RExcel：ExcelでRを使う（4）

㈱農業・食品産業技術総合研究機構
農村工学研究所農村計画部主任研究員

合崎 英男（Aizaki Hideo）

■2000年3月北海道大学大学院農学研究科博士後期課程修了。博士（農学）。農林水産省農業研究センター研究員、農業工学研究所研究員、同主任研究官を経て、06年4月より現職。専門分野は農業経済学（主に環境配慮や食品安全性に関する意思決定分析）。

1. はじめに

RExcelシリーズ第4回では、Excel VBAでRの関数（以下、R関数）を利用する方法について紹介します。

Excelでは、利用者がさまざまな機能を独自のプログラムとして作成・活用できるように、Excel VBA（以下、VBA）というマクロ言語を実装しています。RExcelの導入によってVBAに新たなプロシージャ等が加わり、VBA上でR関数が利用可能となります。

今号では、いくつかの計算例を紹介しますが、VBA自体の説明は最小限となっています。VBAの利用経験によっては、若干わかりにくい部分があるかもしれません。VBAを解説した書籍等は多数刊行されていますので、必要に応じて参照してください。

2. プロシージャ

RExcelを導入することで、VBAに新たなプロシージャが加わりますが、それらの先頭部分には「Rinterface.」が付いています。ここでは、今号で使用する10のプロシージャを簡単に紹介します。

「Rinterface.StartRServer」と「Rinterface.StopRServer」は、それぞれRサーバーを開始/終了させるために使用します。

「Rinterface.RRun」は、R関数で記述した命令文を実行します。たとえば、「一様乱数を1つ生成して変数aに保存する」命令文をR上で実行するには、

```
Rinterface.RRun "a<-runif(1)"
```

と記述します。

ExcelからRに数値・行列/データフレームを渡すためには、「Rinterface.PutArray」と「Rinterface.PutDataframe」を利用します。前者は、Excel上の指定したセル（範囲）の値をR上に変数として保存します。たとえば、セルA1に入力されている数値をR上に変数var1として保存するには、

```
Rinterface.PutArray "var1",
Range("A1")
```

とします。

後者は、Excel 上の指定したセル範囲を R 上にデータフレームとして保存します。たとえば、セル A1 から B10 の範囲の値（1 行目は変数名、それ以外はデータとする）を、R 上にデータフレーム mydf として保存するには、

```
Rinterface.PutDataframe "mydf",
Range("A1:B10")
```

とします。

一方、R から Excel に数値・行列 / データフレームを渡すためには、「Rinterface.GetArray」と「Rinterface.GetDataframe」を利用します。前者は、R 上の変数に保存されたデータを Excel 上のセル（範囲）に返します。たとえば、R 上の変数 var1 に保存されている値をセル A2 に返すには、

```
Rinterface.GetArray "var1",
Range("A2")
```

とします。

後者は、R 上のデータフレームを Excel 上のセル範囲に返します。たとえば、R 上のデータフレーム mydf をセル A1 を左上端とするセル範囲に返すときには、

```
Rinterface.GetDataframe "mydf",
Range("A1")
```

とします。

R 関数を使った命令文を R 上で実行し、その結果を得るためのプロシージャとして、「Rinterface.GetRApply」と「Rinterface.RunRCall」があります。前者は実行結果を Excel に返しますが、後者は返さないという違いがあります（それぞれ前回の 2010 年 10 月号で紹介したワークシート関数 RApply と RCall に対応します）。たとえば、前者を使ってセル A1 に入力された数値の 2 乗を計算し、その結果をセル A2 に返すときには、

```
Rinterface.GetRApply
"function(x)x*x", Range("A2"),
Range("A1")
```

とします。

「Rinterface.InsertCurrentRPlot」は、R 関数で作成した図のコピーを、幅 / 高さ等を指定して、Excel 上の任意のセルを左上端とした領域に貼り付けます。

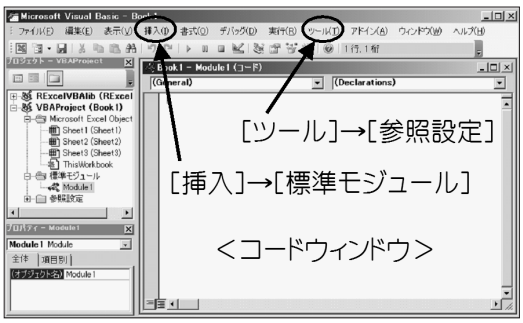
これらのプロシージャの詳細な書式や、そのほかのプロシージャ等については、REXcel のヘルプ（「RExcel」→「RExcel Help」）をご覧ください。

3. 準備

Excel の起動後、ツールバーの「RExcel」→「Start R」を選択して R を立ち上げます。次に、VBA のプログラムを作成するため、Visual Basic Editor（以下、VBE）を立ち上げます。Excel のツールバーから「ツール」→「マクロ」→「Visual Basic Editor」を選択します。

VBE（図 1）が立ち上がりましたら、REXcel

図 1　Visual Basic Editor（VBE）



の機能をVBAで実行できるように設定します。VBEのツールバーの「ツール」→「参照設定」を選択すると、参照可能なライブラリーの一覧が表示されます。その中から「RExcelVBAlib」を探して、選択されているか確認します。ライブラリー名の前のボックス（□）にチェックマーク（✓）が入っていれば、選択済みです。選択されていない場合は、同ボックスをクリックしてチェックマークを入れ、[OK] ボタンを押します。

最後に、VBEのツールバーから「挿入」→「標準モジュール」を選択し、プログラムを入力するコードウィンドウを開きます。

4. プログラム例

今回は、簡単なプログラム例を5つ紹介します。前回（2010年10月号）のワークシート関数を利用した例とほぼ同様な内容を、VBAで行うこととします。

⑴ 数値の計算

例1は、セルA1に入力された数値の2乗した値を、セルA2に出力するプログラムです。

図2にプログラムを示します。1行の「Sub」と7行の「End Sub」は、VBAの命令文です。この2つの間に記述されたコードが、1行のSubの後に記述されている「Example1」という名称のマクロとして処理されます（「Example1」の後の半角両括弧は、どのような名称のときでも付けます）。

2行から6行が、RExcelによって新たに導入されたプロシージャを使ったコード部分です。2行でRServerを開始させています。3行はセルA1に入力されている数値を変数var1としてRに保存するよう指示し、4行はRで変数var1の2乗を計算し、その計算結果を変数var2に保存するよう指示しています。5行は変数var2に保存されている値をセルA2に転送するよう指示しています。以上でR関連のコード部分が終わりであることから、6行でRServerを停止させています。

上記のプログラムをVBEに入力した上で、Excelのシートに戻り、セルA1に「3」を入力して（図3）、このマクロを実行します。マクロの実行は、Excelのツールバーの「ツール」→「マクロ」→「マクロ」から現れるウィンドウで、実行したい「マクロ名」を選択して、[実行] をクリックします。「マクロ名」は、プログラムの1行の「Sub」の後に設定した「Example1」が該当します。実行した結果、3^2の値である「9」がセルA2に出力されます。

なお、図2の3行から5行を、2節で「Rinterface.GetRApply」の例として示した命令文で、置き換えることもできます。

⑵ 複数の変数を使った計算

例2では、2つの変数の相関係数を求めます。使用するデータは、セルA4からB10の範囲に入力されている変数xとyとします（図3）。

図2の9行から14行が、本例のプログラムです。9行、10行、13行、14行は例1と同じですので、説明は省略します。11行から12行で、セルA4からB10の範囲を変数名も含めてデータフレームとして扱い（AsSimpleDF(Range("A4:B10"))）、R上で変数xとyの相関係数を求めて（"function(mydf)with(mydf, cor(x,y))"）、その結果をセルA12に返す（Range("A12")）よう指示しています。

上記のプログラムを「Example2」として、先ほどの「Example1」のプログラムの後に入

図２　Example1 ～５のプログラム

```
 1行 | Sub Example1()
 2行 |     Rinterface.StartRServer
 3行 |     Rinterface.PutArray "var1", Range("A1")
 4行 |     Rinterface.RRun "var2<-var1*var1"
 5行 |     Rinterface.GetArray "var2", Range("A2")
 6行 |     Rinterface.StopRServer
 7行 | End Sub
 8行 |
 9行 | Sub Example2()
10行 |     Rinterface.StartRServer
11行 |     Rinterface.GetRApply "function(mydf)with(mydf, cor(x,y))", _
12行 |     Range("A12"), AsSimpleDF(Range("A4:B10"))
13行 |     Rinterface.StopRServer
14行 | End Sub
15行 |
16行 | Sub Example3()
17行 |     Rinterface.StartRServer
18行 |     Rinterface.PutArray "mat1", Range("A5:B10")
19行 |     Rinterface.RRun "mat2<-t(mat1)"
20行 |     Rinterface.GetArray "mat2", Range("A14")
21行 |     Rinterface.StopRServer
22行 | End Sub
23行 |
24行 | Sub Example4()
25行 |     Rinterface.StartRServer
26行 |     Rinterface.RunRCall "function(mydf)with(mydf, plot(x,y))", _
27行 |     AsSimpleDF(Range("A4:B10"))
28行 |     Rinterface.InsertCurrentRPlot Range("C17"), _
29行 |     widthrescale:=0.3, heightrescale:=0.3, closergraph:=True
30行 |     Rinterface.StopRServer
31行 | End Sub
32行 |
33行 | Sub Example5()
34行 |     Rinterface.StartRServer
35行 |     Rinterface.PutDataframe "mydf", Range("A4:B10")
36行 |     Rinterface.RRun "out<-summary(lm(y~x, data=mydf))"
37行 |     Rinterface.GetArray "out$coef", Range("B30")
38行 |     Rinterface.GetArray "t(colnames(out$coef))", Range("B29")
39行 |     Rinterface.GetArray "rownames(out$coef)", Range("A30")
40行 |     Rinterface.StopRServer
41行 | End Sub
```

力します。セル A4 から B10 のデータを変数名も含めて入力した上で、例 1 と同様な手順で「Example2」マクロを実行すると、図 3 の 12 行に示す結果が得られます。

なお、図 2 の 11 行の末尾にある「_」は、VBA で定義されている記号で、その行に書かれている命令文は、次の行にも続いていることを表します。11 行と 12 行を 1 つの行で記述するときは、この「_」を削除した上で、12 行の内容を 11 行の末尾から続けて記述します。

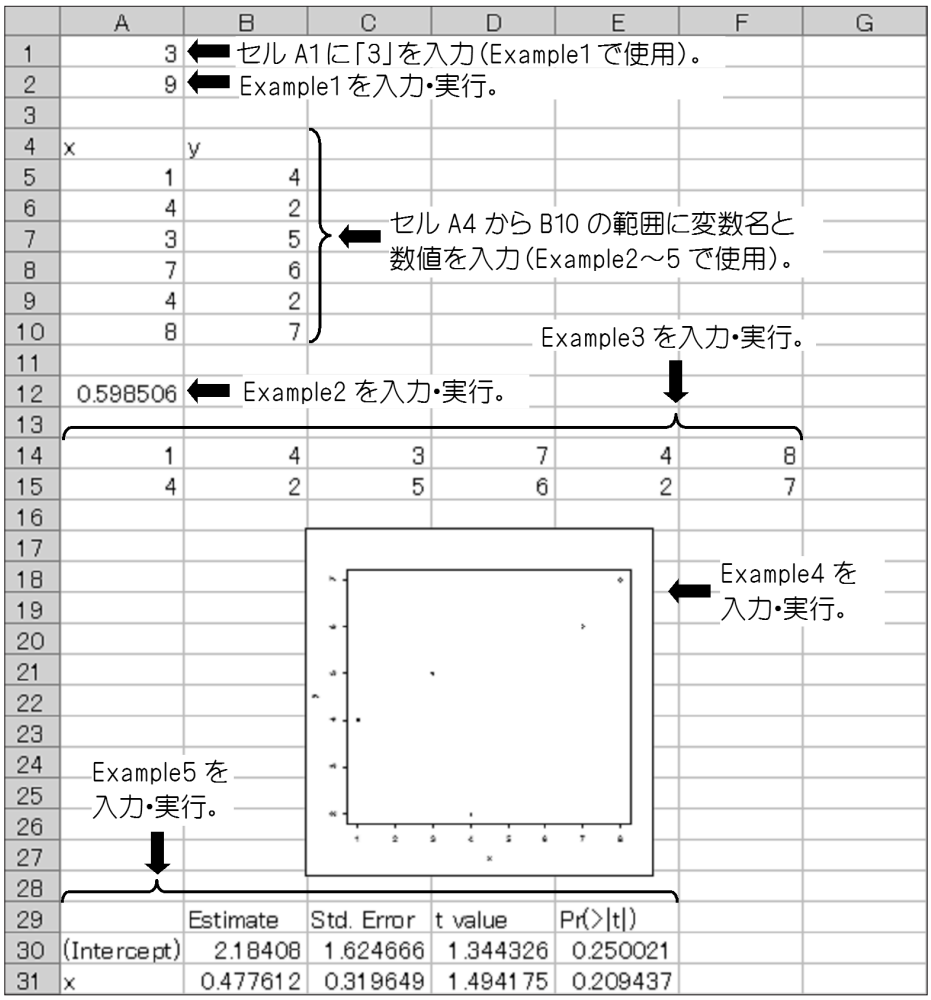
⑶ 行列の計算

例3では、行列の計算を行います。図3の変数xとyの数値部分のみに注目して6行2列の行列とみなし、それをRに転送して転置し、転置行列をExcelに戻します。

図2の16行から22行が、本例のプログラムです。18行はセルA5からB10の範囲のデータ（Range("A5:B10")）をR上に行列mat1として保存し、19行は行列mat1の転置行列（t(mat1)）を行列mat2に保存し、20行は行列mat2をセルA14を左上端とする範囲（Range("A14")）に戻すよう指示しています。

このプログラムを「Example3」として、例2のプログラムの後に入力・実行すると、図3の14行から15行に示す結果が得られます。

図3　各Exampleの実行結果



	A	B	C	D	E	F	G
1	3						
2	9						
3							
4	x	y					
5	1	4					
6	4	2					
7	3	5					
8	7	6					
9	4	2					
10	8	7					
11							
12	0.598506						
13							
14	1	4	3	7	4	8	
15	4	2	5	6	2	7	
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29		Estimate	Std. Error	t value	Pr(>|t|)		
30	(Intercept)	2.18408	1.624666	1.344326	0.250021		
31	x	0.477612	0.319649	1.494175	0.209437		

⑷ 作図

例4では、変数xとyの散布図を描きます。

図2の24行から31行が、本例のプログラムです。26行から27行は、セルA4からB10の範囲にあるデータをデータフレームとしてRに取り込み（AsSimpleDF(Range("A4:B10"))）、関数plotを使って作図（"function(mydf) with(mydf, plot(x,y))"）するよう指示しています。この命令によって、新たなウィンドウに変数xとyの散布図が描かれます。28行から29行は、その散布図のコピーをExcel上のセルC17を左上端とした範囲に、指定した幅と高さ（ともに0.3）で貼り付け、もとの散布図は削除（closergraph:=True）するよう指示しています。

このプログラムをExample4として入力・実行した結果が、図3の17行から27行の範囲にある図です。

⑸ 統計モデル分析

最後の例は、変数xとyを利用した回帰分析です。

図2の33行から41行が、本例のプログラムです。35行は、セルA4からB10の範囲にある変数xとyのデータセット（Range("A4:B10")）を変数名も含めてR上にデータフレーム（"mydf"）として転送するよう指示しています。36行は、R上で関数lmを利用して回帰分析を実行し（lm(y~x, data=mydf)）、その結果の要約（関数summary）をoutに保存するよう指示しています。そして、outに保存されている情報のうち、37行では得られた推定値等（"out$coef"）をセルB30（Range("B30")）を左上端とする範囲に、38行では「Estimate」などの項目名（"t(colnames(out$coef")をセルB29（Range("B29")）から右側のセル範囲に、39行では変数名（"rownames(out$coef)"）をセルA30（Range("A30")）から下の範囲に、それぞれ返すよう指示しています。

上記のプログラムをExample5として入力・実行した結果が、図3の29行から31行の部分になります。

5. おわりに

Excelのツールバーの「RExcel」→「Demo Worksheets」→「Writing macros」にはマクロを使った例が示されています。今号の例と同じR関数を使った作業を、異なるコードで実行しているケースもあります。また、今回は紹介できなかったプロシージャも利用されています。関心のある方は、必要に応じて参照してください。