

Average by Color in Excel (Step-by-Step Example)

Authored by
stats writer

November 17, 2025

RECOMMENDED CITATION

stats writer (2025). *Average by Color in Excel (Step-by-Step Example)*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=95129>

The ability to calculate an average based on cell formatting, specifically color, is a highly requested feature in Excel. While standard functions like `AVERAGEIF` handle criteria based on values (numbers, dates, text), they lack the capability to directly reference visual attributes like background fill or font color. This limitation often necessitates advanced solutions when analyzing visually sorted data.

Analyzing and summarizing data based on color coding is invaluable for identifying immediate trends or patterns--for instance, quickly finding the average revenue generated by "high-priority" tasks highlighted in green, or the average cost of inventory items marked in red. Since Excel does not include a native formula for this specific task, we must leverage VBA (Visual Basic for Applications) to create a custom user-defined function. This comprehensive, step-by-step guide will walk you through setting up the necessary Macro and applying it successfully to calculate the average of numerical values based on cell color.

In practical data analysis, color coding is often used as a preliminary sorting or categorization mechanism. Analysts frequently color-code cells to represent different statuses, priority levels, or outlier identification.

Why Calculate Averages Based on Cell Color?

A common scenario involves a large dataset where certain entries have been manually or conditionally formatted with a specific background color. Suppose you manage sales data where cells are highlighted yellow if sales exceeded a quota, green if they met the quota, and red if they failed. To derive meaningful insights, you need to answer questions like: "What was the average sale amount for all yellow-highlighted entries?"

Attempting to use standard aggregate functions (like **AVERAGE** or **SUM**) on such filtered data is complex and prone to error, especially if the filtering criteria change frequently. Creating a robust, repeatable process using a custom Function simplifies this analysis significantly, ensuring accuracy and efficiency regardless of the dataset size. This methodology moves beyond simple visual interpretation to deliver quantitative results derived directly from visual attributes.

For our demonstration, we will use a straightforward dataset where we want to calculate the average value based on three different cell colors:

	A	B	C	D	E	F
1	Values					
2	20					
3	13					
4	15					
5	18					
6	20					
7	24					
8	26					
9	30					
10	12					
11	15					
12						
13						
14						
15						
16						
17						
18						

As shown above, relying on manual calculation for this task is inefficient. The most effective method is utilizing VBA in Excel to introduce this missing functionality.

Step 1: Preparing Your Data Set

Before implementing any custom code, the first requirement is to organize your source data within the spreadsheet. Ensure your numerical values are entered into a single column (e.g., Column A, starting at row 2) and that the cells requiring analysis are formatted with the desired background colors.

For this specific example, we utilize the following structure, which contains various numeric values that have been color-coded based on certain criteria (which we will define later in Step 4 as light green, yellow, and blue). This data structure will serve as the input range for our custom averaging function.

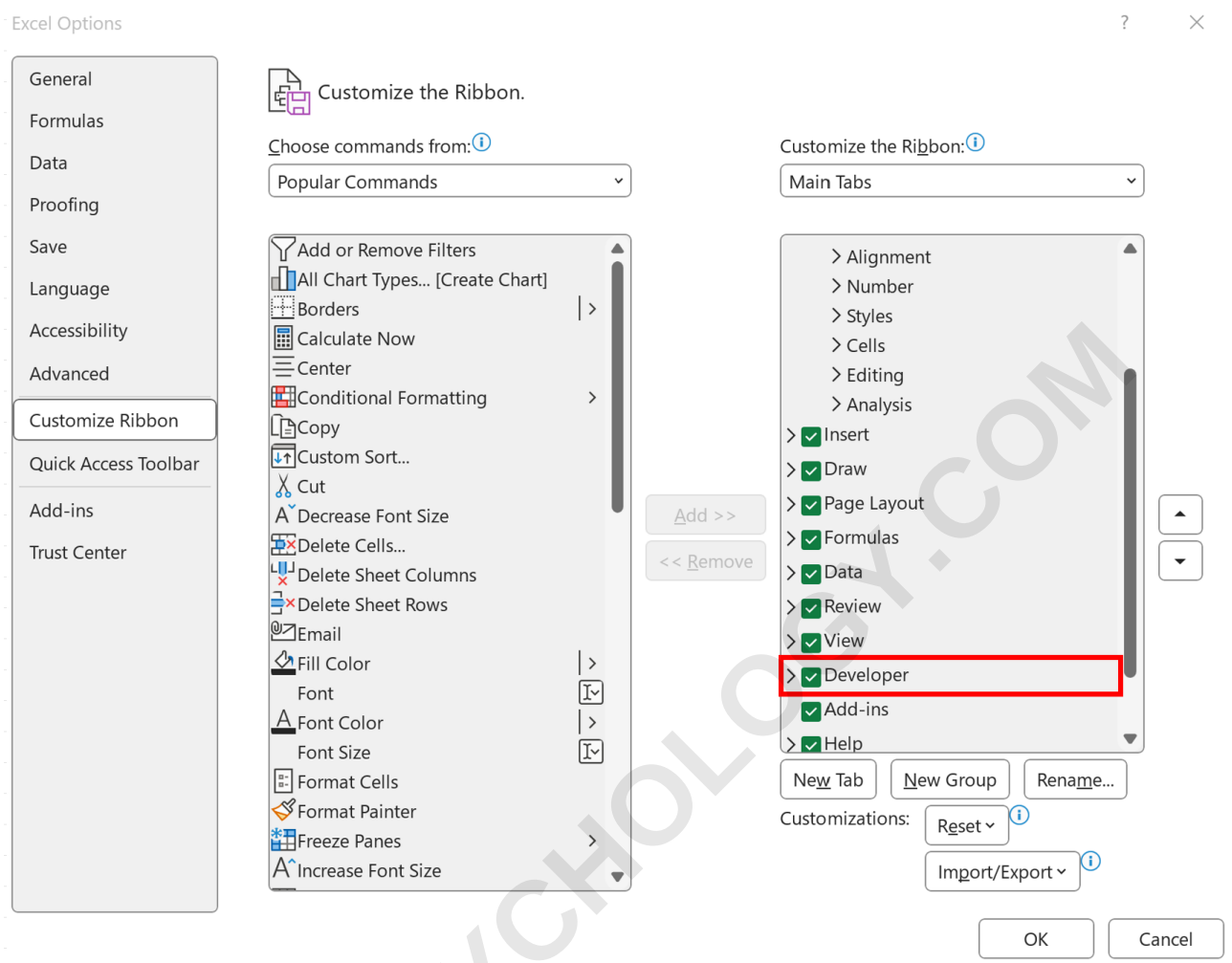
	A	B	C	D	E	F
1	Values					
2	20					
3	13					
4	15					
5	18					
6	20					
7	24					
8	26					
9	30					
10	12					
11	15					
12						
13						
14						
15						
16						
17						
18						

Step 2: Enabling the Developer Tab

To work with VBA code and create custom functions, the **Developer** tab must be visible on the main ribbon interface. By default, this tab is hidden in most Excel installations. If you already see the **Developer** tab, you may skip this step; otherwise, follow these instructions carefully.

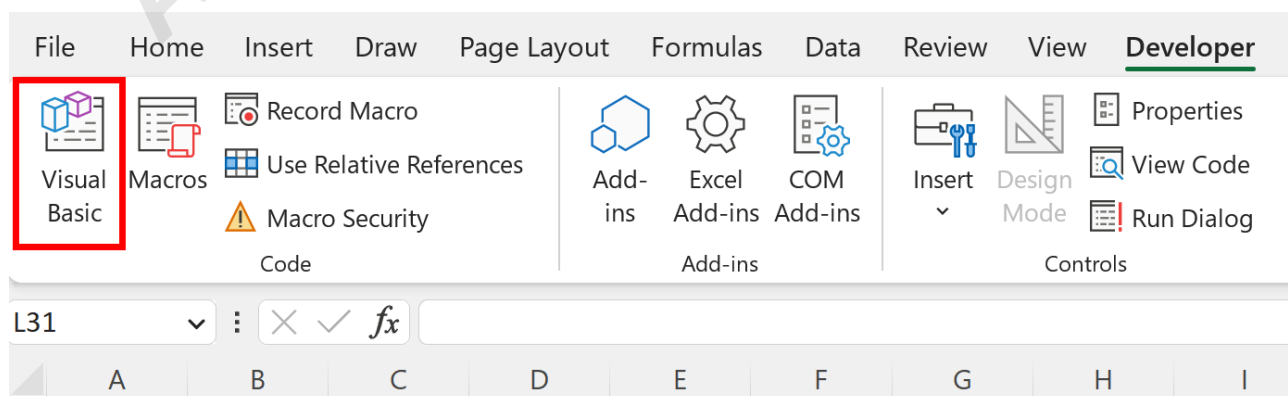
To enable the Developer tab, navigate through the Excel menu options: click the **File** tab, then select **Options** (usually located at the bottom of the left-hand navigation pane).

In the subsequent dialog box, select **Customize Ribbon** from the menu on the left. Under the section titled **Main Tabs**, locate and check the box next to **Developer**. Once selected, click **OK** to save the changes and display the tab on your ribbon:

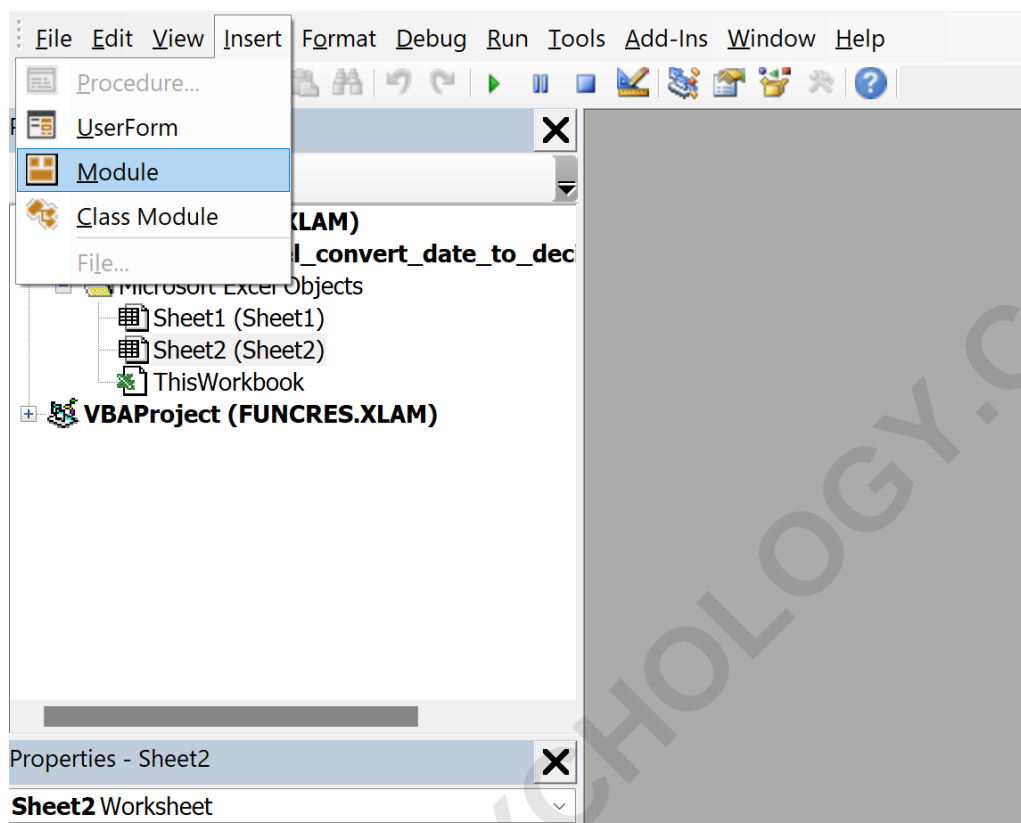


Step 3: Creating the Custom VBA Function

With the **Developer** tab visible, we can now access the Visual Basic Editor (VBE) to write our user-defined Function. Click the **Developer** tab along the top ribbon and then click the **Visual Basic** icon, which typically opens a new window dedicated to code editing:



The VBE environment requires a place to store global code that can be called from any worksheet. To achieve this, click the **Insert** tab within the VBE window and then select **Module** from the dropdown menu. A new, blank Module window will appear, ready to accept the custom code:



Now, paste the following **VBA** code snippet exactly as shown into the Module code editor. This code defines a custom function named `AvgCellsByColor` that accepts two primary arguments: the data range to average and a reference cell whose color should be matched:

Function AvgCellsByColor(CellRange As Range, CellColor As Range)

```
Dim CellColorValue As Integer
```

```
Dim RunningAvg As Long
```

```
Dim RunningSum As Long
```

```
Dim RunningCount As Long
```

```
CellColorValue = CellColor.Interior.ColorIndex
```

```
Set i = CellRange
```

```
For Each i In CellRange
```

```
    If i.Interior.ColorIndex = CellColorValue Then
```

```
        RunningSum = RunningSum + i.Value
```

```
RunningCount = RunningCount + 1
```

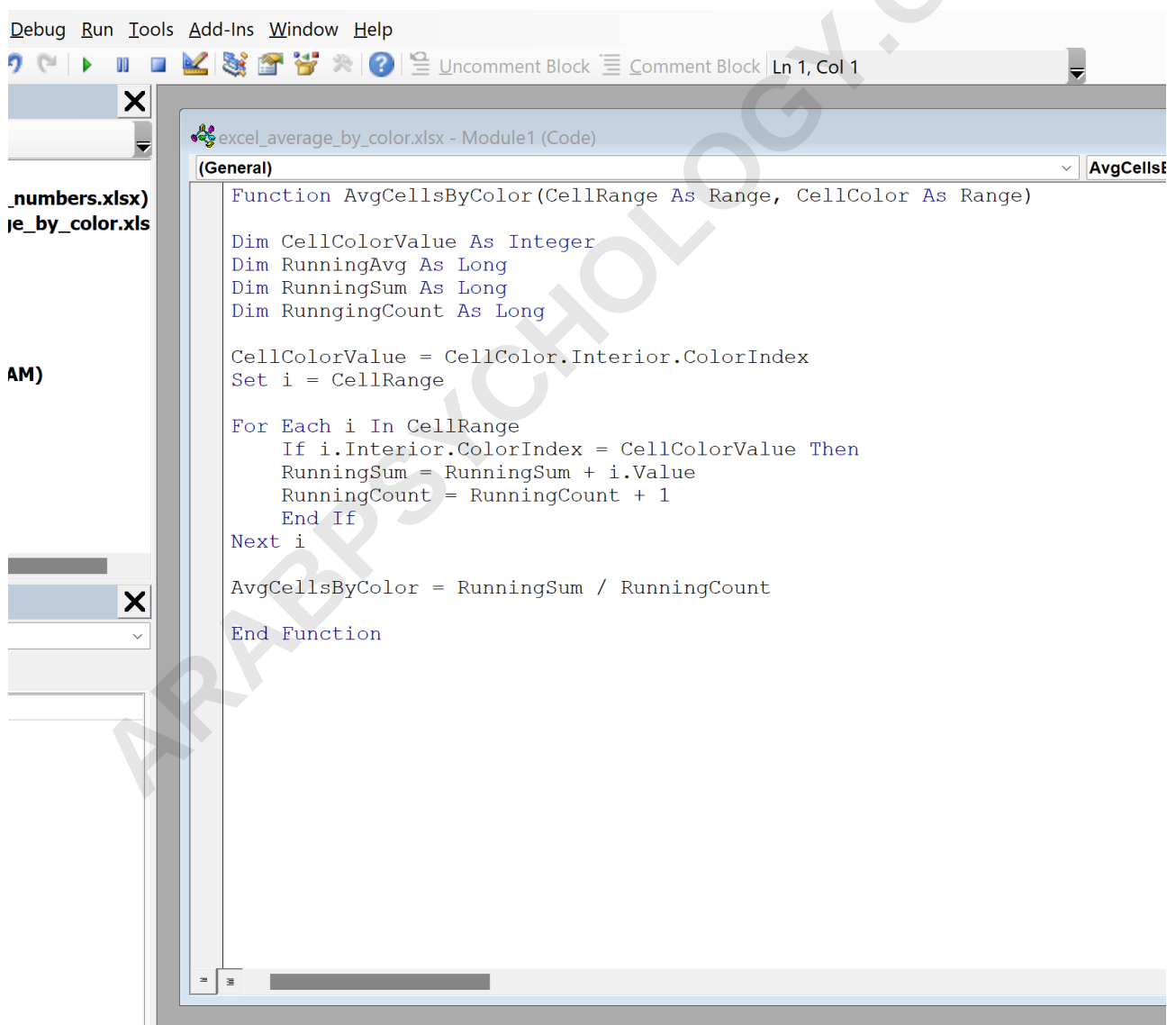
```
End If
```

```
Next i
```

```
AvgCellsByColor = RunningSum / RunningCount
```

```
End Function
```

The structure of the pasted code within the VBE Module should look like the following screenshot. After pasting the code, ensure you save the workbook as an **Excel Macro-Enabled Workbook (.xlsm)** to retain the Macro functionality.



Once the code is verified and saved, close the Visual Basic Editor to return to your main

worksheet. The new function, `AvgCellsByColor`, is now available for use just like any built-in Excel function.

Dissecting the AvgCellsByColor Function

Understanding how the custom Function operates provides clarity on why this VBA approach is necessary. The function iterates through the supplied data range, checking the cell's background color index against a reference color index.

Specifically, the code utilizes several key components:

`CellColorValue = CellColor.Interior.ColorIndex`: This line identifies the unique numerical index of the interior color of the reference cell (`CellColor`). Excel assigns a numerical index to every color palette entry, which is how VBA identifies colors reliably.

`Dim RunningSum As Long` and `Dim RunningCount As Long`: These lines declare variables necessary for aggregating the values and counting the matching cells, which is the core logic of calculating an average.

`For Each i In CellRange`: This loop structure compels the Macro to examine every single cell within the specified input range (`CellRange`).

`If i.Interior.ColorIndex = CellColorValue Then`: This conditional statement checks if the current cell's background color matches the reference color index. Only if this condition is met does the subsequent calculation occur.

The average is finally calculated outside the loop using `AvgCellsByColor = RunningSum / RunningCount`, ensuring that the result is returned as a single value to the Excel worksheet, completing the User-Defined Function.

Step 4: Implementing the Custom Function

The final step involves calling the newly created function, `AvgCellsByColor`, directly within your Excel worksheet. We must first set up a dedicated area to display the results and provide the colored reference cells.

To prepare for the calculation, fill in cells in a secondary column (e.g., cells **C2:C4**) with the exact background colors for which you wish to calculate the average. These cells serve as the "color swatch" references for our Macro.

Next, in the adjacent column (cell **D2**), enter the custom formula. Ensure you use absolute references (using the dollar sign `$`) for the data range (**`$A$2:$A$11`**) so that this range remains fixed when dragging the formula down, but use a relative reference for the color swatch (**`C2`**) so it shifts to **`C3`**, **`C4`**, and so on:

=AvgCellsByColor(\$A\$2:\$A\$11, C2)

After entering the formula in cell D2, utilize the fill handle to drag and apply this formula down to the remaining cells in column D. The formula will automatically calculate and display the average value corresponding to each specific background color defined in column C:

	A	B	C	D	E	F
1	Values					
2	20			17.66667		
3	13			17.75		
4	15			23		
5	18					
6	20					
7	24					
8	26					
9	30					
10	12					
11	15					
12						
13						
14						
15						

Observe the result displayed in column D. For example, the average value for all cells with a light green background is shown as **17.67**. We can quickly verify this result by manually summing the values associated with the light green cells (20, 13, and 20) and dividing by the count (3).

Manual Verification: $(20 + 13 + 20) / 3 = 53 / 3 = 17.67$. This confirms that our custom Function and the Macro setup are working precisely as intended, providing an accurate, automated solution for averaging by color.

Summary and Best Practices

In conclusion, while Excel lacks a native formula to aggregate data based on cell formatting, the problem is elegantly solved using a User-Defined Function created in VBA. This method transforms a typically manual and error-prone calculation into a simple, reusable formula accessible across your workbook.

The process requires four critical steps: data preparation, activating the Developer tab, defining the custom `AvgCellsByColor` Macro within a Module, and finally, deploying the function in the worksheet. This powerful technique is essential for analysts who rely on visual cues in their datasets and require fast, accurate quantitative summaries based on those cues.

ARABPSYCHOLOGY.COM