Excel formula development in the real world

Name field reconcatenation in Excel explained through building worksheet formulae and custom VBA functions.

There must have been times for all of us when we've been presented with a large database containing people's names and other data which needs to be manipulated or reported on in some way, probably in alphabetical order. As the records appear in Excel for the first time, the heart sinks as the first column contains the names:

John Smith
Kevin Adams
-etc

I think that 9 out of 10 such lists I am given contain a name field formatted in this way, first name first. On the other hand, 9 out of 10 clients want their analyses presented in alphabetical order by surname. If you often find yourself in the same boat, this article will help you. We shall start by constructing an Excel worksheet formula to reverse the first name and surname positions in the field. Then we'll work to expand the function so that it can deal easily with initials, multiple forenames or any combination of these features. Finally, we'll consider the potential disadvantages of such a function, and look at an alternative approach, using VBA to construct a custom worksheet formula and overcome those disadvantages.

The basic task is to take the existing entry and re-engineer it, thus:

'John Smith' becomes 'Smith, John'

At first glance, this is an easy problem. All we need to do is look for a space in the text, carve up the string and then recombine it appropriately. If A1 contains the original string, then A2 can contain the formula:

```
=SEARCH(" ",A1)
```

This will return a number representing the first space character within the string in cell A1. In our example, it will give '5' because the fifth character in the string is the first space.

Using this, we can chop up the string into its two parts. In A3, enter:

```
=LEFT(A1,A2-1)
```

This formula tells Excel to return the leftmost 4 characters from the string in A1. It's 4 because that is the result of 5 minus 1, so the formula returns all characters to the left of the space, but not the space itself.

In A4, we need:

=RIGHT(A1,LEN(A1)-A2)

There are two functions here. Firstly, LEN simply returns the length of a string. RIGHT works the same way as LEFT, only… well, you know. So this formula returns just the characters to the right of the space in the string in A1.

Let's summarise what we have so far:

Cell	Formula	Value
A1		John Smith
A2	=SEARCH(" ",A1)	5
A3	=LEFT(A1,A2-1)	John
A4	=RIGHT(A1,LEN(A1)-A2)	Smith

Now, we can put our final version in A5 as follows:

=A4 & ", " & A3

This will return in our example: Smith, John. Our work is done.

Cell	Formula	Value
A5	=A4 & ", " & A3	Smith, John

One word of caution here. If the person entering the data has put extra spaces in at the start or end of the name string, your functions will have a hard time. If you suspect this may be the case, you can replace every reference to A1 with: TRIM(A1). This will have the effect of removing all leading and trailing spaces from the string.

Back to our function. Type into cell A1 the name John James Smith and see what happens.

Cell	Formula	Value
A1		John James Smith
A2	=SEARCH(" ",A1)	5
A3	=LEFT(A1,A2-1)	John
A4	=RIGHT(A1,LEN(A1)-A2)	James Smith
A5	=A4 & ", " & A3	James Smith, John

Things start going wrong in A3, where only the first forename appears, and we end up with 'James Smith, John'. Oops. As it stands, our formula will not cope properly with more than one space in the string.

Here are some more entries which would cause issues:

J J Smith
J. J. Smith
John J Smith
John J. Smith
John James William Albatross Smith

Remember the SEARCH function in A2? That looks for the first space character in the string. We can get it to look not for the first instance of a space, but for the second, by simply adding a third argument to a new formula in A6, thus:

=SEARCH(" ",A1,A2+1)

The effect of this third argument is to search again for a space, but starting to search one character to the right of the first space we found.

If there isn't a second space, the formula will return an error, but we can allow for that like this:

=IF(ISERROR(SEARCH(" ",A1,A2+1)),A2,SEARCH(" ",A1,A2+1))

This looks for the second space. If this would cause an error, then it just returns the position of the first, as found by the formula in A2, but if looking for the second doesn't generate an error, then that's what the formula does. This approach won't always work, though, because there may be more even than two spaces, as in the last example above.

Ideally, we could get the SEARCH command to start at the end of the string and work backwards, but it doesn't work that way and there is no worksheet formula which does.

Instead, we have to find an approach which will find out how many spaces there are in the string and then give the position of the last one. Actually, we can achieve this quite easily using just the formulas we've already met here, plus one other.

In cell C1, enter:

John James William Smith

In C2 enter the new formula:

=SUBSTITUTE(C1," ","")

The first set of quotes in this formula contains a space, the second nothing. The effect of this formula is to return a version of the string without any spaces, thus:

JohnJamesWilliamSmith

In C3, you can enter this formula:

=LEN(C1)-LEN(C2)

This tells you the difference between the number of characters in John James William Smith and the number of characters in JohnJamesWilliamSmith. The result is 3, which is, of course, the number of spaces in the original string.

But the SUBSTITUTE has an optional fourth argument as well, and it is this one which we will now put into use.

In C4, enter:

=SUBSTITUTE(C1," ","^",C3)

That fourth argument is the instance number you want to substitute. If you omit it, this formula will replace every space with a caret (^). But by giving the instance number as the number of spaces we calculated in C3, it will only change the last space to a caret, returning this string:

John James William^Smith

Now, unless your data has a lot of caret characters in it (and if it does, use a different symbol) you can be pretty sure that there will only be one in the string, so it is this character which now becomes the object of your new SEARCH command in cell C5, thus:

=SEARCH("^",C4)

In this case, the formula produces 19, because the caret is the 19th character in the string. From there, it is an easy job to play with the string using the LEFT and RIGHT formulae as we did before. Below is a summary of the full process thus generated:

Cell	Formula	Value
C1		John James William Smith
C2	=SUBSTITUTE(C1," ","")	JohnJamesWilliamSmith
C3	=LEN(C1)-LEN(C2)	3
C4	=SUBSTITUTE(C1," ","^",C3)	John James William^Smith
C5	=SEARCH("^",C4)	19
C6	=LEFT(C1,C5-1)	John James William
C7	=RIGHT(C1,LEN(C1)-C5)	Smith
C8	=C7 & ", " & C6	Smith, John James William

Now, this is a lengthy process, but once you've been through it, you can turn this chain of formulae into one big, Swiss Army Knife of a formula which looks like this:

```
=RIGHT(C1,LEN(C1)-SEARCH("^",SUBSTITUTE(C1," ","^",LEN(C1)-
LEN(SUBSTITUTE(C1," ",""))))) & ", " & LEFT(C1,SEARCH("^",
SUBSTITUTE(C1," ","^",LEN(C1)-LEN(SUBSTITUTE(C1," ",""))))-1)
```

If you've been using the TRIM function as suggested above, you have an even longer formula:

```
=RIGHT(TRIM(C1),LEN(TRIM(C1))-SEARCH("^",SUBSTITUTE(TRIM(C1),
" ","^",LEN(TRIM(C1))-LEN(SUBSTITUTE(TRIM(C1)," ",""))))) & ", " &
LEFT(TRIM(C1),SEARCH("^", SUBSTITUTE(TRIM(C1)," ","^",
LEN(TRIM(C1))-LEN(SUBSTITUTE(TRIM(C1)," ",""))))-1)
```

Scary, huh? Not really. I didn't type that all in by hand, I simply cut and pasted it together. For instance, C8 started off saying:

```
=C7 & ", " & C6
```

What I did was copy the formula in C7 and replace the reference to C7 with the actual C7 formula (discarding the superfluous extra = sign) to get this:

```
=RIGHT(C1,LEN(C1)-C5) & ", " & C6
```

Then I replaced the C6 reference with the formula in C6 and the C4 reference with the formula in C4, and so on until the only cell referenced was cell C1. It's simple, but the resulting formula never fails to impress those not party to this technique!

Still, be that is it may, the formula is still big and unwieldy. If we want to use it again later on another sheet, we'd better pray we have a copy of the formula somewhere to paste in, because it will take a while if we have to start again from scratch.

Also, it's a lovely piece of technical writing, but visible to all users of our spreadsheet and thus copiable by others. This may not be a problem if you don't rely on this sort of work for your living, but that's precisely what some of us do!

And so it makes sense to overcome these disadvantages by creating a custom worksheet formula. If you haven't used one of these before, don't worry. It's quite straightforward, requiring only a little knowledge of VBA to convert the enormous worksheet formula above to the far more manageable:

```
=NAMEREVERSE(C1)
```

This formula behaves just like a regular, built-in Excel worksheet function. If you package your code as an add-in (beyond the scope of this article), you can keep the underlying VBA which makes it work firmly under the bonnet.

With the power of Excel's built in functions, you might imagine that the worksheet version might be far simpler than the VBA routine needed to do the same thing, but you'd be wrong.

Remember how we mourned the lack early on in this article of a worksheet formula which would search for a character within a string, working backwards from the end of that string?

Well, guess what? VBA has such a method. The VBA method InStr does much the same as Excel's SEARCH function, but there is a companion method in VBA called InStrRev, which starts from the other end.

As a result, our VBA code looks like this:

```
Function NAMEREVERSE(strValue As String)
    strLen = Len(strValue)
    strNumSpace = InStrRev(strValue, " ")
    strSurname = Right(strValue, strLen - strNumSpace)
    strRest = Left(strValue, strNumSpace - 1)
    NAMEREVERSE = strSurname & ", " & strRest
End Function
```

Notice how we have used a very similar approach as in our first attempt at a worksheet formula. Instead of SEARCH, we use IsStrRev, allowing us, of course, to start at the end and work backwards. The worksheet formulas LEN, RIGHT and LEFT have direct equivalents in VBA, which are used here to achieve the same effects.

If placed in any module, this will enable NAMEREVERSE to be used in the workbook containing the code just like any regular worksheet function. If you want this new command to be available in all workbooks, you have two choices. You can either package the workbook with the code module as an add-in, or you can paste the code into the PERSONAL.XLS workbook.

So there we have it, a common, real-world problem solved first on a worksheet, then extended to deal with awkward cases. We saw how it easily became a big and frightening single formula, and then saw how to make it more portable and less accessible to the curious by coding a custom function in VBA. Paradoxically, in this case it also became a lot simpler in the transition.