統計数理（1997）
第45巻 第1号 1-11

特集「統計ソフトウェアの新展開」
［原著論文］

重回帰分析支援システムRASSにおけるプログラミング言語

徳島文理大学* 小林郁典
一橋大学** 中野純司

（1996年6月 受付）

1. はじめに

計算機科学におけるいわゆる人工知能やエキスパートシステムの研究の進展にともない，データ解析の分野でも，知識処理機構を持ち，自動解析や手法選択の支援などを行える知的システムの研究が進められてきた．われわれの開発した重回帰分析支援システムRASS（Regression Analysis Supporting System）（中野 他（1991））もそのひとつである．RASSは，グラフィックス表示を含む重回帰分析の基本的な手法を容易に実行でき，またそれらの明らかな誤用をシステム自身ができるだけ防止することを目標として作成された．すなわち，重回帰分析の基本的統計計算機能の他にそれらの利用に関する統計的知識も組み込まれており，利用者が統計手法の選択に困難を感じるような場合には，それまでの解析結果に応じてある程度の助言を行うことができる．また，RASSの知識だけを用いた自動解析も可能である．

ところで，一般にシステム利用者の要求は多様であり，どのように多くの機能をシステムに持たせようと，それですべての利用者を満足させることはできない．したがって多くのシステムはマクロ機能または拡張用言語を持ち，利用者が自分自身のための新しい命令をプログラミングできるようにしている．データ解析においても，ひとつの仮説を検証するためにも複数の統計量が考えられ，それらのすべてをはじめから組み込んでおくことは不可能であるから，拡張用言語は不可欠なものであろう．

しかしながら，これまで発表された知識処理機構を持ったデータ解析システムにおいて，機能拡張用言語を備えたものはあまり見受けられない（Hand（1993））．その理由のひとつとして考えられることは，このようなシステムにおいては新しい計算手法を作成する場合，内部構造に合致するようにプログラムする必要があるということである．すなわち，単に新しい計算手法を定義できるだけでは知的システムの拡張用言語としては不十分であり，その言語でプログラムを書けば自然にそれまでのシステムと整合的になっているようなものでなければならない．ところが，知的システムにおいては統計計算手法とその利用のための知識が複雑に関係付けられているため，そのような言語を実現することが容易ではなかったと思われる．

RASSは，重回帰分析の標準的な順序という知識を表現し，またそれに関する統計手法を解析の段階に応じて適用できるように，オブジェクト指向の考え方を基本として設計されている．そのため，手法と知識は最初から自然に結びつけられている．これらの機能を実現する際，システムの作成には論理型言語であるPrologを用い，数値計算の部分はC言語でプログラムしてそれをPrologの新しい述語とした．したがって，新しい統計計算機能を組み込むことは可能

* 工学部：〒769-21 香川県大川郡志度町堂林1314-1．
** 経済学部：〒186 東京都国立市中2-1．

ではあるが，そのためにはProlog, C両言語とシステム全体に対する詳細な知識が要求される．これでは一般ユーザがそれを実行することは難しい．すなわち，拡張用言語はないと言わざるをえない．

そこでわれわれは，利用者ができるだけ簡単に新しい統計計算機能を追加できるようにする目的でRASS言語を開発した．その設計にあたっては，すでに実現されている部分と自然に整合性を保つこと，重回帰分析において多用される行列計算が容易に記述できること，論理型プログラミングと手続き型プログラミングが混在できること，などに注意した．

本論文では，知的データ解析支援システムに求められるプログラミング言語の条件について考察するとともに，その具体例としてRASS言語を説明する．

2. 知的データ解析システムにおけるプログラミング言語

データ解析は，与えられたデータを加工していくつかの統計量を計算し，それらを見てデータの生成過程に関するなんらかの判断を下すという操作を，満足のいく結果が得られるまで繰り返し実行することである．知的データ解析システムは，このうちの統計計算作業のほとんど全てを実行するとともに，利用者の判断を助け，また，システム独自の判断をも提供する．

データ解析においては，なによりもそのデータに固有の知識が重要である．データ生成過程の固有技術的性質から，データには多くの制約が課されているのが普通であり，それらを考慮しない統計解析には意味がない．このようなデータに関する知識は個々のデータごとに特有のものであり，汎用システムの中に一般的知識として保存しておき，後で再利用することは難しい．また，これを統計解析手法と結びつけ，意味のある解析を行うことは，統計学およびその固有技術分野の広範な知識を持つ“専門家”でなければできない．それと同じ作業を現在の知識工学の手法によって計算機上で実現することはほとんど不可能であろう．このため，知的データ解析システムに可能な判断は，統計学の手法に関する限定的なものとならざるを得ない．したがって，このようなシステムに現在要求されていることは，まったく新しい種類のデータを解析して専門家なみのよい解析結果を得ることではなく，むしろ定型的な統計処理に対して大きな誤用がないようにすることであると考えられる．

ただし，このように“保守的な”システムにおいても，新しい計算手法を組み込む手段としての言語は必要である．新しい有効な統計量が開発されたときにはそれをシステムに付加したいし，システムを利用して解析するデータが同じようなものばかりである場合には，固有技術から示唆される特殊な変数変換を組み込みたいこともあるだろう．

これまで多くのデータ解析システムにおいては，拡張用言語はその記述の自由度をできるだけ大きくする方向に進化してきた．例えばS (Becker et al. (1988)) はもはや統計システムと言うより，統計関数とグラフィックス機能が豊富な汎用言語である．特にデータに対して種々の統計計算を試行錯誤的に容易に行える環境となることを重点において設計されている．そのため，プログラムとしては少々あいまいなものでもシステムが既定値として自動的に補うことによりできるだけ動作させようとする．

知的データ解析システムの拡張用言語としては，それほどの柔軟性は不必要である．思いついた手法をすぐ実行できることより，その手法とシステムにすでに貯蔵されている知識や推論機構との関係に注意しながら，汎用的に使える形で記述することのほうが重要である．それまでのシステムと整合的に新しい手法を組み込むことが比較的簡単に行えるという条件の下でのみ，自由度を高めなければならない．

もちろん，このことは言語の記述能力が低くてもよいということを意味するわけではない．利用者の要求を前もって予測することはできないので，言語は基本的には“なんでもできる”

能力がなければならない．具体的にはシステム自身をその言語で再構成できるくらいの記述能力がなければ不便であろう．

さて，データを加工する際には一般に複雑な数値計算が必要である．その手順はアルゴリズムとして厳密に定義される．それをプログラムとして記述するには，FortranやC言語のような手続き型言語が便利である．また，行列計算は多用されるので，組み込み言語にはそれに対する配慮があることが望ましい．

一方，知的システムとして，計算された統計量からある判断を下すには，統計量の値の判断に関する知識データベースおよび推論のための機構が必要である．一回の判断に関する知識は簡単なアルゴリズムとして定義できるくらいのものであるが，解析が進んでいく過程においては統計量の値によって非常に多くの場合わけを経ることになる．したがって，全体を手続き型のアルゴリズムとして記述することは困難であり，いわゆるエキスパートシステム技術を利用した知識データ表現と推論機構で実現することになる．そして，これらの機能はリスト処理とパターンマッチングによってプログラミングするのが便利であることが知られており，それらの操作が容易であるように設計されたLispやProlog言語などを用いて実現することが多い．

したがって，知的データ解析システムの拡張用言語としては，統計計算のための手続き型の部分と知識処理のための部分が自然なかたちで結び付いているように設計するべきであろう．

3. RASS言語の特徴と文法

RASSはオブジェクト指向の考え方で設計されている．それはオブジェクト指向プログラミングが知識表現として使いやすいということが主たる理由である．重回帰分析で用いられるデータや統計量を解析の段階に応じて分類し，それらをクラスオブジェクト（以後，クラスという）のスロットとしてまとめると，解析の順序に関する知識は自然に表現されることになる．また，適用できる統計計算手法はクラスに付随するメソッドとして定義しており，それは解析のどの段階でどの手法を用いるべきかという知識をあらわすことにもなる．解析を進めるには，クラスの実現であるインスタンスオブジェクト（以後，インスタンスという）にメッセージを送るという操作を繰り返す．

RASS言語も基本的にこの形式をそのまま踏襲する．すなわち，新しい統計手法はどれかのクラスのメソッドとして定義する．そのために必要な統計量はスロットとして保存する．ただし前節で考察したように，統計計算（特に行列演算）とパターンマッチングに関してはそれらが容易に書けるように考慮した．

3.1 構文規則

RASS言語の構文は，図1の形式に従う．図中の〈 〉で囲まれた箇所が，利用者によって定義される部分である．プログラムの最初で，追加しようとする統計量あるいは手続きの属するクラス名を〈CLASS〉で指定する．このクラス名はすでに定義されているもの（主としてRASSで定義されているクラス）でも，新たに定義しようとするクラスでもよい．〈SUPER CLASS〉は対象となるクラスの親クラス（上位クラス）であり，親クラスのすべてのスロットやメソッドが子クラス（下位クラス）でも有効であることはオブジェクト指向言語の特徴である．

スロットはオブジェクトの内部データであり，追加したい場合には，slot( )の中で定義する．ここでは2つのスロットを記述しているが，このように複数のスロットをコンマ(,)で区切って定義できる．〈SLOT NAME〉がスロットの識別子であり，スロット名とも言う．type_〈TYPE〉で，スロットのデータ型（すなわちクラス名）を指定する．指定できるデータ型については後で述べる．スロットのデータ型は省略してもよく，この場合，スロットに最初に代入さ

```
<CLASS> is_subclass_of <SUPER CLASS> : (
  slot(
    <SLOT NAME> type_ <TYPE> slot_comment_ <COMMENT>,
    <SLOT NAME> type_ <TYPE> slot_comment_ <COMMENT>
  ),
  method(
    <METHOD NAME> is_defined_by (
      require( <事前条件> ),
      do( <実行部> ),
      explanation( <説明> )
    ),
    <METHOD NAME> is_defined_by (
      require( <事前条件> ),
      do( <実行部> ),
      explanation( <説明> )
     )
  )
).
```

図1. RASS言語の構文形式.

れるデータの型がそのスロットのデータ型となる. slot_comment_〈COMMENT〉には, コメントを記述する. 〈COMMENT〉は, 説明文字を ' と ' で囲んだものである.

手続きであるメソッドの記述は, method( ) 内で行う. 図中では2つのメソッドを記述しているが, スロットと同様, 1つ以上であれば同時にいくつ記述しても構わない. 〈METHOD NAME〉がメソッドの識別子である. もし, メソッドに引数を持たせたい場合には, メソッド名の後に括弧を用いて記述する. 仮引数としては大文字で始まる名前を用いる. 例えば,

```
method(
    example(X,Y) is_defined_by (
      ...
    ),
    example(X) is_defined_by (
      ...
    )
)
```

のようにする. メソッド名が同じでも引数の数が異なれば, 別のメソッドとして区別される.

メソッド内部の記述は, require( ), do( ), explanation( ) の3つのブロックに分けて行う. require( ) ブロックは, そのメソッドを実行する際に, 守られていなければならない条件を記述する部分である. ここにはスロットの値を変更するような実質的な処理を書くことはできず, 次の do( ) ブロックの処理が行える状態にあることを確認するための文のみを書く. この部分が満足されなければ, いかなるオブジェクトの状態も変化しないでメソッドは終了する. このように処理に入るための条件を明確にしておくことは, 表明を行うことであり, 正しいソフトウエアを書くために有効であることが指摘されている (Meyer (1990)). do( ) ブロックは, 処理を行うためのメソッドの本体を記述するところである. require( ) ブロックとは異なり, 正常終了しない場合にもオブジェクトの状態が変化することがある. すなわち, いくつかのスロットの値が変更されるかもしれない. explanation( ) は, メソッドの説明をする箇所である. このようにコメントをプログラムの一部として持っておくと, 後で説明などに利用することが

できる．なお，require( )，explanation( ) ブロックは省略されることもある．

3.2 実 行 文

do( ) ブロックにおける実行文の主たるものは，メッセージ送信と代入であり，これらはコンマで区切れば複数指定できる．

メッセージ送信は

sm(〈インスタンス名〉，〈メッセージ名〉，〈引数1〉，〈引数2〉，...)

の形をとり，メッセージ名はメソッド名であることが多い．これはRASSにおける命令の基本的な形である．

代入の記号としては：=を使い，左辺には変数名，右辺に数式を書く．変数については次節で述べる．

そのメソッド内だけで利用するローカル変数が必要ならdo( ) ブロックの最初でlocal_variable( ) によって宣言する．変数の型を指定することもできる．

また，Prologのプログラムを書けばそのまま実行されるので，それによって言語の機能を拡張することもできる．

3.3 変　　数

RASS言語における変数には2種類ある．ひとつはオブジェクト変数であり，これはスロットやローカル変数のことである．もうひとつはProlog変数であり，メソッドの仮引数や，sm( ) から返された値を利用するために用いる．

オブジェクト変数は，システムに既存のクラスや利用者が作成したクラスの実体であるインスタンスをさす．統計解析，特に行列計算のプログラムが容易になるようにいくつかのクラスが定義されており，それらをスロットまたはローカル変数の型として使うことができる．その例を表1に示す．

大文字，または下線で始まる文字の並びで識別されるProlog変数は型をもたない．この変数は値が“設定されている”状態か“設定されていない”状態かの2つしかなく，1度ある値に“設定される”とそのルーチン内での設定のやり直しはできない．実際にはメソッド間，またはメッセージ送信の結果を授受するために用いる．またPrologのプログラムを埋め込んだときにそれとの入出力のためにも利用する．したがってこの変数はRASS言語と（それを実現している）Prolog言語の言語間インターフェースと考えてもよい．

表1．変数のためのクラス．

クラス	意味
number	数値 (実数, 整数)
string	文字列
vector	ベクトル (成分は数値)
svector	ベクトル (成分は文字列)
lvector	各成分にラベルを持った vector
lsvector	各成分にラベルを持った svector
matrix	行列 (成分は数値)
smatrix	行列 (成分は文字列)
lmatrix	行と列にラベルを持った matrix
lsmatrix	行と列にラベルを持った smatrix

3.4 行列演算

重回帰分析，または多変量解析においては，行列を頻繁に利用する．RASS言語ではよく利用する行列の操作を関数としてあらかじめ組み込んである．例えば，転置行列を求めるtr( )，逆行列を求めるinv( )，対角成分を取得するdiag( )，行列式を求めるdet( )，各成分の平方根をとるsqrt( )等がある．

さらに，行列の成分を指定するための便利な機能を備えている．ある行列の一つの成分を指定する場合は，一般的なプログラミング言語と同様に，

変数名（行番号，列番号）

で指定する．また，リストを使用して一度に複数の成分を指定することもできる．ここで言うリストとは，[ ] で囲んだ数字や識別子の並びである．例えば，2行目の2列と4列を指定するには，a(2,[2,4]) と記述する．さらに，* や ~ を使用した指定方法もある．* は対応する行や列の全ての成分を指定する．例えば，a(*,2) は，2列目の成分全てを指定する．~ は，行(列)番号やそのリストとともに用いて取り除く成分を指定する．a([1,2],~2) は1行目と2行目の全ての成分の中で，2列目を除いた成分を指定することになる．また，ラベルを持った行列やベクトルでは，成分番号のかわりにラベルで指定することもできる．

3.5 制御構造

RASSの制御構造としてはif文とwhile文がある．

条件判断のためのif文の形式は

if(関係式，文の並び)

であり，括弧内の関係式が成立した時に，文の並びを実行する．文の並びは文をコンマで区切ったもので，文の数に制限はない．関係式は，比較演算子を用いて2つのスカラの大小関係を記述したものか，式を記述したものである．式が与えられた場合は，その評価値がスカラで正の値を持てば文の並びを実行する．

繰り返しにはwhile文を用いるが，その形式は

while(関係式，文の並び)

であり，関係式が満足された時に，文の並びを順に実行する．そして，実行が終れば再び関係式の評価が行われる．関係式の評価は，if文の場合と同じである．

3.6 演 算 子

条件判断や繰り返しでよく利用する比較演算子には，<，<=，>，>=，==，!= がある．算術演算子としては，四則演算用の+，-，*，/，ベキ乗の ^，アダマール積を求める # がある．四則演算用の4つの演算子は行列演算にもそのまま利用することができる．ただし，除算の場合は除数がスカラでなければならない．

4. プログラム例

RASS言語によるプログラムの例を2つ示す．

4.1 回帰診断統計量 DFBETAS

DFBETASはBelsley et al. (1980) によって提案された回帰診断のための統計量で，各観測

値が回帰係数にどれほどの影響を及ぼしているかを表すものである．

従属変数ベクトルを y，定数項を含めた独立変数行列を X とする．回帰モデルとして $y=X\beta+\varepsilon$ を仮定する．ここで β は回帰係数ベクトル，ε は誤差ベクトルである．

このとき，DFBETAS の値は

$$DFBETAS_{\alpha j}=(\hat{\beta}_j-\hat{\beta}_{j(\alpha)})/s_{(\alpha)}\sqrt{[(X'X)^{-1}]_{jj}} \tag{4.1}$$

で定義される．ただし，$\hat{\beta}_j$ は回帰係数の最小二乗推定値，$\hat{\beta}_{j(\alpha)}$, $s_{(\alpha)}$ は，α 番目の観測値を除いた時の回帰係数の最小二乗推定値および標準偏差である．

この計算を RASS 言語により記述したものが図 2 である．

RASS に組み込まれているクラスの中で DFBETAS が属するべきものは，明らかに，回帰診断のためのクラス lSDiag (least square diagnostics) である．このクラスの親クラスは lSRes (least square result) である．

計算結果を保持するために dfbetas というスロットを作成する．メソッド calDFBETAS は式 (4.1) をそのまま RASS 言語の構文規則に従って記述した手続きである．

```
lSDiag is_subclass_of lSRes : (
  slot(
      dfbetas type_ lmatrix slot_comment_ 'DFBETAS'
  ),
  method(
      calDFBETAS is_defined_by (
        require(
          exist_slots(indepVar, depVar, nObs)
        ),
        do(
          local_variable(beta, s, b, xxih, n, x type_ number),
          xxih := sqrt(diag(inv(tr(indepVar)*indepVar))),
          beta := inv(tr(indepVar)*indepVar)*tr(indepVar)*depVar,
          n := nObs,
          dfbetas := indepVar,
          while( n >= 1,
            x := nObs-n+1,
            b := inv(tr(indepVar(~x,*))*indepVar(~x,*))
                 *tr(indepVar(~x,*))*depVar(~x,*),
            s := (tr(depVar(~x,*)-indepVar(~x,*)*b)
                 *(depVar(~x,*)-indepVar(~x,*)*b)
                 /(nObs-indepVar))^(0.5),
            dfbetas(x,*) := tr((beta-b)/(xxih*s)),
            n := n-1
          ),
          sm(dfbetas,dsp)
        ),
        explanation('DFBETAS show how each observation',
                    'influences regression coefficients.')
      )
  )
).
```

図 2.　DFBETAS を求めるプログラム．

require( ) ブロックの exist_slots( ) は親クラスのスロットを参照するという宣言である.

do( ) ブロックの中では, while 文を利用して観測値の数だけ繰り返しを行い, その中で独立変数のラベル付き行列 indepVar と従属変数のラベル付き行列 depVar のそれぞれの一行を削除するために ~ を使用した. このプログラムでは計算量のことをまったく考慮していないが, この例のような場合, まずはわかりやすさのほうを効率より優先してもよいだろう.

計算が終った後で sm(dfbetas,dsp) により, その結果を表示する. メソッド dsp はすべてのオブジェクトに定義されていて, それぞれのオブジェクトにふさわしい形式で表示するためのものである.

4.2 基本統計量を保持するクラス

RASS 言語の自己記述性を調べるために, RASS に組み込まれている基本統計量のクラス bStat (basic statistics) の一部の機能を実現してみる (図3).

RASS では基本統計量はクラス bStat で計算されるが, それは回帰分析データをあらわすクラス regDat (regression data) の子クラスである. ここでも regDat と同じものを RASS 言語で実現したクラス myRegDat を親クラスとして子クラス myBStat を定義する.

myBStat では基本統計量の計算が行われるので, その結果を保持するためのスロットを定義する. そのスロットの値を計算するのは, このクラスのインスタンスが生成されるときである. RASS 言語ではインスタンスの生成は initialize というメソッドで行われる. このメソッドは myRegDat のメソッド, 例えば mkMyBStat (make myBStat) などから呼ばれる. do( ) ブロックでは, まず, 従属変数と定数項を除いた独立変数をまとめた行列 m を作り, メッセージ送信により行列に定義されているメソッドを利用して実際の計算を行い, その結果をスロットに格納する. explanation( ) ブロックは略されている.

メソッド cor は二つの変数名を与えてそれらの間の標本相関係数を返すものである. インスタンスが作成されているならば計算はすでに行われているので, 必要な値をとって来るだけのものである.

5. おわりに

知的データ解析システムを作成しようとするとき, 2通りの方法が考えられる. ひとつは既存のデータ解析システムを利用して作ることであり, もうひとつは最初から作ることである.

既存のデータ解析システムはほとんどが統計計算手法を中心として設計されている. 拡張用言語が組み込まれていることが多いとはいえ, 知的システムの作成に必要となるリスト処理などは扱いにくいのが普通である. したがってこれらを利用して作成された知的データ解析システムでは, 計算処理に対する拡張は容易であるが, 知識構造を自然に反映するような拡張用言語は作成しにくいだろう. ただし, データ解析システムに第一に要求されているのが, 計算の信頼性と多くの統計手法を利用できることであることを考えると, 長く使われてきた既存のデータ解析システムを利用することには大きな利点がある.

それに対して, すべてを最初から設計すれば, 知識構造を自然な制約として持つような拡張用言語を作成できると考えられる. しかし実際には, 知識処理と計算部分が複雑に関係するために, そのような拡張用言語を実現した知的システムはまだほとんどない. そこで本論文ではこのようなシステムのひとつである RASS に対して拡張用言語を作成してみた.

現在のところ, 新たに作成された知的データ解析システムは, RASS も含めて, ほとんどが実験的なものである. それはそのようなシステムでは既存のデータ解析システムに較べて利用

```
myBStat is_subclass_of myRegDat : (
  slot(
    mean        type_ lmatrix slot_comment_ '平均値',
    minimum     type_ lmatrix slot_comment_ '最小値',
    maximum     type_ lmatrix slot_comment_ '最大値',
    skewness    type_ lmatrix slot_comment_ '歪度',
    kurtosis    type_ lmatrix slot_comment_ '尖度',
    covariance  type_ lmatrix slot_comment_ '共分散',
    correlation type_ lmatrix slot_comment_ '相関係数'
  ),
  method(
    initialize is_defined_by (
      require(
        exist_slots(indepVar, depVar)
      ),
      do(
        local_variable(m type_ lmatrix),
        m := colBind(depVar, indepVar(*, ~1)),
        sm(m, mean, Mean),      mean := Mean,
        sm(m, min, Min),        minimum := Min,
        sm(m, max, Max),        maximum := Max,
        sm(m, skew, Skew),      skewness := Skew,
        sm(m, kur, Kur),        kurtosis := Kur,
        sm(m, cov, Cov),        covariance := Cov,
        sm(m, cor, Cor),        correlation := Cor
      )
    ),
    cor(Var1, Var2, Cor) is_defined_by (
      require(
        exist_slots(correlation)
      ),
      do(
        local_variable(c type_ lmatrix),
        c := correlation,
        sm(c, getValue, Var1, Var2, Cor)
      )
    ),
```

図 3. 基本統計量のクラスの再定義の一部.

できる統計手法が限られてしまうことが多いということも理由のひとつであろう. 拡張用言語はそれを改善するためにも重要なものであると考えられる.

謝　辞

山本由和氏 (徳島文理大学) には, RASS 言語を実装する際の Prolog プログラミングについて多くのコメントをいただいたことを感謝する. 本研究の一部は統計数理研究所共同研究(3-共研-7, 4-共研-2) として行われた.

参 考 文 献

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988). *The New S Language: A Programming Environment for Data Analysis and Graphics*, Wadsworth & Brooks/Cole, Pacific Grove, California. (『S言語—データ解析とグラフィックスのためのプログラミング環境』, I, II (渋谷政昭, 柴田里程 訳), 共立出版, 東京)

Belsley, D., Kuh, E. and Welsch, R. (1980). *Regression Diagnostics*, Wiley, New York.

Hand, D. J. (ed.) (1993). *Artificial Intelligence Frontiers in Statistics: AI and Statistics III*, Chapman & Hall, London.

Meyer, B. (1990). 『オブジェクト指向入門』(二木厚吉 監訳, 酒包 實, 酒包順子 訳), アスキー出版局, 東京.

中野純司, 山本由和, 岡田雅史 (1991). 知識ベース重回帰分析支援システム, 応用統計学, **20**, 11-23.

A Programming Language of the Regression Analysis Supporting System RASS

Ikunori Kobayashi

(Department of Mechanical-Electronic Engineering, Tokushima Bunri University)

Junji Nakano

(Department of Economics, Hitotsubashi University)

Not a few intelligent data analysis systems have been developed with the progress of "Artificial Intelligence" or "Expert System" techniques. They can keep and use statistical knowledges together with statistical calculation abilities to advise users for better statistical analyses.

In the process of real data analysis, we sometimes need data handling procedures which are not implemented in the data analysis system we use. Therefore, macro functions or programming languages for defining new procedures are inevitable for statistical software.

Proposed intelligent data analysis systems, however, rarely provide such extension languages. One of the reason is the difficulty of language design to keep the knowledge structure consistently. As statistical calculation abilities and statistical knowledges for using them are strongly connected, it is not easy to arrange newly defined statistical procedures properly in that structure.

In this paper, we consider the design of extension language for intelligent statistical systems and describe its implementation for an intelligent regression analysis supporting system named RASS.

Key words: Extension language, knowledge based system, object oriented programming, statistical software, regression analysis.