Select Every Other Column in Excel (With Example)

Authored by

stats writer

November 17, 2025

Microsoft Excel stands as an indispensable tool in the landscape of modern data management, facilitating everything from basic arithmetic to complex statistical modeling. For professionals working with extensive tables, the challenge often lies not just in calculation, but in efficient data subsetting and viewing. A common requirement in advanced data analysis is the need to isolate specific patterns of information, such as selecting every other column. This technique is invaluable when dealing with alternating metrics, periodic measurements, or simply when condensing a wide report for summary purposes.

Historically, achieving this kind of non-contiguous selection required tedious manual clicking combined with the Control key, or complex combinations of INDEX and COLUMN formulas. These methods were prone to errors and difficult to scale. However, modern versions of Excel have introduced specialized dynamic array functions that simplify these operations dramatically, offering a clean, formula-based approach to reshape and filter data ranges instantly.

This comprehensive guide will explore the precise methodology for selecting every other column within any given range using one of Excel's most powerful recent additions: the **CHOOSECOLS** function. We will break down the function's syntax, walk through a practical, real-world example using a sample dataset, and demonstrate variations that allow you to customize your output based on whether you need to start the selection from the first or second column. Understanding this function is crucial for anyone looking to streamline their data manipulation workflows in Excel.

Understanding the CHOOSECOLS Function

The core mechanism for efficient column selection in recent versions of Excel is the CHOOSECOLS function. This dynamic array function is specifically designed to return a subset of columns from a specified array or range, allowing users to define exactly which columns they wish to retain based on their numerical index (position). This capability moves beyond static cell referencing, enabling sophisticated data reshaping directly within a single formula. It is particularly useful for extracting non-contiguous columns, which is exactly what is required when selecting every other column.

The syntax for the **CHOOSECOLS** function is elegantly simple: `CHOOSECOLS(array, col_num1, , ...)`. The first argument, `array`, defines the complete range of data you are working with. Subsequent arguments, `col_num`, specify the column indices (positions) you want the function to return. When selecting every other column, the key lies in passing an appropriate array of column numbers that skips positions systematically. For example, to select columns 1, 3, 5, and 7 from a range spanning columns A through G, the indices must be explicitly listed.

Consider a scenario where your comprehensive data range spans from **A1:G11**. If the objective is to extract only the odd-numbered columns (the first, third, fifth, and seventh), the formula utilizes a structured array constant to define these indices. This method provides immediate results, spilling

the selected data into adjacent cells, thereby eliminating the need for manual copy-pasting or complex helper columns. The formula structure clearly dictates the desired output:

=CHOOSECOLS(A1:G11, {1,3,5,7})

This powerful application of **CHOOSECOLS** forms the foundation of our technique. The subsequent sections will now demonstrate this formula with concrete data, illustrating exactly how the output is generated and how to modify the approach for different starting points.

Case Study: Isolating Data on Basketball Players

To fully grasp the practical utility of the **CHOOSECOLS** function, let us apply it to a structured dataset containing statistical information related to basketball players. Imagine a scenario where the columns alternate between identifying information (Name, Team) and performance metrics (Points, Rebounds, Assists, etc.). Our starting data, spanning the range **A1:G11**, is presented below, encompassing seven distinct columns of data points.

	A	B	C	D	E	F	G
1	**Team**	**Points**	**Assists**	**Rebounds**	**Steals**	**Blocks**	**Turnovers**
2	Mavs	22	4	5	3	1	5
3	Spurs	19	9	5	0	0	3
4	Rockets	15	3	8	0	0	3
5	Kings	15	8	7	2	2	7
6	Warriors	29	12	9	3	1	5
7	Nets	24	10	12	1	5	1
8	Lakers	40	8	7	2	4	0
9	Thunder	35	3	10	2	2	3
10	Blazers	23	6	5	2	0	0
11	Jazz	33	2	14	1	0	2
12							
13							
14							
15							
16							
17							

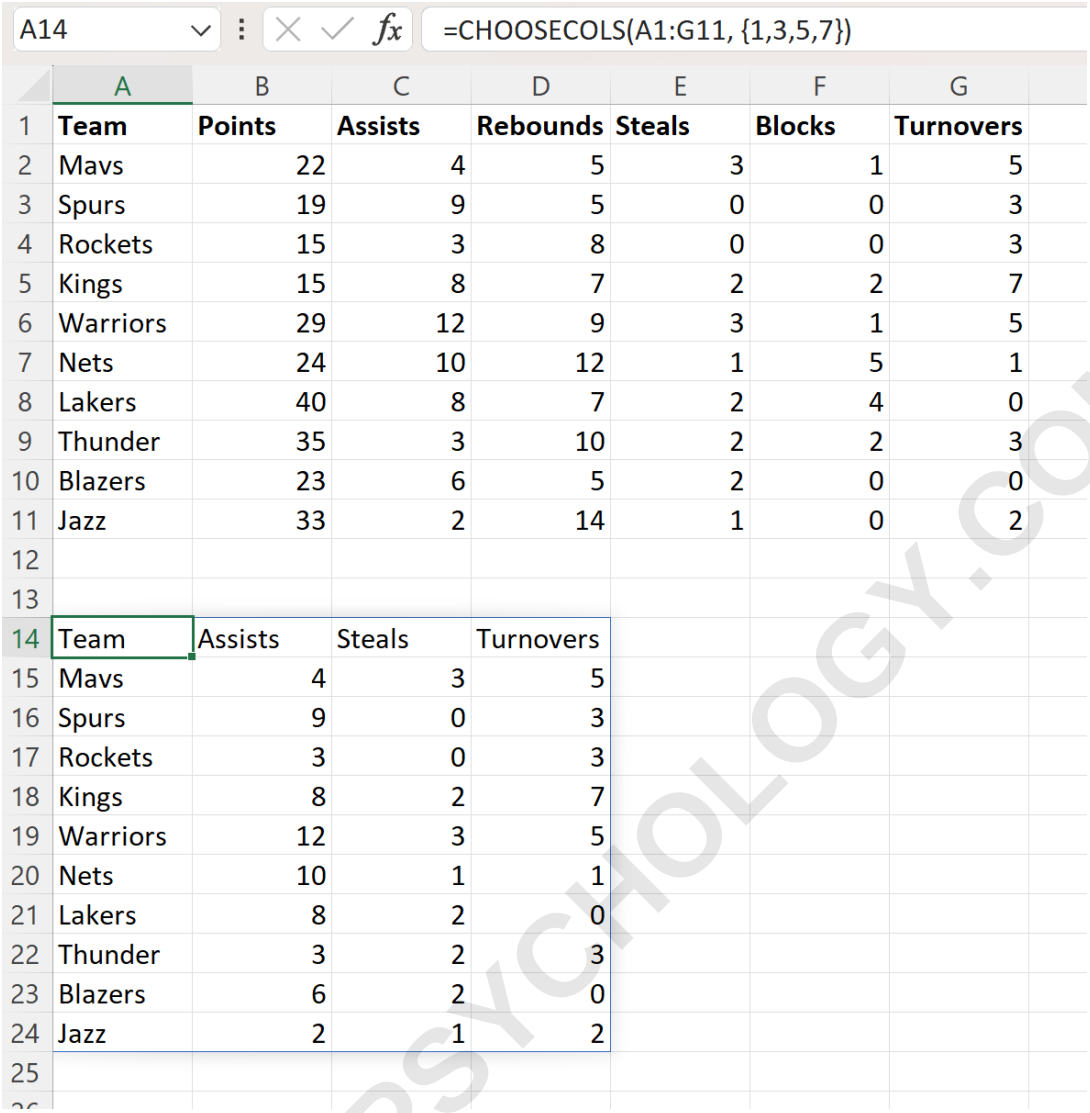
For reporting purposes, we might need to extract only the foundational identification columns and key summary statistics, effectively skipping intermediate metrics that are less critical for the current analysis. This means selecting columns 1, 3, 5, and 7, while ignoring columns 2, 4, and 6. This process of selecting every alternate column allows for immediate simplification of the output table

without disturbing the source data structure. We decide to input the resulting filtered table starting in cell **A14**.

The required formula must reference the entire source range (**A1:G11**) and utilize an array constant listing the precise column indices we wish to keep. Since we are starting with the very first column and selecting every other one, the indices are clearly 1, 3, 5, and 7. The execution of this formula is demonstrated in the subsequent visual representation, showing the formula entered into the target cell:

=CHOOSECOLS(A1:G11, {1,3,5,7})

When confirmed, the function processes the data within **A1:G11** and instantaneously returns a new, dynamic array containing only the specified columns, which then "spills" across the adjacent cells starting from **A14**. The visual result confirms the immediate impact of the **CHOOSECOLS** function on the data structure:



	A	B	C	D	E	F	G
1	**Team**	**Points**	**Assists**	**Rebounds**	**Steals**	**Blocks**	**Turnovers**
2	Mavs	22	4	5	3	1	5
3	Spurs	19	9	5	0	0	3
4	Rockets	15	3	8	0	0	3
5	Kings	15	8	7	2	2	7
6	Warriors	29	12	9	3	1	5
7	Nets	24	10	12	1	5	1
8	Lakers	40	8	7	2	4	0
9	Thunder	35	3	10	2	2	3
10	Blazers	23	6	5	2	0	0
11	Jazz	33	2	14	1	0	2
12							
13							
14	Team	Assists	Steals	Turnovers			
15	Mavs	4	3	5			
16	Spurs	9	0	3			
17	Rockets	3	0	3			
18	Kings	8	2	7			
19	Warriors	12	3	5			
20	Nets	10	1	1			
21	Lakers	8	2	0			
22	Thunder	3	2	3			
23	Blazers	6	2	0			
24	Jazz	2	1	2			
25							

Analyzing the Output: Starting from Column 1

The successful application of the formula `=CHOOSECOLS(A1:G11, {1,3,5,7})` yields a filtered table that perfectly isolates the intended data sequence. The output clearly demonstrates that the **CHOOSECOLS** function accurately indexed the specified range, extracting columns based on their relative position within the defined **A1:G11** block. Specifically, the function retrieved data corresponding to the first, third, fifth, and seventh internal columns, fulfilling the requirement of selecting every alternate column beginning with the first element of the source array.

This method is highly scalable. If the original data contained fifty columns, instead of manually listing all odd numbers, a more advanced dynamic approach using functions like **SEQUENCE** could be employed to generate the necessary indices dynamically. However, for smaller ranges, the explicit array constant provides the clearest and most straightforward implementation for selecting specific alternating columns. The resulting table is a clean subset, ready for immediate

presentation or further data analysis.

The resulting visual confirmation below validates the effectiveness of the formula, showing a simplified, four-column table derived from the original seven-column dataset. This process avoids data duplication and ensures that the summarized view remains dynamically linked to the original source data. Any change in the source range **A1:G11** will automatically cascade and update the output starting at **A14**.

	A	B	C	D	E	F	G
1	**Team**	**Points**	**Assists**	**Rebounds**	**Steals**	**Blocks**	**Turnovers**
2	Mavs	22	4	5	3	1	5
3	Spurs	19	9	5	0	0	3
4	Rockets	15	3	8	0	0	3
5	Kings	15	8	7	2	2	7
6	Warriors	29	12	9	3	1	5
7	Nets	24	10	12	1	5	1
8	Lakers	40	8	7	2	4	0
9	Thunder	35	3	10	2	2	3
10	Blazers	23	6	5	2	0	0
11	Jazz	33	2	14	1	0	2
12							
13							
14	Team	Assists	Steals	Turnovers			
15	Mavs	4	3	5			
16	Spurs	9	0	3			
17	Rockets	3	0	3			
18	Kings	8	2	7			
19	Warriors	12	3	5			
20	Nets	10	1	1			
21	Lakers	8	2	0			
22	Thunder	3	2	3			
23	Blazers	6	2	0			
24	Jazz	2	1	2			
25							
26							

Adjusting the Sequence: Starting from Column 2

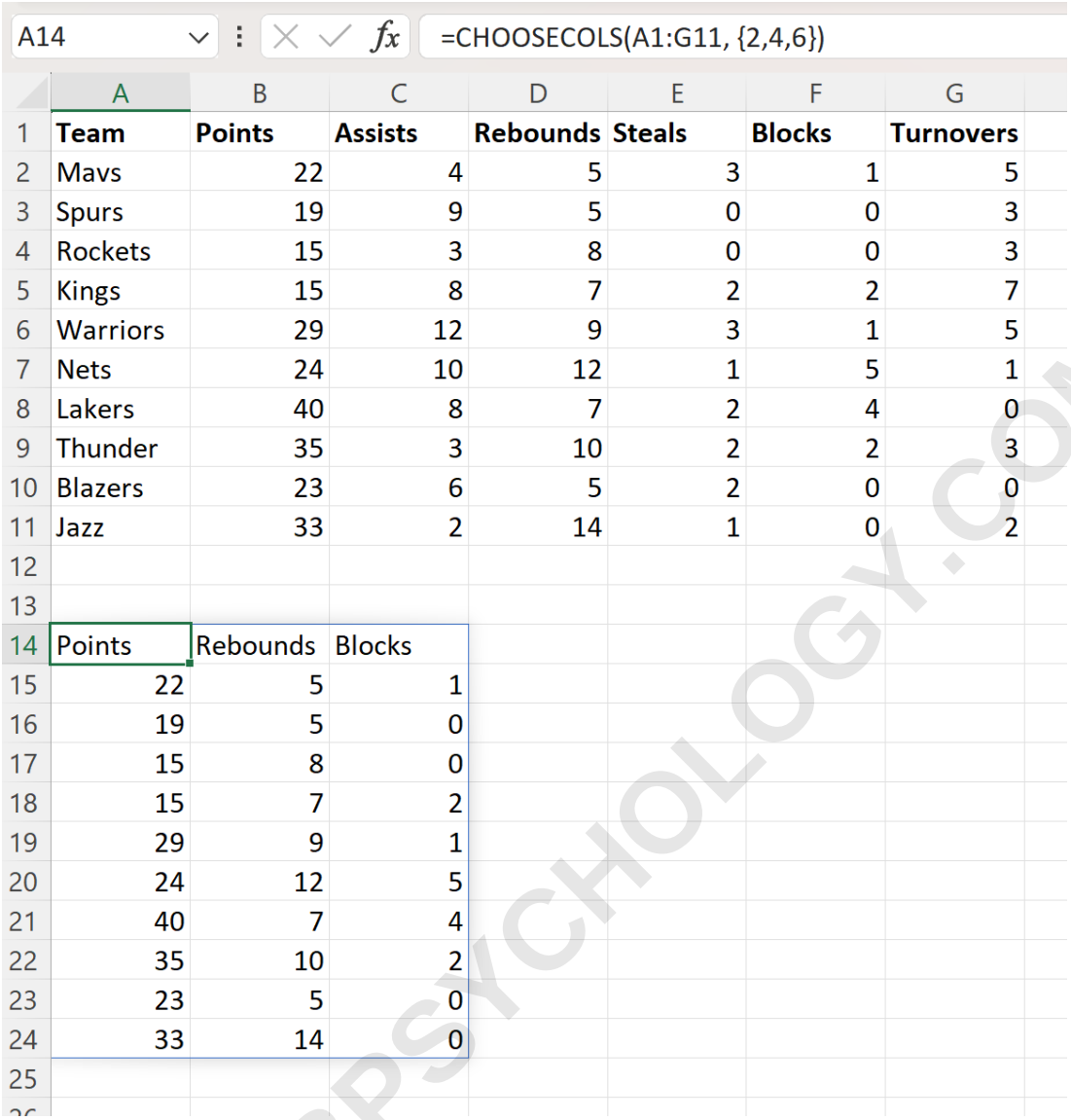
The versatility of the **CHOOSECOLS** function allows for immediate modification of the column sequence simply by adjusting the input array constant. In the previous example, we initiated the selection with the index 1, yielding columns 1, 3, 5, and 7. However, if the analytical requirement changes, and we need to isolate the even-numbered columns--perhaps corresponding to different

sets of alternating metrics--we simply shift our starting index to 2 and maintain the interval of two columns.

For the same range **A1:G11**, which encompasses seven columns, starting the selection at the second column means we are interested in columns 2, 4, and 6. Note that since the total number of columns is seven, the sequence naturally terminates at index 6. The elegance of the CHOOSECOLS function is demonstrated here, as the formula remains structurally identical; only the indices within the curly braces are modified to reflect the desired sequence:

=CHOOSECOLS(A1:G11, {2,4,6})

Executing this revised formula instantly generates a new output. This three-column table now consists exclusively of the data originally located in columns B, D, and F. This manipulation highlights the power of using explicit index arrays to target specific, repetitive data patterns efficiently. The visual result below confirms that every other column was selected, beginning specifically with the second column in the source range.



A14 =CHOOSECOLS(A1:G11, {2,4,6})

	A	B	C	D	E	F	G
1	**Team**	**Points**	**Assists**	**Rebounds**	**Steals**	**Blocks**	**Turnovers**
2	Mavs	22	4	5	3	1	5
3	Spurs	19	9	5	0	0	3
4	Rockets	15	3	8	0	0	3
5	Kings	15	8	7	2	2	7
6	Warriors	29	12	9	3	1	5
7	Nets	24	10	12	1	5	1
8	Lakers	40	8	7	2	4	0
9	Thunder	35	3	10	2	2	3
10	Blazers	23	6	5	2	0	0
11	Jazz	33	2	14	1	0	2
12							
13							

14	Points	Rebounds	Blocks
15	22	5	1
16	19	5	0
17	15	8	0
18	15	7	2
19	29	9	1
20	24	12	5
21	40	7	4
22	35	10	2
23	23	5	0
24	33	14	0
25			

This ability to easily switch between odd and even column selection provides immense flexibility when segmenting large financial models, scientific data logs, or any structure where key information alternates with auxiliary data points. This simplicity is a hallmark of modern Excel formula design.

The Importance of Dynamic Array Functions in Excel

The transition from traditional cell-by-cell formulas to dynamic array functions marks a major evolutionary step for Excel users. Functions like **CHOOSECOLS** eliminate the cumbersome requirement of dragging formulas or manually defining ranges for every row. When using **CHOOSECOLS**, the output automatically "spills" into the necessary number of cells, both horizontally and vertically, based on the dimensions of the source data and the requested columns. This minimizes formula complexity and significantly improves spreadsheet performance, especially

when handling large volumes of information.

Prior to the introduction of functions like CHOOSECOLS, achieving conditional or sequential column selection required nested functions, often involving combinations of **INDEX**, **MATCH**, and **COLUMN**, typically entered as Ctrl+Shift+Enter array formulas. These legacy methods were notoriously difficult to debug and maintain. The modern approach, encapsulated by **CHOOSECOLS**, simplifies the logic down to two core components: the source array and the column indices, making the intent of the formula clear and immediately understandable.

Furthermore, dynamic arrays enhance the stability of complex models. If the underlying dataset expands--for example, if a player is added to the basketball roster in row 12--the spilled output array automatically adjusts its height to incorporate the new data, without any modification to the formula itself. This robustness is critical for reliable and automated data analysis pipelines built within the spreadsheet environment.

Advanced Applications and Use Cases

While selecting every other column is a straightforward application, the power of **CHOOSECOLS** extends into more specialized use cases. This function can be seamlessly integrated with other dynamic array functions like **FILTER**, **SORT**, or **UNIQUE** to perform complex data manipulations in a single step. For instance, you might first use **FILTER** to select only players who scored above a certain threshold, and then use **CHOOSECOLS** on the filtered result to extract only specific metrics (i.e., every other column) for those high-performing individuals.

Another powerful application involves data transformation for compatibility with other systems. Often, raw data is exported in formats that include alternating metadata columns (e.g., Data Point, Status Flag, Data Point, Status Flag, etc.). By using **CHOOSECOLS** to select only the data points (indices 1, 3, 5, ...), you can quickly cleanse and prepare the data for import into databases or other analytical platforms that require a lean, metric-only structure. This step saves significant time compared to manual cleanup or scripting external tools.

The ability to handle column selection based on calculated indices is also pivotal. Instead of hardcoding `{1,3,5,7}`, users can generate these index arrays using formulas like **SEQUENCE** combined with **COLUMNS** or **COUNT**. This makes the column selection truly dynamic, adapting instantly if columns are inserted or deleted in the source range. This level of automation ensures that the formula remains relevant and functional even as the structure of the underlying dataset evolves.

Prerequisites and Compatibility Notes

It is important for users to understand the compatibility requirements for utilizing the

CHOOSECOLS function and other dynamic array capabilities discussed here. These functions are part of the modern calculation engine introduced by Microsoft. Specifically, **CHOOSECOLS** is available in Excel for Microsoft 365, Excel for Microsoft 365 for Mac, Excel for the web, and Excel for iPhone, iPad, and Android.

Users operating on older, perpetual license versions of Excel, such as Excel 2019, 2016, or earlier, will not have access to **CHOOSECOLS**. In these legacy environments, achieving the selection of every other column requires resorting to the older, more complex methods, such as utilizing the array combination of `INDEX(A1:G11, ROW(A1), N)` where N is derived from a helper column or a complex modulo calculation based on the **COLUMN** function. Therefore, access to a current Microsoft 365 subscription is a key prerequisite for implementing the clean, formula-based technique described in this guide.

Furthermore, when applying **CHOOSECOLS**, users should always ensure that the output range does not overlap with existing data. Because **CHOOSECOLS** spills its results dynamically, if the destination cells are not empty, the formula will return a `#SPILL!` error. Before entering the formula, verify that there are enough adjacent blank cells to accommodate the full height and width of the calculated result to ensure successful data extraction.

Summary of Best Practices

The **CHOOSECOLS** function offers a dramatically streamlined solution for a common data manipulation task: selecting every other column from a range in Excel. By understanding how to define the appropriate column index array--whether starting with `{1, 3, 5, ...}` for odd columns or `{2, 4, 6, ...}` for even columns--users can efficiently create dynamic subsets of their source data. This method avoids traditional complexities and ensures that data extraction is both rapid and robust.

To maximize efficiency, always remember that the column indices specified in the CHOOSECOLS function refer to the position within the defined `array`, not the absolute column letters of the spreadsheet. When used correctly, this function is a key component in advanced spreadsheet modeling and efficient data analysis, providing a powerful means to reshape and present data exactly as required.

For those seeking comprehensive details regarding parameters, error handling, and alternative indexing methods for the **CHOOSECOLS** function, consulting the official Microsoft documentation is highly recommended. Mastering this function is essential for anyone aiming to leverage the full capabilities of dynamic arrays in their daily Excel work.