A quick guide to macros

Ed Wilson, University of East Anglia
March 2010

Naming cells

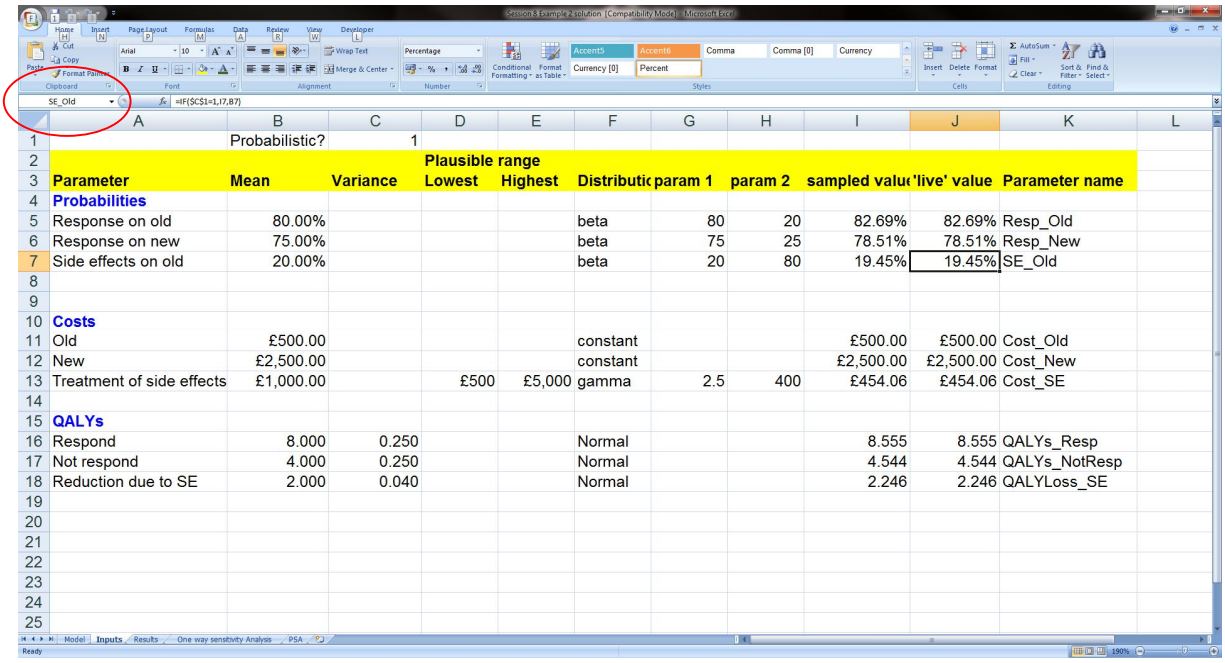
Before getting on to macros, firstly it's a good idea to work with named cells, rather than referring to them by their row and column number as it makes error checking a lot simpler. For example, entering a formula as:

= *QALYs_Resp - QALYLoss_SE*

is a lot more meaningful than

= *Inputs!J16 - Inputs!J18*

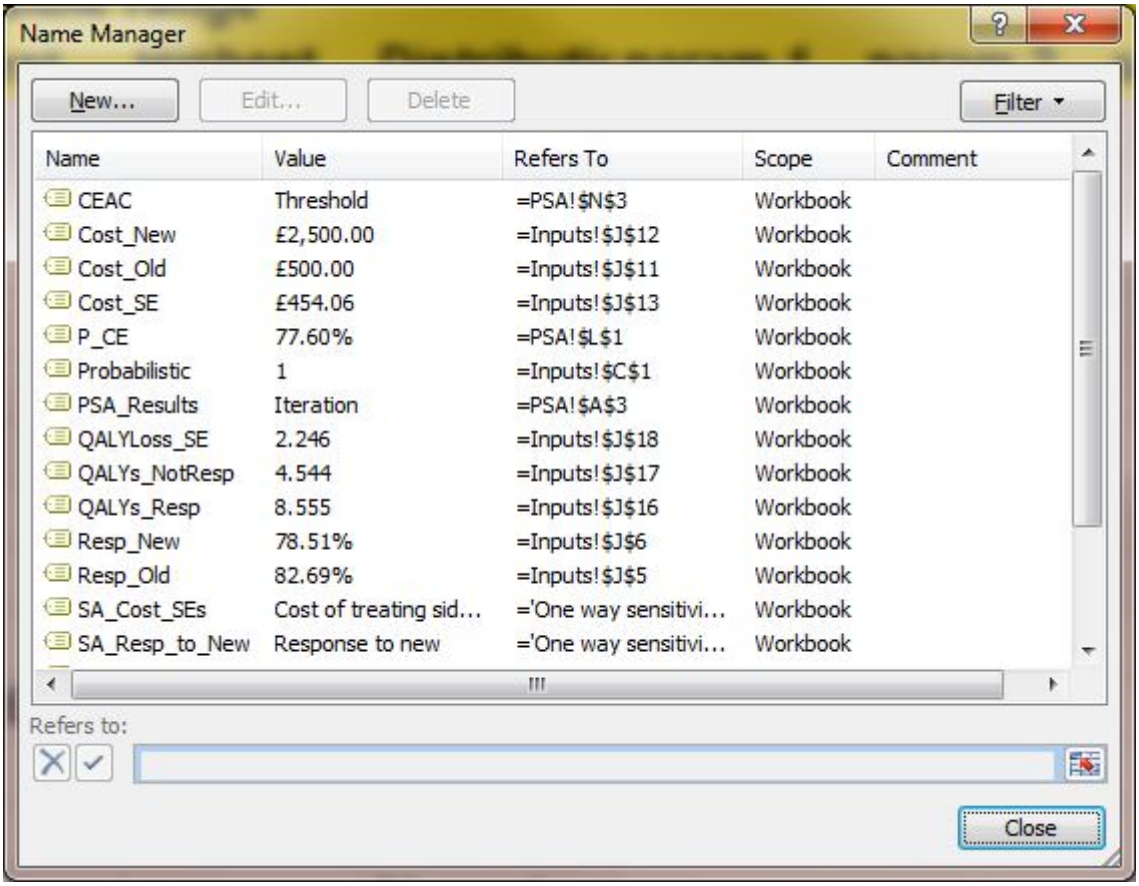
To name a cell, make sure it is selected, and then click on the box to the top left of the screen, just to the left of the formula bar. Enter a meaningful name for the cell and hit enter:



	A	B	C	D	E	F	G	H	I	J	K	L
1		Probabilistic?	1									
2				Plausible range								
3	**Parameter**	**Mean**	**Variance**	**Lowest**	**Highest**	**Distributic**	**param 1**	**param 2**	**sampled valu**	**'live' value**	**Parameter name**	
4	**Probabilities**											
5	Response on old	80.00%				beta	80	20	82.69%	82.69%	Resp_Old	
6	Response on new	75.00%				beta	75	25	78.51%	78.51%	Resp_New	
7	Side effects on old	20.00%				beta	20	80	19.45%	19.45%	SE_Old	
8												
9												
10	**Costs**											
11	Old	£500.00				constant			£500.00	£500.00	Cost_Old	
12	New	£2,500.00				constant			£2,500.00	£2,500.00	Cost_New	
13	Treatment of side effects	£1,000.00		£500	£5,000	gamma	2.5	400	£454.06	£454.06	Cost_SE	
14												
15	**QALYs**											
16	Respond	8.000	0.250			Normal			8.555	8.555	QALYs_Resp	
17	Not respond	4.000	0.250			Normal			4.544	4.544	QALYs_NotResp	
18	Reduction due to SE	2.000	0.040			Normal			2.246	2.246	QALYLoss_SE	
19												
20												
21												
22												
23												
24												
25												

Note there are restrictions on what you can call cells. E.g. you are not allowed spaces or certain special characters. Use an underscore character instead of a space if you wish.

To see all the cells you've named, and to edit, delete or create new ones, go to Formulas -> Name manager:



Setting up a Macro

We're going to write a macro that simply changes the value of one cell, and copies the results of a calculation to another cell.

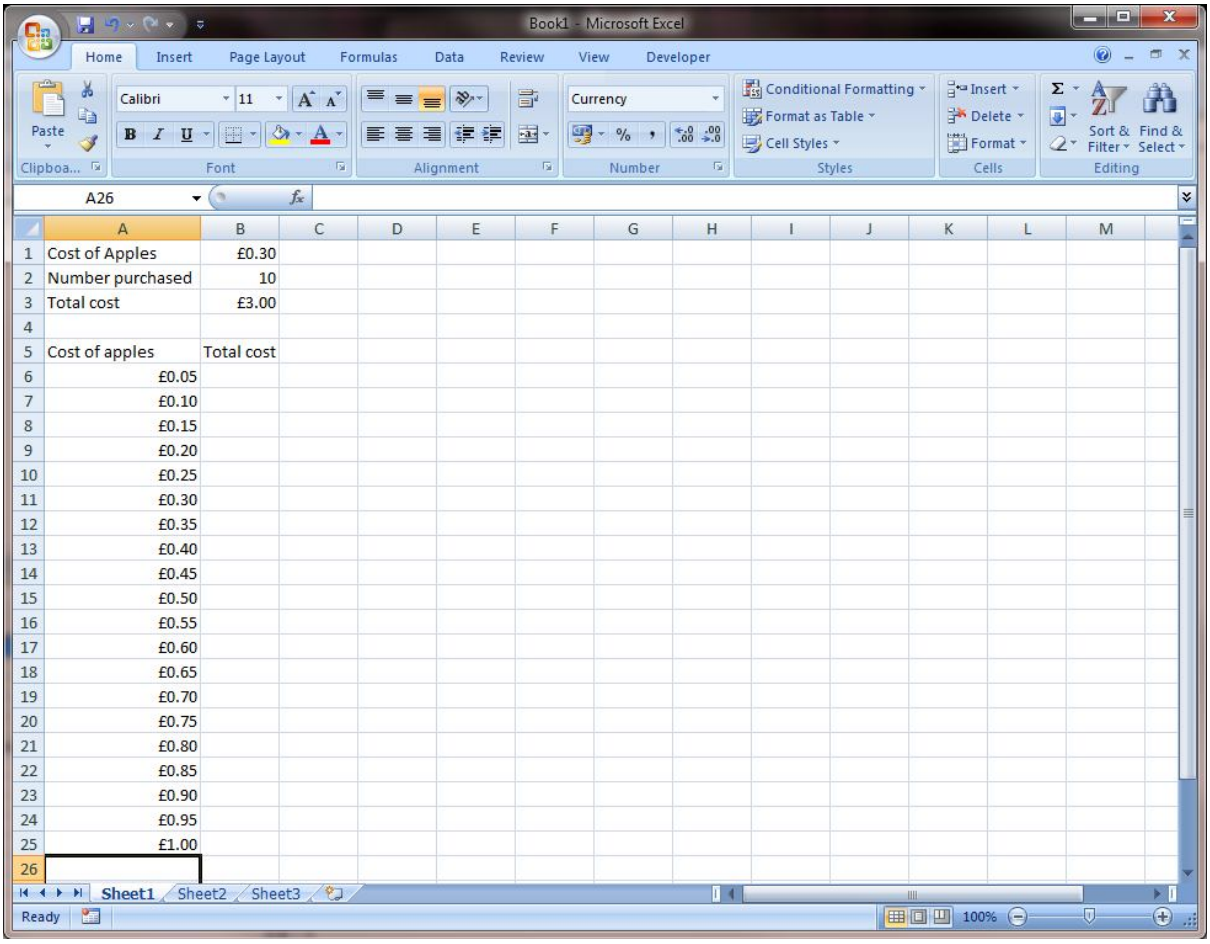
Make sure you have no workbooks open in excel, and open a new one.

* In cell A1 type 'Cost of apples'
* In cell A2 type 'Number purchased'
* In cell A3 type 'Total cost'
* In cell B1 type 10
* In cell B2 type £0.30
* In cell B3 type '=B1*B2' (without the inverted commas)

Make sure the result in cell B3 is correct (i.e. £3!)

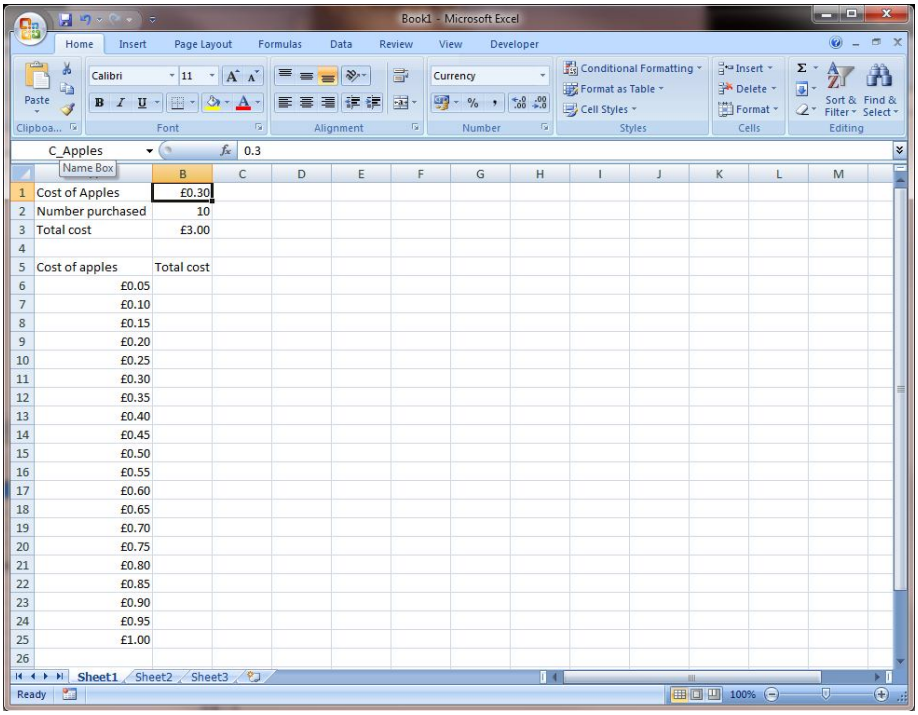
* In cell A5 type 'Cost of Apples'
* In cell B5 type 'Total cost'
* In cell A6 type '£0.05'
* In cell A7 type '£0.10'
* Enter '£0.15', '£0.20' etc in cells A8 to A25 until you get to £1.00

You should have something that looks like this:



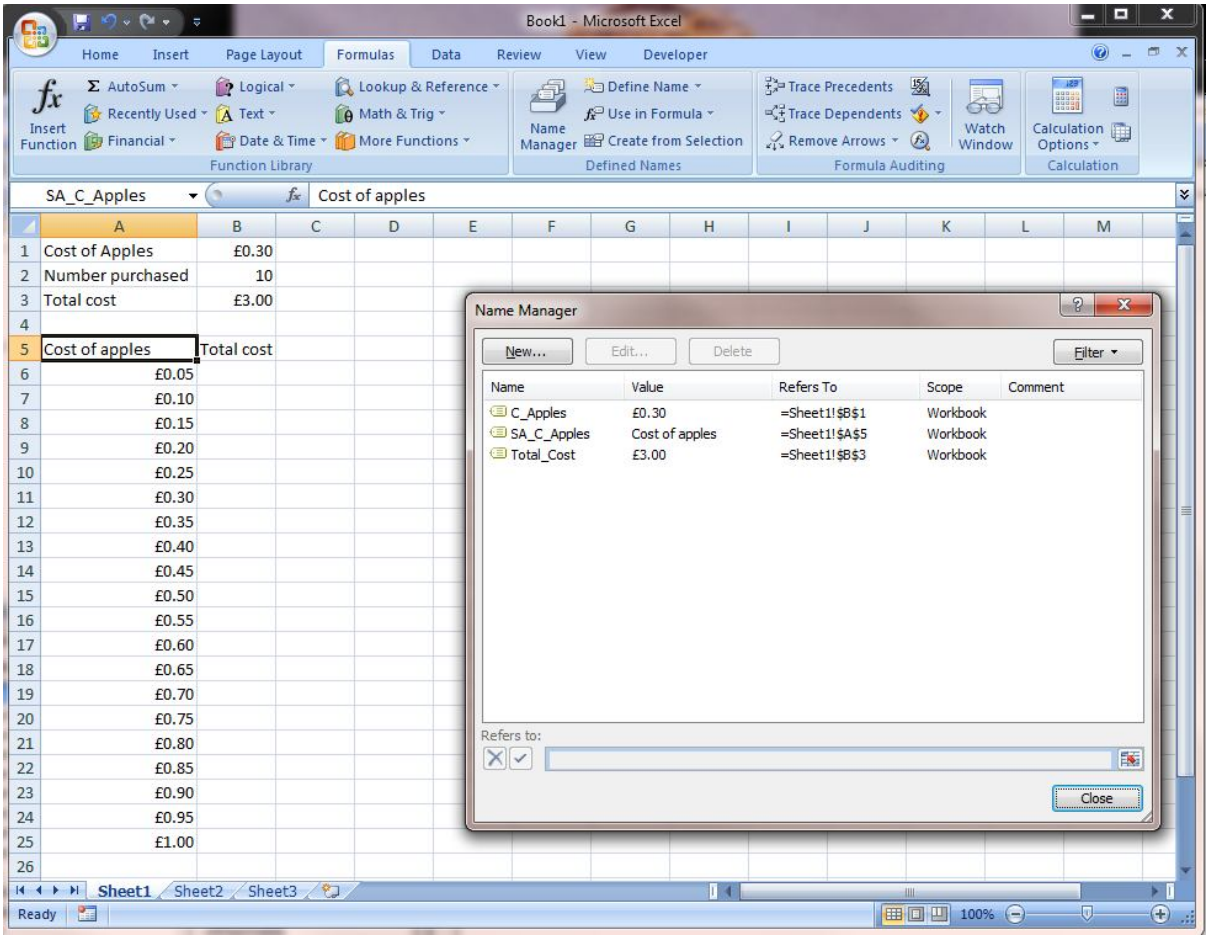
We're now going to name cells B1, B3 and A5 to make them a bit more memorable.

* Click on cell B1.
* Click on the name box and enter 'C_Apples'
* hit enter:



* Now name cell B3 'Total_Cost' and cell A5 'SA_C_Apples' (this stands for 'sensitivity analysis on the cost of apples').

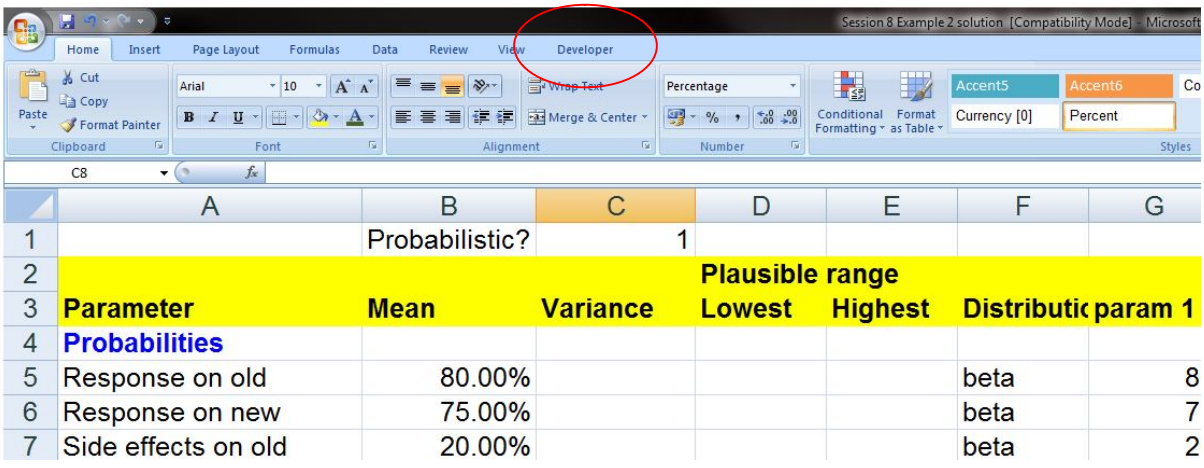
* Check the names on the name manager (Formulas -> Name manager):



* When you're sure you've set it up correctly, close the name manager.

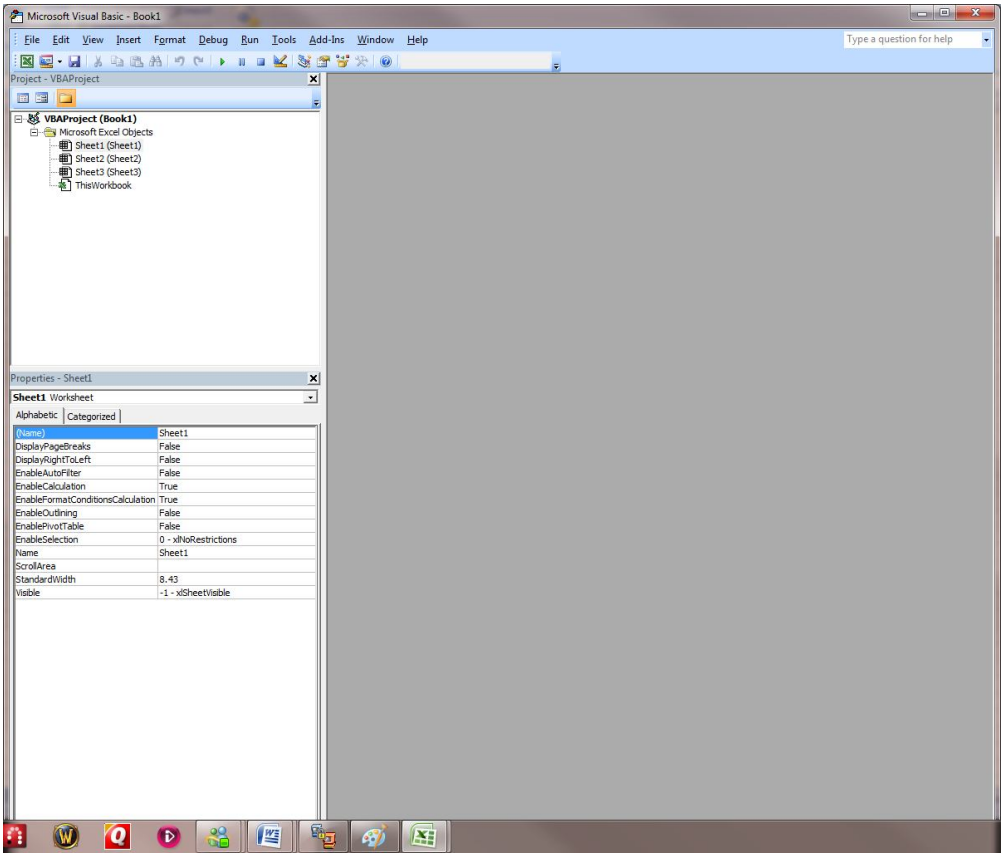
Opening the VB interface

Macros are written in Visual Basic. To open the Visual Basic editing window, first check to see if you have the 'developer' tab available:



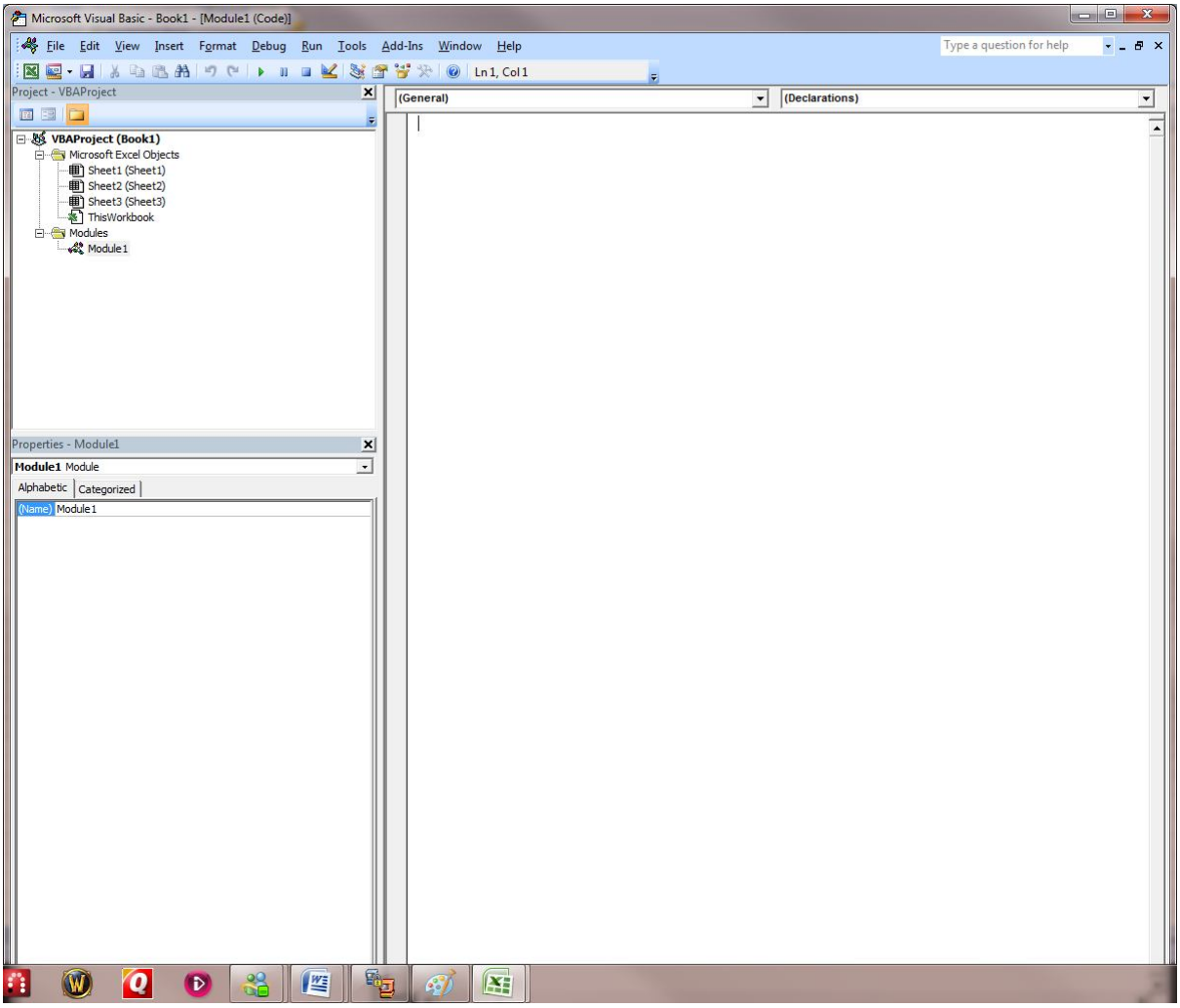
If not, do the following:
* click on the Office button (top left with the windows logo)
* click 'Excel options'
* on the 'popular' tab, check the box next to 'show developer tab in the ribbon'.
* click ok

You will now be able to see the developer tab. Click on it and click 'visual basic'. The window will open and look something like this:



Adding a module

This is not really necessary, but good practice. Click on Insert -> Module. Notice the tree diagram on the upper left side of the screen. This shows the objects within your excel workbook, one for each worksheet, another one called 'ThisWorkbook', and a section called 'Modules' with your new module added (called Module1). The module will have opened and you can start to type code in the text area:



Writing a macro

We are going to write a macro works out how total cost changes with the price of apples.

Make sure the main text area is selected (the caret should be flashing in it, just below where it says '(General)'.

All macros have to be written as 'subroutines', which have to be named.

Type:

sub SA_Cost_Apples

and hit enter.

Notice a pair of brackets and the word 'end sub' have appeared, and the lowercase 's' has changed to uppercase.

Now we want to tell excel to take each of the values we entered for the cost of apples in cells A6:A25, and put them into cell B1 one at a time. We then want it to record the result in cells B6:B25.

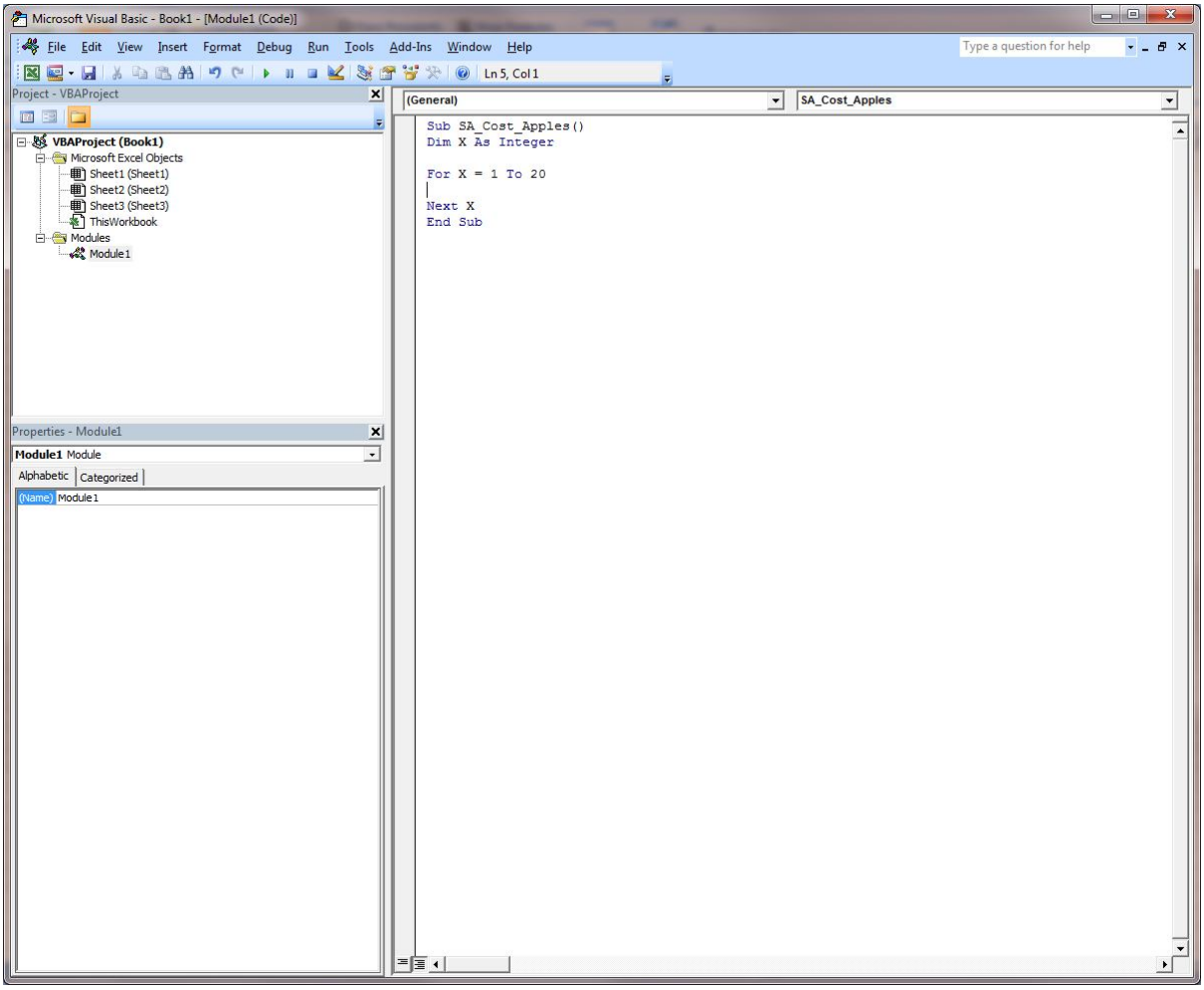
First, we'll need a count variable so excel can keep track of where it is.

Type:

Dim X as integer
For X = 1 to 20
next x

Notice it changes the 'x' in 'next x' to uppercase (unless you used a lowercase X in the 'dim' statement).

Your code should look like this:



(note you can add a few extra lines to space things out a bit more clearly if you want).

Make sure you type the next bit of code in between the 'For X = 1 to 20' and 'Next X' lines. We want excel to put the value of cell A6 into B1. This is where the named cells come in useful as we can use the names rather than the row and column numbers. We will also use the 'offset' command. Type:

range("C_apples").value = range("SA_C_Apples").offset(X,0)

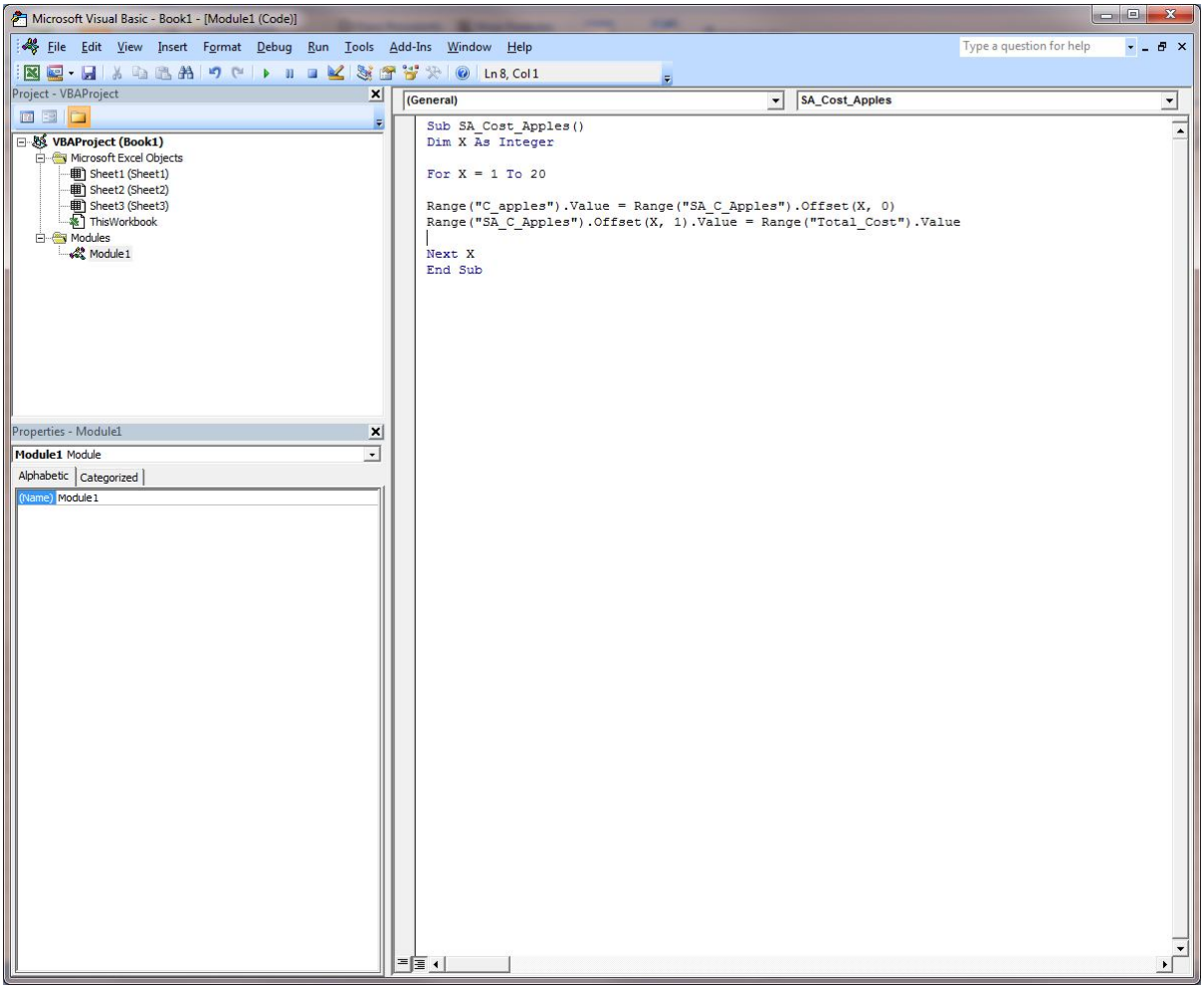
This tells excel to set the value of the cell called 'C_apples' to the value of another cell. That cell is the cell called 'SA_C_Apples' plus X rows and zero columns. On the first loop, X takes the value 1, so it's the cell 'SA_C_Apples' plus 1 row and zero columns, which is cell A6.

Now enter:

range("SA_C_Apples").offset(x,1).value = range("Total_Cost").value

This tells excel to set the value of cell SA_C_apples plus 1 row and 1 column to the value of the cell called 'Total_Cost'.

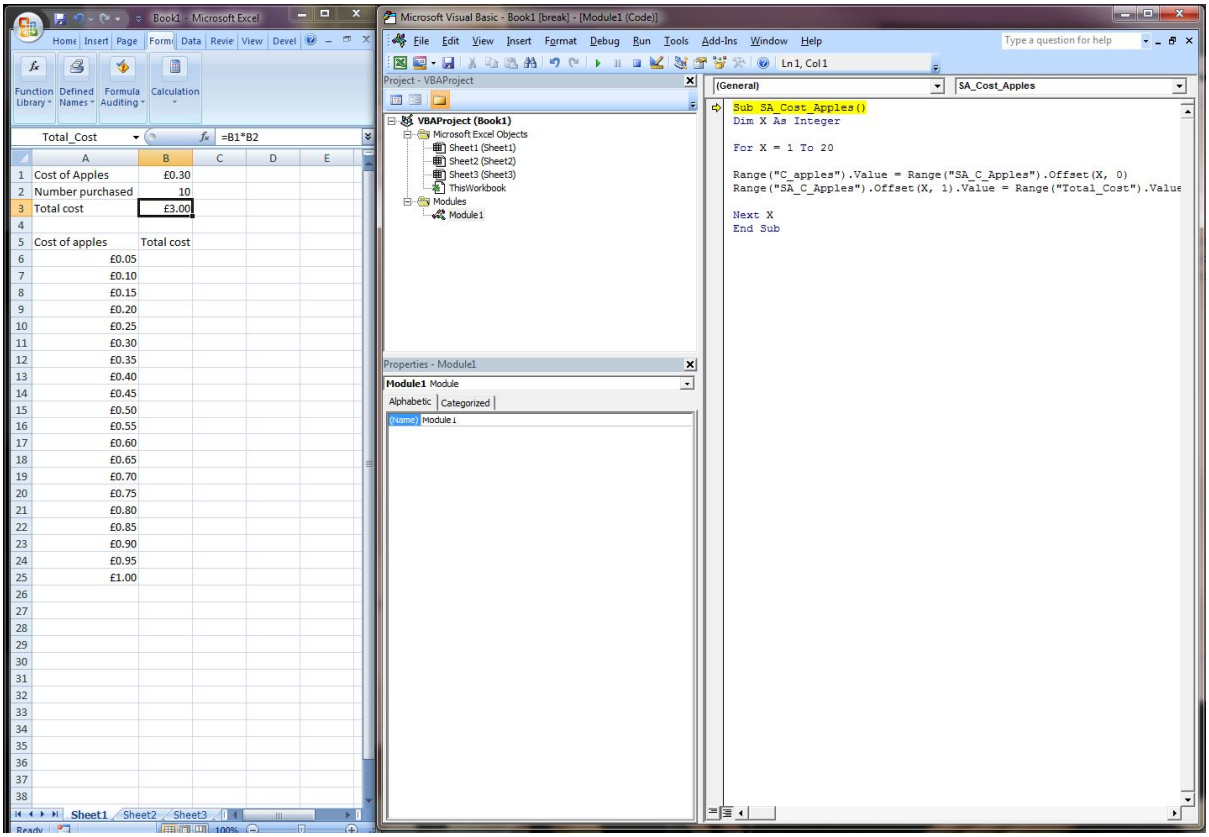
Your code should look like this:



The macro is now complete. All you need to do is run it.

There are two ways of doing this: pressing F8 will step through the macro one line at a time. This is very useful for debugging as you can see what happens with each line of code. Hitting F5 will run the entire macro.

Before running this, try and arrange your windows so you can see both the code and the excel spreadsheet at the same time like this:



Make sure the caret is flashing within the subroutine (i.e. not after the 'End Sub' line or before the 'Sub SA_Cost_Apples' line), and hit F8. The first line will be highlighted in yellow as above. Hit F8 repeatedly and watch what is happening on the spreadsheet. Once you get the picture, hit F5 to complete the run.

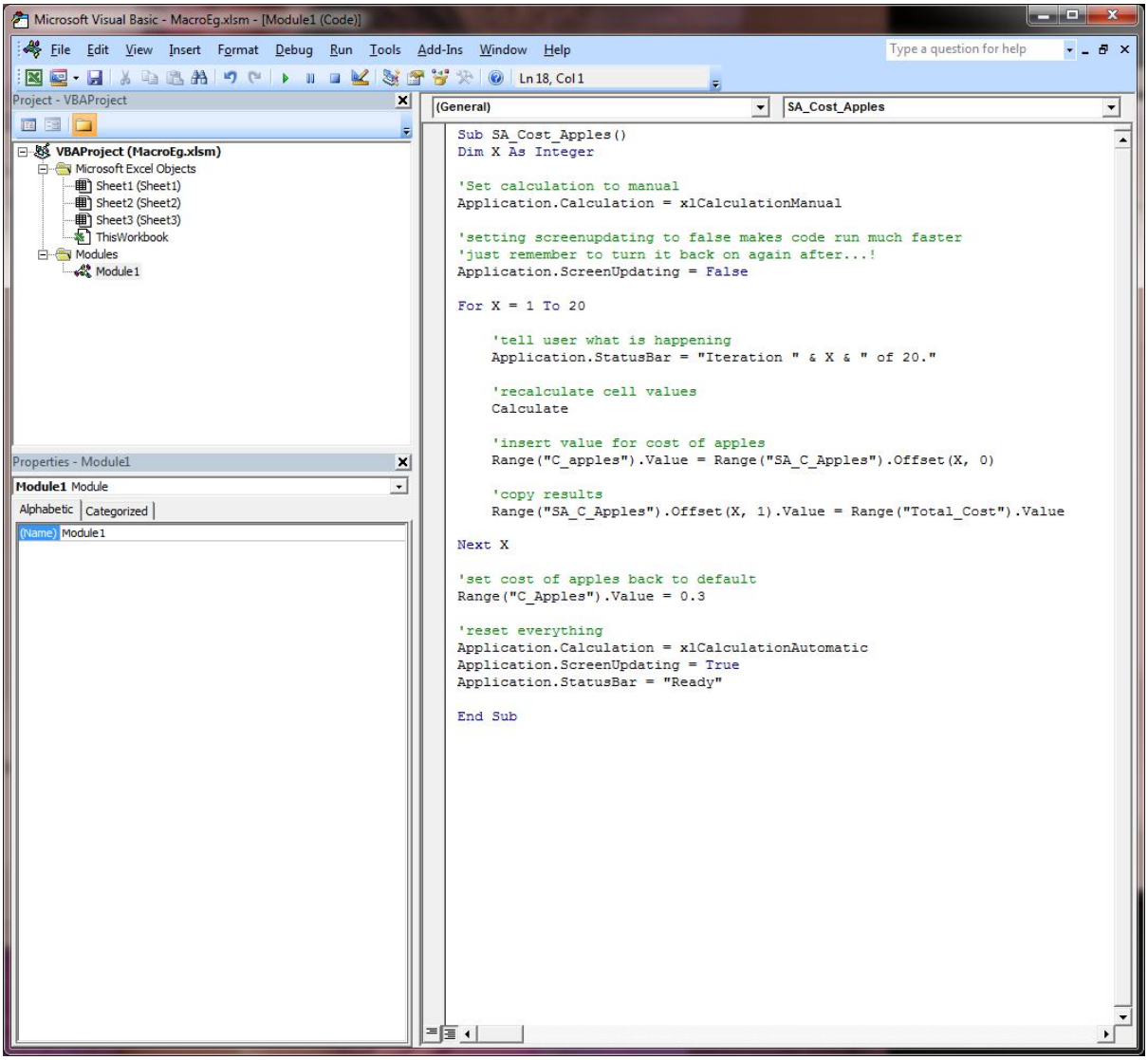
You've just done a one-way sensitivity analysis showing the effect of changing the cost of apples on total cost.

Note that the cell "C_Apples" still has the value '£1.00'. If you were to do a sensitivity analysis on 'number purchased', you'd have to remember to reset the cost of apples to their base case value (£0.30).

Finally, save your work. You'll need to save it as a 'macro enabled workbook'.

Some useful functions

I've added some useful bells & whistles to the code in the image below:



Comments

The first thing to note is comments. If you begin a line with a single inverted comma: ' excel ignores everything after that on the same line and colours the text green. Use this to comment your code. ALWAYS do this A LOT. It makes life a lot easier if you come back to the code after a few months.

Automatic vs Manual Calculation

If you're doing a probabilistic sensitivity analysis, you should set excel to manual calculation. The default is automatic, meaning every time a cell changes somewhere, excel will recalculate the whole sheet. You don't necessarily want this, so set it to manual with the line

Application.Calculation = xlCalculationManual

Note it's critical to use the code

Calculate

to tell excel to recalculate all the cells, and it's a good idea to reset calculation to automatic afterwards:

Application.Calculation = xlCalculationAutomatic

Note that if you quit a macro half way through, calculation may still be set to manual. To set it to automatic, click the office button in the top left corner of the window then -> Excel options -> Formulas and select 'automatic' on workbook calculation.

Screen updating

If you've got a complex model to calculate, you can speed things up no end by telling excel not to update the screen every time it calculates, using the code:

application.screenupdating=false

Always remember to turn it back on again though with:

application.screenupdating=true

If excel is behaving oddly after you've run a macro, it's probably either because you've got calculation set to manual, or screenupdating turned off.

StatusBar

The status bar is in the bottom left corner of the excel window. It's default value is 'Ready'. If you've got a lot of code or a lot of iterations, use the status bar to tell your users what is happening. E.g.

Application.StatusBar = "Iteration " & X & " of 20."

tells the user how far the macro has got. Remember to reset it to 'Ready' afterwards to keep things neat.

Probabilistic sensitivity analysis

Think about how you can use the same code to run a probabilistic sensitivity analysis on trial data or a decision model. With trial data you want excel to firstly generate a 'new' dataset from the existing data. Then you will want to use a macro to copy the mean cost and QALYs from each treatment and store them somewhere on a spreadsheet. After that, you want it to resample another dataset, copy the results and so on.