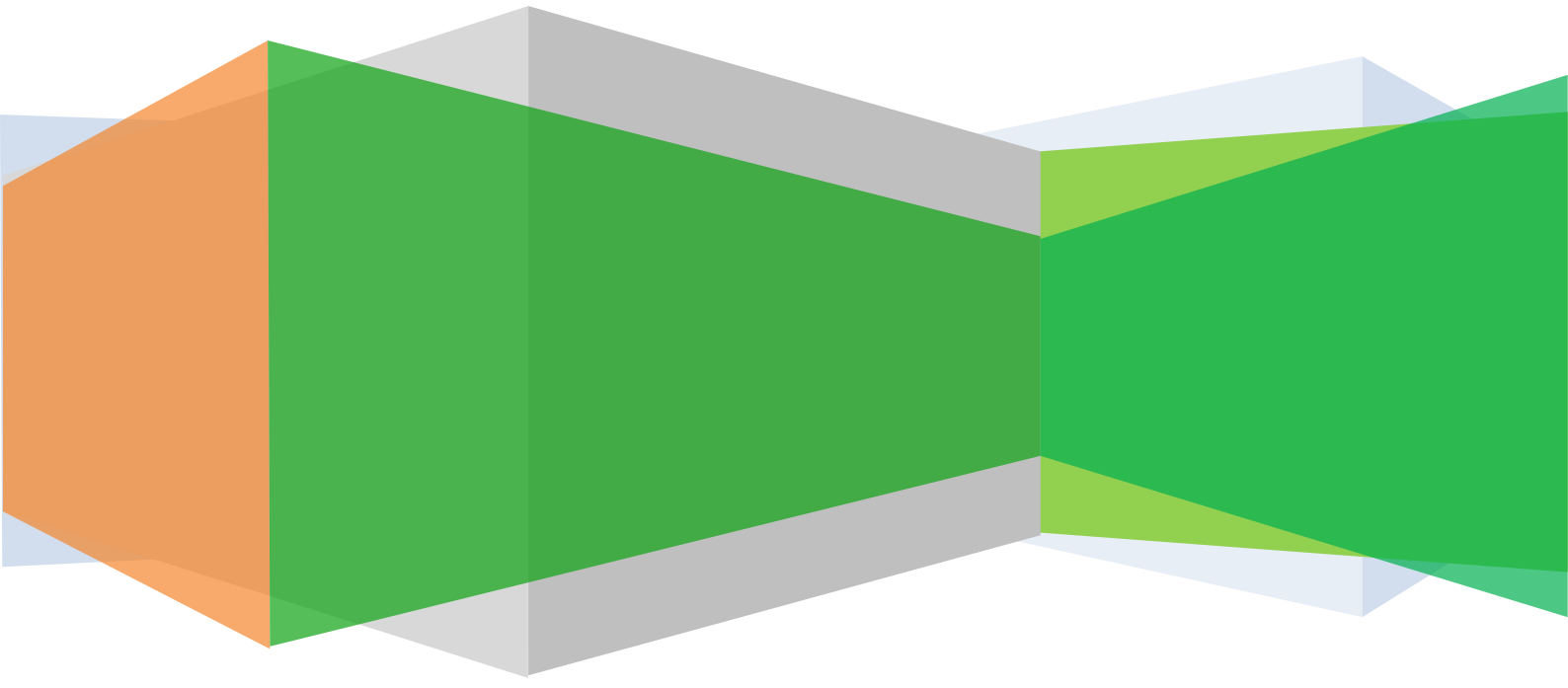
Excel-lentsolutions.co.uk

Do's and Don'ts of Macro Programming in Excel

A Methodical Approach

by Farzad Afkham



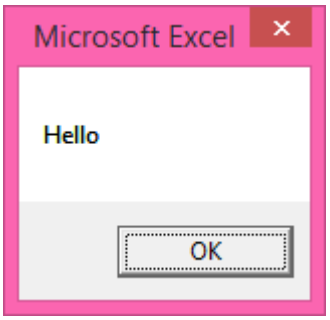
When I meet clients in Excel, usually they fall into two categories:

1) Those who have done some programming

2) Those who would never do it.

While this article is dedicated to category 1's, it is actually very beneficial (further down - section 9) to the category 2's.

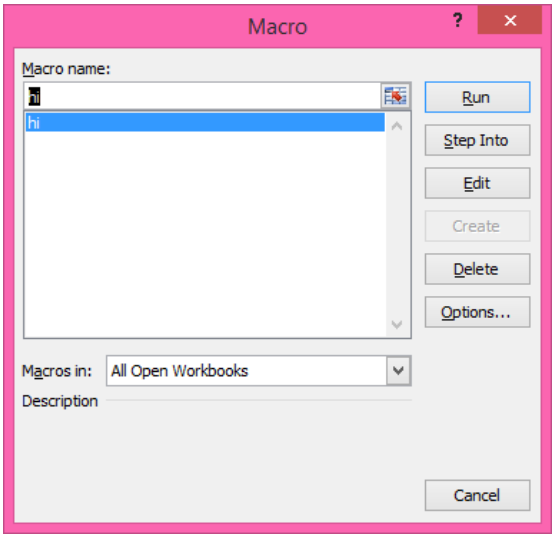
So what can I tell you about programming in Excel that you don't already know.



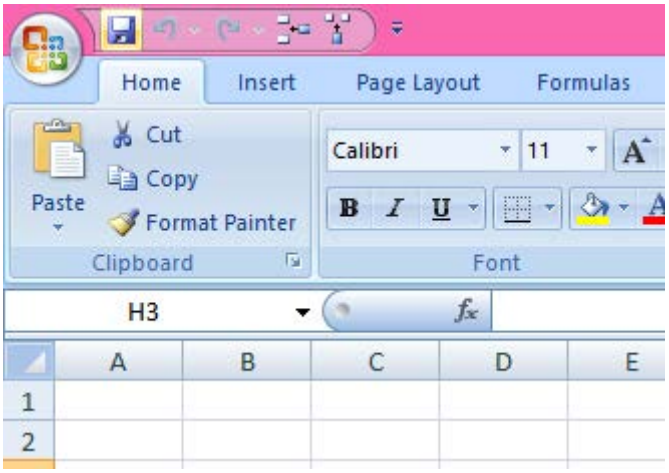
Well, let's start with the basics.

I'll go over the reasons a little further on.

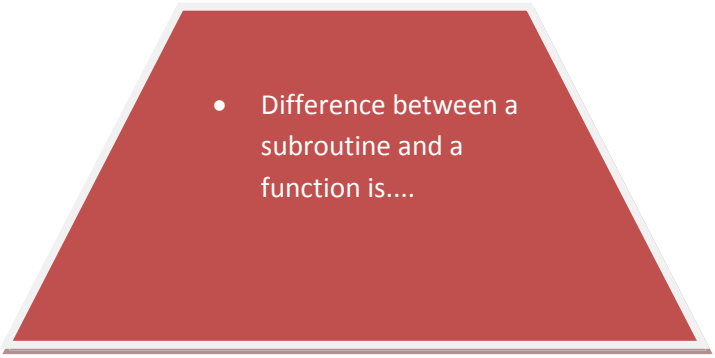
1) Never run a macro on your only copy.



2) Always save your work PRIOR to running a macro.

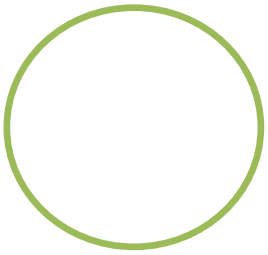


3) If you've never taken a course in programming, I recommend brushing up on some basics. It will come in massively useful when your code doesn't work and you want to find out why.



4) If you're more of an Excel super-user than a programmer and you've learned to write macros by using the macro-recorder, big kudos to you, but there are a few limitations you should be aware of. Read the limitations of this approach below.

5) The loop that never ends.

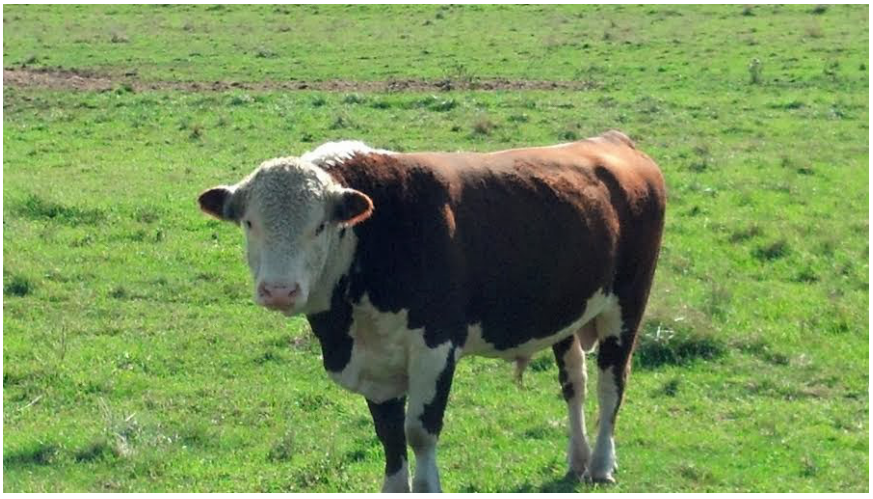


6) When Excel has become corrupted.

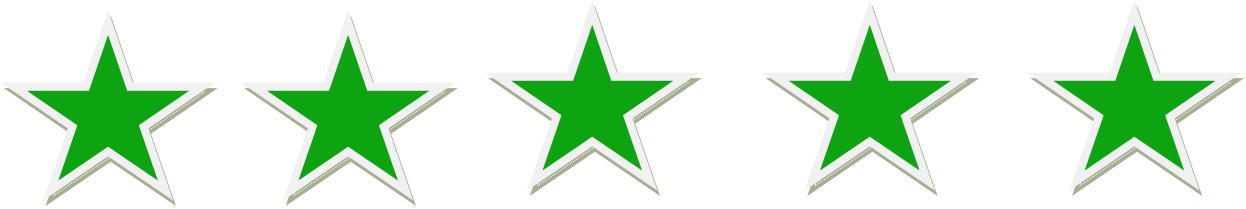
7) Mac Macros - for those programming for Excel on the Macintosh.

8) When it takes longer to program than to do it manually.

9) Farming out your automation requirements and what to expect.

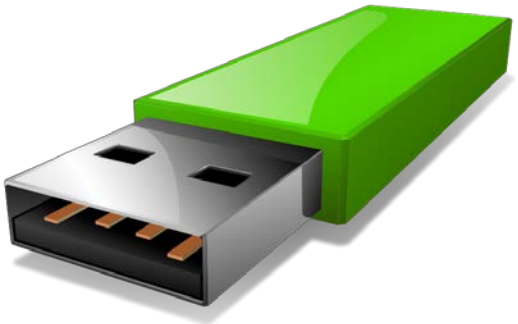


10) Cheaper versus Better.



ONE

Never run a macro on your only copy.



Macros have been known to corrupt a file. So it is always worth backing up a file on which you're planning on working prior to creating a macro or running vba scripts (aka macros), preferably on a removable drive, eg. a flash drive/memory stick. That way should anything go wrong, you'll be able to open your file again, maybe even on another computer.

TWO

Always save your work PRIOR to running a macro.

Macros cannot be undone. With Excel, usually you have a get out of jail free card if something goes wrong. For instance if you over-type a data cell. You can always go to the Edit menu and click the undo button or Press the Ctrl button on your keyboard and while holding it down, press "Z" key. This will restore your last edit to how it was priorly. If you keep undoing, it will keep going back.

No such luck with the Macro you just ran. If you expected the Macro to have done one thing, and you got something else, you're stuck with it. THE ONLY WAY AROUND this is to close down the file WITHOUT saving it. Re-opening it will hopefully get you back to the file prior to the macro being run.

Why do I say hopefully? Because if in your macro you've written in a SAVE command, then it has already saved your workbook and there's no return. This should hopefully high-light point 1: running a macro on your only copy should NEVER be done!

THREE

If you've never taken a course in programming, I recommend brushing up on some basics. It will come in massively useful when your code doesn't work and you want to find out why.

Whether you consider yourself a programmer or not, as you delve into the world of the macro recorder and begin to look at the code that the Excel Application has written for you, you'll soon start wondering what happens if you begin to edit the code. And before you know it, you're programming.

However the best way to write a program isn't through the recorder. That's just the beginning of your journey. And while it may save you time today because it does exactly what you need it to do, there will come a time tomorrow where you data starts at a different point, it ends at a different point, there are more rows to your new data and more columns, there are different data now in the columns than expected. This is because perhaps you've inserted a column or you've added a subtotal row. Do you really want to go into the code and change it 'every time'?

Code that is well-written will look at your data every time. It will determine what information is in which columns, it determines how far down the rows it should look. It's intelligent. You write the code just the once and it serves you every time.

So if you're serious about saving yourself time and errors, get yourself on a programming course of some kind. Better that, than wonder why your data has gone awry. And if this has been happening for months, you might have bigger problems than you initially thought as essentially the original data may have been over-written. The major advantage of improving your understanding of programming comes when your macro doesn't produce the expected results and you want to a)Learn why, and b), and more importantly rectify it.

FOUR

If you're more of an Excel super-user than a programmer and you've learned to write macros by using the macro-recorder, big kudos to you, but there are a few limitations you should be aware of.

```
Function Factorial(x As Integer) As Integer
If x <= 1 Then
    Factorial = 1
Else
    Factorial = x * Factorial(x - 1)
End If
End Function
```

Apart from what has been said above, another advantage of well-written code is speed of execution. If you need to run the macro and get the results, reports, etc. right away, do you really want to wait minutes each time for it? A well-written macro usually delivers your information in seconds!

Programmers take advantage of something called a loop, that enables a process to be iterated as many times as is necessary in order to complete the job. Now the Macro recorder does not do that. Essentially if you've used the recorder, it has written the code again and again and again. And so what looks like 30 pages of recorded code, could have been only 6-8 lines neatly placed in one or two appropriate loops.

FIVE

The loop that never ends.

If you're already using loops in your macros beware of the loop that you thought would end at a certain point, but then it doesn't because you didn't take something into account and it goes on and on forever.

Use the ESC key or the Fn + Break key on your keyboard to put a stop to it. Here's an example of a loop that never ends:

Look through the data and stop looking when you see the name Jill!

But Jill has left the company and her name no longer appears in the list! Whoops. The computer will keep looking!

Jill's name was only recently removed!

SIX

When your Excel file has become corrupted.

This doesn't happen very often, but sometimes your file, for whatever reason becomes corrupt and nothing works as expected. At times like these calculations go awry. I've encountered 2+2 = 6. That's the time to throw in the towel and realise, you're not going to get plausible results or a recurring experience, which as a programmer you count on. So here's the first thing I recommend:

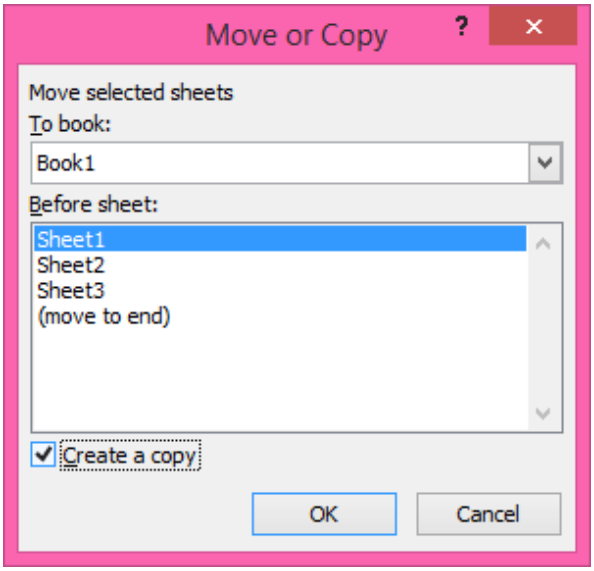
a) Save as [Another name] and close Excel. Then reopen and see if that has fixed it.

If not

b) Copy and paste all the VBA modules into a word document and recreate the data files by copying the sheets into a brand new workbook and then adding the VBA code into brand new modules. Save and see if that has fixed the problem.

If not

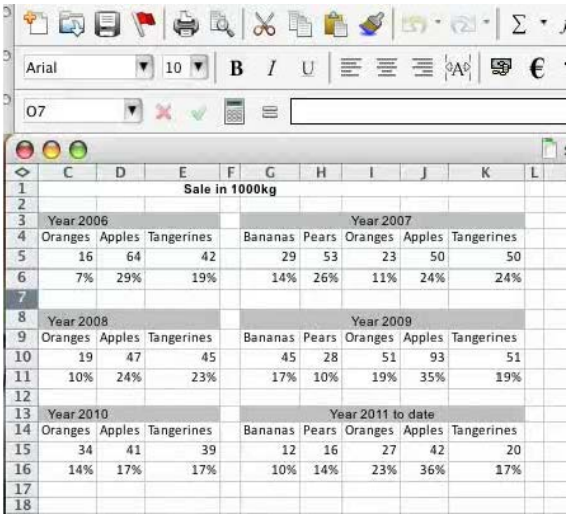
c) Copy the data from each tab from the Excel workbook, then copy and paste the VBA code from the aforementioned word document. This should hopefully fix the problem.



Be sure and check the 'Create a copy' box!

SEVEN

Mac Macros - for those programming for Excel on the Macintosh.



I've successfully created Macros for Mac user clients, but there are a few points worth bearing in mind when creating macros for the Macintosh. Certain commands will not work. So if the macro was initially written for the PC, it is high advisable to test the macro in the same environment on the MAC.

This not only means the particular MAC architecture, but also the version of Excel being used. Certain menus or buttons or active x controls may not work on the MAC. Windows API commands more than likely will not work on the MAC. So if your macro allows you to browse for a file on your PC computer, the chances are the browsing function won't work on the MAC and will cause it to crash.

Dates work differently on the MAC. Excel is a windows-based application, so the user experience will always be superior on the PC, but within limitations, there's much that can be achieved in the MAC environment.

EIGHT

When it takes longer to program than to do it manually.

When I'm asked to consult on a project, it is foremost on my mind, how long it is all going to take to provide the client with an end deliverable that is bug-free, user-friendly, efficient, quick, does what the client intended and is extensible. And this is the same advice I'm going to pass on to you. You might find that you need something done that clearly should be automated. So as you start going through the motions, you figure why not turn on the macro recorder and I can then do the same thing for the rows below. But then you realize that you have to go into the code and change the cell reference of the starting cell and then subsequent cells and lo and behold what should have taken you 5 minutes has taken about 15 minutes of fiddling about with the code. This idea can be extended to something that could have taken 1 hour but because of the programming and debugging has now taken around 3 hours.

So my advice is this: Programming is a very exact discipline. The computer will do what you ask it to do. And EXACTLY what you ask it to do. It won't do more and it won't do less. Whether you have miscalculated what you asked of it is irrelevant to the computer. It won't intuit for you. There's a certain amount of debugging that goes with every bit of code that you either record or write from scratch, if you're going to reuse that code. And apart from the writing of the code, much of your time will be spent debugging. So,

IF you're going to use your code again and again - That is when you should consider getting a macro to do the job for you.

IF you are going to do a job that you foresee taking hours or days by a human, that is again when you should consider a macro for the task.

Those are the only times you should consider writing a macro or getting someone else to write one.

Only 2 reasons to consider writing macros:

IF you're going to use your code again and again...

IF you're going to do a job that you foresee taking hours or days by a human...

Nine

Farming out your automation requirements and what to expect.

I'm often asked the question: How long will it take? And after what I've just said, you know where I'm going with this. The whole process takes a very detailed understanding. My clients know that I'm going to give them a button or a menu that will do magic, that once they click on, will do all the tasks for them in the blink of an eye. But that comes at a cost.

a) I need to spend time to understand what it does already and what needs to be added or automated, if it already exists. If I'm building something from scratch, then I need to get an understanding of what needs to happen.

b) I then need to translate that understanding into an e-mail or letter of agreement, so to be sure that my understanding of what needs to happen (in my words) does not conflict with the clients' understanding of what needs to happen.

c) If the understanding is correct, I then need to create a road map of what processes need to take place and in which order.

Let me give you an example. Let's say you are asked to go and make a cup of coffee. For the human brain that is a fairly simple task. You can describe it in a sentence and the execution of that, although complex, for an able-bodied human it is a series of instructions from the brain that most humans can follow.
Now, let's put that same task into a series of instructions. a) Get up, b)go to the kitchen, c)make a cup of coffee, d)Return with the coffee, e)Give it to the person who requested it.

But that is a simplified list of instructions really, as you can break a) down in the following fashion. i)Lean forward, ii)Put your weight on both legs, iii)push until your legs straighten, and

b) can be broken down: iv) In the standing position, roll your chair back with the back of your legs, v)put your right leg to the right, vi)put your left leg to the right. vii)Once you've cleared your desk, put your right leg forward, viii)Put your left leg forward. REPEAT steps vii & viii until you reach the hallway, ix)Turn left into the kitchen. etc.

And this could be broken down even further, because the brain is a marvellous piece of organic machinery. Now you have some idea as to what a programmer has to do. And if there's an obstacle on the way to the kitchen, did the programmer think of providing instructions to get around that, or will the person walk straight into that obstacle? This is what is known as a bug. Something unexpected that the programmer did not foresee. Sometimes it's because the programmer didn't know that a chair might be rolled out into the hallway and didn't plan for it. Sometimes it's because the requester didn't know at the time of providing the brief that there would ever be a chair in the hallllway.

In either case, the person getting the coffee will not reach the kitchen without instructions for circumventing the chair in the hallway. Of course I've provided an over-simplified example of what I do, but essentially my time is spent planning what needs to be done and planning what could go wrong and what to do in case it does, and that is what takes time. In general, I can provide a fairly good estimation of the time it will take, but each job is different and estimation is not an exact science by definition. I hope this has given you a pretty good idea of what to expect. In the long run, you have a piece of software written for you that is tailor-made, that is going to make your life easy for years to come and the long-term benefit of going the macro-route, far outweighs having you or your staff doing it manually both in terms of time and in terms of man-hour cost.

TEN

Cheaper versus Better.

I was once told that there's a student who is willing to do what I do for a few pounds per hour less. And I explained that the student isn't doing what 'I' do. He's doing what 'he' does. And they are different! But essentially writing a programmed macro. So why not go for the student versus the experience of a professional?

Well, hopefully the professional, if he has been in business for several years, has learned tricks, solutions, strategies etc. that the student just starting out hasn't. The professional, was once that student and had to discover certain things, had to research certain solutions, had to seek help, had to make mistakes and learn from them.

What does that mean to the buyer?

Well if the student charges half of what the professional charges per hour and it takes him twice as long as the professional, what is the net price to the buyer?

The difference is that the buyer has waited twice as long for the product and paid the same amount.

But the difference doesn't just end there. The professional's code will stand up to the chairs in the hallway because his experience tells him there are likely to be chairs in the hallway. The student won't know that and after waiting twice as long the buyer will be calling the student to say the program is falling over!

Beware of those who offer programming services for low 'flat' rates. We've all been enticed by those low bids that seem too good to be true. They do this to win the job and then when they run into trouble, they know they won't be earning more for the extra 20 hours they are spending on the work, and soon their motivation begins to fade and what happens to the buyer?

The buyer waits and waits...and waits. Sometimes the buyer ends up hiring the professional to either complete the job or to start the job from scratch! So don't put yourself or your business through that! A good programmer will prove invaluable. They might even reuse code they've already written for someone else to save hours on the job and save you ££££s.

The moral of this story? Hire a professional. To get what you expected reliably, in a timely fashion, for a product that has longevity, for customer service that attends to your concerns and for your peace of mind.

Farzad Afkham is a Mathematician and Senior Excel VBA Developer. He is the Managing Director of Excellentsolutions.com and has consulted at companies like Ernst & Young in the Financial Sector and the BBC in Media, as well as many other SMEs. He has written original algorithms for many businesses' models in a career spanning over 10 years.

Here's what some of our clients have said about us:

"When a client needed some sophisticated Excel programmes written I knew Farzad was the man to help us.

The routines Farzad created were both effective and efficient, saving the owner of the business the best part of a day every month, as well as allowing us to very quickly create a database of all the related transactions for the previous 9 months (a massive task if we'd tried to do it manually).

I would highly recommend Farzad's skills to anyone who does time-consuming tasks through spread-sheets - I know he will automate them to take a fraction of the time it currently takes you now, and probably more accurately too!"

- Stuart Kerslake, On behalf of Interpost

"We contracted Farzad to update our risk management tool and he did an excellent job for us. He came in to our business, rapidly understood our needs, delivered what he said he could deliver in less time than we had anticipated. We won't be looking any further afield for future updates."

- Philippa Klein, Head of Business, BBC Factual Production

"Farzad has been instrumental in significantly adding value to our data products by increasing our security as well as allowing us to market our data flexibly and simply to demanding clients. he is truly an expert in Excel and would add value and save costs in any process that requires repetition."
Service Category: Business Consultant
Year first hired: 2010 (hired more than once)
Top Qualities: Great Results, Expert, High Integrity"

- Simon Bliss, CEO of Principal People and Market My Business