Learn How to Select Every Other Column in Excel Using CHOOSECOLS

Authored by

Mohammed looti

November 10, 2025

Harnessing the CHOOSECOLS Function for Targeted Data Extraction in Excel

The ability to efficiently manipulate and reshape data is paramount in modern spreadsheet analysis. When working with extensive tables, analysts often encounter scenarios where they only require a subset of columns, particularly non-contiguous ones. Microsoft Excel provides a powerful, modern solution for this specific task: the **CHOOSECOLS function**. This function is a key component of Excel's dynamic array capabilities, allowing users to return specific columns from a given array or range without the need for complex nested formulas or manual data rearrangement. Utilizing **CHOOSECOLS** streamlines the process of extracting alternating columns, transforming what was once a tedious operation into a simple, elegant formula.

To demonstrate its utility, consider a common requirement: selecting every other column within a defined area, such as the range **A1:G11**. This action is frequently necessary when a source data table contains alternating helper columns, calculation fields, or irrelevant timestamps that need to be excluded from a summary report or further analysis. By supplying **CHOOSECOLS** with an array of column indices, we instruct Excel precisely which data streams to retrieve. The function then dynamically spills the resulting array into adjacent cells, providing an immediate, clean output that preserves data integrity while discarding unwanted information.

The basic structure for achieving this specific odd-column extraction (1st, 3rd, 5th, and 7th columns) is concise and highly effective. This formula explicitly maps out the positional indices we wish to keep, allowing for granular control over the data selection process. The use of an array constant (the curly braces containing the indices) is central to defining the selection pattern.

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

The following comprehensive examples will walk through the application of this formula, illustrating exactly how to employ the **CHOOSECOLS** function to selectively isolate alternating columns, thereby drastically improving the efficiency of data preparation and analysis workflows.

Understanding the Syntax and Application of CHOOSECOLS

The structure of the **CHOOSECOLS** function is straightforward, requiring two fundamental arguments. The first argument specifies the source data, which can be either a traditional cell range reference (like **A1:G11**) or a named array. This argument is crucial as it defines the universe of data from which columns will be selected. It is imperative that this range is accurately defined to ensure the indices provided in the second argument correspond correctly to the columns within that specific data block.

The second argument, often referred to as `col_num`, is a list or array of numerical indices that

dictate which columns should be returned from the source array. These indices are relative to the input array, meaning the first column in the source range is always index 1, the second is index 2, and so on. To select non-contiguous columns, such as every other column, we supply this argument with an array constant--a sequence of numbers enclosed in curly braces (e.g., **{1, 3, 5, 7}**). Furthermore, **CHOOSECOLS** allows for negative indices, which count backwards from the last column in the array, offering additional flexibility for complex data structuring needs.

For our purpose of selecting every other column, we must carefully construct the array constant to skip the required columns. If our input range has seven columns (A through G), and we start with the first column (A), we must skip the second (B), select the third (C), skip the fourth (D), and so forth. This results in the sequential indices 1, 3, 5, and 7. The output of the **CHOOSECOLS** function is a dynamic array, meaning the result automatically "spills" across the necessary number of rows and columns, eliminating the need to manually copy the formula or pre-define the output area. This dynamic nature is one of the greatest advantages of using modern Excel functions for data manipulation.

Case Study 1: Selecting Odd-Numbered Columns from a Dataset

To solidify the understanding of **CHOOSECOLS**, let us examine a practical scenario. Suppose we are analyzing a dataset containing information on various basketball players. This dataset includes seven columns: Player Name, Position, Height, Weight, Games Played, Points Per Game (PPG), and Rebounds Per Game (RPG). For a specific report, we only need the Player Name, Height, Games Played, and Rebounds Per Game--which conveniently correspond to the odd-numbered columns (1, 3, 5, and 7) of the source range.

The initial dataset, spanning the range **A1:G11**, is structured as follows:

	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							

Our objective is to extract every other column, starting with the first one. This task requires careful selection of the indices. If the source range **A1:G11** contains seven columns, we need to specify all four odd positions: 1, 3, 5, and 7. We will place the resulting formula in an empty cell, such as **A14**, allowing the results to spill downwards and across the requisite number of columns.

We implement the following formula in cell **A14** to execute the extraction:

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

This approach is fundamentally robust because it is entirely non-destructive. The original data remains untouched, and the output is a live, dynamic reference. Should the source data in **A1:G11** change, the spilled array beginning at **A14** will automatically update, ensuring the derived summary remains current and accurate. This characteristic makes **CHOOSECOLS** exceptionally useful in creating dashboard views or summary tables that rely on frequently updated primary datasets. The following illustration demonstrates the precise input of the formula within the Excel environment:

A14 =CHOOSECOLS(A1:G11, {1,3,5,7})

	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							

Visualizing and Interpreting the Results

Upon entering the **CHOOSECOLS** formula, the function immediately processes the request and returns the specified columns. Specifically, the columns corresponding to positional indices **1** (Player Name), **3** (Height), **5** (Games Played), and **7** (RPG) are extracted and displayed dynamically, starting from cell **A14**. The columns that were skipped--Position (2), Weight (4), and PPG (6)--are effectively excluded from the resulting array, achieving the goal of selecting every other column starting from the beginning of the range.

The resulting output confirms that the powerful **CHOOSECOLS** function successfully isolated the desired data streams. This result set is a perfect, filtered representation of the original dataset, containing only the columns necessary for the intended downstream analysis or reporting. The efficiency provided by this single formula cannot be overstated, particularly when compared to older methods which might involve manual copying, pasting, or the use of more cumbersome array

formulas like `INDEX` combined with `COLUMNS`.

The resulting filtered table is displayed below, highlighting the successful extraction of the alternating columns:

	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							

It is important to reiterate that since the first column index specified in the array constant was **1**, the function established a pattern of extraction starting with the first column and proceeding by skipping one column thereafter. Understanding the starting index is critical, as modifying this single number fundamentally changes the extraction pattern, allowing for flexible adaptation to various data restructuring requirements, as demonstrated in the next case study.

Case Study 2: Selecting Even-Numbered Columns

While the previous example focused on selecting odd-numbered columns, data requirements often necessitate the opposite: selecting every other column starting with the second column. If, using

the same basketball dataset (**A1:G11**), we now needed to focus solely on the positional and performance metrics, we would need columns 2, 4, and 6, corresponding to Position, Weight, and Points Per Game (PPG).

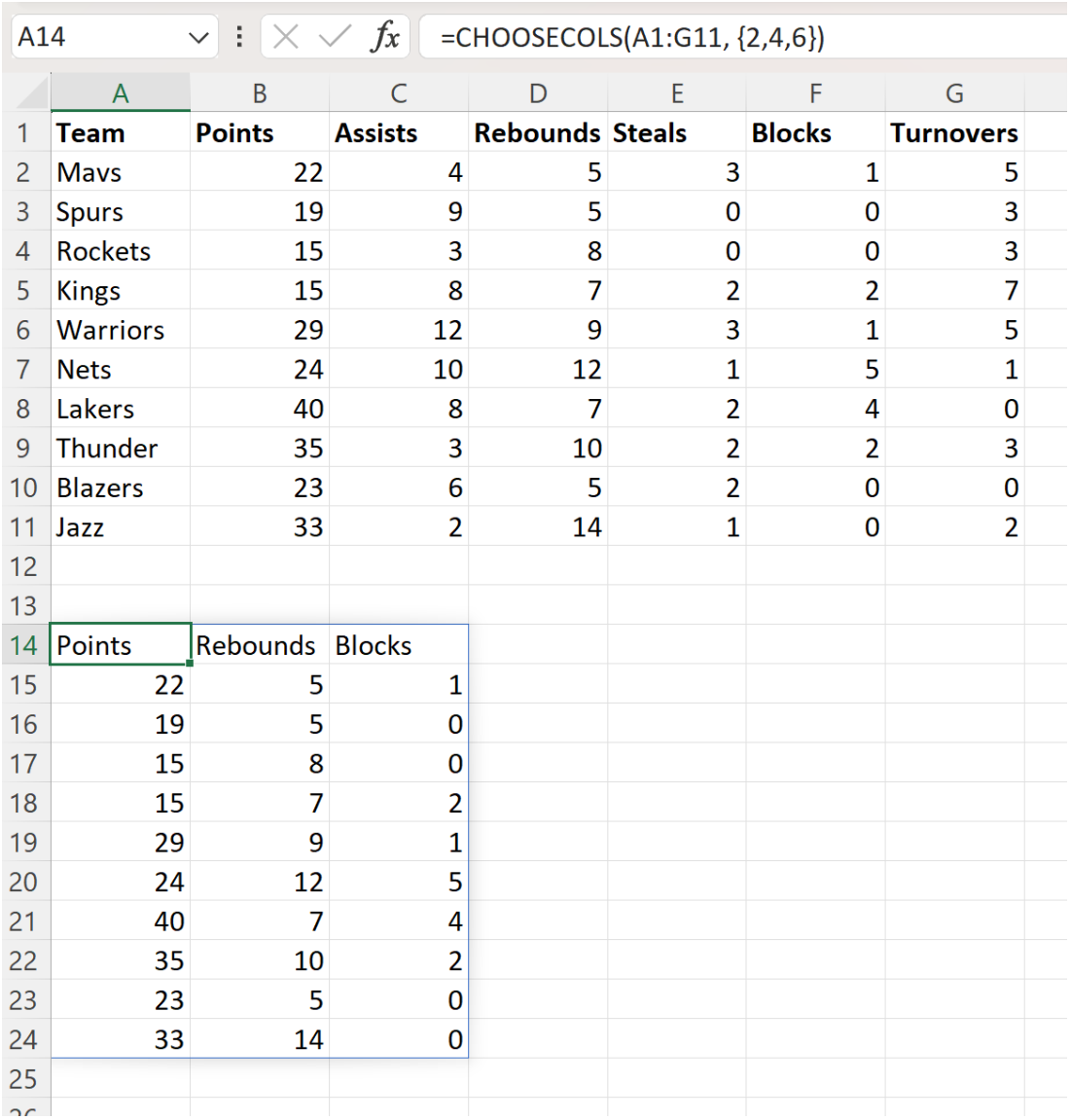
To achieve this alternate selection pattern, we simply adjust the array constant within the **CHOOSECOLS function**. Instead of starting with 1, we begin with 2 and continue the sequence of alternating indices: 2, 4, and 6. Note that because our source range **A1:G11** only contains seven columns, the highest even index is 6, resulting in a slightly shorter array constant compared to the odd-numbered selection.

The modified formula to return every other column starting with the second column is as follows:

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

Implementing this change demonstrates the versatility of the function. By merely changing the indices within the curly braces, we pivot the selection mechanism instantly. The function maintains the integrity of the selected data while efficiently discarding the columns that are not required (Player Name, Height, Games Played, and Rebounds Per Game). The resulting output, starting from cell **A14**, displays the three selected even-numbered columns: Position, Weight, and PPG.

The subsequent screenshot illustrates the execution and result of this formula in Excel:



	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							

Observe that every other column in the source range is selected, commencing precisely from the second column. This flexibility highlights why the **CHOOSECOLS function** is an indispensable tool for analysts routinely faced with restructuring complex tables. It offers a clear, declarative method for column selection, far surpassing the complexity and fragility of older array methods used for similar tasks.

Advanced Considerations and Efficiency

While the examples above utilize static array constants (e.g., **{1,3,5,7}**), more advanced scenarios may require dynamically generating the column indices. For large datasets where the number of columns changes frequently, manually typing out hundreds of indices is impractical. In such cases, the **CHOOSECOLS** function can be combined with other dynamic array functions, such as **SEQUENCE** and **FILTER**, to automatically generate the required alternating indices based on the

total column count of the input array.

For instance, to dynamically generate the odd indices for a range with *N* columns, one could use a combination of **SEQUENCE(N)** to get all indices, followed by a filtering mechanism (like **MOD(index, 2)=1**) to isolate only the odd numbers. This approach ensures that the formula remains robust even when the source data structure evolves, eliminating the need for manual updates to the index array. This level of automation is critical in professional reporting environments where data integrity and efficiency are paramount concerns.

The introduction of the **CHOOSECOLS function** marks a significant improvement over traditional methods for column extraction in Excel. Previously, users might have relied on complex `INDEX` formulas combined with `ROW` or `COLUMN` functions, often requiring Ctrl+Shift+Enter entry and lacking the intuitive dynamic spilling behavior. **CHOOSECOLS** simplifies data selection, making the process faster, less error-prone, and significantly easier to audit and maintain. Its utility extends far beyond simple alternating column selection, providing a versatile foundation for any task requiring the selective reorganization of columns within an array or range.

Additional Resources and Further Learning

Mastering the **CHOOSECOLS** function is merely one step in leveraging the full power of Excel's dynamic array capabilities. Exploring related functions and techniques will further enhance your ability to manipulate and analyze data efficiently. For those seeking to deepen their understanding of data transformation in Excel, several tutorials and official documentation resources are highly recommended.

For comprehensive details regarding the syntax, limitations, and advanced usage scenarios of the **CHOOSECOLS** function, users should consult the official Microsoft documentation, which provides granular technical details essential for expert application.

Beyond selecting columns, similar dynamic array functions exist for row manipulation (e.g., **CHOOSEROWS**) and general data reshaping (e.g., **FILTER** and **SORT**). Integrating these tools allows for the creation of sophisticated, fully automated data models directly within the spreadsheet environment.

Explore tutorials explaining how to perform other common operations in Excel:

Learning how to use the **CHOOSEROWS** function for selective row extraction.

Understanding the application of the **FILTER** function for criteria-based data subsetting.

Implementing the **SEQUENCE** function to dynamically generate numerical lists for complex indexing.