイス・メソッド

多変量のグラフとしてChernoffのフェイス・メソッドがあるので、これを適用して同じくコンピュータに対する意見(Q8-1、ただし前半の15項目)を代表例として取り上げ、各学部の相違を検討してみる。

ユーザ関数(Face)が作成されているので、たとえばQ8 学部別平均値を用いて、

```
Face(mq81, "Q8. 1-15")
```

で、図13を描くことができる。顔の容貌に割り当てた変数は 表3の通りであり、これらを総合して各学部の顔が描かれた。ただし、顔の全体的(たとえば感情)印象と変数の示す内容とは必ずしも対応していない。両者をマッチさせるためにはさらに細かい変数の検討が必要であり、ここでは項目順に変数が機械的に割り当てられたにすぎない。しかし、((人・法・医), 文)が類似し、さらに((理・経), (基・工))が接近して一群をなしている様子が顔の相貌から分かる。

他の項目についても同様に、データ・オブジェクト名 mq* と "タイトル文字列" を入力することにより、容易にフェイス図を描くことができる。学部間の類似性や特徴などを全体的な印象として把握することができるので便利であろう。

表3:フェイス図に割り当てられた意見項目

Q8 コンピュータについていろいろな意見がありますが、どう思いますか.	
1.顔の面積	コンピュータはなんとなく親しみにくい.
2.顔の形	コンピュータが好きだ.
3.鼻の長さ	コンピュータには近寄り難い冷たさがある.
4.口の位置	自分にとってコンピュータといっても遠くの存在だ.
5.笑いの曲線	コンピュータはドライな感じがする.
6.口の幅	コンピュータの処理結果はすべて正しい.
7.両目の位置	コンピュータを盲信するのは危険だ.
8.両目の分離	コンピュータに対して恐怖感を持っている.
9.両目の角度	コンピュータを扱うのは不安だ.
10. 両目の形	コンピュータは健康に害をもたらす.
11. 両目の幅	キーボードの位置を知らないので、コンピュータに向かうのがおっくうである.
12. 瞳の位置	コンピュータの画面は小さくて不便だ.
13. 眉の位置	コンピュータの反応が速いのが快適だ.
14. 眉の角度	コンピュータを使うには覚えなければならないことが多すぎる.
15. 眉の幅	キーボードを叩いているとストレスの解消になる.

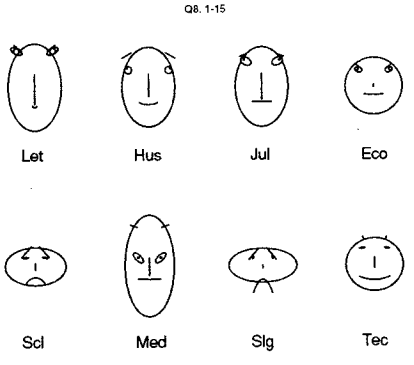

図13: フェイスメッソド結果

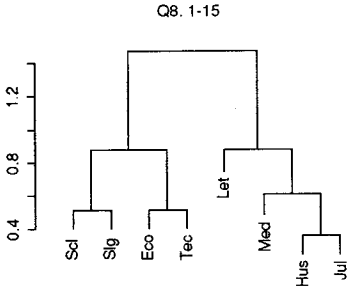

図14: クラスター分析結果

6.3.3 クラスター分析

フェイスメソッドで得られたのと同様のことを、クラスター分析により検討してみる。各学部間のユークリッド距離を求め、最長距離法(complete linkage)を用いてクラスター分析を行

った。ユーザー関数が作成されているので、入力は

```
Cluster(mq81, "Q8. 1−15")
```

でよい。

結果は 図14 であるが、両結果は平行関係にある。まず、人·法が類似しており、最短距離でクラスターをなし、それに医、文が接近していく。他方のグループとして、理·基、経·工が類似し、両者でまた一つのまとまりを示す。こうしたクラスター形成の経過は先のファイス図から読み取れるものと同じである。

クラスター分析では、距離の算出方法やクラスターの定義方法などにより、いろいろな方法が提案されているが、ここでは最もポピュラーな方法を用いた。オプションの設定により他の方法による分析ももちろん可能である。

6.3.4 星型図

いまひつつの多変量を図示する方法は星型図を描くことである。中心点から放射状に多変量の値を延ばしてグラフ化すればよいので、何変数でも取り扱うことができる。変数値が小さいときは原点に近く、大きいときは長い線を原点から引けばよい。

一例として、情報処理教育に関する意見をグラフ化したものが 図15である。文科系学部は総じて情報処理教育に関してネガティブであり、スコアが低い(原点からの距離が小さい)。それに対して理科系学部は、星型の面積が大(原点からの距離が長い)であり、情報処理教育に関してポジティブである。特に工·基礎工学部が大であり、医·理学部がこれに続く。文科系学部の中では、人間科学部がやや好意的であることがわかる。

変数は 図16に工学部の例を拡大して示すごとく、0度の位置から反時計回りで項目番号が増大する。各変数の内容は、実際には略号でラベルをふることにより、図形の中に記入することができるが、ここでは省略し、全体的印象を眺めるに止めることとする。

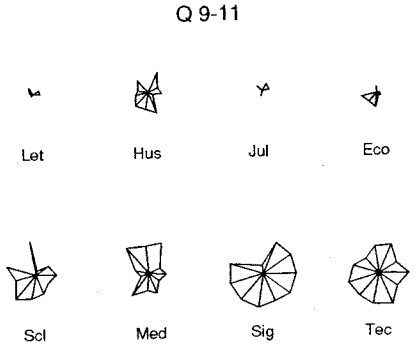

図15: 星型図(全体)

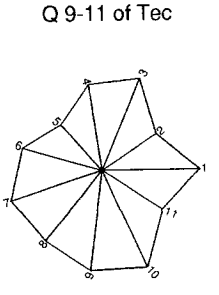

図16: 星型図(特定グループ)

こうした分析はその一例であって、本視覚化データベースは調査データの保存と同時にそうした作業を支援するための環境である。S と対話しつつ試行を繰り返し、有用な情報が得られれば、それを基盤にさらに次の分析へと進めばよい。プレゼンテーションよりも分析を深化させるのが、本来の目的である。

7 Appendix: ユーザー関数ソース·プログラム

本データベースのために作成されたユーザー関数のソース·プログラムを以下に記載する。内蔵の標準関数と区別するため、大文字で始まる名前である。いずれもデバッグずみであり、ディレクトリ ～/ s /myfunc(統計処理関係)または ～/info(情報処理教育データベース関係)に保存されている。

```
# *** Statistis Functions ***
  Range  ← function(x){max(x)-min(x)}
  RangeQ ← function(x,q){quantile(x,q/2)-quantile(x,(1-q)/2)}
  Var  ← function(x){sum(x^2)/length(x)-(sum(x)/length(x))^2}
  Std  ← function(x){sqrt(sum(x^2)/length(x)-(sum(x)/length(x))^2)}
  Uvar ← function(x){(sum(x^2)-(sum(x)^2/length(x))/(length(x)-1)}
  Ustd ← function(x){sqrt((sum(x^2)-(sum(x)^2/length(x))/(length(x)-1))}
  Cov  ← function(x,y){sum(x*y)/length(x)-sum(x)*sum(y)/length(x)^2}
  Cor  ← function(x,y){Cov(x,y)/(Std(x)*Std(y))}

# *** Matrix Operations ***
# "Matp.x.s" Matrix percent divided by row(x)-total
   Matp.x ← function(x)
     {sx1 ← x%*%c(rep(1,ncol(x))); t(t(x)%*%solve(diag(c(sx1))))}

# "Matp.y.s"  Matrix percent divided by column(y)-sum
   Matp.y ← function(x)
     {sx2 ← t(x)%*%c(rep(1,nrow(x))); x%*%solve(diag(c(sx2)))}

# "Matp.xy.s"  Matrix percents divided total(xy)
  Matp.xy ← function(x)
     {ns ← sum(x%*%c(rep(1,ncol(x)))); x/ns}

# "Det.s"  Absolute(determinant) of symmetric matrix
    # Det ← function(mat) {prod(eigen(mat)$values) } # symmetry
    Det ← function(mat) {prod(svd(mat)$d)} # asymmetry

# "Tr.s" Trace of matrix
    Tr ← function(mat) {sum(diag(mat))}

# "Makemat.s"
  # Make minor matrix from mat by pick up c(i,...)
  # cc ← c(i,...)
```

```
    Makemat ← function(mat,cc) {mm ← mat[cc,cc]}

# "Minor.s" Minor matrix, delete of i-th row and j-th column
  # Minor(mat,i,j) dim=ncol-1
    Minor ← function(mat,i,j) { mm ← mat[-i,-j]}

# "Cofactor.s" Cofactor of symmetric matrix
  # Cofactor(mat,i,j); delete i-th row and j-th column, dim=ncol-1
    Cofactor ← function(mat,i,j) {
        mm ← Minor(mat,i,j); Det(mm)*(-1)^sum(i+j)}

# *** Multivariate Correlations ***
# "M.cor.s" Multiple correlation
  # M.cor(cor,i,vec), multiple cor(i ← vec)
    M.cor ← function(cor,i,vec) {cc ← Makemat(cor,c(i,vec))
      mcor ← sqrt(1-Det(cc)/Cofactor(cc,1,1))}

# "P.cor.s" Partial correlation
  # P.cor(cor,i,j,c(k,...)), partial cor(i,j) with c(k,...)=const
    P.cor ← function(cor,i,j,vec) {
      pvec ← c(i,j,vec); cc ← Makemat(cor,pvec)
      pcor ← -(Cofactor(cc,1,2))/sqrt(Cofactor(cc,1,1)*Cofactor(cc,2,2))}

# "p.cor.s" Partial correlation by R.Shibata
  # p.cor(x,i:j), partial cor(i:j) with others=const of data-matrix x(n*p)
    p.cor ← function(x,among=1:2) {
      y ← x[,among]; varname ← dimnames(y)[[2]]
      g ← x[,-among]; s ← cor(lsfit(g,y)$resid)
      dimnames(s) ← list(varname,varname); s}

# "Can.cor.s"  Cannonical correlation
  # Can.cor(mat,vec1,vec2), length(vec1) < length(vec2)
    Can.cor ← function(mat,vec1,vec2) {
      r11 ← mat[vec1,vec1]; r12 ← mat[vec1,vec2]
      r21 ← mat[vec2,vec1]; r22 ← mat[vec2,vec2]
        cr ← solve(r11)%*%r12%*%solve(r22)%*%r21
        cc ← sqrt(eigen(cr)$values)}

# *** Graphics ***
# "Bar.1.s" Barchart of 1-keta data
  # x=Category Data given by Seisu(1..30) vector
  # class=automatically counted
  Bar.1 ← function(x) {xmax ← max(x); data ← tabulate(x,xmax)
    # print(table(data))
    print(data)
     cat("Mean=",mean(x),"\n"); cat("SD=",Std(x),"\n")
     cat("Max=",max(x),"\n"); cat("Min=",min(x),"\n")
  barplot(data,names=as.character(1:max(x)))
```

```
    # box(2,col=1)
    # barplot(data,col=seq(data),names=as.character(1:max(x)))
    # box(3,col=4)
  }

# "Bar.mat.s" Bar chart of x=matrix frequency data
   # Row=Category,Column=sub.category,x=data matrix
   # fp="f" frequency,fp="p" percent
   Bar.mat <- function(x,fp) {
     if(fp=="f") { print(x); barplot(x,names=as.character(1:ncol(x))) }
     if(fp=="p") { sm <- t(x)%*%c(rep(1,nrow(x)))  # sum of column
       xx <- x%*%solve(diag(c(sm)))   # percent matrix
       print(x);  print(xx)
         #  print("Raw Matrix Data\n",x)
         #  print("Raw Percent Data\n",xx)
       barplot(xx, names=as.character(1:ncol(x)))
         # barplot(xx,col=seq(data), names=as.character(1:ncol(x)))
     }}

# "Hist.1.s" Histgram of x=1-keta data vector
   # class=automatically counted
   Hist.1 <- function(x) {data <- x
     print(table(data)); options(digits=8)
       cat("Mean=",mean(data),"\n"); cat("SD=",Std(data),"\n")
       cat("Max=",max(data),"\n"); cat("Min=",min(data),"\n")
     hist(data)}

# "Hist.cont.s" Histgram of x=continuous data vector
   # class=class upper bound, e.g. class <- c(-3:4)
   Hist.cont <- function(x,class) {data <- cut(x,class)
     print(table(data))
       cat("Mean=",mean(x),"\n"); cat("SD=",Std(x),"\n")
       cat("Max=",max(x),"\n"); cat("Min=",min(x),"\n")
     hist(data)}

# "Pie.1.s" Pie chart of 1-kata vector
    # x=data vector, cname=automatically assigned
    Pie.1 <- function(x){xmax <- max(x); data <- tabulate(x,xmax)
      print(data)
        cat("Mean=",mean(x),"\n"); cat("SD=",Std(x),"\n")
        cat("Max=",max(x),"\n"); cat("Min=",min(x),"\n")
      pie(data,name=as.character(1:max(x)),rotate=F,col=seq(data),inner=1.5)}

# "Pie.cont.s" Pie chart of continuous data
    # x=continuous data vector, class=class upper bound, e.g. class <- c(-3:4)
    Pie.cont <- function(x,class){xdata <- cut(x,class)
      xmax <- max(xdata); fdata <- tabulate(xdata,xmax)
      print(fdata); print(table(xdata))
```

```
      cat("Max(xdata)=",max(xdata),"\n"); cat("Min(xdata)=",min(xdata),"\n")
      cat("\nMean=",mean(x),"\n"); cat("SD=",Std(x),"\n")
      cat("Max=",max(x),"\n"); cat("Min=",min(x),"\n")
    pie(fdata,name=as.character(1:max(xdata)),
      rotate=F,col=seq(fdata),inner=1.5)}

# "Cplot.s" Cross tabulation of category vectors(x,y)
  # cc=color(1=black,2=red,3=green,4=blue,5=yellow,6=cyan,7=magenta)
  Cplot ← function(x,y,cc) {
    print("Xsum="); print(table(x))
    print("Ysum="); print(table(y))
    print("Total n=")
    ns ← sum(table(x)); print(ns)
    xy ← table(x,y); print("Cross-Tabulation"); print(xy)
    xy1 ← xy/ns; print(xy1)
    persp(xy1,eye=c(10,0,5),col=cc)}

# "Tsplot2.s" Graph of 2-groups(vec1,vec2)
  # leg ← c("name1","name2"), head ← "Header"
  Tsplot2 ← function(vec1,vec2,leg,head) {
    tsplot(vec1,vec2,lwd=2)
      legend(locator(1),legend=leg,fill=1:2); title(main=head)
    postscript("tsplot.ps",width=6,height=5.5)
    tsplot(vec1,vec2,lwd=3)
      legend(1,max(vec1,vec2),legend=leg,fill=1:2); title(main=head)
    graphics.off()}

# "Tsplot4.s" Graph of 4-groups(vec1,vec2,vec3,vec4)
  # leg ← c("name1","name2","name3","name4"), head ← "Header"
  Tsplot4 ← function(vec1,vec2,vec3,vec4,leg,head) {
    tsplot(vec1, vec2,vec3,vec4,lwd=2)
      ymax ← max(vec1,vec2,vec3,vec4)
      legend(1,ymax,legend=leg,fill=1:4,ncol=4)
        # legend(locator(1),legend=leg,fill=1:4,ncol=4)
      title(main=head)
    postscript("tsplot.ps",width=6,height=5.5)
    tsplot(vec1,vec2,vec3,vec4,lwd=4)
      legend(1,ymax,legend=leg,fill=1:4); title(main=head)
    graphics.off()}

# "Face.s" Face method
  # x=matrix(obs,variables), xhead="Header"
  Face ← function(x,xhead) {
    postscript("face.ps",width=6,height=5)
    faces(x,byrow=T,ncol=4,head=xhead,
      labels=c("Let","Hus","Jul","Eco","Sci","Med","Sig","Tec"))
    graphics.off()}
```

```
# "Cluster.s" Cluster analysis
  # x=matrix(obs,variables), xhead="Header"
  # x.dist=distance between row
  Cluster ← function(x,xhead) {x.dist ← dist(x)
    postscript("clust.ps",width=6,height=5)
    plclust(hclust(x.dist),
      labels=c("Let","Hus","Jul","Eco","Sci","Med","Sig","Tec"))
      title(xhead)
    graphics.off()}

# "Stars.s" Stars diagram
  # x=matrix(obs, variables), xhead="Header"
  Stars ← function(x,xhead) {
    lab ← c("Let","Hus","Jul","Eco","Sci","Med","Sig","Tec")
    stars(x,ncol=4,byrow=T,labels=lab)
    postscript("stars.ps",height=5,width=6)
    stars(mat,ncol=4,byrow=T,labels=lab); title(main=xhead)
    starsymb(x,sample=2); title(main=paste(xhead," of ",lab[2]))
    graphics.off()}

# *** Infomation Database ***
# initialization
    .First ← function() {attach("~/s/myfunc"); attach("~/info")
       x11(); print(attach())}

# remove objs
    RemoveData ← function(x) {
      rem(bunka,rika,let,hus,jul,eco,sci,med,sig,tec) }

# "Mvec.sing.s"
  # Mvec.sing(xvec,group)
  # xvec=items vector, group=sample obj(group)
     Mvec.sing ← function(xvec,group) {
       q1 ← NULL; for(el in xvec) q1 ← c(q1,mean(group[,el]))}

# "Qmeanvec.s"
  # Mean scores of Q-items
    Qmeanvec ← function() {vall ← c(1:103)
       g1 ← Mvec.sing(vall,let); g2 ← Mvec.sing(vall,hus)
       g3 ← Mvec.sing(vall,jul); g4 ← Mvec.sing(vall,eco)
       g5 ← Mvec.sing(vall,sci); g6 ← Mvec.sing(vall,med)
       g7 ← Mvec.sing(vall,sig); g8 ← Mvec.sing(vall,tec)
     meanall ← matrix(c(g1,g2,g3,g4,g5,g6,g7,g8),ncol=103,byrow=T)
       mq1 ← meanall[,6:15];   mq2 ← meanall[,17:20]
       mq3 ← meanall[,21:25];  mq4 ← meanall[,26:31]
       mq5 ← meanall[,33:40];  mq6 ← meanall[,42:49]
       mq81 ← meanall[,52:66]; mq82 ← meanall[,67:81]
       mq9  ← meanall[,82:92]; mq11 ← meanall[,93:103]
```

```
    mq34  ← cbind(mq3,mq4)
    mq911 ← rbind(meanall[1:4,82:92], meanall[5:8,93:103]) }

# "Q8prcomp.s" PCA of Q8
  # the first 5 compornents and pairs plot
  Q8prcomp ← function() {xdata ← all[,52:81]; dim(xdata)
    fscore ← prcomp(xdata)$x[,1:5]
    cc ← cor(cbind(fscore,xdata))[,1:5]
      dump("cc", fileout="fscorr.out")
    g1 ← fscore[1:111,]; dim(g1);   g2 ← fscore[112:193,]; dim(g2)
    g3 ← fscore[194:398,]; dim(g3); g4 ← fscore[399:510,]; dim(g4)
    g5 ← fscore[511:663,]; dim(g5); g6 ← fscore[664:729,]; dim(g6)
    g7 ← fscore[730:881,]; dim(g7); g8 ← fscore[882:999,]; dim(g8)
    postscript("pairs.ps", height=6, width=6)
      pairs(g1); title(main="Let"); pairs(g2); title(main="Hus")
      pairs(g3); title(main="Jul"); pairs(g4); title(main="Eco")
      pairs(g5); title(main="Sci"); pairs(g6); title(main="Med")
      pairs(g7); title(main="Sig"); pairs(g8); title(main="Tec")
    graphics.off()}
```

注

1）大阪大学大型計算機センターではワークステーションの最新版が利用可能である.

2）ユーザーが特定のディレクトリを付置した場合にはシステム関数より高順位となる.

3）発生方法については後述の統計的分布を参照.

4）データの様子を探索的に眺めるときは作図のみ、プレゼンテーション用のグラフの場合にはグラフの重ね描き、説明の追加などを行う.

5）S に内蔵の関数に汎用性があり、特に他の処理を追加して自作する必要もないように思われるので、標準関数としてそのまま使用することとする.

6）データの使用を御快諾下さいました情報処理教育センター・松浦敏雄助教授に感謝いたします.

7）ODINS(大阪大学総合学術情報通信システム)の稼働後はそれに接続することにより公開可能である.

8）以下、データベース用のユーザー関数を *Name*() とするとき、そのソースプログラムは *Name.s* であり ～/info に保存されている.

9）Q 3 :自宅, Q 4 :大学.

10）Q 9 :文科系の学生が文科系の学生のための教育をどう評価するか, Q11:理科系の学生が理科系の学生のための教育をどう評価するか.

11）バリマックス回転などの回転を行っていないので、寄与率は急速に減少する。しかし、これらの2つの主成分に類似するものは因子分析の結果[16]からも得られている.

文　　献

[1]アイザック (1989)S 言語(入門編), ISAC.

[2]アイザック (1991)S Version 3 (使用の手引き, 定型統計処理), ISAC.

[3]安芸重雄·地道正行 (1994)データ解析言語 S について, 大阪大学大型計算機センター S 講習会資料, 1－46.

[4]Becker,R.A. & Chambers,J.M. (1984) *S: An Interactive Environment for Data Analysis and Graphics,* Wadsworth.

[5]Becker,R.A., Chambers,J.M. and Wilks,A.R. (1988) *The New S Language,* Wadsworth & Brooks, 渋谷政昭·柴田里程, S 言語 I, II, 共立出版.

[6]Chambers,J.M. (1977) *Computational Methods for Data Analysis,* John Wiley.

[7]Chambers,J.M. et al (1983) *Graphical Methods for Data Analysis,* Chapman.

[8]Chambers,J.M. & Hastie,T.J. (1992) *Statistical Model in S,* Wadsworth & Brooks,柴田里程, S と統計モデル, 共立出版.

[9]Cleveland,W.S. (1985) *The Elements of Graphing Data,* Wadsworth.

[10]Cleveland,W.S. & McGill,M.E. (1988) *Dynamic Graphics for Statistics,* Chapman.

[11]Cook,R.D. & Weisberg,S. (1994) *An Introduction to Regression Graphics,* John-Wiley.

[12]原田章·松浦敏雄 (1994)1993年度オンラインアンケート調査結果報告, 大阪大学情報処理教育センタ広報, 11, 59–74.

[13]河合徹 (1990)データ解析システム S の機能拡張, 計算機統計学, 3, 73–78.

[14]Kernighan,B.W.& Pike,B. (1984) *The UNIX Programming Environment,* Englewood Cliffs.

[15]松浦敏雄 (1994)「情報活用基礎」必修化への道, 大阪大学情報処理教育センター広報, 11, 3－8.

[16]松浦敏雄他 (1994)教育用計算機システムの運用と学生の意識－大阪大学情報処理教育センターにおける事例－, 行動計量学, 40, 17–31.

[17]水田正弘 (1990)視覚的操作によるデータ解析システムについて, 計算機統計学, 3, 23–29.

[18]柴田里程 (1991)S の開くニューパラダイム, 行動計量学, 18, 57–63.

[19]渋谷政昭·柴田里程 (1992)S によるデータ解析, 共立出版.

[20]Tierney,L. (1990) *Lisp-Stat: An Object-oriented Environment for Statistical Computing and Dynamic Graphics,* Wiley.

[21]Turkey,J.W. (1977) *Exploratory Data Analysis,* Addison-Wesley.

[22]上野喜正·町井昌徳 (1992)SToolkit: S言語上のGUI構築ツール, 計算機統計学, 5, 59–62.

[23]Venables,B. & Smith,D.(1992) *Notes on S-PLUS: A Programing Environment for Data Analysis and Graphics,* Department of Statistics, The University of Adelaide.

[24]吉田光雄 (1993) *Mathematica* による統計学入門, 大阪大学情報処理教育センター広報, 10, 146–170.

[25]吉田光雄 (1994)SAS による社会調査結果の視覚化データベース, 大阪大学人間科学部紀要, 20, 107–142.

[26]吉田光雄 (1994) *Mathematica* による社会調査結果の視覚化データベース, 第 8 回日本計算機統計学会論文集, 51–54, 1994.

[27]脇本和昌他 (1979)多変量グラフィックス解析法, 朝倉書店.

[28]渡辺洋他 (1985)探索的データ解析入門, 朝倉書店.

Visualized Social Survey Database by S Language

Mitsuo YOSHIDA

Social survey results are usually preserved as database to consult them afterwards and to do another analysis. If it can be browsed in a visual format, such a database seems to be a convenient tool for many purposes of social studies. One of the current topics in statistics is VDA (Visual Data Analysis) and many methods are available in computers.

Called as "*A Programming Environment for Data Analysis and Graphics*" and because of its high-level language and interactive programming environment for UNIX system, the *S Language* is now considered one of the largest and most powerful applications for advanced statistical usage. It provides an integrated environment for data analysis, numerical calculations, graphics and related computations.

Computations and programming take place through the S-System, but it is easily extended to user specific environment. Although fundamental statistical techniques are constructed as commands or functions in the system, many other advanced or user-oriented statistical methods are programmed and preserved systematically as user functions.

This report summarized how to save and edit social survey data in *S* and how to construct statistics and visualized database.

As an example, a social survey result concerning computer education at Osaka University is adopted and demonstrated.

This database system would be useful, not only to preserve all of the data, but as a visual database that contributes as an EDA (Exploratory Data Analysis) revealing covert structures of the data.