Rのプログラミング言語としての側面について

岩　橋　俊　哉

The statistical language "R" as a programming tool.

Toshiya Iwahashi

序

前回の研究ノート（岩橋，2007）では、統計言語Rの概要と電卓的なデータ処理方法を示したが、今回はその続きとして、Rのプログラミング言語としての側面に焦点をあて、その特徴とプログラミング方法について整理してみたい。

R（Sも同様）には、電卓的に使って、統計処理が容易にできるという特長がある。さらに統計処理専用の関数も数多く用意されており、統計処理用のツールとして魅力的な処理環境を提供している。しかし、Rにはプログラム言語としての側面もある。ライブラリーなど拡張性も確保されており、プログラミング言語としても十分な機能を備えている。基本コンセプトの多くは、本家であるSの特徴でもあるが、ここではRを利用するという前提で、特に区別せずに特徴を整理してみたいと思う。また、ここでも前回と同様に、心理学分野でのデータ処理に利用することを前提としている。

Rのコンセプト

Rは、Sと同様にインタプリターであることから、プログラムを書いた後、すぐに実行して処理するというような環境で、短いプログラムを実行する場合に威力を発揮する。この点は最近のスクリプト言語に通じるものがある。インタプリターには、処理速度が遅いという欠点もあるが、最近のコンピュータを用いて、心理学領域のデータを処理する程度であれば、この欠点は、ほとんど問題にならないであろう。

プログラムの記述時に省略できる要素

汎用言語の場合には、多くの場合、あらかじめ変数の型などを定義してから設計する必要があるが、Rでは、細かく変数（Rではオブジェクト）の型を定義したり、あらかじめメモリーの領域を確保するなどといった作業をする必要はない。これらは最近の汎用のスクリプト言語にも共通する特徴である。

変数の最小単位もスカラーではなく、ベクトルであるという特徴を持ち、APL（プログラム言語）のように行列演算が簡単にできる。短いプログラムを書いて処理したい場合にはこれは大きな長所である。

また、変数名だけ入力することで、その内容をコンソールで容易に確認できるようになっている。これはインタプリターの長所を利用したもので、ここでは関数 print（ ）が自動的に補われている。さらに標準の統計処理関数には、出力のフォーマットも設定されているので、出力の書式に注意を払う必要はない。ただし、その出力をなんらかの書式で処理しようとする場合には、少し手間がかかることになる。これは、SAS や SPSS の場合と同様である。

グラフィックス機能の中心はグラフ描画であり、グラフに図形やテキストを配置する（いわゆる低水準）関数の場合にも、点や線などをグラフ座標上にプロットするという描画モデルであり、汎用言語のような柔軟性はない。

他の長所と短所も補足的に挙げておくと、その他の特徴として、（S は異なるが）R の場合、GPL に従って配付されており、ソースコードも公開されている。これにより R 内部のアルゴリズムなどが確認でき、計算精度などデータ処理の信頼性も確認できる。万が一バグなどあれば、R のソースコードを変更することも可能である。

R を使う上で一番の障害は、コマンド入力方式が基本であることだろう。コマンド入力方式では、命令をあらかじめ覚えておかないと、操作することがまったくできないという問題がある。統計処理のツールとして使うだけという場合に、この方式は（グラフィックユーザーインターフェイスが一般的な今日では）、かなり敷居が高いといわざるを得ない。ただし、この問題に対して、最近になって R commander（舟尾；2007を参照）というアプリケーションが開発され、メニューで選択する手段も提供されるようになってきている。R commander は、まだ完璧ではないようであるが、これにより使い勝手も SPSS や S-Plus とほとんど遜色もなくなり、統計処理ツールとしての魅力はさらに増している。

R 言語を理解する上で、（S や）R に特有の（プログラミング）用語に注意が必要である。特徴的な点を以下に挙げておく。

- ファクター（=factor カテゴリ変数のこと。数字ではあるが、数値ではないデータ。心理学では評価尺度など）を指定する場合に用いる。関数 factor で変換する必要がある。
- ワークスペース（workspace）オブジェクト情報の設定とコマンド履歴の記録。保存したり、必要に応じて呼び出して設定することができる。作業ディレクトリーとは異なることに注意。
- 代入演算子では、<- を持ちいる。等号 = あるいは、逆方向の -> も違反ではないが、プログラムの可読性が低下するという理由から推奨されていない。
- グラフィックスの関数は、低水準描画関数と高水準描画関数の大きく 2 種類に分けられる。高水準描画関数でグラフそのものを作成し、低水準描画関数でタイトルのテキスト、

凡例などを追加して表示する。

・使われている括弧の種類は、C 言語などとほぼ同様であり、

（と）は算術式の優先順位の設定、または if 構文の中での条件式の記述,

｛と｝は制御構造の構文の中での複数の式をまとめる役割,

［と］は行列変数で配列を示す。

プログラムの書き方について，基本書式を以下にまとめてみた。他の汎用言語との共通点も多い。

・以下の語は代表的な予約語であり、オブジェクト名としてユーザーが新規に定義することはできない。

break, else, for, function, if, in, next, repeat, return, while, TRUE, FALSE, NA, NULL、pi, Inf, -Inf, NaN, NA, NULL 他

・自由書式を採用しており、空白は原則として無視されるので、字下げなどを入れて読みやすいように記述することができる。

・複数の命令を 1 行に書く場合には、途中の命令の最後にセミコロンを入れる。

例、a <- 10 ; b <- "AA"

・コメントの記述方法　# の後に記述した文字はコメントとして解釈され、実行されない。ただし、複数行にまたがるコメントは通常記述できない。なお if (FALSE) {コメント} を利用する方法もあり、Ligges（2006）で紹介されている。

・日本語の文字は変数や関数などのオブジェクト名として、原則用いることはできない。

R のデータ・変数

R（S も同様に）では、変数と関数を総称してオブジェクトと呼んでいる。その命名のルールは、他の言語とほぼ共通であり標準的な書式が採用されている。

・使える文字は、英数と .（ピリオド）である。

・大文字と小文字は区別される。

・最初の文字は、英字でなくてはならない。途中に数字が入ることは差し支えない。

・変数名の長さは256文字までに制限されている。

・（上記の）予約語は使えない。

変数の型（データ型）について

・変数にデータを代入するとその変数のデータ型が決定される。ただ以下のファクター（=factor）型のみ明示的に変換する必要がある。

・ファクターは、カテゴリー変数のこと。数字で表記されるが、数値ではない。心理尺度な

どを処理する場合に用いる。

・ベクトルは、１次元の配列変数で、変数の基本型である。

・リストは、数字、文字など異なる型のデータが混在する配列である。

また、SPSS, SASなどほとんどの統計処理アプリケーションで用いられているデータ書式と同じものにデータ・フレームがある。二次元に配列された表のデータであるが、行頭にラベルをつけてデータ処理を行うことができる。Rで作成する場合には、ベクトル変数を合成することで可能であるが、多くの場合、Excelなど表計算アプリケーションでデータをあらかじめ作成してから読みこむことになろう。ちなみに、統計計算に必須の行列計算の関数もRには用意されている。

アルゴリズムの記述

演算子の概要

・代入演算子は、<- として記述される。

・基本的な算術演算子

べき乗：^or**, 乗算：*, 除算：/, 加算：+, 減算：-, 剰余：％％　など

・基本的な論理演算子。括弧もふくめた演算子の優先順位は、他の一般的な言語とほぼ同じと考えて良い。

等しい：==, 等しくない：!=, 大きい：>, 以上：>=, 小さい：<, 以下：<=, 否定：!　など

条件分岐構文として、if文, ifelse文, switch文が用意されている。

・ifの書式は、if（条件）{式} else {式} であり、C言語と同様の書式が採用されている。分岐が３つ以上になる場合には、switch文が用意されている。

・switchの書式は、例えばswitch（条件，式，式，...，デフォルト値）であり、条件の結果によって実行する式が変わる。

なお用途が限られるが、この他にベクトルに対して条件を指定することができるifelseもある。

繰り返し制御文には、repeat, while, forがある。ただし、もともと変数がベクトルであることから、この特徴を生かすことで、繰り返し制御を特に使わなくとも処理ができる場合も多い。また、この方が処理速度も速くなるということである。

・repeatの書式は、repeat {式} であり、内部のnext文で上に戻り、break文でループから出て次の処理に進む。

・whileの書式は、while（条件）{式} で、条件を真である限り、{ } 内の命令を実行する。

・for の書式は、for（繰り返し指定）{式} であり、繰り返し指定に従って { } 内の命令を繰り返し実行する。ただし、指定方法は、他の言語とは異なり、あらかじめベクトル変数を、例えば M <- c(1,2,3) のように定義しておき、繰り返し指定で、(i in M) と指定する。

関数

関数は自作することもできる。この場合の書式は、関数名 <- function () { 式 } となる。ソースとして、管理することもできる。またワークスペースには、オブジェクト（データと関数）が保存される。

グラフィックス

描画モデルは、グラフ作成機能に特化していると考えてよい。直線などの作図は低水準作図関数で可能であるが、凡例やタイトルといった要素と同様にグラフ表示を補うという位置づけであり、汎用の描画システムではない。描画の手順は、通常はスクリーンに描画して確認した後、画像ファイルとして保存して利用したり、印刷するなどという手順になるだろう。

a．出力先デバイスの指定

任意のフォーマットの画像に変換：bitmap ()，JPEG：jpeg ()，PDF：pdf ()，LaTeX用の書式：pictex ()，png用画像：png ()，ポストスクリプト書式：postscript ()，XFigのフォーマットに沿った画像：X11 () などが可能である。なお、出力先デバイスを閉じる関数は、dev.off () であり、描画の最後には、この関数を実行することが必要である。

b．描画領域の設定

マージンなどの描画環境は、関数 par () で設定する。さまざまなオプション設定が可能である。

高水準グラフ描画関数だが、

1．汎用関数として、plot () のオプション指定でさまざまなグラフを作成可能である。

2．個別関数としては、

棒グラフ：barplot ()，箱ヒゲ図：boxplot ()，関数プロット：curve ()，ヒストグラム：hist ()，モザイクプロット：mosaicplot ()，散布図行列：pairs ()，鳥瞰図：persp ()，Q-Qプロット：qqplot () などが標準で使用できる。

3．以下の低水準描画関数を用いて、タイトル、凡例などを追加することができる。

直線の描画 abline ()，矢印 arrows ()，軸 axis ()，格子 grid ()，凡例 legend ()，個別の線 lines ()，マージンにテキスト mtext ()，グラフィックスの初期化 plot.new ()，座標系を初期化 plot.window ()，点 points ()，ポリゴン polygon ()，テキスト text ()，ラベル

title ()

開発環境

デバッグの方法は、変数の中身の変化を逐次追うというように基本的な方法の他に、デバッグ用関数として、browser ()，traceback () などが用意されている。また作成したオブジェクトの表示は ls ()、削除は rm () で行える。

拡張性とパッケージの利用

標準では用意されていない描画関数、統計処理関数などが多く公開されていることから、それをパッケージ単位で読み込んで利用することができる。もちろん自身で開発環境を作成して、それらを利用することができる。例えば、心理学で良く用いる統計処理をセットにすることなどが考えられる。

成績データの処理例

以下のイメージのようなデータを Excel などの表計算アプリケーションで作成する。

ID	Sex	Sansuu	Kokugo
1	F	20	45
2	M	40	70
3	F	60	20
4	F	55	50
5	M	70	50
6	F	80	30
7	M	40	60
8	M	60	80
9	F	80	20
10	M	100	90

このファイルを関数 read.table () 関数でオブジェクト（例えば、Seiseki）に読みこむ。行頭のラベルはあらかじめ入力しておくことができるが、後から追加することも可能である。

例、Seiseki <- read.table（ファイル名，header=T，sep=","）

なお、上の文はカンマ区切りの場合であり、タブ区切りのファイルでは、sep="¥t" と指定する。また読み込んだデータは、関数 edit () を利用することにより R 上でも編集できる。

データの集計では、関数 summary で、summary（Seiseki）と指定すると Seiseki データの要約統計量をまとめて算出することができる。個々の例えば、Sansuu の平均点を求めたい場合には、ドル記号 $ で変数をつないで mean（Seiseki$Sansuu）のように指定する。同様な書式で標準偏差は関数 sd ()，最大値は max ()，最小値は min ()，中央値は median () で求めることができる。

度数分布表の作成 table（cut（ヒストグラムを書く　例、hist（Seiseki , seq（0,100, by =10）

科目同士の相関係数は cor（）で算出でき、Plot（）で散布図を描くことになろう。

おおよそ、以下のような処理プログラムになる。上から順に実行される。

```
Seiseki <- read.table（ファイル名，header=T, sep=","）
summary（Seiseki）
mean（Seiseki$Sansuu）; sd（Seiseki$Sansuu）; max（Seiseki$Sansuu）
hist（Seiseki , seq（0,100, by =10）
plot（Seiseki$Sansuu, Seiseki$Kokugo）  #  散布図の指定
cor（Seiseki$Sansuu, Seiseki$Kokugo,use = "all.obs", method = "pearson"）
    # x. y の指定、欠損値の扱い、相関係数の算出方法の指定
```

文献

Ligges, U.（石田基広訳）2006「R の基礎とプログラミング技法」シュプリンガー・ジャパン.

舟尾暢男　2005「The R Tips　データ解析環境 R の基本技・グラフィックス活用集」九天社.

舟尾暢男　2007「R commander ハンドブック」九天社.

岡田昌史　2004「The R Book －データ解析環境 R の活用事例集」九天社.

舟尾暢男・高浪洋平　2005「データ解析環境『R』－定番フリーソフトの基本操作からグラフィックス、統計解析まで」工学社.

竹内俊彦　2005「はじめての S - PLUS/R 言語プログラミング－例題で学ぶ S - PLUS/R 言語の基本」オーム社.

岩橋俊哉　2007「統計言語 R の心理統計での利用について」大東文化大学紀要　45号　15-22.

（2007年 9 月25日受理）