Visual Basic – Macro recording

Introduction

Before you start working with Excel macros, you will need to do two things:

1. *Display the Visual Basic toolbar.* To do this, right-click on any of the other Excel toolbars, then select the checkbox next to the "Visual Basic" entry. By default, the Visual Basic toolbar will appear as a 'floating toolbar' in the main body of the Excel window. If you prefer, you can drag it to the top or the left-hand edge of the window so it becomes 'attached' there like the other commonly displayed toolbars.
2. *Reduce your security level.* Because of the prevalence of macro viruses, the default security settings for Excel prevent you from running any macros into the program along with a workbook. (Even with the default settings, you can still run macros you *created* in a workbook during the same Excel session). To change this setting, select Security on the VB toolbar (or select Tools|Options and choose the Security tab and click on the Macro Security button in the lower right corner), and then choose the Medium security level. This level will still prompt you before running any macros that do not come from a 'trusted source' (generally a software developer who possesses a digital certificate guaranteeing authenticity), but will allow you to run such macros if you so choose. You are recommended *not* to agree to let Excel run macros unless you know you wrote them yourself, or they came from a source that you trust.

Note that macros and user-defined functions from each open workbook will be available for use in all open workbooks.

Recording a simple macro

The most straightforward way to construct a simple macro is to "record" it from normal Excel keystrokes. In this exercise, you will record, and then look at the code in a simple macro.

Your macro will sum each row of two columns into a third column using formulae, and copy the values of the sums into a fourth column. To prepare for your macro, create a new workbook and enter ten numbers into cells A2:A11 and a different set into B2:B11. Label the columns with "A" and "B" in cells A1 and B1 respectively. Now you are ready to record your macro.

1. From the menu bar select Tools/Macro/Record New Macro. In the dialogue box enter Macro Name = SumToValues; for the shortcut key enter "g", then click OK. The macro recording button (the circle in the Visual Basic toolbar) changes to a square and an extra floating toolbar with a "Stop" button appears.
2. Select cell C1 on your spreadsheet and enter the text "Sum of A+B".
3. In cell C2 enter "=A2 + B2"
4. Select C2.
5. Use Ctrl+C to copy contents of C2 to the clipboard.

6. Select cells C3 to C11 and use Ctrl+V to paste a copy of the formula on the clipboard to cells C3:C11. (The same formula now exists in C2:C11).
7. Select cells C2:C11 and use Ctrl+C to copy this collection to the clipboard.
8. Move cursor right to cell D2 and click the right-button of the mouse and select "Paste Special". In the dialogue box tick the "Values" button. Press OK.
9. Press ESC to get out of copy mode.
10. Select cell D1 and enter "Values".
11. Press Stop on the floating macro record button.

You can see the code by using Alt+F11 which invokes the Visual Basic Editor. Alternatively, select the Visual Basic Editor on the VB Toolbar. If successful your macro should look similar to the one below, but without the comments, indicated by a line starting with an apostrophe, after the instructions.

```
Sub SumToValues()
'
' SumToValues Macro
' Macro recorded by …
'
' Keyboard Shortcut: Ctrl+g
'
' Select cell C1
    Range("C1").Select
' Enter header text
    ActiveCell.FormulaR1C1 = "Sum of A+B"
' Select cell C2
    Range("C2").Select
' Enter formula to sum cells A2 and B2. Notice that the
' row-column syntax has been used by Excel. The cells to
' add are give an address relative to the cell taking the
' formula, i.e. cell A2 is 0 rows more than C2 and -2
' columns after it.
    ActiveCell.FormulaR1C1 = "=RC[-2]+RC[-1]"
' select C2 again
    Range("C2").Select
' Copy selection (cell C2) to Clipboard
    Selection.Copy
' Select range of cells to get copy of formula
    Range("C3:C11").Select
' Paste formula into this range of cells
    ActiveSheet.Paste
' Quit cut/copy mode (remove moving border)
    Application.CutCopyMode = False
' Select cell C2 again and cells down to C11
    Range("C2:C11").Select
' Copy this selection to clipboard
    Selection.Copy
```

```
' Select cell D2 to paste data
    Range("D2").Select
' Paste block of data copied to clipboard
    Selection.PasteSpecial Paste:=xlPasteValues, _
                Operation:=xlNone, SkipBlanks:=False, _
                                        Transpose:=False
' Quit cut/copy mode
    Application.CutCopyMode = False
' Select cell D1
Range("D1").Select
' Enter text "Values"
    ActiveCell.FormulaR1C1 = "Values"
    Range("D2").Select
'
End Sub
```

Now test your new macro. Delete the contents pf cells C1:D11 then press the Run button on the VB toolbar and choose SumToValues from the list of available macros. You should find that the contents of your cells C1:D11 reappear.
You might think it's rather tedious to go to the VB toolbar each time you want to perform a simple action like this. Since you associated a shortcut with the macro when it was created you can run it by simply typing the shortcut. In this example type Ctrl+g. (If you didn't associate a shortcut when the macro was created one can be added at a later time. Select Run from the VB toolbar, then select Options. You will see a box that allows you to associate a shortcut with the macro; click in this box and type a letter, say g. The key sequence Ctrl+g is now associated with the SumToValues operation. Avoid choosing a letter which Excel has already chosen for its own in-built shortcuts).

VBA

VBA is an object-orientated programming language. It concerns itself with "Objects", such as `Range("C1")` and "Methods", such as Select or Copy, and "Properties". These items will be discussed briefly later in the course. There is much more about them in the third-year course on Object-Oriented Programming.

The VB editor

When you run you macro you could chose "Step Into" from the "Run" dialogue box. The use of the F8 key will step the macro through each instruction in turn. If you arrange your windows so that both the VB Editor and Excel spreadsheet are visible side-by-side, you will see the actions of the macro as they are applied in the spreadsheet. This can be very useful in debugging a macro. There are, as you might expect, many other useful features available in the VB Editor but we don't have time to discuss them in this course. You can, of course, investigate them yourself from the Debug entry in the VB Editor.

Function macros

The above example records a macro as a sequence of keyboard keystrokes. The VB Editor can be used to edit the macro that has been recorded. However sometimes one creates a macro directly by typing the instructions into a module in the VB Editor. This is the case if you create a user-defined function. These are like mathematical functions, e.g. sin(x). Functions cannot be recorded; they have to be created directly, even if parts of them may contain code created by using the macro recorder facility.