Excel Blackbelt? Solve this core Excel bug

23 September 2021

1. SUMMARY

Autofilters were the predecessors to Excel tables and are still sometimes used to filter data. An Excel bug means autofilter ranges can be misstated, hindering automated spreadsheet management.

This article looks at:

1. Why this matters
2. An Excel Blackbelt challenge: identify and solve the bug
3. The circumstances in which this bug appears
4. A solution
5. Some near misses and fails

You may want to skip from 1 to 3.

1.1 Background on autofilters and tables

Filtering data is useful, e.g. as one part of exploratory data analysis. You can use autofilter to filter data but Excel 2007 introduced Excel Tables (tables) which have many advantages including automatic resizing of tables and associated formulae as data expands. But autofilters are still around and are sometimes used.

You can have at most one autofilter on a sheet and many tables. Tables can “collide” with each other or with an autofilter e.g. where an autofilter and a table(s) share common rows. Data can all-too-easily be inadvertently hidden or even deleted because more than one object can be affected by a filter or delete.

So it’s important to know about table ranges, autofilter ranges and whether their rows or columns overlap. This should be part of an automatic audit of all important spreadsheets.

1.2 The autofilter address bug

Microsoft says that VBA can return a range object representing the range to which the specified AutoFilter applies. The same applies to the corresponding range address. Microsoft gives the following example:

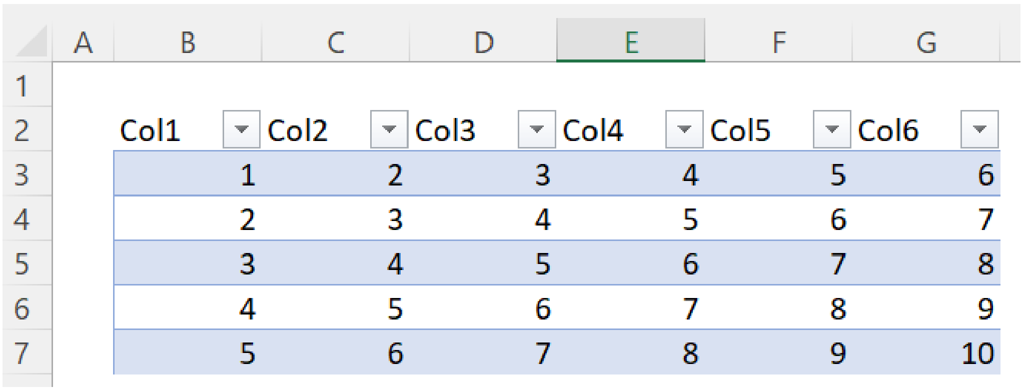
```
rAddress = Worksheets("Crew").AutoFilter.Range.Address
```

The bug is that if the active worksheet – the sheet you are looking at – has a table in the “wrong place” the above code will return incorrect results: it will overstate the number of rows in the autofilter.

This note shows the obscure circumstances in which this bug becomes apparent and supplies a workaround.

2. THE BUG IN DETAIL

We have a sheet AF, which contains a simple autofilter as follows:



	A	B	C	D	E	F	G
1							
2		Col1	Col2	Col3	Col4	Col5	Col6
3		1	2	3	4	5	6
4		2	3	4	5	6	7
5		3	4	5	6	7	8
6		4	5	6	7	8	9
7		5	6	7	8	9	10

We expand the Microsoft code into a user-defined function (UDF) and sub:

```
Function AutofilterAddress(wsname as String) as String
	AutofilterAddress = Worksheets(wsname).AutoFilter.Range.Address(False, False) ' No dollars
End Function
```

```
Sub FixedSheetAutofilterAddress()
	MsgBox Worksheets(“AF”).AutoFilter.Range.Address(False, False) ' No dollars
End Sub
```

We create a sheet Code1, on which we will use the AutofilterAddress UDF. Clearly AutofilterAddress(“AF”) will return “B2:G7”. Won’t it?

The rather nasty truth seems to be that what gets returned depends on the existence and location of any table(s) on the active sheet i.e. the sheet you’re looking at - Code1 for us.

3. TAKE THE BLACKBELT CHALLENGE

- With just the above, can you identify the circumstances in which the UDF gives the wrong address?
- Can you fix the UDF i.e. provide a workaround in the function?

The workbook excel-autofilter-bug-solved.xlsm provides a major clue and full solution.

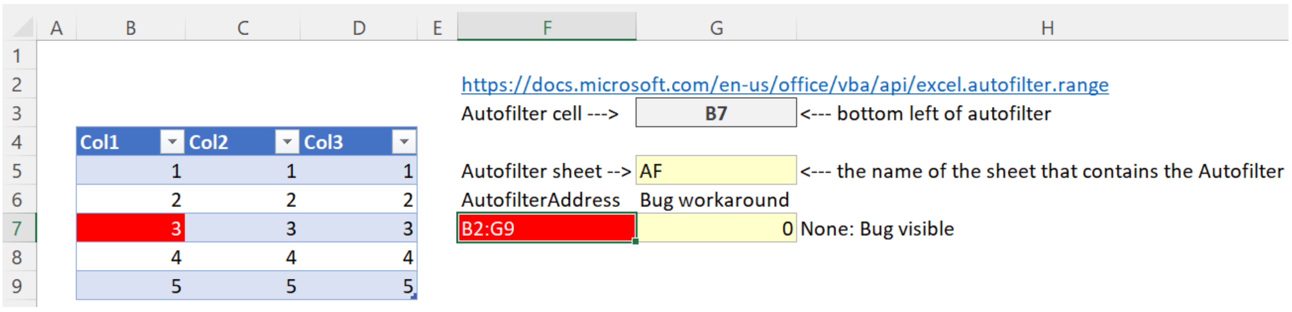
4. EXAMPLES: WHEN THE BUG REVEALS ITSELF

More specifically, the bug is about the location of tables on the active sheet versus the address of the bottom left hand cell of the targeted autofilter(!) – B7 in our example. If B7 on the active sheet falls anywhere in a table on that sheet – apart from the last row – the autofilter address will be reported incorrectly. Astounding.

Let's get more practical with some examples, noting that the autofilter address is B2:G7.

Example 1

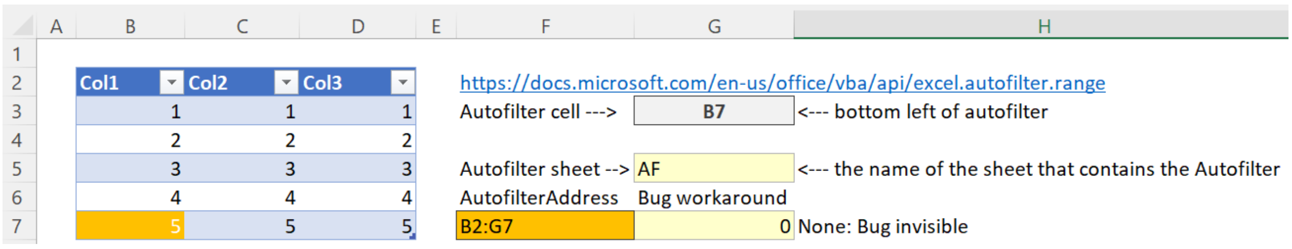
The table on the active sheet has address B4:G9. B7 lies within the table, but not on its last row. The autofilter address is incorrectly reported as B2:G**9**, not B2:G7. Note that the 9 corresponds to the last row of the table on the active sheet.



	A	B	C	D	E	F	G	H
1								
2						https://docs.microsoft.com/en-us/office/vba/api/excel.autofilter.range		
3						Autofilter cell --->	B7	<--- bottom left of autofilter
4		Col1	Col2	Col3				
5		1	1	1		Autofilter sheet -->	AF	<--- the name of the sheet that contains the Autofilter
6		2	2	2		AutofilterAddress	Bug workaround	
7		3	3	3		B2:G9	0	None: Bug visible
8		4	4	4				
9		5	5	5				

Example 2

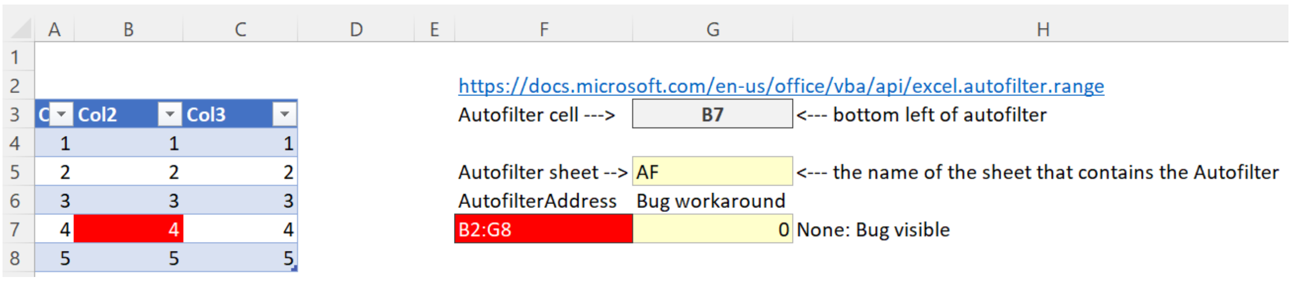
Move the table 2 rows up. The bug is hidden i.e. the autofilter address is reported correctly.



	A	B	C	D	E	F	G	H
1								
2		Col1	Col2	Col3		https://docs.microsoft.com/en-us/office/vba/api/excel.autofilter.range		
3		1	1	1		Autofilter cell --->	B7	<--- bottom left of autofilter
4		2	2	2				
5		3	3	3		Autofilter sheet -->	AF	<--- the name of the sheet that contains the Autofilter
6		4	4	4		AutofilterAddress	Bug workaround	
7		5	5	5		B2:G7	0	None: Bug invisible

Example 3

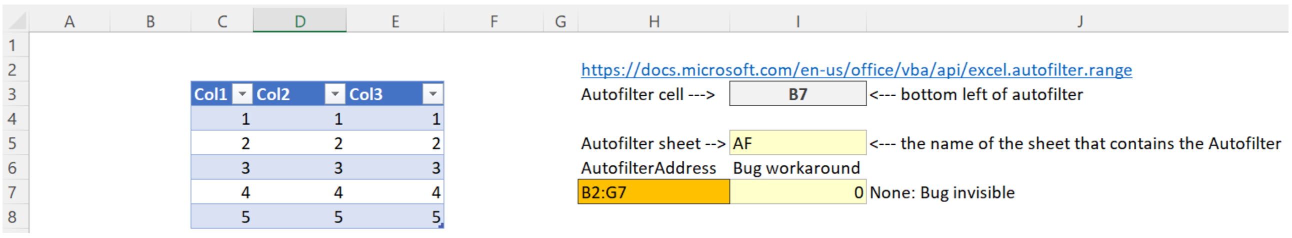
Move the table 1 column to the left and the bug becomes visible.



	A	B	C	D	E	F	G	H
1								
2						https://docs.microsoft.com/en-us/office/vba/api/excel.autofilter.range		
3	C	Col2	Col3			Autofilter cell --->	B7	<--- bottom left of autofilter
4	1	1	1					
5	2	2	2			Autofilter sheet -->	AF	<--- the name of the sheet that contains the Autofilter
6	3	3	3			AutofilterAddress	Bug workaround	
7	4	4	4			B2:G8	0	None: Bug visible
8	5	5	5					

Example 4

Insert two columns before the table and the bug becomes hidden:

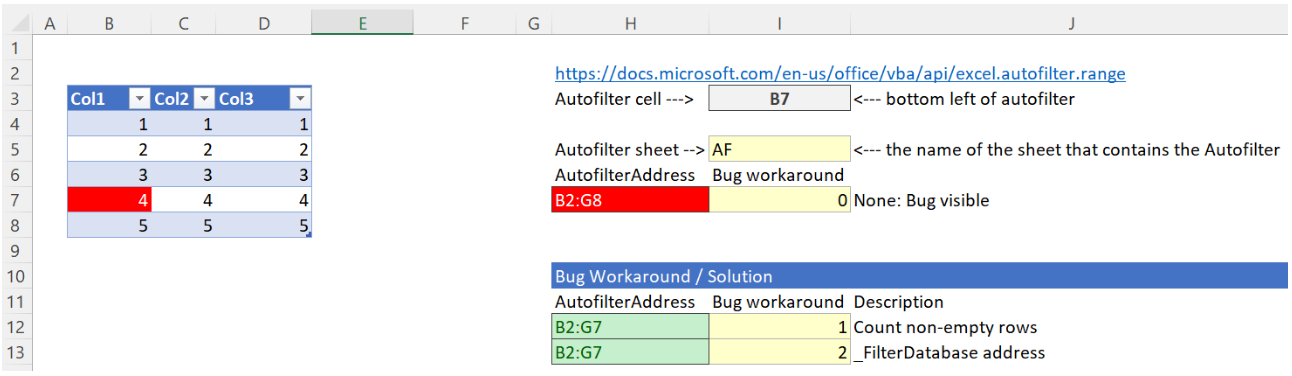


	A	B	C	D	E	F	G	H	I	J
1										
2								https://docs.microsoft.com/en-us/office/vba/api/excel.autofilter.range		
3			Col1	Col2	Col3			Autofilter cell --->	B7	<--- bottom left of autofilter
4			1	1	1					
5			2	2	2			Autofilter sheet -->	AF	<--- the name of the sheet that contains the Autofilter
6			3	3	3			AutofilterAddress	Bug workaround	
7			4	4	4			B2:G7	0	None: Bug invisible
8			5	5	5					

It gets worse. While the above is phrased in terms of the UDF, running the Sub FixedSheetAutofilterAddress while you are on the sheet Code1 will also report the wrong autofilter address.

5. A SOLUTION

Code for the solutions are provided in an associated workbook.



	A	B	C	D	E	F	G	H	I	J
1										
2								https://docs.microsoft.com/en-us/office/vba/api/excel.autofilter.range		
3		Col1	Col2	Col3				Autofilter cell --->	B7	<--- bottom left of autofilter
4		1	1	1						
5		2	2	2				Autofilter sheet -->	AF	<--- the name of the sheet that contains the Autofilter
6		3	3	3				AutofilterAddress	Bug workaround	
7		4	4	4				B2:G8	0	None: Bug visible
8		5	5	5						
9										
10								Bug Workaround / Solution		
11								AutofilterAddress	Bug workaround	Description
12								B2:G7	1	Count non-empty rows
13								B2:G7	2	_FilterDatabase address

Solution: non-empty row count

The idea is that although the address of the autofilter may be incorrectly reported, the first row is correct. We can use our UDF to count the number of non-empty rows in the autofilter and use the result to correct the number of rows and hence the address.

6. SOME NEAR MISSES and FAILURES

Near miss 1: CurrentRegion

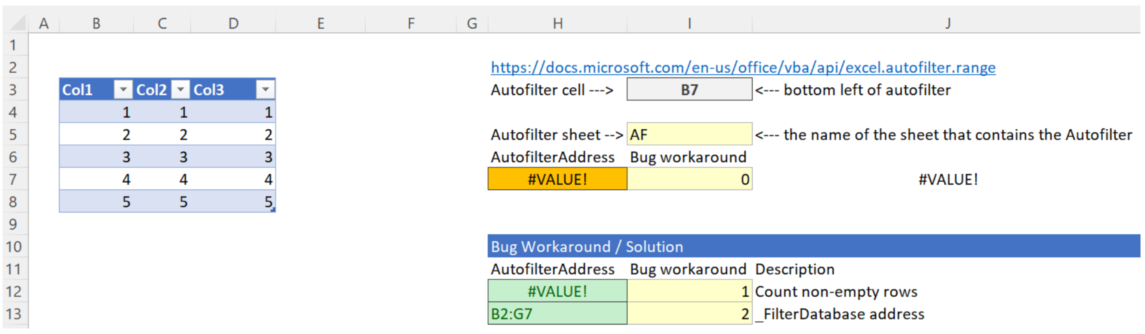
Given that we can correctly identify the first row of the autofilter, can we use the VBA CurrentRegion property to simplify the solution above? Disappointingly it turns out that the following reasonable code doesn't work:

```
Public Function foo(ByVal rng As Range)
   foo = rng.CurrentRegion.Address
End Function
```

Why it should fail logically I don't know; it's not as if we're trying to alter or select an Excel object.

Near miss 2: _FilterDatabase

When you create an autofilter Excel automatically creates a hidden named range, _FilterDatabase. This is local to the sheet on which the autofilter was created. The range is identical to that of the autofilter. Its address can be reliably obtained i.e. it doesn't suffer from the bug. This sounds promising, but the problem is that _FilterDatabase persists even after you've removed the autofilter – we want instead an error:



	A	B	C	D	E	F	G	H	I	J
1										
2								https://docs.microsoft.com/en-us/office/vba/api/excel.autofilter.range		
3		Col1	Col2	Col3				Autofilter cell --->	B7	<--- bottom left of autofilter
4		1	1	1						
5		2	2	2				Autofilter sheet -->	AF	<--- the name of the sheet that contains the Autofilter
6		3	3	3				AutofilterAddress	Bug workaround	
7		4	4	4				#VALUE!	0	#VALUE!
8		5	5	5						
9										
10								Bug Workaround / Solution		
11								AutofilterAddress	Bug workaround	Description
12								#VALUE!	1	Count non-empty rows
13								B2:G7	2	_FilterDatabase address

Failure 1: Selecting / activating sheets

It was suggested to me via the LinkedIn Excel Blackbelts group that activating or selecting the sheet with the target autofilter might work, but there are several problems with this:

- You can't use a function to select a range or sheet (typically this would be a recipe for disaster if such a function ran simply through a sheet recalculation)

  You really don't want to be manually running Subs – and even if you do the then activated sheet might have a table whose presence and location meant the bug came back into play.

Failure 2: Range.End

This is effectively a more naïve version of the CurrentRegion idea. While in easy circumstances the code

```
Worksheets("AF").range("B2").End(xlDown).Address
```

might appear to work, there are problems if (in our case) row 7 of the autofilter is hidden (through using the autofilter) or one of the cells in B2:B7 is empty.