香 川 大 学 経 済 論 叢
第90巻 第2号 2017年9月　93－124

研究ノート

データ分析へのExcel VBAの活用(2)

大　野　拓　行

はじめに

前稿，大野(2017a)，においてはワークシートを用いたデータ分析の枠組みを考え，マクロの雛形を作成した。本稿においては，作成したマクロの雛形を拡張することを通して，さらにExcel VBAの特徴をみていくことにする。

Ⅲ．IsNumeric関数，IsNumber関数の特性と数値データの抽出

1．IsNumeric関数，IsNumber関数の特性

VBAにおいてデータが数値かどうかの判別によく使用されるのが，IsNumeric関数とIsNumber関数である。ここでは，先ず，２つ関数の特性をみておく。また，データ分析のマクロではSpecialCellsメソッド（指定された条件を満たしているすべてのセルを返す）も利用されるので，どのようなデータをSpecialCellsメソッドでは数値とみなしているかを合わせてみておくことにする。(1)

まず，MSDN（Microsoft Developer Network）などの情報をもとに，IsNumeric関数，IsNumber関数とSpecialCellsメソッドをまとめておく。

・IsNumeric関数（VBA関数）：

書式　IsNumeric(expression)

（1）　SpecialCellsメソッドには，ほかにも留意すべき特性があるので節を改めて考察する。

引数 *expression* が数値として認識できる場合は真（True）を返す。それ以外は偽（False）を返す。桁区切り文字や全角の数字も数値として認識される。引数 *expression* が日付式の場合は，偽（False）を返す。MSDN から引用，一部改変。

・IsNumber 関数（ワークシート関数）：

書式　WorksheetFunction. IsNumber（expression）

引数 *expression* が数値を参照するとき True を返す。引数値は変換しない。二重引用符で囲まれた数値は文字列として扱われる。「Office のヘルプ」（https://support.office.com/ja-jp/）から引用，一部改変。

・SpecialCells メソッド：

書式　Range オブジェクト. SpecialCells（xlCellTypeConstants, xlNumbers）

IsNumeric 関数，IsNumber 関数のように，引数 *expression* が数値（数値とみなすことができる）かを判定する関数ではなく，指定された Range オブジェクトに数値が入力されているセルがあれば，それを Range オブジェクトとして返す。(2)

表 1 は様々なデータにおいて，IsNumeric 関数，IsNumber 関数と SpecialCells メソッドの差異をみた結果である。ワークシートのセルに「対象」列のデータを代入しておき，IsNumeric 関数，IsNumber 関数を利用して，戻り値が True の時は CDbl 関数により，Double 型にし，セルに表示した。表において数値が表示されていない欄は数値とは認識されなかったと判断できる。また，別途，SpecialCells が対象データを数値として扱うかを調べた。(3)

表に基づいて，IsNumeric 関数，IsNumber 関数の特性をまとめてみると，以下のようになる。

1）IsNumeric 関数は MSDN の説明にあるように，数値として認識できる桁区切り文字や全角の数字も数値として認識する。対象セルの Value プロパティを適用

（2） SpecialCells メソッドでは，「数値セル」と「数式の結果が数値になっているセル」を区分しており，数式の結果が数値になっているセルを対象とする場合は，第 1 引数を変更して，SpecialCells（xlCellTypeFormulas, xlNumbers）とする。

（3） 表 1 のために作成したマクロと結果を付図 1 に示す。

表 1　IsNumeric 関数，IsNumber 関数，SpecialCells メソッド

	対象	説明	TypeName	IsNumeric		IsNumber	SpecialCells
				. Value	. Text		
①	2	数値	Double	2	2	2	○
②	1	数字（半角文字列）	String	1	1		
③	１０．５	数字（全角文字列）	String	10.5	10.5		
④	17	数式	Double	17	17	17	
⑤	￥100	通貨型	Currency	100	100		○
⑥		空白	Empty	0			
⑦	10,5	数字（半角，カンマ付）	String	105	105		
⑧	１０，５	数字（全角，カンマ付）	String	105	105		
⑨	香川	文字列	String				
⑩	－	文字列	String				
⑪	"10.5"	“数値”	String				
⑫	FALSE	Boolean 型	Boolean	0			
⑬	TRUE	Boolean 型	Boolean	－1			
⑭	2017/3/20	日付型	Date				○

すると，空白セル（値 0），Boolean 型データも数値として認識する。また，Text プロパティを適用すると，空白セル，Boolean 型データは数値として認識されない。

2）通貨型データは IsNumeric 関数では，Value プロパティ，Text プロパティともに，数値として認識されるが，IsNumber 関数では数値として認識されない。

3）IsNumeric 関数はカンマ（,）を桁区切り文字として認識するので，ピリオドを誤ってカンマとして入力されたデータは注意が必要である。

4）IsNumber 関数は，IsNumetric 関数に比較して厳格だといえる。桁区切り文字や全角の数字に加え，通貨型データも数値としては認識されない。

5）対象が数式の結果が数値になっているセルの場合，IsNumeric 関数，IsNumber 関数ともに数値として認識する。

2．数値データの抽出

Excel VBA によるデータ分析では，以上のような特性を念頭において IsNumeric 関

数，IsNumber 関数を利用することが重要である。本稿では，さまざまな型のデータが混在するセル範囲から数値データを抽出するマクロの一例を作成することを通して，Excel VBA の特徴をみていくことにする。

まず，ここでは，数値データの抽出の方針を次のように考えることにした。

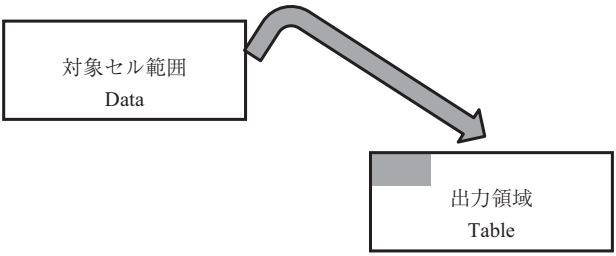
・Text プロパティを利用した IsNumeric 関数で認識されたデータを数値とすることを基本とする。すなわち，対象が数式の結果が数値になっているセル（表における④），通貨型データ（⑤），に加え，文字列として入力した数字（②，③）も数値として扱うことにする。

・ただし，カンマ付数字（⑦，⑧）は，ピリオドを誤ってカンマと入力した可能性があるので，数値と扱うかどうかを確認する。(4)

・Text プロパティを利用した IsNumeric 関数では空白セルは数値として認識されないが（⑥），空白セルを数値 0 として扱いたいケースも多いので，その点も考慮する。

以上の方針に基づいて作成したマクロ（数値の抽出）を付図 2 に掲げる。マクロのイメージは図 1 のようになる。大野（2017a）で説明した Function プロシージャ F_Input1，F_Table，F_Check1(5) を利用して対象セル範囲への参照（Range 型変数 Data）

図 1　マクロ（数値の抽出）のイメージ



（4） カンマを桁区切り文字として，入力データを数値として認識されると，「10,5」は数値 105 として扱われることに注意が必要である。

（5） 大野（2017a）の付図 1 を参照。

図 2　空白セルの処理(2)

```
50  If k > 0 Then
51      myMsg = "空白セルを数値（0）としますか？"
52      myBtn = MsgBox(myMsg, vbYesNo, myTitle)
53      Table. SpecialCells(xlCellTypeBlanks). Interior. ColorIndex = 0
54      If myBtn = vbYes Then
55          Table. SpecialCells(xlCellTypeBlanks). Value = 0
56      End If
57  End If
```

を取得し，数値とは認識できないデータは空白セルとして，対象セル範囲と同じ大きさの出力領域（Range 型変数 Table）に書き出す。

マクロの 30 行目で，Function プロシージャ F_Input1 を使用して，対象セル範囲への参照を Range 型変数 Data に Set している。また，34～35 行目で，Function プロシージャ F_Table を使用して，出力領域の準備をしている(6)。出力領域の大きさは対象セル範囲の大きさと同じなので，F_Table の引数が，Data.Rows.Count（Data の行数），Data.Columns.Count（Data の列数）となっている(7)。38 行目で，対象セル範囲の値（Data.Value）を出力領域の値（Table.Value）に代入して(8)，以後は出力領域のデータを操作することにしている。

（空白セルの処理：43～57 行）

空白セルの処理は 2 つの部分から成る。

(1)　43～49 行：For～Next ステートメントを使用して，空白セルの数（k）を数えながら，空白セルがあったら，セルの背景色を薄水色（ColorIndex = 34）にする。

（6）　大野（2017a）の「Ⅱ．データ分析の手順から見る Excel VBA の特徴」を参照。

（7）　Count プロパティは大野（2017 a）の付表 5 にあるように，コレクションオブジェクトに含まれるオブジェクト数を返す。Range オブジェクトに適用すると，その要素数を返す。Data. Rows が Range 型変数 Data で参照されている領域の全ての行を表す Range オブジェクトを返し，その要素数（行数）を Count プロパティで取得している。大野（2017a）の図 1 を参照。

（8）　Set Table = Data との差異が重要である。Set Table = Data とすると，Data が持っているセル範囲への参照が，Table に代入され，図 1 における 2 つの領域が 1 つ（Data）になってしまう。

⑵ 50～57行：空白セルが存在したら（k > 0），52行目で数値（0）として扱うかどうかをMsgBox関数で問い，その戻り値がvbYesであったら，55行目で，空白セルの値を数値（0）としている。ここで，Excel VBAに特徴的な，SpecialCellsメソッドを使用している。Table.SpecialCells（xlCellTypeBlannks）は出力領域（変数Tableで参照されるRangeオブジェクト）において，空白セルからなるRangeオブジェクトを戻り値として返す。SpecialCellsメソッドを使うと，個々の空白セルの位置に考慮することなく，空白セル全体に対しての操作が可能となる。(9) 53行目では背景色を色無しに，55行目では値を0としている。

（カンマ付き数字の処理：60～87行）

カンマ付き数字の処理も空白セルの処理と同様に2つの部分からなるが，少し複雑になる。

⑴ 60～76行：空白セルの処理と同様に，For～Nextステートメントを使用している。変数kを，数値とすべきかどうかを確認するデータ（カンマ付き数字）のカウンターとして使用している。

62行目で，IsNumeric関数を利用して，Table(i).Textが数値と見なせるかを判定し，その結果をBoolean型変数c1に代入している。IsNumeric関数で数値として認識されるデータに対してはc1 = Trueとなる（表1における「. Text」の欄を参照）。

63行目で，IsNumber関数を利用して，Table(i).Valueが数値かを判定し，その結果をBoolean型変数c2に代入している。IsNumber関数で数値と認識されるデータに対してはc2 = Trueとなる（表1における「IsNumber」の欄を参照）。

図3 カンマ付き数字の処理⑴

```
60    k = 0
61    For i = 1 To Table.Count
62        c1 = IsNumeric(Table(i).Text)
63        c2 = WorksheetFunction.IsNumber(Table(i).Value)
64        If Not c1 Then Table(i).Value = ""
```

（9） SpecialCellsメソッドは，対象となるRangeオブジェクトに，引数で指定したセルが存在しないケースではエラーとなることに注意が必要である。

```
65          If c1 And (Not c2) Then
66              If TypeName(Table(i).Value) = "Currency" Then GoTo Label1
67              Table(i).Interior.ColorIndex = myColor
68              k = k + 1
69              If k = 1 Then
70                  Set r = Table(i)
71              Else
72                  Set r = Union(r, Table(i))
73              End If
74          End If
75  Label1:
76      Next i
```

64行目で，IsNumeric関数で数値とみなされない（Not c1）セルには空白が代入される。[10] 65～76行目において，数値とすべきかどうかを確認するデータ（カンマ付き数字）を探し出し，そのセルの背景色を薄水色にしているのであるが，以下のことに留意が必要である。

1）65行目の条件「c1 And (Not c2)」に該当するデータは，表1における②，③，⑤，⑦，⑧である。

2）このうち，文字列としての数字（②，③）は38行目のDataの値をTableに代入する時に，数値として代入されるので，処理済みである。[11]

3）通貨型データ（⑤）は66行目で処理の対象外としている。

4）処理を要するデータはカンマ付き数字（⑦，⑧）となり，67行目で背景色を薄水色とし，68行目でカウンターに1を加えている。

5）空白セルの処理と大きく異なるのは69～73行目までの処理である。空白セルの処理では，SpecialCellsメソッドに引数xlCellTypeBlannksがあったので，

(10)　64行目の条件判断部分は「c1 = False」でもよい。

(11)　大野（2017a）の付表5にあるように，MSDNでは，Range.Valueプロパティは，「指定されたセル範囲の値を表すバリアント型（Variant）の値を設定します。値の取得および設定が可能です。」と説明されている。一つの解釈としては文字列としての数字には接頭辞（'）が付いているが，Valueプロパティには接頭辞は含まれないと考えることができる。セルのデータの接頭辞を取得するには，別にPrefixCharacterプロパティがある。Mougモーグ（http://www.moug.net/）の「Q&A掲示板」で教授いただいた。

SpecialCells メソッドを利用して，該当するセルを一括処理できたが，今回は，条件がカンマ付き数字で，SpecialCells メソッドは利用できない。この問題を Union メソッド[12]で処理している。最初にカンマ付き数字が出てきた時（k ＝ 1），セルへの参照を Range 型変数 r にセットし（70 行目），２度目からは Union メソッドで，r の参照するセルを拡張していく（72 行目）。最終的には r は条件を満たすセル参照の集合（Range オブジェクト）が入ることになる。

(2) 77～87 行：カンマ付き数字のセルが存在したら（k > 0），79 行目で数値として扱うかどうかを MsgBox 関数で問い，その戻り値が vbYes であったら，For Each～Next ステートメント（83～85 行目）で，CDbl 関数により数値に変換する。なお，メッセージボックスにおいて，［いいえ］を選択すれば，該当セルの背景色は薄水色のままでマクロは終了する（80 行目）ので，手作業でデータを修正できる。

図 4　カンマ付き数字の処理(2)

```
77    If k > 0 Then
78        myMsg = "これらのセルを数値としますか？"
79        myBtn = MsgBox(myMsg, vbYesNo, myTitle)
80        If myBtn = vbNo Then Exit Sub
81        If myBtn = vbYes Then
82            r. Interior. ColorIndex = 0
83            For Each rl In r
84                rl. Value = CDbl(rl. Value)
85            Next rl
86        End If
87    End If
```

この項の最後に，マクロ（数値の抽出）の実行例を図 5 に示す。

(12) Application のメソッドで，２つ以上のセル範囲の集合（Range オブジェクト）を返す。

図 5　例示(数値の抽出)

	A	B	C	D	E	F	G	H	I	J
1										
2		1	2	3	0		1	2	3	0
3		7月20日								
4		3	1	10,5	10．5		3	1	10,5	10.5
5		10，5	18		香川		1 0，5	18		
6		－	FALSE	TRUE	¥100					¥100.00
7										
8			Data					Table		

対象となるデータはセル範囲 B2:E6 で，B4 セルと，C5 セルには数式が入っている。出力領域の左上隅セルとして G2 セルを指定し，空白セルは数値（0）とせず，また，カンマ付き数字も数値に変換していない。

セルにあるどのデータを数値と見なすかの議論は別にしても，図 5 の Table のように空白セルと数値セルが混在している状態では分析がやりづらいので，次節で数値セルの整形を考える。

Ⅳ．数値セルの整形と SpecialCells メソッドの特性

1．数値セルの整形

Excel でデータを扱う時は，データをリスト形式で記述することが多い。しかし，単一系列を扱う場合などには，縦長のリスト形式は不便で，矩形にデータを記述するケースも多い。手作業で縦 1 列あるいは横 1 列（以後は 1 列と表記する）データを矩形データに整形する，あるいは矩形データを 1 列データに整形する作業は単調であり，空白など，不要なセルを除きながらの作業は間違いも多くなりがちである。

前節（数値の抽出）を前提として，数値セルと空白セルからなるセル範囲から数値セルを取り出し整形することを目的としたマクロ（数値セルの整形）を付図 3 に掲げる。主要な変数は図 6 のとおりである。

図 6　主な変数（数値セルの整形）

```
10    Dim Data As Range: '対象セル参照
11    Dim Table As Range: '出力領域セル参照
12
13    Dim n As Long: 'Data のセル数
14    Dim c As Long, r As Long: 'Data の列数（c），行数（r）
15    Dim nl As Long: '数値セルの数
16    Dim myCut As Long: ' 1 列データ => 矩形での区切り数
17    Dim iEnd As Long: '整形後の矩形の行数
```

31 行目で，Function プロシージャ F_Input1 を用いて，対象セル範囲の参照を変数 Data にセットしている。また，38 行目で SpecialCells メソッドを利用して数値セルの数（nl）を取得している。SpecialCells メソッドは領域内に指定したセルが存在しない場合はエラーとなるので，エラートラップで処理している。

マクロの主要部分は，53 行目以降で，⑴ 1 列→矩形への整形の準備（55～68 行目），と⑵数値データの書き出し（71～83 行目），の 2 つから成る。以下，それぞれの特徴的な部分をみていく。

⑴　1 列→矩形への整形[13]の準備（55～68 行目）

53 行目で，1 列データを矩形データに整形するのか（c = 1 Or r = 1），矩形データを縦 1 列データに整形するのかを判別している。

56～63 行目において InputBox メソッドを利用して，区切り数（整形後の矩形の列数）を取得し（myCut），それを用いて，66～67 行目で，整形後の矩形の行数（iEnd）を計算している。

図 7　整形後の矩形の行数（iEnd）の計算

```
66    iEnd = nl ¥ myCut
67    If nl Mod myCut > 0 Then iEnd = iEnd + 1
```

(13)　矩形へのデータの書き出しは行方向（横方向）優先に行われる。書き出した後，Excel の「行列を入れ替えてコピー」機能を使用すれば，列方向（縦方向）にできる。

n1 は 38 行目で計算される数値セルの数であり，¥，Mod は Visual Basic における整数除算演算子，剰余（余りだけを返す除算）演算子である。(14)

(2) 数値データの書き出し（71〜83 行目）

大野（2017a）の図 16（Function プロシージャ F_Check1）の説明でもみたように，Excel VBA では，矩形（Range オブジェクト）の要素は 2 元配列として（行 index，列 index）として扱う以外にも，要素を行方向の連続した index で順序づけた 1 元配列としても扱うことが可能である。この Range オブジェクトの性質を利用すれば，1 列→矩形の整形と，矩形→ 1 列の整形を一体的に取り扱うことが可能となる。

Function プロシージャ F_Table を用いて，1 列→矩形の整形では 76 行目で，出力領域 Table（iEnd, myCut）を準備し，矩形→ 1 列の整形では 78 行目で，出力領域 Table（n1, 1）を準備している。あとは，数値セルの参照 Data の内容を出力領域 Table に書き出すのみであるが，ここでは，それをサブルーチン化（S_Format）している。

図 8 数値の書き出し（S_Format）

```
 1  Sub S_Format(Source As Range, Target As Range)
 2  '
 3  '空白セルを除外しながらの整形(Source -> Target)
 4  '
 5    Dim i As Long, k As Long
 6
 7    k = 1
 8    For i = 1 To Source.Count
 9        If Source(i).Value <> "" Then
10            Target(k).Value = Source(i).Value
11            k = k + 1
12        End If
13    Next i
14  End Sub
```

(14) 例えば，5 ¥ 3 は 1，5 Mod 3 は 2 となる。

2．SpecialCells メソッドの特性

大野（2017a）の付表5にあるように，SpecialCells メソッドは Range オブジェクトのメソッドで，指定された条件を満たしているすべてのセル（Range オブジェクト）を返す。[15]

書式　Range オブジェクト. SpecialCells（Type, Value）

第1引数 Type には取得するセルの種類を指定する。指定できる種類は10種類あり，付図2で掲げたマクロ（数値の抽出）においては，空白セルを指定する定数 xlCellTypeBlanks を用いている（53，55行目）。

53行目　Table.SpecialCells（xlCellTypeBlanks）. Interior. ColorIndex ＝ 0
55行目　Table.SpecialCells（xlCellTypeBlanks）. Value ＝ 0

または，第1引数が xlCellTypeConstants（定数が含まれているセル），あるいは xlCellTypeFormulas（数式が含まれているセル）のケースでは，第2引数 Type で，特定の種類の定数や数式を含むセルだけを取得するように指定可能である。付図3のマクロ（数値セルの整形）においては，第1引数を xlCellTypeConstants として，第2引数で，xlNumbers（数値）を指定している（38行目）。

n1 ＝ Data. SpecialCells（xlConstants, xlNumbers）. Count

SpecialCells メソッドの使用方法については，解説書や Excel VBA 関連のウェブサイトに詳しいので[16]，本稿では省略し，主に SpecialCells（xlCellTypeBlanks）と SpecialCells（xlConstants）の使用上の留意点についてみていくことにする。

（1） 対象となる Range オブジェクトに，引数で指定したセルが存在しないケースではエラーとなるので対処法を講じておく必要がある。

付図2のマクロ（数値の抽出）では，43～49行目で空白セルの存在を確認した上

（15） Range オブジェクトのメソッドなので，MSDN などでは，「Range.SpecialCells メソッド」と表記する場合もある。
（16） 大村（2010）「8-10 特定の種類のセルを参照する」，Moug モーグ（http://www.moug.net/）「即効テクニック：特定のセルを選択する」などを参照。

で，SpecialCells(xlCellTypeBlanks）を使用している。また，付図 3 のマクロ（数値セルの整形）では，37～43 行目のエラートラップで，数値セルが存在しない場合はマクロを終了させている。

(2)　SpecialCells メソッドの特性①：複数セル範囲を指定した場合

SpecialCells メソッドは Range オブジェクトに対して，指定された条件を満たしているすべてのセル（Range オブジェクト）を返すが，その使用に当たっては，「使われたセル範囲」に留意する必要がある。

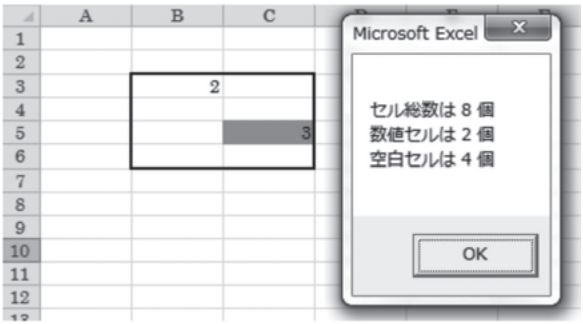
図 9　SpecialCells メソッドの特性①

```
Sub Sample12()
'
'SpecialCells の特性①
'（注意）新規ワークシートが追加されます
'
  Dim r As Range
  Dim n1 As Long:'セル総数
  Dim n2 As Long:'数値セル数
  Dim n3 As Long:'空白セル数

  Worksheets.Add

  Range("B3").Value = 2
  Range("C5").Value = 3

  Set r = Range("B3:C6"):'対象とする範囲

  'Range("E9") = 10:Range("E9").Delete:'E9 セルに入力し，削除
  'r.Borders.LineStyle = xlContinuous:'罫線
  'r.Interior.ColorIndex = 4:'塗りつぶし

  'LastCell を塗りつぶす
  Range("A1").SpecialCells(xlCellTypeLastCell)._
      Interior.ColorIndex = 48

  n1 = r.Count
  n2 = r.SpecialCells(xlCellTypeConstants, xlNumbers).Count
```

```
28    n3 = r.SpecialCells(xlCellTypeBlanks).Count
29
30    r.BorderAround Weight:= xlMedium
31    MsgBox "セル総数は" & n1 & "個" & vbCrLf &_
32        "数値セルは" & n2 & "個" & vbCrLf &_
33        "空白セルは" & n3 & "個"
34
35    Range("B8:E10").SpecialCells(xlCellTypeBlanks).Select
36
37  End Sub
```

SpecialCells(xlCellTypeBlanks) を例に，このことを説明するために作成したマクロが Sample12 である。11～14 行目で，新規ワークシートを挿入し，B3，C5 セルに数値を代入している。この時点で，使われた最終セル（LastCell）は C5 となり[(17)]，「使われたセル範囲」は A1:C5 となる。この状態で，セル範囲 B3:C6 を Range 変数 r にセットし（16 行目），セル総数（26 行目），数値セル数（27 行目），空白セル数（28 行目）を取得し，表示させてみると，セル総数 8，数値セル数 2 は正しく表示されるが，空白セル数は 6 であるはずなのに，表示は 4 となる（図 10）。

図 10　Sample12 の実行結果



(17)　23～24 行目において，Specialcells(xlCellTypeLastCell) メソッドを利用して，LastCell を探し出し，グレーに塗りつぶしている。また，Worksheet.UsedRange プロパティを使用すれば，「使われているセル範囲」の Range オブジェクトを取得できる。

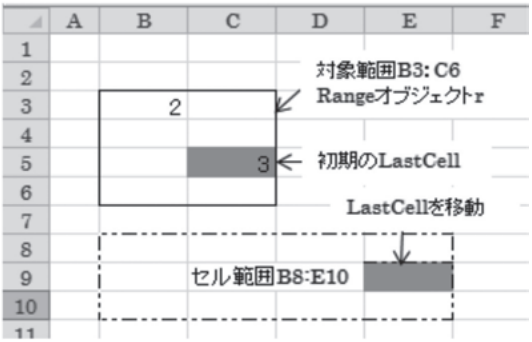
これは，「使われたセル範囲」が A1:C5 であるため，指定セル範囲 B3:C6 との共通部分が Specialcells メソッドの対象となっており，空白セル数が 4 となるのである（図 11 を参照）。また，「使われたセル範囲」が A1:C5 である状態で，35 行目のように，「使われたセル範囲」に含まれないセル範囲に対して SpecialCells（xlCellTypeBlanks）メソッドを適用しようとするとエラーとなる。(18)

「使われたセル」という表現は，数値や数式の代入だけではなく，罫線や塗りつぶしなども含まれることにも留意する必要がある。(19)

次に，18 行目を非コメント化してマクロを実行する。18 行目は E9 セルに数値を代入して削除している。これで LastCell は E 9 に移動して，「使われたセル範囲」は A1:E9 に拡大されるので，セル範囲 B3:C6 に対する SpecialCells(xlCellTypeBlanks) メソッドは正しい結果を返す。

35 行目もエラーで止まることはないが，しかしながら，期待されるセル範囲 B8:E10 は選択されず，「使われたセル範囲」A1:E9 との共通部分であるセル範囲 B8:E9 が選択されることに注意が必要である。

図 11　SpecialCells(xlCellTypeBlanks) の特性



(18)　初期の状態ではこのマクロは 35 行目で「該当するセルが見つかりません」のエラーメッセージを出して止まる。

(19)　コメントアウトしている 19 行目や 20 行目を非コメント化して確かめることができる。

以上から，複数セル範囲を対象として SpecialCells メソッドを使用する場合には，常に，使われた最終セル（LastCell）に留意する必要があり，使用が難しいとも考えられるが，ワークシートで使用しないであろう遠くのセルを LastCell としておけば，簡単に問題は解決できる。例えば，

Cells(1000, 1000). Value ＝“LastCell”

を SpecialCells メソッドを実行する前に追加しておけばよい[20]。

SpecialCells メソッドの対象とする範囲が指定した範囲と「使われた範囲」の共通部分であることから，新規ワークシートにおいては，全てが空白セルではあるが，SpecialCells(xlCellTypeBlanks）メソッドを使用すると，実行時エラー（該当するセルが見つかりません）となることにも注意が必要である。

(3) SpecialCells メソッドの特性②：単一セルを指定した場合

一般に， SpecialCells メソッドは複数セル範囲に対して使用されるケースが多いが，単一セルを対象として，SpecialCells メソッドを使用した場合には，どの位置にある単一セルを指定しても，「使われた範囲」が SpecialCells メソッドの対象となることにも注意が必要である。このことを説明するために作成したマクロが Sample13 である。

図 12 SpecialCells メソッドの特性②

```
Sub Sample13()
'
'SpecialCells の特性②
'(注意)新規ワークシートが挿入されます
'      終了はダイアログボックスで［Cancel］
'
  Dim r As Range
  Dim myTitle As String
  Dim myMsg As String

  Worksheets.Add
```

(20) 論理的には，ワークシートの最終セル Cells(Rows. Count, Columns. Count）を LastCell に設定することが望ましいが，実行時に無駄な時間がかかる。

```
12
13      Range("B3").Value = 2
14      Range("C5").Value = 3
15
16      '単一セルの指定
17  Label1:
18      myTitle = "SpecialCellsの特性②"
19      myMsg = "単一セルを指定して下さい"
20      Set r = F_Input1(myMsg, myTitle)
21      If r Is Nothing Then Exit Sub
22
23  '   Set r = r.SpecialCells(xlCellTypeConstants)
24      Set r = r.SpecialCells(xlCellTypeBlanks)
25
26      r.Select
27      MsgBox r.Count
28      GoTo Label1
29
30  End Sub
```

14行目まではSample12（図9）と同一であるので，「使われた範囲」はA1:C5となる。20行目で，大野（2017a）で作成した，セル範囲を取得するFunctionプロシージャF_Input1を使用して単一セルを指定して，24行目で，そのセルにSpecialCells（xlCellTypeBlanks）を適用し，26行目で戻り値（Rangeオブジェクト）を選択し，27行目で空白セルの数を表示させている。

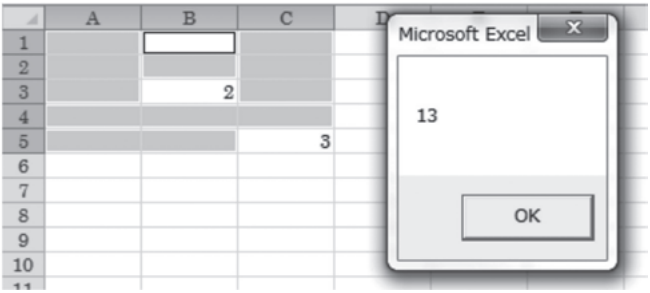
1）「使われた範囲」内の単一の空白セルを指定

2）「使われた範囲」内の単一の数値セルを指定

3）「使われた範囲」外の単一セルを指定

のどのケースにおいても，実行結果は図13のようになり，空白セル数は13個となる。

図 13　Sample13 の実行結果



次に，24 行目をコメントアウトし，23 行目を非コメント化して，SpecialCells（xlCellTypeConstants）について調べてみると，どの位置の単一セルを指定しても，該当するセルは B3 と C5 の 2 つのセルとなる。

先に述べたように，意図して，単一セルに対して SpecialCells メソッドを使用することはないが，対象とするセル範囲を InputBox メソッドで指定するマクロでは，単一セルが指定された場合の対応を考慮しておく必要がある。最も簡単な対応は，単一セルが指定された場合には，再入力を促すことであろう。しかし，ここでは，Excel VBA の特徴をみることも目的なので，単一セルが指定された場合にも SpecialCells メソッドが正しく機能するコードを考えてみた。

SpecialCells メソッドの使用では「使われた範囲」が大きな制約になっているが，これを先に述べた，ワークシートの遠くのセルを LastCell とするコードを付加することで解消できる。また，単一セルが指定された時，SpecialCells メソッドの対象が「使われた範囲」になることは，複数のセル範囲の共有セル範囲（Range オブジェクト）を返す Application.Intersect メソッドを利用することで解消できる。以上の 2 点を考慮して，Sample 13 を拡張したマクロを付図 4 に掲げる。

⑷　SpecialCells メソッドの戻り値（Range オブジェクト）の利用について

本節では，前節（数値の抽出）を前提として，数値セルと空白セルからなるセル範囲から数値セルを取り出し整形することを目的としたマクロ（数値セルの整形）を提示した。そこでは，数値の書き出し部分をサブルーチン化したが（図 8 を参照），こ

れをSpecialCells（xlCellTypeConstants）の戻り値を利用したコードに置き換えたものがS_Format1（図14）である。

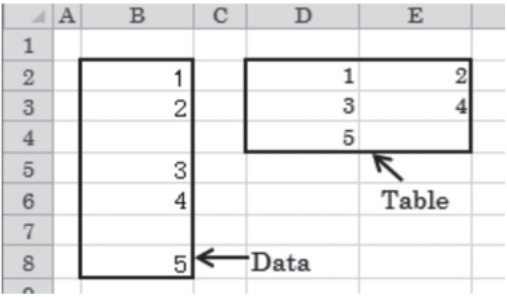
図14　SpecialCellsメソッドを利用した数値の書き出し

```
1   Sub S_Format1(Source As Range, Target As Range)
2   '
3   'SpecialCells(xlCellTypeConstants)の利用
4   '   1列(行) -> 矩形では予想通り
5   '   矩形 -> 1列では予想外
6   '
7     Dim r As Range
8     Dim rl As Range
9     Dim i As Long
10    Dim k As Long
11
12    Set r = Source.SpecialCells(xlCellTypeConstants)
13
14    k = 1
15    For Each rl In r
16        Target(k).Value = rl.Value
17        k = k + 1
18    Next rl
19  End Sub
```

12行目で，SpecialCells(xlCellTypeConstants）により取り出した数値セルへの参照を変数rにセットし，14～18行目でFor Each～Nextステートメントを利用して書き出している。4～5行目のコメントで記述しているようにこのサブルーチンは1列(行)のデータを矩形に整形するケースでは上手くいくが，矩形データを1列に整形するケースでは上手くいかない。

付図3に掲げたコードの83行目をCall S_Format1(Data, Table）に置き換え，1列（行）のデータを矩形に整形した実行例を図15に示す。

図 15　1 列→矩形の実行例



空白セルを含む B2:B8 の範囲を指定し，区切り数を 2，出力先の左上隅を D1 と指定した結果である。正しく，数値のみが取り出されて整形されていることがわかる。

同じマクロで，矩形データを 1 列データに整形を試みた実行例が図 16 である。

図 16　矩形→ 1 列の実行例

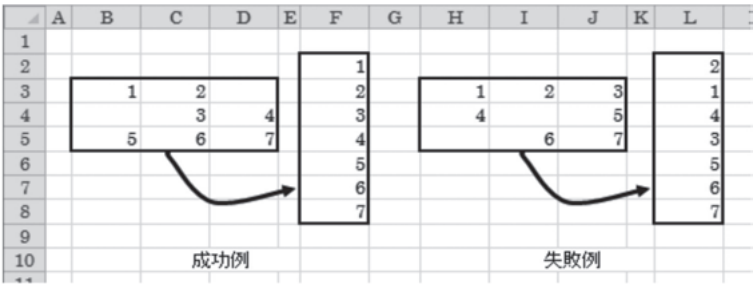


図 16 の左図では，行方向に数値のみが抽出され，1 列に整形されているが，図 16 の右図においては抽出される順序に規則性は見いだせない。先にみたように，二元配列の Range オブジェクトは，行方向の連続の index で順序づけられる一元配列として扱えることの類推から，どんなケースでも，図 16 の左図のように行方向に参照がなされそうであるが，そうではなさそうである。

このことを，より単純な例で確かめたのが，図 17 である。

図 17　2 × 2 のセル範囲における SpecialCells メソッドの戻り値

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1																
2			①			②			③			④			⑤	
3		1	2			2		1			1	2		1	2	
4		3	4		3	4		3	4			4		3		
5																
6		1	B3		2	F3		1	H3		1	K3		2	O3	
7		2	C3		3	E4		3	H4		2	L3		1	N3	
8		3	B4		4	F4		4	I4		4	L4		3	N4	
9		4	C4													
10																
11			⑥			⑦			⑧			⑨			⑩	
12						1			1	2			2		1	
13		3	4				4				3			3		
14																
15		3	B14		1	E13		1	H13		2	L13		1	N13	
16		4	C14		4	F14		2	I13		3	K14		3	N14	
17																
18			⑪													
19			2													
20			4													
21																
22		2	R13													
23		4	R14													

数値セルを含む 2 × 2 のセル範囲に SpecialCells(xlCellTypeConstants) を適用して，その戻り値（Range オブジェクト）の参照先セルの値とセル番地をみたものである。[21] 例えば，セル範囲 B3:C4 に SpecialCells(xlCellTypeConstants) を適用して，その戻り値のセル番地と値が B6:C9 に表示されている。このケース①では，予想通り，行方向に参照が行われている。図 17 に示したように試みた 11 ケースの内で，唯一，ケース⑤が予想外の結果となった。3 × 3 のセル領域について同様の検証を行ってみると，セル領域全てが数値であるケースからセル領域の 1 つのセルのみに数値がある場合まで，511 ケース（$=\sum_{i=1}^{9} {}_9C_i$）で，SpecialCells メソッドを適用し，その戻り値を調べたところ，167 ケース (32.7%) で，規則的な参照が行われていなかった。[22] さらに，4 × 4 のセル領域では，65,535 ケース（$=\sum_{i=1}^{16} {}_{16}C_i$）中，41,459 ケース（63.3%）に

(21)　2 × 2 のセル領域の 1 つのセルに数値があるケース（4 通り）もあるが，ここでは省略している。

(22)　検証のため使用したマクロを付図 5 に掲げる。

おいて，規則的な参照が行われていなかった。

今回のまとめ

前稿，大野拓行(2017a)で作成したマクロの雛形を拡張して，データ分析へのExcel VBAの活用を考察した。先ず，IsNumeric関数とIsNumber関数の特性を調べ，数値データの抽出や整形を行うマクロの作成を通して，Excel VBAの特徴をみた。

その考察の過程において，SpecialCellsメソッドの働きに興味をいだき，調べた結果，「使われたセル範囲」が重要であることが分かった。

また，マクロを作成していく過程で，SpecialCellsメソッドの戻り値（Rangeオブジェクト）の参照順序に疑問が生じて考察したが，規則的な参照順序を見出すことはできなかった。

本稿においては，SpecialCellsメソッドに深入りした感がある。次稿では，もう少し，利用頻度が高いマクロの作成を通して，Excel VBAの特徴をみていきたい。

参 考 文 献

大野拓行（2017a）「データ分析へのExcel VBAの活用」『香川大学経済論叢』第90巻第1号 173-207頁
大村あつし（2011）『Excel 2010 VBA基礎編』技術評論社
大村あつし（2015）『Excel VBA本格入門』技術評論社
門脇加奈子（2016）『かんたんExcelマクロ＆VBA』技術評論社
田中　亨（2016）『Excel VBA逆引き辞典パーフェクト　第3版』翔泳社
土屋和人（2016）『Excel VBAパーフェクトマスター』秀和システム

参考ウェブサイト

インストラクターのネタ帳　http://www.relief.jp/itnote/
エクセルの神髄　http://excel-ubara.com/
Office TANAKA　http://officetanaka.net/
Moug モーグ　http://www.moug.net/
MSDN（Microsoft Developer Network）　https://msdn.microsoft.com/ja-jp/

付図 1　IsNumeric 関数，IsNumber 関数の特性

```
Sub Sample11()
    Dim i As Integer

    Range("B2:F15").Clear: '表示範囲クリア
    For i = 2 To 15
        With Cells(i, 1)
            Cells(i, 2).Value = TypeName(.Value)
            If IsNumeric(.Value) Then
                Cells(i, 3).Value = CDbl(.Value)
            End If
            If IsNumeric(.Text) Then
                Cells(i, 4).Value = CDbl(.Text)
            End If
            If WorksheetFunction.IsNumber(.Value) Then
                Cells(i, 5).Value = CDbl(.Value)
            End If
            If (VarType(.Value) >= 2) And (VarType(.Value) <= 5) Then
                Cells(i, 6).Value = CDbl(.Value)
            End If
        End With
    Next i

End Sub
```

	A	B	C	D	E	F
1	対象	TapeName	.Value	.Text	IsNumber	VarType
2	2	Double	2	2	2	2
3	'1	String	1	1		
4	'10．5	String	10.5	10.5		
5	17	Double	17	17	17	17
6	¥100	Currency	100	100		
7		Empty	0			
8	10,5	String	105	105		
9	10，5	String	105	105		
10	香川	String				
11	－	String				
12	"10.5"	String				
13	FALSE	Boolean	0			
14	TRUE	Boolean	-1			
15	2017/3/20	Date				

付図２　マクロ（数値の抽出）

```
Sub 数値の抽出()
'
'指定するセル領域から数値を抽出します
'　・対象が数式の結果が数値になっているセル，通貨型データ，に加え，
'　　文字列として入力した数字も数値とします。
'　・空白セルを数値（０）とするかを選択できます。
'　・カンマ付き数字を数値とするかを選択できます。
'
'                              2017.4　大野

  Dim Data As Range:'入力データ
  Dim Table As Range:'出力領域

  Dim i As Long
  Dim k As Long
  Dim c1 As Boolean
  Dim c2 As Boolean
  Dim r As Range
  Dim r1 As Range

  Dim myTitle As String
  Dim myMsg As String
  Dim myBtn As Integer

  Const myColor As Integer = 34:'薄水色
  myTitle = "数値データの抽出"

'入力データへの参照の取得
  myMsg = "データ範囲を選択して下さい"
  Set Data = F_Input1(myMsg, myTitle)
  If Data Is Nothing Then Exit Sub

'出力領域の準備
  Set Table = _
  F_Table(myTitle, Data.Rows.Count, Data.Columns.Count)
  If Table Is Nothing Then Exit Sub

  Table.Value = Data.Value
  Table.Worksheet.Select
  Table(1, 1).Select

'空白セルの処理
  k = 0
  For i = 1 To Table.Count
      If Table(i).Value = "" Then
          Table(i).Interior.ColorIndex = myColor
```

```
47              k = k + 1
48          End If
49      Next i
50      If k > 0 Then
51          myMsg = "空白セルを数値（0）としますか？"
52          myBtn = MsgBox(myMsg, vbYesNo, myTitle)
53          Table.SpecialCells(xlCellTypeBlanks).Interior.ColorIndex = 0
54          If myBtn = vbYes Then
55              Table.SpecialCells(xlCellTypeBlanks).Value = 0
56          End If
57      End If
58
59  'カンマ付き数字の処理
60      k = 0
61      For i = 1 To Table.Count
62          c1 = IsNumeric(Table(i).Text)
63          c2 = WorksheetFunction.IsNumber(Table(i).Value)
64          If Not c1 Then Table(i).Value = ""
65          If c1 And (Not c2) Then
66              If TypeName(Table(i).Value) = "Currency" Then GoTo Label1
67              Table(i).Interior.ColorIndex = myColor
68              k = k + 1
69              If k = 1 Then
70                  Set r = Table(i)
71              Else
72                  Set r = Union(r, Table(i))
73              End If
74          End If
75  Label1:
76      Next i
77      If k > 0 Then
78          myMsg = "これらのセルを数値としますか？"
79          myBtn = MsgBox(myMsg, vbYesNo, myTitle)
80          If myBtn = vbNo Then Exit Sub
81          If myBtn = vbYes Then
82              r.Interior.ColorIndex = 0
83              For Each r1 In r
84                  r1.Value = CDbl(r1.Value)
85              Next r1
86          End If
87      End If
88
89  End Sub
```

付図３　マクロ（数値セルの整形）

```
Sub 数値セルの整形()
'
'数値セルの整形
'　（空白セルを除外します）
'　　　　2017.4　大野
'
'１列（１行）データ　=>　矩形　:横方向に配置していきます
'矩形　=>　縦１列　:横方向に取っていきます
'
  Dim Data As Range:'対象セル参照
  Dim Table As Range:'出力領域セル参照

  Dim n As Long:'Data のセル数
  Dim c As Long, r As Long:'Data の列数（c），行数（r）
  Dim n1 As Long:'数値セルの数
  Dim myCut As Long:'１列データ　=>　矩形での区切り数
  Dim iEnd As Long:'整形後の矩形の行数

  Dim myTitle As String
  Dim myMsg As String
  Dim myBtn As Integer

  Dim i As Long, j As Long, k As Long
  Dim v As Variant

  On Error GoTo myError
  myTitle = "数値セルの整形"

  '入力データへの参照の取得
  myMsg = "対象データの範囲を選択して下さい"
  Set Data = F_Input1(myMsg, myTitle)
  If Data Is Nothing Then Exit Sub

  Data.Select

  '数値データ存在のチェック
  On Error Resume Next
  n1 = Data.SpecialCells(xlConstants, xlNumbers).Count
  If Err.Number > 0 Then
      MsgBox "数値セルがありません"
      Exit Sub
  End If
  On Error GoTo 0

  'Data のセル数，列数，行数の取得
  n = Data.Count
```

```
47    c = Data.Columns.Count
48    r = Data.Rows.Count
49
50    MsgBox "選択範囲のセル数は" & n & "です。" & vbCrLf &_
51        "その内，数値データ数は" & n1 & "個です"
52
53    If c = 1 Or r = 1 Then
54  Label1:
55        '区切り数の入力
56        myMsg = "区切り数を指定して下さい(Cancel->Exit)"
57        v = Application.InputBox(myMsg, myTitle, Type:=1)
58        If TypeName(v) = "Boolean" Then Exit Sub
59        myCut = v
60        If myCut > n1 Or myCut <= 0 Then
61            MsgBox "数値が正しくありません"
62            GoTo Label1
63        End If
64
65        '整形後の矩形の行数の計算
66        iEnd = n1 ¥ myCut
67        If n1 Mod myCut > 0 Then iEnd = iEnd + 1
68    End If
69
70    '整形後の矩形の出力先の準備
71    myMsg = "新規シートに出力しますか？"
72    myBtn = MsgBox(myMsg, vbYesNo, myTitle)
73    If myBtn = vbYes Then Worksheets.Add
74
75    If c = 1 Or r = 1 Then
76        Set Table = F_Table(myTitle, iEnd, myCut)
77    Else
78        Set Table = F_Table(myTitle, n1, 1)
79    End If
80    If Table Is Nothing Then Exit Sub
81
82    '数値データの書き出し
83    Call S_Format(Data, Table)
84
85    Exit Sub
86
87  myError:
88    MsgBox "エラー番号:" & Err.Number & vbCrLf &_
89        "エラーの種類:" & Err.Description, vbExclamation
90
91  End Sub
```

```
1  Sub S_Format(Source As Range, Target As Range)
2  '
3  '矩形データを１列データへの整形(Source->Target)
4  '
5      Dim i As Long, k As Long
6
7      k = 1
8      For i = 1 To Source.Count
9          If Source(i).Value <> "" Then
10             Target(k).Value = Source(i).Value
11             k = k + 1
12         End If
13     Next i
14 End Sub
```

付図 4　SpecialCells メソッドの特性

```
Sub Sample14()
'
'SpecialCells の特性
'(注意) 新規ワークシートが挿入されます
'      終了はダイアログボックスで［Cancel］
'
  Dim r As Range
  Dim r1 As Range

  Dim myTitle As String
  Dim myMsg As String

  Worksheets.Add

  Range("B3").Value = 2
  Range("C5").Value = 3

  'LastCell の設定
  Cells(1000, 1000).Value = "LastCell"

  'セル範囲の指定
Label1:
  myTitle = "SpecialCells の特性"
  myMsg = "セル範囲を指定して下さい"
  Set r = F_Input1(myMsg, myTitle)
  If r Is Nothing Then Exit Sub

'     Set r1 = r.SpecialCells(xlCellTypeConstants)
  Set r1 = r.SpecialCells(xlCellTypeBlanks)

  'Intersect メソッドの利用
  Set r = Intersect(r, r1)

  r.Select
  MsgBox r.Count
  GoTo Label1

End Sub
```

付図５　SpecialCells メソッドの戻り値の検証（３×３）

```
Sub Sample15()
'
'   A1 セルを基準として，n＊n のセル領域に数値をセットし，
'   SpecialCells メソッドを適用した戻り値の参照の仕方を
'   調べるためのマクロ
'
'                                   2017.5 大野
'n = 4 で相当の時間を要します
'出力先は A1 セルを基準として，n＊n のセル領域より右を指定


    Dim r2 As Range: 'Data 領域
    Dim r3 As Range: 'SpecialCells メソッドの適用
    Dim r4 As Range: 'For Each～Next で使用
    Dim r5 As Range: '出力領域

    Dim s As String: 'Function Combination からの戻り値
    Dim r As Variant: '戻り値を Split し，r(0)～に格納
    Dim r1 As Variant: 'ケース毎に数字を Split し，同上
    Dim i As Long, j As Long
    Dim n As Long: 'n＊n の矩形を調べる
    Dim n1 As Long: '出力領域の最大の行数

    Dim l As Long: '数値セル数（１～ n＊n）
    Dim myMsg As String

    Dim k1 As Long: '並びが昇順でないケースの数
    Dim k2 As Long
    Dim k3 As Long: 'ケース総数
    Dim myBtn As Long: '並びが昇順かの判定

    Cells.Clear

    n = 3
    'Data 領域の確保
    Set r2 = Cells(1, 1).Resize(n, n)
    '出力領域の最大の行数の計算
    n1 = WorksheetFunction.Combin(n * n, Int(n * n / 2))
    '出力領域の確保
    Set r5 = F_Table("組み合わせ", n1, n * n)

    k1 = 0
    k3 = 0
    For l = 1 To n * n
        s = Combination(n * n, l): 'Function Combination の呼び出し

```

```
47          r = Split(s, vbCrLf):'改行コードで Split し，結果を Variant 変数へ
48
49          For i = 0 To UBound(r) − 1
50              r2.Clear:'Data 領域をクリア
51              r1 = Split(r(i), ""):'ケース毎に数字を Split
52              For j = 1 To UBound(r1)
53                  r2(r1(j)) = r1(j):'Data 領域に数字をセット
54              Next j
55
56              'SpecialCells メソッドの適用
57              Set r3 = r2.SpecialCells(xlCellTypeConstants)
58
59              myMsg = ""
60              k2 = 1
61              myBtn = 0:'順番が昇順
62          'SpecialCells メソッドの適用した参照の順番を調べ,
63          '昇順ではない場合は，myBtn = 1 とする。
64              For Each r4 In r3
65                  myMsg = myMsg & r4.Text & ""
66                  If r4.Value >= k2 Then
67                      k2 = r4.Value
68                  Else
69                      myBtn = 1
70                  End If
71              Next r4
72      '結果の書き出し，昇順でないケースは色を付ける
73              r5(i + 1, 1).Value = myMsg
74              k3 = k3 + 1
75              If myBtn = 1 Then
76                  k1 = k1 + 1
77                  r5(i + 1, 1).Interior.ColorIndex = 42
78              End If
79          Next i
80
81          Next l
82
83      r5.EntireColumn.AutoFit
84
85      Range("A6").Value = "ケース総数": Range("C6").Value = k3
86      Range("A7").Value = "昇順でないケース数": Range("C7").Value = k1
87
88  End Sub

1   Function Combination(Max As Long, Count As Long, Optional Min As Long = 1, _
2           Optional Preceding As String = "") As String
3   '
4   'このプロシージャは下記のウェブサイトからの借用です。
```

```
'なお，コメントなど，筆者が，一部改変しました。
'
'「組み合わせ計算アルゴリズムについて教えて下さい」
'https://detail.chiebukuro.yahoo.co.jp/qa/question_detail/q1120855461
'                                        閲覧日：2017.5.2
'
'使用例：Combination(4, 2)
'        1～4までの数字から2つを取る組み合わせを，改行コードを区切りとした，
'        文字列として返す。
'

  Dim i As Long, Buffer As String

  If Max - Min + 1 < Count Then
      Combination = "Error":'不可能なケース（4つの数字から6つ取るなど）の処理
  ElseIf Count = 0 Then
      'Count = 0で一つできあがり
      Combination = Preceding & vbCrLf
  ElseIf Max - Min + 1 = Count Then
      'ここは最後のケース
      Buffer = Preceding

      For i = Min To Max
          Buffer = Buffer & "" & i
      Next i
      Combination = Buffer & vbCrLf
  Else
      'ここがポイント(再帰処理)
      Buffer = ""
      For i = Min To Max - (Count - 1)
          Buffer = Buffer & Combination(Max, Count - 1, i + 1, Preceding & "" & i)
      Next i
      Combination = Buffer
  End If

End Function

Sub 使用例()

  MsgBox Combination(4, 2)

End Sub
```