# Jasp 言語と XML 利用のための拡張

小 林 郁 典 † 中 野 純 司 ††
山 本 由 和 † 藤 原 丈 史 †††

Jasp 言語は，ネットワークに対応し，GUI 環境を持つ統計システム Jasp のためのプログラミング言語である．それは，データ解析に関する手続きを直感的に記述できるような簡単な文法を持ち，さらに，Jasp 自体を拡張することができるような記述能力を持つ．関数を基本構成要素とする Jasp 言語は，その関数を自然にオブジェクトへ拡張する枠組みを利用者に提供する．さらに，Java 言語のクラスオブジェクトを直接プログラムの中で扱うことができる．本稿では，Jasp 言語の特徴と，この拡張の一例として，XML 文書を Jasp 上で利用するための方法について述べる．

# The Jasp Language and its Extension to use XML Documents

Ikunori Kobayashi ,† Junji Nakano ,†† Yoshikazu Yamamoto †
and Takeshi Fujiwara †††

## 1. はじめに

近年のコンピュータ技術の発展は，データ解析に新しい可能性を提供した．例えば，ハードウェアの発達によって一度に処理できるデータ量が増し，より複雑な解析ができるようになった．また，ネットワークの発達，とりわけインターネットの爆発的な普及により，必要なデータを手軽に検索することもできるようになった．このようなコンピュータを取り巻く環境の変化の中で，統計システム（あるいは統計パッケージ）もこれに応じた変化が求められるようになっている．

コンピュータの黎明期より，データ解析のためのソフトウェアは開発され続けている．そして，その時代のコンピュータの能力に比例して，それらのソフトウェアも発展してきた．比較的新しいものとして，例えば，S[2] や LISP-STAT[7]，XploRe[3] が挙げられる．これらも今なお拡張は進められているが，一方で，OMEGA プロジェクト [6] のような最新のコンピュータ技術に基づいた新しい統計システムの開発も進められている．我々も，近年のネットワークを中心としたコンピュータ技術の発展をひとつのターニングポイントとして捉え，従来の統計システムにはない操作性と機能を持ったネットワーク対応型の統計システム Jasp を開発している．

統計システム，あるいは統計パッケージと言われるデータ解析のためのアプリケーションソフトウェアは，データ解析の作業をいかに直感的かつ正確に行えるのか，さらに複雑なこと（様々なこと）ができるのかという視点で評価されがちである．前者については，システムにおけるグラフィカルユーザインターフェースの設計とシステムが持つ潜在的な機能に強く関係する．後者については，システムが利用者に対して提供するプログラミング機能の記述能力に強く関係する．

統計システムにとってプログラミング環境は非常に重要である．一般に統計システムの利用者のシステムに対する要求は多様であり，システムの開発者がどのように多くの機能をシステムに追加しようと，それですべての利用者を満足させることは難しい．従って，利用者が統計システムに対して実行させたい命令を正しく表現することができるプログラミング環境とその専用言語が要求される．また，そのプログラミング言語は，統計システムが備えている機能を十分に活用・制御できる記述能力を備えていなければならない．

近年のインターネットの普及によるコンピュータのグローバル化により，統計システムに対する新しい可能性と要望が誕生した．例えば，インターネット上で

† 徳島文理大学工学部
Faculty of Engineering, Tokushima Bunri University
†† 統計数理研究所
Institute of Statistical Mathematics
††† 総合研究大学院大学
Graduate University of Advanced Studies

公開されている多量なデータをそのまま利用したり，インターネット上で公開されている統計解析上有益なソフトウェアをシステムに組み込んで利用してみたいということが挙げられる．また，解析の分散処理とギガのオーダに向上したハードウェアの高性能化により統計システムが解析できるデータ量も利用者が把握できないほどまでに増加した．データを記述するフォーマットも XML など様々なものが考案されている．このような状況を踏まえれば，今後さらにデータ解析を取り巻く環境は大きく変わっていくことが予想される．

現在，普及している統計システムや統計パッケージもこれらの影響を強く受け，利用者にそのメリットが活かされるように拡張や変更が行われている．しかしながら，ほとんどの統計システムでは，この作業がそのシステムの開発者に委ねられていて，一般の利用者がシステムを拡張することは容易ではない．それらに備えられているプログラミング環境を用いても，多くのシステムでは，実装することが不可能であることが多い．この理由として，多くの統計システムが利用者レベルでの拡張性を考慮していないことと，さらに，拡張性よりも正確な計算結果を算出することに重点を置いてきたからであろう．正確な統計結果を算出することは，今なお重要な要求事項であることに変わりはないが，統計システムを取り巻く状況が多様化していく中で，統計システムの利用者レベルでシステムを拡張できる枠組みが要求される．

## 2. 基本設計方針

我々が Jasp 及び Jasp 言語を設計するときに，特に気を付けた点は，次の通りである．

(1) これまでの統計解析用のソフトウェアや統計システムの共通点や，それらに関する広く評価されている特徴をできるだけ取り入れる
(2) 有益なプログラム資産を活用する
(3) システムとして拡張性を高める
(4) 直感的に操作できるインターフェースを備える

このような条件をできるだけ達成させるために，我々はシステム開発に Java 言語を採用した．

Java 言語は，汎用型のオブジェクト指向プログラミング言語である．これは，1991 年にサン・マイクロシステムズによって発表された．Java のプログラムは，バイトコードと呼ばれる形態にコンパイルされ，Java バーチャル・マシン (Java VM) 上で稼動する．従って，Java VM が稼動するコンピュータ上であれば，OS に依存することなく Java のプログラムを実行させることができる．Java 言語は，ネットワーク機能やそれに付随するセキュリティ機能，グラフィックス，リモート操作，データベースアクセスのための豊富なライブラリ群を標準で持っている．近年，Java で開発されたプログラムがインターネット上で数多く公開されるようになってきた．

このような背景に基づいて，我々は，Jasp を Java 言語を利用して開発することにした．さらに，Jasp の言語機能としては，Java 言語のスクリプト言語である Pnuts[8] を利用することにした．ただし，Pnuts は汎用の言語なので，特に統計計算向きに設計されているわけではない．従って，我々は，Jasp と Pnuts の間にプリプロセッサを置き，Pnuts の長所を活かしながら Jasp 言語を実装している．

本章では，Jasp 言語の基本的な要素である関数（Jasp 関数）とデータ，Java クラスの取扱いについて述べる．

### 2.1 Jasp 関数

Jasp 言語は，これまで多くのデータアナリストが利用してきた統計システムの言語（例えば S 言語や XploRe 言語）と同じように手続き型の言語とした．手続き型言語は，最も利用されているコンピュータ言語の形態であり，目的の処理の流れに基づいて，そのままプログラムを記述できる特徴を持つ．また，Jasp 言語には，いくつかの手続きをまとめる枠組みとして Jasp 関数と呼ばれる関数を利用することができる．Jasp 関数は，入力としていくつかの引数を与え，出力としてひとつの計算結果を戻す．ただし，プログラムの流れを理解しやすくするために，関数の内部で外部のデータ（変数）の値を変更することができない．

Jasp 関数のフォーマットは次の通りである．

```
function NAME ( ARGUMENTS ) BODY
```

ここで，`NAME` は関数の名前，`ARGUMENTS` は引数のリスト，`BODY` は関数の本体である．引数のリストは空でもよいし，”,”で区切られたいくつかの引数でもよい．もし同名の関数が存在しても引数の数でそれらは区別される．関数の本体は，いくつかの文 (statement) で構成される．もし複数の文が必要である場合には，それらを”{”と”}”で囲む．また，その際，各文は”;”か改行で区切られる．他の手続き型言語のように，Jasp 言語は”`if`”文や”`for`” 文などを利用することができる．関数の呼出し元への値の返却には，”`return`”文が使われる．

Jasp 関数の内部で宣言された変数は一時的なものである．それらの有効範囲は，その関数の内部に限定され，他の関数によって変更されるようなことはない．呼出しに利用された引数の値が Jasp 関数内で変更さ

れても，その変化は呼出し側に影響することもない．

### 2.2 データの操作

Jasp 言語は，データの型指定を省略することができる言語として設計されている．変数は，はじめに設定される値の型によってそのデータ型が決定される．

Jasp 言語で扱えるデータ型は，次の 3 種類に大別される．

(1) 基本データ型
(2) Jasp クラス
(3) Java クラス

基本データ型は，一般的なプログラミング言語と同じように，整数，実数，文字列などを表すものである．Jasp クラスは，Jasp 言語専用のクラス定義機構を利用して宣言されたクラスのことである（3章参照）．Java クラスは，Java 言語を用いて作成されたクラスのことである（2.3節参照）．利用者は，これらのデータ型を同等に扱うことができる．

データ解析では，行列演算が多用される．従って，Jasp 言語では行列を表現するためのデータ型 (JaspMatrix) が用意されている．JaspMatrix は，正確には Java クラスであるが，利用者が扱いやすいように特殊な表記法が用意されている．例えば，次の式は，`mat` と名付けられた 2 行 3 列の行列を生成する．

```
mat = JaspMatrix(2,3)
```

要素を指定するには `[ ]` 演算子を利用して，例えば，

```
mat[1,2]
```

のように記述する．この他にも，第 1 行の要素を削除し，第 1 列から第 3 列を選択するためには，

```
mat[!1,1:3]
```

と入力する．さらに，JaspMatrix は，行と列のラベルを持つことができ，それらを使用して次のように要素の指定ができる．

```
mat["no2", "x3"]
```

行列の演算子には，次のようなものがある．

```
x + y        ; 要素同士の和
x - y        ; 要素同士の差
x * y        ; 積
y = x.inv    ; x の逆行列
y = x.trans  ; x の転置行列
y = x.det    ; x の行列式
```

ここで，利用例として，Durbin-Watson 統計量 [9] を求める Jasp 関数を示す．この統計量は次の式で計算される．

$$D = \sum_{i=2}^{n} (e_i - e_{i-1})^2 / \sum_{i=1}^{n} e_i^2$$

```
-----------------------------------------
function Durbin_Watson( res ){
  dw = 0.0
  for(i=2; i<=res.nr; i++){
    dw = dw + pow(res[i,1]-res[i-1,1],2.0)
  }
  dw = dw / (res.trans * res)
  return dw
}
-----------------------------------------
```

**図 1** Jasp 言語による Durbin-Watson 統計量

ここで，$e_i$ は $i$ 番目の残差である．これを Jasp 関数で記述したプログラムが図 1である．図 1の `res` は残差を表す JaspMatrix で，`res.nr` はその行数を表す．

### 2.3 Java クラス

Jasp 言語の基本である Pnuts は，Java のクラスを変数として直接利用することができる．この機能を活かして，Jasp 言語でも Java のクラスを変数として利用することができる．Java 言語で記述されるクラスは，高機能な能力を提供することができるので，Jasp 上で Java クラスを活用することができれば，必然的に Jasp の能力や拡張性を高めることになる．

Java クラスを Jasp 言語で利用するためには，識別子 `class` か関数 `import` を使用する．例えば，次は識別子 `class` を利用して Java のクラス `BigDecimal` を `bd` という変数に代入させている．

```
bd = class java.math.BigDecimal
```

これ以降，`bd` は Java のクラス `BigDecimal` を表し，そのインスタンスを生成するには，例えば，

```
bd1 = bd(1.1e32)
```

と入力する．ここで，`bd1` はクラス `BigDecimal` のインスタンスオブジェクトとなる．同じ作業を関数 `import` を利用した場合には，次のように記述できる．

```
import("java.math.BigDecimal")
bd1 = BigDecimal(1.1e32)
```

関数 `import` では，複数のクラスを指定するために，次のようにワイルドカードを使うことができる．

```
import("cern.jet.random.*")
```

Jasp プログラムでは，インスタンス・フィールドとインスタンス・メソッドのアクセスのために，ドット演算子が使われる．また，クラス・フィールドとクラス・メソッドのアクセスのために，ダブルコロン演算子が使われる．

図 2に，Java クラスを導入した Jasp プログラムの例（一部省略）を紹介する．このプログラムは，最尤

```
-------------------------------------------
function mle(func, data, para){
  size = para.length
    .....
  opti = Opti_F9::define(model, size)
                // set a target model
  opti.initialize(para)
                // set initial values
  ans = opti.optimize()
  return ans
}
-------------------------------------------
```

**図 2** 最尤推定値を求める Jasp 関数

```
-------------------------------------------
function nor(x, p){// normal distribution
  pi = 3.141593
  exp(-1*(x-p[0])*(x-p[0])
     /(2*p[1]*p[1]))/(sqrt(2*pi)*p[1])
}
       // make random data for simulation
data = normal(50)
       // initial values of parameters
para = [0.1, 1.1]
ans = mle(nor, data, para)
-------------------------------------------
```

**図 3** 関数 mle の使用例

推定値を計算する関数である．非線形最小二乗アルゴリズムの部分（クラス Opti_F9）は，インターネット上で（無料で）公開されているライブラリ [12] を利用している．図中の model は，func，data，para によって計算された尤度関数を表している．size はパラメータ数であり，para はパラメータの初期値である．

次に，この関数 mle を利用したプログラム例を紹介する．ここでは，乱数を 50 個生成し，それを使って正規分布のパラメータの最尤推定値を求めるためのプログラムである．図 3に示す．このプログラム中における p[0] と p[1] は，それぞれ正規分布の平均値と標準偏差を表す．

## 3. 統計システムにおけるオブジェクト指向プログラミング

オブジェクト指向プログラミングは，互いに関係するデータ（属性）と手続き（メソッド）をまとめたオブジェクトによりシステムを構築することである．これを用いるメリットは，従来の手続き主体のプログラムよりもオブジェクトを用いた方が我々の思考形態に近い形でシステムを捉えやすいこと，継承機構により拡張性が高いこと，などが挙げられる．逆に，

(1) オブジェクトを作成したり利用したりするには，オブジェクトの枠組みを熟知しておかなければならない
(2) 常に対象となるオブジェクトと送りつけるメッセージの両方を指定しなければならない
(3) オブジェクトの階層が大きくなりすぎると，目的のオブジェクトを見つけにくくなる

という運用上の問題点が指摘されている．

オブジェクト指向プログラミングを可能にした統計システムとしては LISP-STAT が有名である．これは，リスト処理を得意とする純粋な関数型言語である LISP 言語を開発プラットフォームとして採用している．LISP 言語は，対話型操作が可能なインタプリタであり，標準ではオブジェクト指向プログラミングの枠組みは提供していないが，その言語仕様の柔軟性から比較的容易にオブジェクト指向プログラミングを実現できる．実際，CLOS[4] のようなシステムが発表されている．LISP-STAT の基本要素は関数である．データ解析の多くは実装されている関数を利用することで解析を進め，必要に応じてオブジェクトを利用するというハイブリッド型を採用している．すべてをオブジェクトにすると，利用者はどのオブジェクトの何と言うメソッドを使えばよいのかをいつも気にしなくてはならなくなるので，システムの初心者には不適であると言われている．従って，LISP-STAT のようなハイブリッド型は，汎用統計システムの設計方針としては適切であろう．

統計システムのプログラミング環境でオブジェクトを取り扱うことができれば，各種統計手法の枠組みに応じて解析データからオブジェクトを用意し，それにメッセージを送る方法で解析を進めることができる．従って，利用者は，手続き主体で解析を進めるのではなく，データを主体とした概念的な操作で解析を進められる．また，近年の統計システムは，GUI が標準である．そのウィンドウを多用する GUI の構築には，オブジェクト指向プログラミングが効果的なのは明らかである．我々の Jasp プログラミング環境は，このようなハイブリッド型でオブジェクト指向プログラミングを利用者に対して提供することができるように設計された．

汎用型の統計システムでは，あらかじめ組み込まれている手法は多くなる．システムの開発者，あるいは

システムを拡張しようとする利用者は，それらを使いやすいようにうまくシステム上で分類し，実装することが要求される．例えば，関数のような手法を基準としたケースでは，それらの名前が衝突しないように名前付け規則などに従って開発し，そして，そのリストをウィンドウなどを利用して利用者に見せる方法が一般的な方法であろう．ある程度の数であれば，このような問題は起こりにくいが，利用者が覚えきれないくらいの関数がシステム上に存在するようになれば，このような問題が深刻になってくる．これに対して，オブジェクトでは，概念的に分類して手続きやデータを管理させることができる．

Jasp クラスの定義のためのフォーマットは次の通りである．

```
jaspclass NAME(SUPER) BODY
```

ここで，NAME はクラスの名前，SUPER は親クラスの名前である．親クラスの指定が不要な場合には (SUPER) の部分を省くことができる．BODY はコンストラクタ，メソッド，プライベート関数から構成される．プライベート関数は Jasp 関数のことであり，メソッドはプライベート関数と外部とのインタフェースとして利用される手続きである．Jasp クラスの中では，内部データ（属性）の型宣言をする必要はない．また，Jasp 関数をまったく修正することなく Jasp クラスの中にプライベート関数として取り込むことができるという特徴を持つ．

図 4に Jasp クラスの簡単な宣言例を示す．図中のクラスは線形回帰を表すものである．このクラスは，2 つのメソッドと 1 つのプライベート関数から構成される．beta と estimated は計算結果を保持するための内部データである．メソッドの中の this. で始まる変数は，属性として扱われ，this. で始まる手続きの呼出しは，同じクラス内のメソッドが呼ばれる．this. が付かない手続き呼出しは，そのクラス内のプライベート関数が呼び出される．

このクラスから実体であるインスタンスを作成するには，次のコマンドを利用する．

```
reg1 = LinearRegression("datafile.dat")
```

そして，そのインスタンスの内部データを参照するには，

```
reg1.estimated
```

と入力する．この場合，従属変数の予測値が返される．

## 4. プログラミング環境

### 4.1 プログラミングのためのウィンドウ

Jasp を起動すると 2 つのウィンドウが現れる．それぞれ CUI (Character User Interface) ウィンドウと GUI (Graphical User Interface) ウィンドウと呼ばれている．CUI ウィンドウは，プログラミングのためのウィンドウである．Jasp のプログラミング環境とは直接的にこのウィンドウのことを指す．GUI ウィンドウは，結果の表や図を出力したり，グラフィカルな操作などのためのアイコンが表示される．

```
-----------------------------------------
jaspclass LinearRegression {
  method LinearRegression( file ) {
    data = read(file)
    y = data[,1]
    t = Vector(data.nr,1.0)
    x = c_bind(t,data[,!1],"RIGHT")
    this.LinearRegression(y,x)
  }
  method LinearRegression(y, x) {
    this.beta = ols(y, x)
    this.estimated = x * beta
  }
  function ols(y, x) {
    coeff = (x.trans*x).inv*x.trans*y
    return coeff
  }
}
-----------------------------------------
```

**図 4** Jasp クラスの宣言例

CUI ウィンドウにはエディタエリアと入出力エリアの 2 つの領域がある．エディタエリアでは他のエディタを別に起動することなくプログラムの入力を行うことができる．ここでは，利用者がプログラムの一部をマウスで選択して実行に移すこともできる．この機能は，プログラムの開発時のデバッグなどで有効である．入出力エリアは，1 命令ずつの実行を与え，それに応じて Jasp からの結果が表示されるところである．ここでの Jasp からの結果はテキストベースの部分だけに限定される．

Jasp では，GUI ウィンドウ上での操作は入出力エリアで与える 1 つの命令文と 1 対 1 に対応するように実現している．従って，GUI ウィンドウでの操作はすべて入出力エリアに解析ログとして記録される．CUI ウィンドウ上のプルダウンメニューの中に解析ログをファイルに書き出す命令があるので，Jasp 上での解析手順の再現を可能にする．

```
-------------------------------------------
/**
 * @summary "estimate coefficients"
 * @param JaspMatrix y "dependent variable"
 * @param JaspMatrix x "independent variables"
 * @return "coefficients"
 */
function estimate_coefficients(y, x){
  beta = (x.trans * x).inv * x.trans * y
  return beta
}
-------------------------------------------
```

**図 5** GUI のためのコメント

### 4.2 GUI のためのコメント

Jasp 言語では，変数や引数のデータ型を指定する必要はない．これは，利用者，特にシステムの初心者が，プログラムを簡単に作成できるようにするためである．多くの統計システム言語でこのような仕組みは見受けられる．しかしながら，データ型を宣言することのメリットも存在する．例えば，すでに存在するプログラムの内容についてよく知らない利用者にとってプログラムの構造を理解しやすくしたり，そのプログラムを GUI 上で利用する際に，そのプログラムの呼出しに関する事前確認を行わせることが可能になる．また，データ型をたくさん定義できるシステムにおいては，何らかの方法で変数や引数のデータ型を利用者がわかるようにすることは，そのプログラムをより読みやすくする．そこで我々は，関数の引数のデータ型を定義することができるようにコメントを利用することにした．

コメント部の中で，`@summary` や`@param`，`@return` で始まる文があれば，それは GUI にとって特別な意味を持つ．例えば，図 5において，"`/**`" と"`*/`" で囲まれた箇所が Java 言語と同等にコメントとして扱われるが，先ほどの 3 つの識別子を含む行はその情報が GUI のために利用される．`@summary` で始まる文は，コメントの下で定義している関数の説明をするところである．`@return` で始まる文は，その関数の戻り値の説明をするところで，さらに，`@param` で始まる文は，引数の説明をするところである．`@param` 行は次のフォーマットに従う．

```
@param TYPE NAME "STRINGS"
```

ここで，`TYPE` は，`NAME` と名付けられた引数のデータ型を意味している．`STRINGS` は，説明である．

これらの行は，Jasp サーバ上では，一般的なコメントとして無視されるが，Jasp クライアント上の言語処理系では，この情報が活用される．利用者が Jasp を使用するとき，Jasp クライアントは 2 つのウィンドウ（Jasp の CUI ウィンドウと GUI ウィンドウ）を表示する．さらに，GUI ウィンドウは 3 つのサブウィンドウを持つ．その中の左下のウィンドウが，利用可能なコンストラクタと関数を表示するが，この時，コメント情報を記述されていないものは表示されないようにした．コメント情報のないコンストラクタや関数は，利用者の個人的な使用目的のために定義されたものであると見なし，利用者が安易に実行に移せないことにした．GUI ウィンドウの左上のサブウィンドウでは，アイコンとそれらの木構造表現により解析の履歴が分かるようにしている．アイコンは，データや適用したモデル，解析結果を表している．利用者がアイコンをマウスで指定すると，利用可能な手続きがポップアップメニューとして表示される．このメニューの中で，コメント情報（その手続きの説明）を見ることができる．さらに，その一覧から目的の手続きを選択すると，ポップアップウィンドウが開き，その中で引数の情報が表示される．この仕組みは，利用者の間違った操作を未然に防ぐ役割を担う．

## 5. XML 文書に関する Jasp 上での操作方法

XML (eXtensible Markup Language) は，W3C (World Wide Web Consortium) が標準化を行っている次世代の構造化文書規約である．Web site 作成のための機能も兼ね備えているので，HTML (Hyper Text Markup Language) の次の記述言語と位置づけられている．マークアップ言語は，タグと呼ばれる構造により値に属性を付加する形でデータを記述する．マークアップ言語はテキストなので，利用者（人間）とコンピュータの両方にとって読みやすく，分かりやすい．

XML は今後ますますデータ記述フォーマットとして活用されていくと予想される．特に，XML を取り扱えるアプリケーションソフトがいくつか発表されてきているので，XML がデータ解析システムの中間フォーマットとして活用されていくことが明らかになってきている．Jasp で XML 文書を扱うことができれば，解析データや解析結果の共有を容易にすることができ，システムとしての汎用性を高めることになる．

XML でどのようなタグが使えるのか，あるいは，どのような属性を与えられるかは DTD (Document Type Definition) と呼ばれる定義ファイルで規定される．ただし，XML 文書には必ず DTD が必要ということではなく，構造のチェックの必要がなければ DTD

を省略することができる．DTD の無い XML 文書を well-formed XML 文書と呼ぶ．

ただし，XML の自由度の高さとデータの種類の多さから，データ記述の構造を規定することが難しく，DTD に関してさまざまな提案や勧告が行われている．例えば，解析データの記述に関与しているものでは，DDI[11] や DandD[10] などが挙げられる．Jasp では，XML に関する Jasp 言語の API を利用することでできるので，比較的容易に各種の DTD に従った XML 文書を読み込ませることを可能にする．

具体的には，Jasp 上で XML 文書を扱うために，DOM (Document Object Model) と呼ばれるプログラミングインタフェースを使用する．特に，DOM 対応の Java 言語用 API (Application Programming Interface) が Web 上で無料で公開されている [1] ので，Jasp ではこれを利用する．

Jasp 言語は，Java クラスを変数としてそのまま扱うことができるので，この API を使用することで XML 文書を表す XML インスタンスを Jasp プログラミング環境で作成することができる．ただし，API が提供する機能は多種多様であり，そのままの形で Jasp の利用者に提供しても使いこなすことが困難である．そこで，Java 言語のレベルでこれらの API のデータ解析上必要な機能だけを抽出し，使いやすくするための Wrapper クラスを用意した．これを Jasp プログラミング上で利用することで，利用者が XML を容易に扱うことを可能とする．

### 5.1 XML 文書の読込み

ここでは，Jasp 上で XML 文書を取り込む例を紹介する．DTD として DandD を採用している．例えば，DTD に従って記述されたデータ `acidj.xml` を Jasp 上に読み込ませるためには，次のコマンドを入力する．

```
-------------------------------------------
  acid = JaspSimpleXML("acidj.xml")
  acid_tree = acid.printTree()
  acid_tree
  acid_matrix = importDataVector(acid, 3)
  acid_matrix
-------------------------------------------
```

1 行目では，XML 文書から Jasp 言語の変数を作成している．Jasp 関数 `JaspSimpleXML()` の第 1 引数には，XML 文書ファイル名を渡す．`acid` は DOM オブジェクトを表す変数である．2 行目は，XML 文書の構造を文字列で返すメソッド `printTree()` を呼び出している．`acid_tree` は String クラスのインスタンスである．3 行目は，その文字列を GUI ウィンドウ上に表示させるためのコマンドである．Jasp は，変数だけの入力で，その内容を GUI ウィンドウに表示する．4 行目の `importDataVector()` は Jasp 関数である．この関数の内容は後で説明する．第 1 引数には XML インスタンス名を，第 2 引数には，インスタンス内にあるタグ `DataVector` の内容を何個取ってくるのかを指示している．この戻り値は，JaspMatrix のインスタンスとなる．5 行目は，4 行目で求めた行列の内容を GUI 上に表示させている．

```
-------------------------------------------
function importDataVector(dom, colNum){
  dvec = dom.getDataVector(1)
  len = dvec.length
  ans = JaspMatrix(len, colNum)
  for(i=1; i<=len; i++){
      ans[i,1]=dvec[i-1] }
  label = dom.getDataVectorAttr(1, "Id")
  ans.setColLabel(label, 1)
  for(j=2; j<=colNum; j++){
      dvec = dom.getDataVector(j)
    for(i=1; i<=len; i++){
      ans[i,j]=dvec[i-1] }
    label = dom.getDataVectorAttr(j-1,"Id")
    ans.setColLabel(label, j)
  }
  return ans
}
-------------------------------------------
```

**図 6** 関数 importDataVector

プログラム中の関数 `importDataVector()` は，Jasp 言語利用者が記述できるレベルのものである．2 行目と 9 行目の `getDataVector()` は Java レベルで実装したものである．具体的には，`JaspDandD` クラスのメソッドである．6 行目と 11 行目の `getDataVectorAttr()` は，`DataVector` タグの属性を参照するためのメソッドである．この場合，属性 ID の内容を行列の列ラベルに設定している．

### 5.2 JaspMatrix を XML 文書へ

JaspMatrix のインスタンスにメッセージ `writeXML()` を送れば well-formed な XML 文書を作成できる．`writeXML()` には引数として書き出すファイル名を指定する．例えば，図 7の要素を持つ JaspMatrix があるとする．インスタンス名が test であれば，

```
test.writeXML("exMatrix.xml")
```

と入力すれば XML 文書”`exMatrix.xml`”が作成され

```
-----------------------------------------
        x  y  z
    No1 1  2  3
    No2 4  5  6
-----------------------------------------
```

**図 7** JaspMatrix の例

```
-----------------------------------------
    <JaspMatrix>
        <observation>
                <x>1</x>
                <y>2</y>
                <z>3</z>
        </observation>
        <observation>
                <x>4</x>
                <y>5</y>
                <z>6</z>
        </observation>
    </JaspMatrix>
-----------------------------------------
```

**図 8** exMatrix.xml の内容

る．図 8に `exMatrix.xml` の中身を記す．

## 6. おわりに

本論文では，汎用統計システム Jasp の専用言語の設計と特徴について述べた．さらに，XML 文書を Jasp 上で操作できるようにするための方法を簡単に紹介した．情報技術の躍進と伴に，システムとして要求される能力は大きくなっている．統計システムについても例外ではなく，より効果的に統計解析を支援する役割を担い続け，今後も発展していくであろう．このためには，これまでに蓄積されてきた既存の有益なプログラム資産を活かしながら統計システムをより良く拡張していていくことが求められる．従って，既存の技術と新しい技術をひとつのシステム上で整合性や一貫性を保ちながらうまくインテグレートすることが必要になる．Jasp 言語は，その枠組みを具体的に提供するひとつの実例である．

## 参考文献


1) Apache XML Project (2000): `http://xml.apache.org/`
2) Chambers, J. M., Hastie, T. J. (1992): *Statistical Models in S*, Eadsworth.
3) Härdle W., Klinke S., Müller M. (2000): *XploRe Learning Guide*, Springer.
4) Keene E. S. (1989): *Object-Oriented Programming in COMMON LISP*, Symbolics.
5) Nakano J., Fujiwara T., Yamamoto Y., Kobayashi I. (2000): A statistical package based on Pnuts, COMPSTAT2000.
6) Omega Project (2001): `http://www.omegahat.org`
7) Tierney L. (1990): *LISP-STAT*, John Wiley & Sons.
8) 戸松豊和 (2001): Pnuts, `http://javacenter.sun.co.jp/pnuts/`
9) Ryan T. P. (1997): *Modern Regression Methods*, John Wiley & Sons.
10) 柴田里程 (2001): データリテラシー, 共立出版.
11) Society Science Community (2001): Data Documentation Initiative, `http://www.icpsr.umich.edu/DDI/`
12) Verrill, S. (1998): *Package-optimization*, Forest Products Laboratory, `http://www1.fpl.fs.fed.us/`