How to Easily Adjust Row Height in Excel VBA

Authored by

stats writer

November 19, 2025

Introduction to Controlling Row Height in VBA

Microsoft Excel is a powerful tool for data visualization and analysis. While manual formatting is straightforward, automating repetitive tasks is best achieved using VBA (Visual Basic for Applications). One common formatting requirement is adjusting the height of rows to optimize readability or standardize document appearance. By leveraging specific properties and methods within the VBA environment, we gain precise programmatic control over row dimensions.

To change the row height in VBA, you primarily utilize the **RowHeight property** to set the desired height (measured in points) for a selected row or range of rows on the active Worksheet. You can use this property to make a row taller or shorter depending on the numerical value you assign. Alternatively, you can use the **AutoFit** method to automatically adjust the row height based on the content of the cells, ensuring full visibility. Finally, methods targeting the `EntireRow.RowHeight` property can be applied to efficiently adjust the height for multiple rows simultaneously.

This comprehensive guide explores the primary techniques used to manipulate row height, moving beyond manual adjustments and offering robust, reusable code solutions. Whether you need to set fixed dimensions for aesthetic consistency or dynamically resize rows based on content, VBA provides the necessary tools. Understanding these methods is fundamental for any developer looking to master automated Excel formatting via custom macros.

You can use the following methods to change the row height in Excel using VBA:

Understanding the RowHeight Property in VBA

The core mechanism for setting a row's size explicitly is the RowHeight property. This property belongs to the **Range object** in Excel VBA and allows you to read or write the height of one or more rows, measured in points. A point, in this context, is a standard typographical measurement unit, where 72 points equal one inch. It is crucial to remember that the standard default row height in Excel is approximately **14.4 points**, depending on the default font size used in the workbook.

When setting the RowHeight property, you assign a numerical value to it. Assigning a larger value will make the row taller, while assigning a smaller value will make it shorter. If the cells in the targeted row contain content that exceeds the specified height, the text may appear truncated or hidden, especially if text wrapping is not enabled.

Unlike the AutoFit method, which dynamically calculates the required size, using the RowHeight property requires you to define the exact desired measurement. This is ideal when strict document standards or precise aesthetic requirements must be met, ensuring that all designated rows maintain a uniform appearance regardless of their underlying content.

Method 1: Setting a Fixed Height for a Single Row

The most straightforward application of row height adjustment involves targeting a single row on the active **Worksheet** and assigning it a specific height value. This is useful for highlighting header rows or separating data sections visually. We reference the row index directly using the `Rows` collection and then apply the RowHeight property.

In the example below, we are creating a procedure named `ChangeRowHeight` that specifically targets the third row (`Rows(3)`) and sets its height to **40 points**. This value is significantly larger than the default, ensuring the row stands out. This demonstrates the fundamental syntax for manual height setting.

```
Sub ChangeRowHeight()
Rows(3).RowHeight = 40
End Sub
```

This particular macro changes the height of the third row to a precise value of **40 points**. Remember that the default row height in Excel is typically **14.4 points**.

Method 2: Adjusting Height for a Range of Multiple Rows

When dealing with larger datasets or tables, it is necessary to apply the same fixed height to a contiguous range of rows rather than iterating through them individually. VBA allows us to specify a range using standard Excel notation (e.g., "1:5") within the `Rows` collection. When the RowHeight property is assigned a value on such a range, that value is applied uniformly to every single row within the selection.

This approach ensures visual uniformity and is much more efficient than looping through individual rows, making it the preferred method for bulk formatting operations. The range notation must be provided as a string argument to the `Rows()` method.

```
Sub ChangeRowHeight()
Rows("1:5").RowHeight = 40
End Sub
```

Executing this macro changes the height of each row falling within the designated range (rows one through five) to a fixed measurement of **40 points**.

Method 3: Dynamically Resizing Rows Using the AutoFit Method

While fixed height settings offer control, the most practical solution for ensuring all cell content is visible is using the AutoFit method. The AutoFit method automatically adjusts the row height(s) to perfectly accommodate the largest element (such as the tallest font or the longest wrapped text) present in any cell within that row or range of rows.

The key advantage of AutoFit is its reliance on actual content metrics. It calculates the necessary dimensions, preventing text truncation and dramatically improving data clarity, especially when column widths are constrained or variable text wrapping is utilized.

To use this method, you simply call `.AutoFit` on the targeted `Rows` range object. No argument or value assignment is needed, as the height calculation is handled entirely by Excel's rendering engine, making the code clean and highly efficient for formatting tasks.

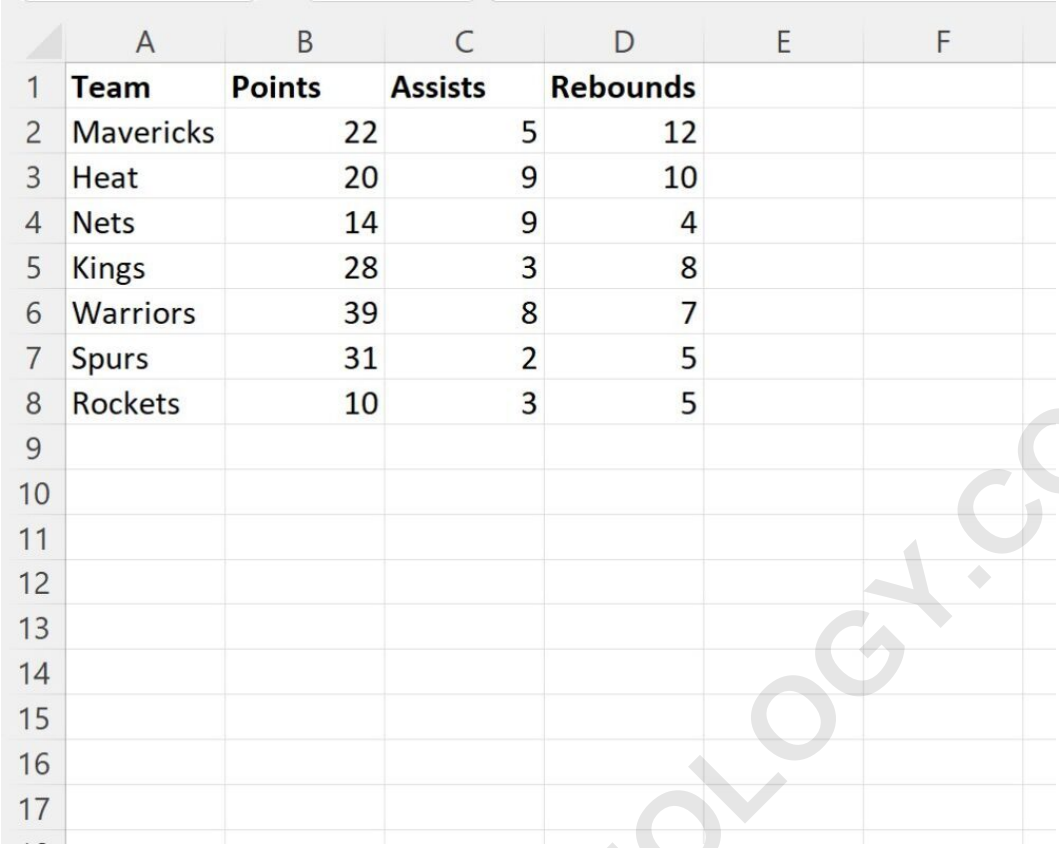
```
Sub ChangeRowHeight()
Rows("1:8").AutoFit
End Sub
```

This procedure utilizes the AutoFit method to automatically adjust the height of all rows from one through eight. The resulting height for each row will be precisely tall enough to display the entirety of the tallest text in that respective row.

Practical Implementation: Step-by-Step Examples

To demonstrate the functionality of these three distinct methods, we will use a standardized starting dataset. Note that this dataset includes varying amounts of text, and some rows might have truncated content at the default height, which highlights the difference between fixed setting and dynamic AutoFit.

The following image represents the initial dataset used across all examples:



	A	B	C	D	E	F
1	**Team**	**Points**	**Assists**	**Rebounds**		
2	Mavericks	22	5	12		
3	Heat	20	9	10		
4	Nets	14	9	4		
5	Kings	28	3	8		
6	Warriors	39	8	7		
7	Spurs	31	2	5		
8	Rockets	10	3	5		
9						
10						
11						
12						
13						
14						
15						
16						
17						

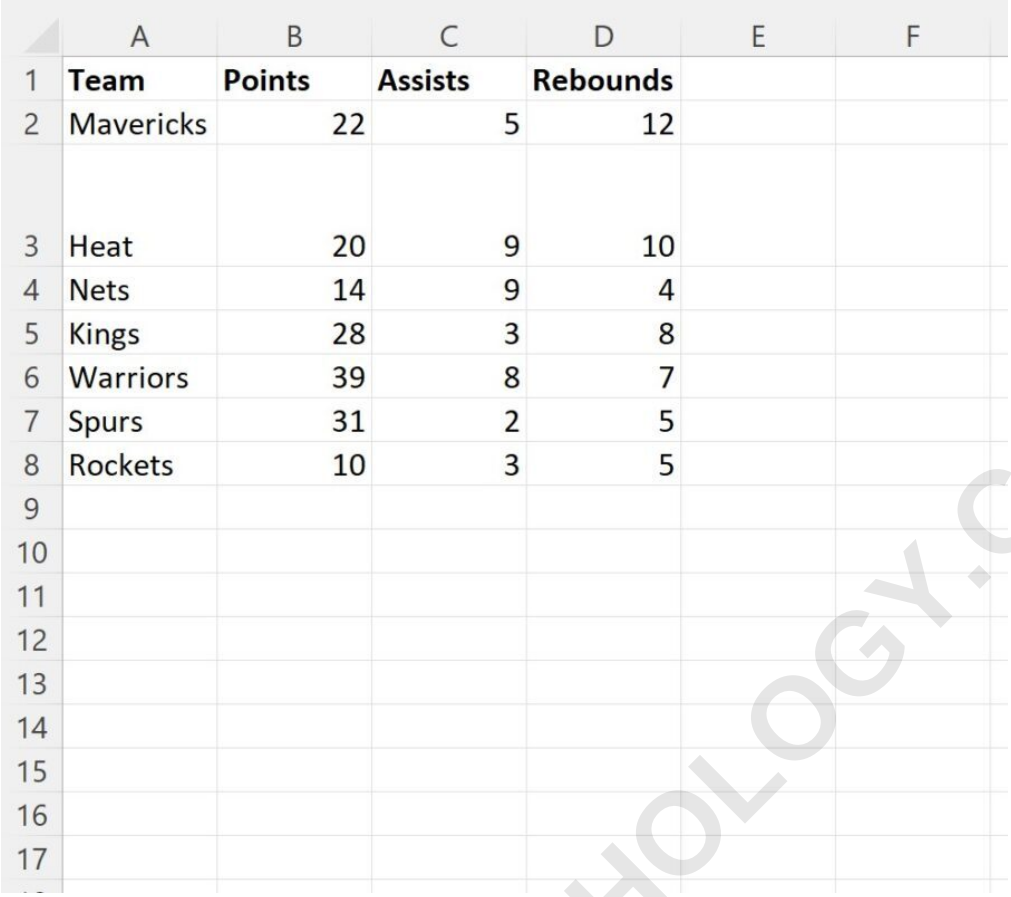
Example 1: Setting Height for a Single Row

This demonstration applies a specific, fixed height to row 3, showcasing how to manually override the existing default dimensions for a targeted row.

We create and execute the following procedure to change the height of the third row to 40 points:

```
Sub ChangeRowHeight()
Rows(3).RowHeight = 40
End Sub
```

After running the code, observe the output:



	A	B	C	D	E	F
1	**Team**	**Points**	**Assists**	**Rebounds**		
2	Mavericks	22	5	12		
3	Heat	20	9	10		
4	Nets	14	9	4		
5	Kings	28	3	8		
6	Warriors	39	8	7		
7	Spurs	31	2	5		
8	Rockets	10	3	5		
9						
10						
11						
12						
13						
14						
15						
16						
17						

Notice that only the height of the third row has been increased to 40 points while the height of all other rows remained the same.

Example 2: Setting Height for a Range of Rows

Here, we apply the fixed height setting to a range of rows to ensure uniformity across a section of the data. This is efficient for formatting entire tables.

We create the following macro to change the height of each row from one through five to 40 points:

```
Sub ChangeRowHeight()
Rows("1:5").RowHeight = 40
End Sub
```

When we run this macro, we receive the following output:

	A	B	C	D	E	F	G
1	**Team**	**Points**	**Assists**	**Rebounds**			
2	Mavericks	22	5	12			
3	Heat	20	9	10			
4	Nets	14	9	4			
5	Kings	28	3	8			
6	Warriors	39	8	7			
7	Spurs	31	2	5			
8	Rockets	10	3	5			
9							
10							
11							
12							
13							
14							

Notice that the height of each of the first five rows has increased uniformly to 40 points, demonstrating the effective use of the RowHeight property on a defined range.

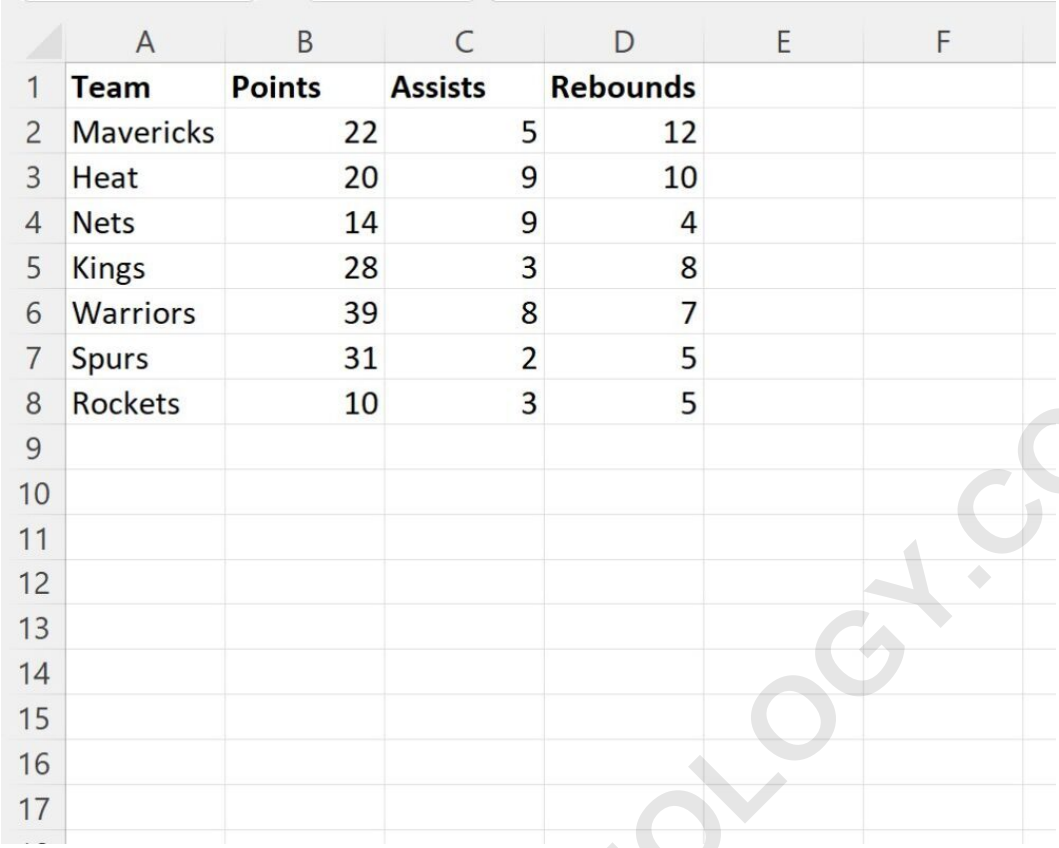
Example 3: Dynamic Adjustment using AutoFit

In this final example, we utilize `.AutoFit` to ensure all text, including wrapped text, is visible. This is crucial for reports where content length is variable.

We create the following macro to automatically adjust the height of the first eight rows:

```
Sub ChangeRowHeight()
Rows("1:8").AutoFit
End Sub
```

When we run this macro, we receive the following output:



	A	B	C	D	E	F
1	**Team**	**Points**	**Assists**	**Rebounds**		
2	Mavericks	22	5	12		
3	Heat	20	9	10		
4	Nets	14	9	4		
5	Kings	28	3	8		
6	Warriors	39	8	7		
7	Spurs	31	2	5		
8	Rockets	10	3	5		
9						
10						
11						
12						
13						
14						
15						
16						
17						

Notice that the height of each row has automatically been adjusted to be as tall as necessary to display the tallest text in each row. Row 5, which previously had truncated content, is now resized to fully display its wrapped text.

Best Practices and Troubleshooting When Adjusting Row Heights

While manipulating row dimensions in VBA is straightforward, following certain best practices ensures robust and professional code.

Ensure Text Wrapping is Enabled for AutoFit: The AutoFit method relies heavily on cell formatting. If text wrapping (`Range.WrapText = True`) is not active for cells containing long text, AutoFit will only adjust the height based on a single line of text, potentially leaving truncated content.

Handle Multiple Worksheets: Always qualify your range references with the specific Worksheet object (e.g., `Worksheets("DataSheet").Rows("1:10").RowHeight = 25`). If you omit the sheet qualification, the code will execute on the currently active sheet, which may lead to errors or unintended formatting changes.

Prioritize AutoFit over Fixed Heights for Variable Data: Use the RowHeight property only when absolute visual uniformity is mandatory. For dynamic data entry, AutoFit provides superior usability and scalability by adapting automatically to new content.

Resetting Manual Adjustments: If a user has manually dragged a row height, Excel sometimes marks that row, which can interfere with subsequent programmatic `.AutoFit` calls. If you suspect this is causing issues, sometimes you can force a reset by setting the height to a default value before running `.AutoFit`.

By mastering the `.RowHeight` property and the `.AutoFit` method, you gain complete command over the visual presentation of your Excel data, moving towards fully automated and professional report generation workflows.