統計解析環境 R について

鑓　田　　亨

はじめに

R は統計計算とグラフィックスのための言語および環境である。現在 R Project[1]によって活発に開発が進められており、2004年11月には Ver. 2.0.1が公開されている。R のソースコードおよびバイナリーは CRAN（The Comprehensive R Archive Network）[2]から入手することができる。

R の利点としては、以下のようなことがあげられるだろう。

⑴　一般に統計分析のためのソフトウェアは高価なものが多く、個人での利用は難しい。これに対し R は GPL（GNU General Public License）にしたがって配布されており、自由に利用することができる。

⑵　ベル研究所で開発された S のクローンであり、S 用に書かれたプログラムはほとんど修正なしで動作する。したがって S 向けに書かれたドキュメントを参考にすることができる。

⑶　他のフリーソフト（Octave[3]など）が数値計算ソフトとしての性格が強いのに対し、R では古典的な検定から、最小二乗法、主成分分析、因子分析、クラスター分析などの豊富な統計的手法が提供されている。

⑷　分析結果が、単なる画面出力ではなく、R 内で操作可能なオブジェクトとして得られる。そのため分析結果の再利用が容易である。

⑸　Microsoft Windows, Mac OS X, 各種 Linux など多くのシステムで動作する。

この研究ノートでは、因子、行列、データフレームなど R が扱うデータの構造、添字操作によるデータの抽出など、R 言語の基本的な部分について利用の際に個人的にとまどった部分を中心にまとめることにする。

1　オブジェクト

R が処理する対象はオブジェクトと言われる。オブジェクトは型（type）を持つ。オブジェクトの型は関数 `typeof( )` で知ることができる。また、オブジェクトはモード（mode）と長さ（length）を持つ[4]。オブジェクトのモード、長さを求めるには、それぞれ関数 `mode( )`, `length( )` を用いる[5]。

オブジェクトは名前（name）[6]によってアクセスされる。オブジェクトの名前には大小の英字[7]、

1）http://www.r-project.org/

2）http://cran.r-project.org/

3）http://www.octave.org/

4）他の S 言語の実装と互換性を持つ。

5）モードや長さは変更可能である。例えば、x というオブジェクトの長さを10に変更するのであれば、`length(x) <- 10` とすれば良い。後に説明する属性についても同じ方法で設定可能である。

数字、それに加えてピリオドが用いることができる。ただし、名前は数字以外で始まらなければならない。`FALSE,Inf,NA,NaN,NULL,TRUE,break,else,for,function,if,in,next,repeat,while` は予約語となっている。また `c,q,t,C,D,F,I,T,diff,mean,pi,range,rank,var` といったシステムオブジェクトの名前も避けることが望ましい。

R において、オブジェクトは、付値演算子 `<-` によって定義され、変更される[8)]。オブジェクトの定義に際して、モード、長さなどの宣言はなく、付値によって動的に変化する。

1.1 基本型

ここでは、基本型のうちデータの保管に用いられるベクトルとリストについて説明する。

ベクトル 同じ型からなるオブジェクトの集りをベクトルという。R は論理値、整数値、倍精度実数値、複素数値、文字、raw の 6 つの基本的なベクトルの型を持つ。関数 `typeof( )` と `mode( )` が返す値とは以下のように対応している。

リスト もう一つのデータ保管手段としてリスト（総称的ベクトル）がある。リストは要素

	型	モード	例
論理値	`logical`	`logical`	`TRUE, FALSE`
整数値	`integer`	`numeric`	`1, -2,3`
倍精度実数値	`double`	`numeric`	`6.2,3.4e8,0.5e-5`
複素数値	`complex`	`complex`	`4+6.7i`
文字	`character`	`character`	`"character", 'string'`
raw	`raw`	`raw`	生バイト（16進数表記）

(elements) を持ち、その各々は任意の型の R オブジェクトを含むことができる。

1.2 属性

`NULL`[9)] 以外のすべてのオブジェクトは、それらに付随する属性 (attributes) を持つことができる。属性はリストとして保管され、その全ての要素は名前を持つ[10)]。オブジェクトの属性は関数 `attributes( )` を用いて得ることができる。R はオブジェクトに種々の属性を付加することにより、オブジェクトを構造化する。代表的な属性としては以下のものがある。

名札 名札属性 (`names`) は、もし存在すれば、ベクトルあるいはリストの個々の要素のラベルとなる。また、名札属性は添字操作にも使われる。名札属性は、関数 `names( )` によっても得ることができる。

6）オブジェクトの名前は型 `symbol`、モード `name` を持つ。
7）大文字と小文字は区別される。
8）等号=が用いられるのは、関数の仮引数の省略時の値の定義、および関数呼出しの際の仮引数名と実引数名の対応をつけるために限られており、代入のためには用いられない。
9）存在しないオブジェクトを示すために用いられるオブジェクトである。
10）名前を持つ要素は成分 (components) と言われる。

次元　次元属性は配列あるいは行列(2 次元の配列)を実装するために使われる。配列・行列は次元属性を持つベクトルである。次元属性は関数 `dim( )`[11] によっても得ることができる。

```
› x ‹- 1:4
› dim(x) ‹- c(2,2)
› x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
› dim(x)
[1] 2 2
```

次元名　配列は文字ベクトルのリストである次元名属性を用いて各次元に名前を持つことができる。次元名は関数 `dimnames( )`によって得ることができる[12]。

クラス(class)　クラス属性は関数 `class( )`によって得ることができる。明示的にクラス属性を持たない場合、オブジェクトは暗黙のクラスを持つ[13]。

クラス属性は、関数にそのオブジェクトをどのように処理するかについての情報を与える。例えば線形回帰関数 `lm( )`はリストを返り値とするが、その返り値には自動的にクラス属性 `"lm"` が付け加えられる。R 言語で頻繁に使われる `plot( )`などの関数は総称的関数(generic function)と呼ばれる。これらの関数は引数であるオブジェクトのクラス属性に応じて、実際に使われる関数が決まる。クラス属性 `"lm"`を持つオブジェクトであれば、`plot( )`関数は線形回帰オブジェクトのプロット用に設計されたプロット関数 `plot.lm( )`を起動する。関数 `plot.lm( )`は関数 `plot( )`の一つのメソッド(method)と呼ばれる。`plot( )`関数はその他にも様々なクラス属性に応じたメソッドを持つ。この機構は、多くの同種の関数を同じ関数で代表させることで、直観的なデータの操作が可能になる。

時系列　時系列属性は、時系列データの始点、終点、周期を保管するために用いられる。時系列属性は関数 `tsp( )`によって得ることができる。例えば、関数 `ts( )`によって時系列属性をもつデータを作成する。

---

11) 関数 `dim( )`はデータフレームのためのメソッドを持つ。データフレーム `x` が与えられたとき、関数 `dim( )` は `c(lenght(row.names(x)),length(x))`を返す。

12) 2 次元以上の配列 `x` の `dimnames(x)[[1]],dimnames(x)[[2]]`はそれぞれ `rownames(x),colnames(x)`によっても得ることができる。

13) これは `“matrix”,“array”`、またはそのオブジェクトのモードである。

```
› x ‹- ts(c(1:24)、start=c(2004,11)、frequency=12)
› x
      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2004                                          1   2
2005   3   4   5   6   7   8   9  10  11  12  13  14
2006  15  16  17  18  19  20  21  22  23  24
› tsp(x)
[1] 2004.833 2006.750   12.000
```

1.3 特殊な複合オブジェクト

因子 因子(factor)は有限個の値を持つ項目(性別、社会的クラス等)を記述し、同じ長さを持つ他のベクトルの要素を、その項目ごとにグループ化するために用いられるベクトルオブジェクトである。因子は項目を表わす `levels` 属性[14]とクラス属性 `"factor"`を持つ。

例えば男性を 0、女性を 1 として表わしたベクトル `x` があったとしよう。x から因子オブジェクトをつくるには関数 `factor( )`を用いる。

```
› x ‹- c(1,0,0,1,1,0)
› y ‹- factor(x)
› y
[1] 1 0 0 1 1 0
Levels: 0 1
› attributes(y)
$levels
[1] "0" "1"
$class
[1] "factor"
```

項目を表わす `levels` 属性は、`x` のユニークな要素を昇順にソートしたものである。もし、男性を `M`、女性を `F` で表わしたいのであれば次のようにする。

```
› y ‹- factor(x,labels=c("M","F")) # あるいは
› y ‹- factor(x); levels(y) ‹- c("M","F")
› y
[1] F M M F F M
Levels: M F
```

データフレーム データフレームはクラス属性 `"data.frame"`を持つリストで、同じ長さを持つ数値ベクトル、因子[15]などからなる。データフレームは変量のラベルである名札属性と場合のラベルである属性 `row.names` を持つ。R での統計分析では主にデータフレームを操作することになる。

14) 関数 `levels` によって得ることができる。

15) データフレームが作成されるとき、非数値ベクトルは因子ベクトルに強制変換される。

データフレームは行列形式で表示され、その行や列は行列と同じ添字操作で抽出できる。したがって特に数値ベクトルのみからなるデータフレームは行列と同じように見えるが、クラス属性が異なるため、データフレームを行列演算の対象とすることはできない[16]。

データフレームを作成するには、関数 `data.frame( )`を用いる。ただし実際にデータを分析する際には、外部のファイルを読み込むことになるだろう。例えば、タブ区切りのテキストデータとしてデータファイルが与えられたとしよう。ファイル名を `file. txt` とする。カレントディレクトリにそのファイルがある場合、

```
read.table("file.txt",sep="\t")
```

とする。空白のセルは欠損値（NA）として扱われる。1 行目が変数名の場合、引数として `header=TRUE` を指定すると、名札属性がついた形でファイルが読み込まれる。

2　オブジェクト内の要素の抽出

2.1　単一の添字

以下のオブジェクトを例とする。

```
› x.matrix ‹-matrix(c(1:4),nrow=2);colnames(x.matrix) ‹- c("a","b")
› x.data.frame ‹- data.frame(a=1:2,b=3:4)
› x.list ‹- list(a=1:2,b=3:4)

› x.matrix    › x.data.frame    › x.list
     a b        a b             $a
[1,] 1 3      1 1 3            [1] 1 2
[2,] 2 4      2 2 4
                                $b
                                [1] 3 4
```

[および [[　‘オブジェクト名[2]’はオブジェクトの 2 番目の要素を抽出する。

```
› x.matrix[2]
[1] 2
› x.data.frame[2]
  b
1 3
2 4
› x.list[2]
$b
[1] 3 4
```

R では、行列は次元属性を持つベクトルであり、各要素は列主導で並んでいる。したがって `x.`

16）数値ベクトルのみからなるデータフレームを行列演算の対象とするためには、関数`as.matrix( )`でクラス属性を“matrix”に変換する必要がある。

matrix[2]は 2を抽出する。一方、データフレームおよびリストの要素はベクトルとなっている。したがって抽出される 2 番目の要素もベクトルである。また、抽出された要素は元のオブジェクトのモードおよび属性を継承している。

‘オブジェクト名[[2]]’ はオブジェクトの 2 番目の要素それ自体を抽出する。

```
› x.matrix[[2]]
[1] 2
› x.data.frame[[2]]
[1] 3 4
› x.list[[2]]
[1] 3 4
```

‘オブジェクト名 [2]’ の場合と異なり、元のオブジェクトのモードおよび属性は失なわれている。

名前による指定　オブジェクト内の要素は、その名前[17]によって選択することもできる。

```
› x.matrix["b"]
[1] NA
› x.data.frame["b"]
  b
1 3
2 4
› x.list["b"]
$b
[1] 3 4
```

x.matrix が持つのは次元名属性であり、要素の名前は持っていない。したがって x.matrix["b"]は NA となる。x.data.frame["b"],x.list["b"]はそれぞれ x.data.frame[2], x.list[2]と同じである。

同様に、x. data. frame [[2]], x. list [[2]] は x. data. frame [["b"]], x. list [["b"]]と書くこともできる。また

```
› x.data.frame$b
[1] 3 4
› x.list$b
[1] 3 4
```

という形式での指定も可能である。

[,, …] 形式の指定　行列（配列）やデータフレームでは、次のような形式での指定も可能である。

```
› x.matrix[1,2]
[1] 3
```

17）名札属性(names)によって与えられる。

```
› x.data.frame[1,2]
[1] 3
```

リストの場合､3を抽出したい場合は `x.list[[2]][2]`あるいは `x.list[[c(2,1)]]`とする必要がある。

添字を省略した場合は､すべてを指定したことになる。

```
› x.matrix[,2]
[1] 3 4
```

データフレームについて行のみを指定した場合、元のオブジェクトのデータ構造は継承される。逆に列のみを指定した場合、元のオブジェクトのデータ構造は失なわれ、要素そのものが抽出される。

```
› x.data.frame[2,]
  a b
2 2 4
› x.data.frame[,2]
[1] 3 4
```

2.2 添字ベクトル

`[` による抽出では、単一の添字だけでなく、複数の要素を持つベクトルによる抽出も可能である。添字ベクトルには論理ベクトル、正の整数値ベクトル、負の整数値ベクトル、文字列ベクトルが用いられる。

論理ベクトル この場合、添字ベクトル中の TRUE に対応する値が選択され、FALSE に対応する値が無視される。添字ベクトルとして論理ベクトルを用いる場合、その長さは選択対象のオブジェクトと同じ長さでなければならない。

```
› x ‹- c(20,9,16,15,1,5,13,17,12,18)
› x › 10
[1]  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
› x[x › 10]
[1] 20 16 15 13 17 12 18
```

正の整数値ベクトル この場合、添字ベクトル内の値に対応する番号の要素がその順序で選択される。

```
› x ‹- c(20,9,16,15,1,5,13,17,12,18)
› names(x) ‹- 1:10 # 結果を見やすくするために名札属性をつける
› x[c(8,3,2)]
 8  3  2
17 16  9
```

負の整数値ベクトル この場合、添字ベクトル内の（負号をとった）値に対応する番号の要素以外が選択される。

```
› x ‹- c(20,9,16,15,1,5,13,17,12,18)
```

```
› names(x) ‹- 1:10 # 結果を見やすくするために名札属性をつける
› x[-c(8,3,2)]
 1  4  5  6  7  9 10
20 15  1  5 13 12 18
```

文字列ベクトル　オブジェクトが名札属性を持っている場合には、その名前を添字ベクトルとして使うことができる。

```
› x ‹- c(20,9,16,15,1,5,13,17,12,18)
› names(x) ‹- letters[1:10] # 名札属性として a から j を付値
› x
 a  b  c  d  e  f  g  h  i  j
20  9 16 15  1  5 13 17 12 18
› x[c("f","b","i")]
 f  b  i
 5  9 12
```

3　制御構造

3.1　グループ化

R における計算は文（statements[18]）を順次評価していくことによってなされる。文は改行あるいはセミコロンによって区切ることができる。#から行末まではコメントとなる。

複数の文は括弧 ‘{’ と ‘}’ を用いてグループ化できる。文のグループはブロックと呼ばれることもある。単一の文は構文的に完全な文の最後に改行が入力されると評価されるが、ブロックは閉じ括弧の後に改行が入力されるまで評価されない。ブロックはブロック内の最後の文の値を返す。

以下、文は単一の文かブロックを意味する。

3.2　if/else 文

if/else 文は次のような形式を持つ。

```
if ( statement1 )
        statement2
else
        statement3
```

最初に statement1 が評価され、その値が論理値ベクトルで、その最初の要素[19]が TRUE であれば statement2 が評価され、FALSE であれば statement3 が評価される[20]。else 節は省略可能である。

18）オブジェクトの型の 1 つである式（expressions）との混合を避けるため、式という言葉を避けている。

19）statement1 の返り値は最初の要素だけが使われ、他の要素は警告とともに無視される。

20）statement1 が数値ベクトルの場合、0 は FALSE に、それ以外は TRUE に強制変換される。

R には if/else 文をベクトル化した関数 ifelse がある。

3.3 反復

R には tapply( ),apply( ),lapply( )といった暗黙に反復を行なうための関数が存在する。加えて多くの関数はベクトル化されているため、for,while,repeat といった明示的な反復を用いる機会は少ない。

for 文 for による反復の構文は次のような形式を持つ。

```
for (name in vector)
        statement
```

vector はベクトル（あるいはリスト）オブジェクトであり、その各要素の値が変数 name に設定され、statement が評価される。

while 文 while による反復の構文は次のような形式を持つ。

```
while ( statement1 ) statement2
```

statement1が評価され、その値が TRUE であれば statement2が実行される。
この過程は statement1の値が FALSE になるまで続く。

repeat 文 repeat による反復の構文は次のような形式を持つ。

```
repeat statement
```

この場合、statement の評価が限りなく続けられる。反復から抜けるためには break が評価されなければならない。

4 関数

4.1 関数定義

他の言語と同様に、R ではユーザーが関数を定義できる。また R システムの一部として提供される関数のほとんどは R 言語によって書かれており、このことは R 環境の柔軟性を意味している。

関数は通常、次のような付値で定義される。

```
function.name <- function ( arguments ) statements
```

arguments は仮引数をコンマで区切って並べたものである。仮引数はシンボル、または‘symbol=expression’形式の文[21]、または特殊形式引数‘…’[22]である。

statements が関数本体である。関数本体は任意の R 文であり、通常、括弧‘{’と‘}’を用いてグループ化される。関数本体での‘<-’による付値は局所的であり、関数外のオブジェクトに影響

21）この場合、シンボルで表わされる引数に expression で与えられる既定値が設定され、関数を呼び出す際にその引数の値を指定しなければ既定値が用いられる。システムが提供する多くの関数の引数には適切な既定値が設定されている。

22）特殊形式引数‘…’は任意の数の引数を含むことができる。

を与えない。

このように定義された関数は‘function.name(arugments)’と入力することで呼び出される。

4.2 引数の照合

関数が呼び出されるとき、仮引数と実引数が照合される。これは次の 3 つの段階を経る。

⑴ 実引数の名前が仮引数の名前に一致すれば、それと適合させる。

⑵ 実引数の名前が仮引数の名前と少なくとも 1 文字以上一致すれば、それと適合させる。

⑶ 名前なしの実引数を、まだ適合していない仮引数と順に適合させる。

したがって以下の関数呼び出しはすべて同等である。

```
› qnorm(p=0.99,mean=0,sd=1)
[1] 2.326348
› qnorm(m=0,s=1,0.99)
[1] 2.326348
› qnorm(0.99,0,1)
[1] 2.326348
```

5 おわりに

本来は R による具体的なデータの操作や、Emacs や LaTeX などの他のソフトの連携といった実践的な内容をとり入れる予定であったが、字数の関係から削らざるを得なかった。そのために R 言語の基本的な説明という形になってしまったが、コンパクトな分、R を利用しはじめようという人には有益な点もあるかもしれない。

多くの研究者および学生にとって、統計分析を利用することの重要性は増してきているように思う。そうした人々にとって R は貴重な道具となるだろう。情報交換ができれば幸いである。

参考文献

[1] Richard A. Becker, John M. Chambers, and Allan R. Wilks. *The New S Language—A Programming Environment for Data Analysis and Graphics—*. Wadsworth & Brooks/Cole, California,1988.(渋谷、柴田(訳)『S言語—データ解析とラフィックスのためのプログラミング環境—Ⅰ、Ⅱ』. 共立出版、1991).

[2] Peter Dalgaard. *Introductory Statistics With R*. Springer Verlag, New York, 2002.

[3] John Fox. *An R and S-Plus Companion to Applied Regression*. Sage Publications, 2002.

[4] John H. Maindonald and John Braun. *Data Analysis and Graphics Using R—an Example-based Approach*. Cambridge University Press, 2003.

[5] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2004. ISBN 3-900051-07-0.

[6] W. N. Venables, D. M. Smith, and the R Development Core Team. *Introduction to R*. Network Theory Limited, 2002.

[7] W.N. Venables and B.D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, 4th edition, 2002.

[8] 岡田昌史（編）『The R Book—データ解析環境 R の活用事例集』九天社、2004.

[9] 渋谷政昭、柴田里程『S によるデータ解析』共立出版、1992.

[10] 坪田信孝『データ解析言語 S』科学技術出版、1998.

[11] 時永祥三『S による経営情報解析』経済の情報と数理. 牧野書店、1993.

[12] 中澤港『R による統計解析の基礎』ピアソン・エデュケーション、2003.
[13] 間瀬茂、神保雅一、鎌倉稔成、金藤浩司『工学のためのデータサイエンス入門―フリーな統計環境 R を用いたデータ解析』数理工学社、2004.
[14] 渡辺利夫『使いながら学ぶ S 言語』オーム社、1994.

また、R 付属のドキュメントである「R Language Definition」および RjpWiki[23)]を参考にした。

23）http://www.okada.jp.org/RWiki/