Learning VBA: How to Change Column Width in Excel

Authored by

Mohammed looti

November 15, 2025

Introduction to Column Manipulation in VBA

Mastering the adjustment of column widths is a fundamental skill when automating data presentation within Excel using VBA (Visual Basic for Applications). While manually resizing columns is tedious, VBA provides powerful, efficient methods to ensure your spreadsheets are perfectly formatted, regardless of the data volume. This guide details the essential techniques for controlling column dimensions programmatically, ranging from setting fixed widths for single columns to dynamically fitting content across vast ranges.

The three primary approaches to modifying column size in Excel using VBA are focused on precision and efficiency. Whether you need to enforce a standard width across a report or ensure that all text is fully visible, understanding these methods is crucial. We will explore setting a fixed width for an individual column, applying a fixed width across a contiguous range of columns, and utilizing the highly effective **AutoFit** method for dynamic sizing.

Before diving into the practical examples, it is important to remember that column width in Excel is measured in units corresponding to the number of characters of the Normal style font that can fit into the column. For reference, the standard default width for any column in an Excel sheet is **8.29** units.

Understanding the ColumnWidth Property

The most direct way to control column size is by setting the ColumnWidth property. This property accepts a numeric value, defining the fixed size of the selected column or range. This is invaluable when standardized reporting templates require exact dimensions. We will examine two ways to apply this property: targeting a single column and targeting a range of columns.

Method 1: Change Width of One Column

To adjust the width of a single, specific column, you reference the column letter within the `Columns()` collection and assign the desired numeric value to the ColumnWidth property.

```
Sub ChangeColumnWidthSingle()
Columns("B").ColumnWidth = 20
End Sub
```

This particular macro changes the width of column B to a fixed value of **20** units, providing

significantly more horizontal space than the default setting.

Method 2: Change Width of Multiple Columns

When dealing with multiple columns that require the same fixed width, referencing a range dramatically simplifies the code. Instead of iterating through each column individually, you can specify the range (e.g., "B:D") directly within the `Columns()` collection.

```
Sub ChangeColumnWidthRange()
Columns("B:D").ColumnWidth = 20
End Sub
```

Executing this routine will set the width of all columns spanning from B through D to a uniform size of **20**. This method is highly efficient for applying consistent formatting across an entire section of your data table.

Automating Sizing with the AutoFit Method

While setting fixed widths is useful for consistency, often the content within your columns changes dynamically. In such scenarios, fixed widths can lead to truncated text (if the width is too small) or wasted space (if the width is too large). The solution is the AutoFit method.

Method 3: Auto Adjust Width of Multiple Columns

The AutoFit method is unique because it removes the guesswork, automatically adjusting the column width just enough to perfectly accommodate the longest cell entry within that column. This ensures maximum readability without manual intervention.

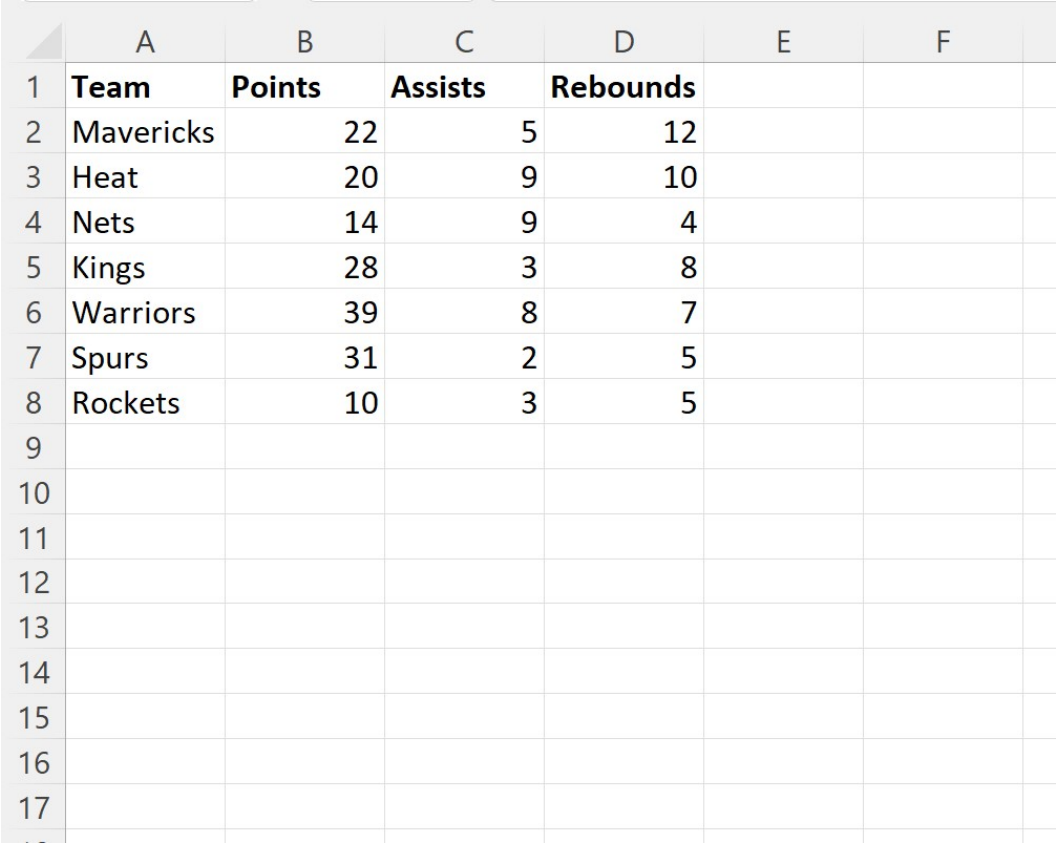
```
Sub AutoAdjustColumnWidth()
Columns("B:D").AutoFit
End Sub
```

This macro automatically adjusts the width of each column in the range from B to D to be exactly as wide as necessary to display the longest cell content contained within those respective columns. This is the preferred method for dealing with variable text data.

Practical Application: Preparing the Dataset

To illustrate the efficacy of these three VBA methods, we will work with a sample dataset. This dataset features varying lengths of text and numeric values, allowing us to clearly see the impact of fixed sizing versus dynamic fitting.

The following image displays our initial dataset in Excel, where all columns retain the default width of 8.29. Note that the text in Column A (Names) is currently truncated because the default width is insufficient.



	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						

Our goal across the following examples is to use VBA to reformat this table, ensuring all data is fully visible and presented neatly.

Practical Application: Changing a Single Column

In our first practical scenario, we aim to standardize the width of a single column that contains numeric data. Specifically, we will target Column B (the "Points" column) and set its width to 20.

This width is chosen arbitrarily to demonstrate how the ColumnWidth property fixes the dimension, regardless of the actual data size.

We create the following macro within the VBA editor:

```
Sub ChangeColumnWidth()
Columns("B").ColumnWidth = 20
End Sub
```

Upon running this macro, the specified code is executed, resulting in the following output visualization:

	A	B	C	D	E
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					
18					

Observe that only the width of column B (the "Points" column) has increased to a fixed width of 20 units. Crucially, the widths of all other columns, including Column A which still has truncated text, remain unchanged at the default 8.29. This demonstrates the precise, targeted control offered by the single-column width adjustment method.

Practical Application: Managing Column Ranges

For our second example, imagine we need to apply the same fixed width to multiple adjacent data columns--say, Columns B through D. Using the range notation within the `Columns()` collection allows us to achieve this formatting consistency with a single line of code, significantly improving script efficiency compared to applying the change column by column.

The necessary macro is structured as follows, targeting the range "B:D" and setting their collective ColumnWidth to 20:

```
Sub ChangeColumnWidth()
Columns("B:D").ColumnWidth = 20
End Sub
```

Executing this routine yields a visually consistent output across the targeted range:

	A	B	C	D
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				
18				
19				

This result confirms that the width of each column from B to D has been uniformly increased to 20.

Importantly, Column A, which falls outside the specified range, retains its original, inadequate width. This technique proves indispensable when standardizing data fields like numeric scores, dates, or short categorical variables across a report section.

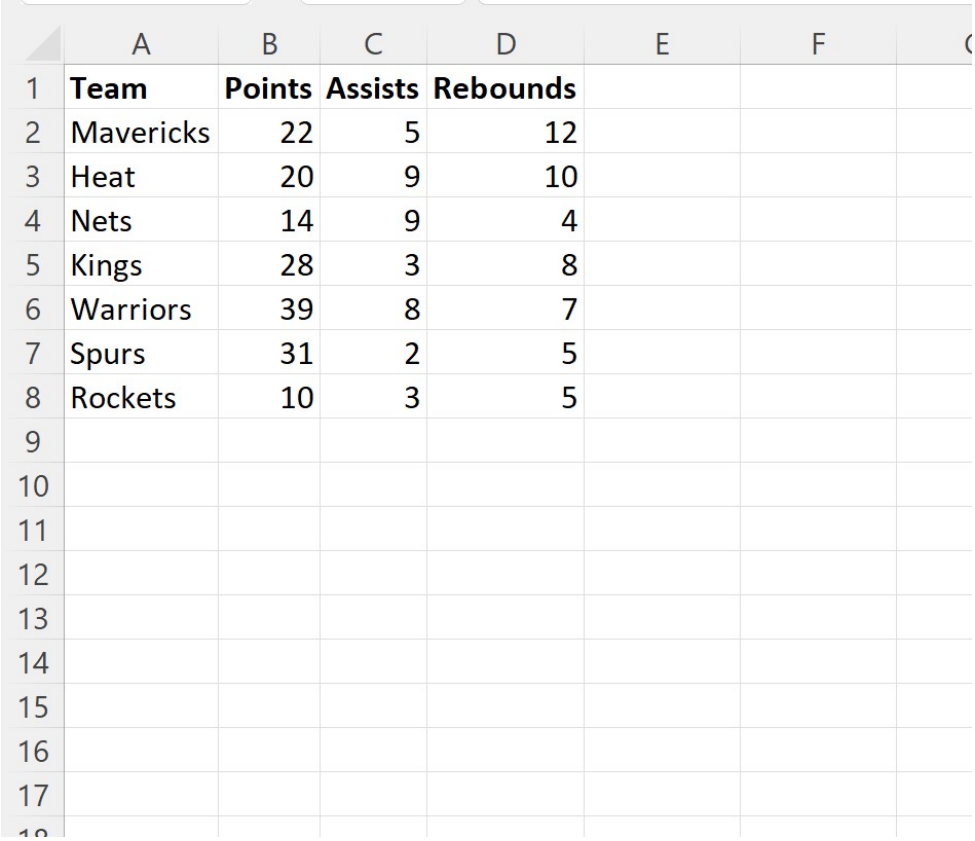
Practical Application: Dynamic Resizing using AutoFit

For data containing text or long descriptive names, fixed widths are impractical. This is where the **AutoFit** method provides the ideal solution. In this final example, we will apply dynamic resizing to all columns containing data (A through D) to ensure that even the longest cell entry--such as the full names in Column A--is entirely visible.

We create the following macro, utilizing the AutoFit method on the range "A:D":

```
Sub ChangeColumnWidth()
Columns("A:D").AutoFit
End Sub
```

The execution of this code dynamically assesses the content of each column individually and adjusts the width accordingly, producing the following clean 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 the distinct result: Column A has expanded significantly to accommodate the longest name, while Columns B, C, and D, which contain shorter numeric values, have only widened slightly beyond the default to fit their headers or largest number. This dynamic adjustment is often the most visually appealing and user-friendly approach for presenting complex datasets.

Best Practices and Important Considerations in VBA Resizing

While the three methods demonstrated provide complete control over column width, adopting certain best practices ensures robust and professional VBA code, especially when dealing with complex workbooks or shared environments.

One crucial consideration is scoping. When running column adjustments, always explicitly state the worksheet you are targeting. Although our examples implicitly target the active sheet, in a professional macro, you should use the structure `Worksheets("Sheet1").Columns("A:D").AutoFit` to prevent accidental formatting changes on the wrong sheet. Furthermore, remember that the AutoFit method applies to visible cells only; if you are running code on filtered data, only the visible rows and columns will be taken into account

for sizing calculations.

Finally, for extremely large datasets where performance is paramount, resizing the entire worksheet at once using `Cells.AutoFit` is generally fast, but if only a small portion of the sheet needs adjustment, always limit your scope to the minimum necessary range (e.g., `Columns("E:G").AutoFit`). Using the targeted approach conserves processing power and makes your code easier to debug and maintain.

Additional Resources

The following tutorials explain how to perform other common tasks in VBA, building upon the foundational knowledge of manipulating column properties:

Tutorial on adjusting row height in VBA.

Guide to formatting cells based on specific conditions.

Advanced techniques for looping through large datasets in Excel.