How to extract a year from a date in Google Sheets?

Authored by

stats writer

November 21, 2025

Extracting specific components from a date is a fundamental requirement in data analysis and management within spreadsheet environments. Whether you are sorting historical records, calculating elapsed time, or preparing data for reporting, isolating the year is often the crucial first step. In Google Sheets, this process is streamlined and efficient, relying primarily on a dedicated function designed specifically for this task. Understanding how to correctly implement this function ensures data integrity and accelerates your workflow, moving beyond manual data entry and manipulation.

The primary method for extracting the year utilizes the built-in **YEAR function**. This function is straightforward: it requires a valid date serial number or a cell reference containing a date, and in return, it provides the four-digit year corresponding to that date. For instance, if you were working with the date May 13th, 2016, the formula `=YEAR("5/13/2016")` would immediately return `2016`. While this simple application demonstrates its power, real-world use often involves applying this function across large datasets housed within specific cell references.

Mastering date extraction is essential for analysts who regularly handle time-series data or historical information. By converting complex date formats into simple numerical years, you enable powerful capabilities such as grouping data by year for summary statistics, creating pivot tables to track annual trends, or applying conditional formatting based on specific time periods. This guide will provide a detailed look into using the YEAR function, illustrating practical examples, and exploring advanced techniques for combined date component extraction in Google Sheets.

Understanding the Core YEAR Function in Google Sheets

The simplest and most direct way to isolate the year component from a complete date value in Google Sheets is by employing the YEAR function. This is a crucial temporal function designed to parse the underlying date serial number--which represents the number of days since a specific epoch date (usually January 1, 1900)--and extract only the calendar year. Its syntax is remarkably simple, requiring just one argument: the date you wish to analyze.

The structure of the function is `=YEAR(date)`. The `date` argument can be supplied in several formats. Most commonly, it is a direct reference to a cell containing a valid date (e.g., **A1**), but it can also be a date entered directly as a string (e.g., `"2023-10-27"`) or the result of another function that returns a date serial number (like `TODAY()` or `DATEVALUE()`). When referencing a cell, the formatting of the cell content must be recognized by Google Sheets as a valid date; otherwise, the function will likely return an error.

To illustrate the basic application, consider that we have a date stored in cell **A1**. We can utilize the following formula in any other cell (e.g., **B1**) to immediately return the four-digit year. This efficiency is the reason the YEAR function is preferred over complex string manipulation formulas, as it directly handles the underlying numerical date format.

You can use the following formula to extract the year from a date in Google Sheets:

=YEAR(A1)

This particular formula will return the year from the date in cell **A1**.

Practical Example: Extracting the Year from a Data Column

When working with real-world data, it is rare to analyze only a single date. Typically, data analysts deal with extensive columns of dates, such as transaction dates, enrollment dates, or project start dates. The true power of spreadsheet functions like the YEAR function lies in their ability to be efficiently applied to an entire column, saving significant time compared to manual entry or individual calculations.

Let us assume we have a table containing raw date information in Column A, starting from cell **A2** downwards. Our goal is to create a new, clean column (Column B) dedicated solely to the year of each corresponding entry. This setup allows for non-destructive data manipulation--the original date column remains untouched, ensuring source data integrity while providing the necessary derived value for analysis.

The following example demonstrates this common scenario. We begin by setting up the initial formula in the first target cell, **B2**, referencing the date located immediately to its left in **A2**. Once the formula is confirmed to be working correctly for the first row, we can leverage the powerful autofill features inherent in Google Sheets to quickly populate the rest of the column.

The following example shows how to use this function in practice. Suppose we have the following list of dates in Google Sheets:

	A	B	C	D	E
1	**Date**				
2	1/1/2014				
3	3/14/2016				
4	3/22/2017				
5	5/1/2018				
6	5/2/2018				
7	10/12/2019				
8	8/12/2019				
9	1/1/2020				
10	12/3/2021				
11	11/5/2022				
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					

We can type the following formula into cell **B2** to extract the year from the date in cell **A2**:

=YEAR(A2)

Scaling the Extraction: Dragging and Filling Formulas

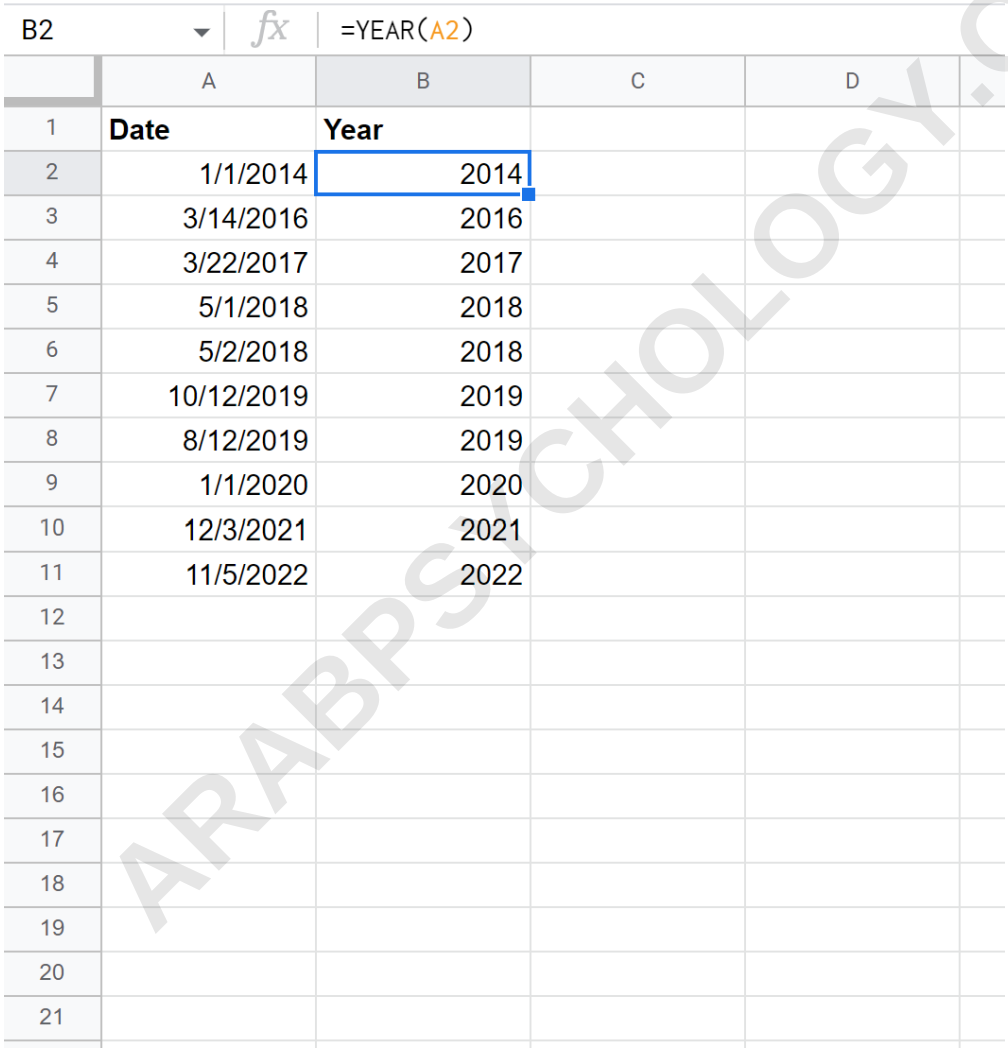
After implementing the initial **YEAR function** in cell **B2**, the subsequent step is to efficiently apply this formula down the entire column corresponding to our dataset. Spreadsheet programs like Google Sheets use relative referencing by default. This means that when a formula like `=YEAR(A2)` is copied down one row, the cell reference automatically adjusts to `=YEAR(A3)`, then `=YEAR(A4)`, and so on. This mechanism is crucial for handling large volumes of data without manually rewriting hundreds or thousands of individual formulas.

The primary method for scaling this operation is the "drag and fill" feature. By hovering the cursor over the bottom-right corner of cell **B2** until it turns into a crosshair, the user can click and drag the formula down to the last row where dates exist in Column A. Alternatively, in modern spreadsheet

software, a simple double-click on that crosshair often triggers an automatic fill down to the last continuous row of data in the adjacent column (Column A).

Executing this fill operation instantly transforms the raw date column into a dedicated year column, facilitating immediate data segmentation. The resulting column (Column B) now contains pure numerical values representing the year, which are ready for aggregation, sorting, or use as a grouping variable in complex analyses. It is important to verify the last few entries to ensure the formula correctly propagated through the entire range of data.

We can then drag and fill this formula down to each remaining cell in column B:



	A	B	C	D
1	**Date**	**Year**		
2	1/1/2014	2014		
3	3/14/2016	2016		
4	3/22/2017	2017		
5	5/1/2018	2018		
6	5/2/2018	2018		
7	10/12/2019	2019		
8	8/12/2019	2019		
9	1/1/2020	2020		
10	12/3/2021	2021		
11	11/5/2022	2022		
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				

The values in column B display the year of the date in the corresponding cell in column A.

Advanced Extraction: Combining Month and Year using TEXT

While the YEAR function is excellent for numerical extraction, analytical requirements often

demand a more descriptive output, such as displaying the year alongside the abbreviated month name (e.g., "Oct 2023"). Extracting and formatting combined date components requires the use of the versatile TEXT function, which converts a number (or a date serial number) into a formatted text string based on a specified format code.

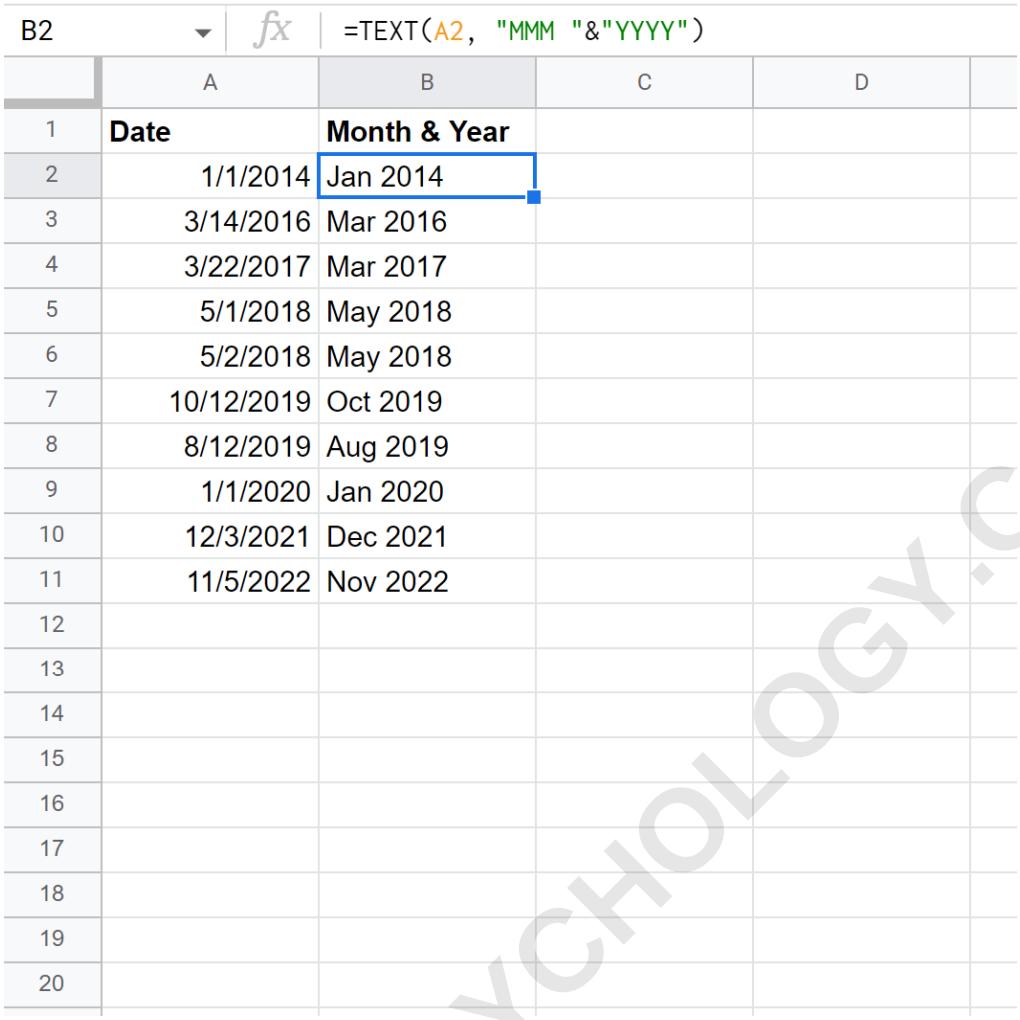
The syntax for the **TEXT function** is `=TEXT(value, format)`. In this context, the `value` is the date cell reference (e.g., **A2**), and the `format` is a string defining how the date should appear. To achieve the "Month Year" format, we use specific codes: `MMM` for the three-letter month abbreviation and `YYYY` for the four-digit year. We must also concatenate these elements if we want to include custom separators or spaces, though often the format string handles the spacing internally.

The provided example demonstrates a powerful concatenation approach using the ampersand operator (`&`) to combine two distinct format strings, although a simpler single format string (e.g., `"MMM YYYY"`) would typically suffice for direct formatting. Using the formula structure below ensures that the output is not a number but a human-readable text string, suitable for reports and labels where numerical sorting is not the primary requirement.

If you'd like to extract the month along with the year for each date, you can type the following formula into cell **B2**:

=TEXT(A2, "MMM "&"YYYY")

We can then drag and fill this formula down to each remaining cell in column B:



	A	B	C	D
1	**Date**	**Month & Year**		
2	1/1/2014	Jan 2014		
3	3/14/2016	Mar 2016		
4	3/22/2017	Mar 2017		
5	5/1/2018	May 2018		
6	5/2/2018	May 2018		
7	10/12/2019	Oct 2019		
8	8/12/2019	Aug 2019		
9	1/1/2020	Jan 2020		
10	12/3/2021	Dec 2021		
11	11/5/2022	Nov 2022		
12				
13				
14				
15				
16				
17				
18				
19				
20				

The values in column B now display the month and year of the date in the corresponding cell in column A.

Leveraging Format Codes for Custom Output

The true versatility of the TEXT function lies in the extensive set of format codes available for dates and times. While `YYYY` and `MMM` are commonly used for year and month, knowing the full range of codes allows for highly specific and customized output strings, tailored exactly to reporting requirements. For instance, you might need the full month name spelled out, or perhaps only the last two digits of the year.

Here is a breakdown of essential format codes related to year and month extraction:

`YY`: Displays the year as a two-digit number (e.g., 23).
`YYYY`: Displays the year as a four-digit number (e.g., 2023).
`M`: Displays the month as a number (1 through 12).

MM: Displays the month as a two-digit number (01 through 12).
MMM: Displays the month as a three-letter abbreviation (Jan through Dec).
MMMM: Displays the month as a full name (January through December).

By combining these codes, you can craft powerful expressions. For example, `=TEXT(A2, "MMMM YYYY")` would return "October 2023".

It is critical to remember that the output of the **TEXT function** is always a text string, not a numerical value. While this is perfect for display purposes, it restricts subsequent mathematical operations or specific filtering that relies on the numerical properties of a date or year. If you need the year numerically (for calculations or sorting), you must revert to the **YEAR function** or convert the output back to a numerical type using functions like **VALUE()**, though this is rarely necessary if the initial goal was purely extraction and display.

Alternative Methods: ArrayFormula and QUERY

For users managing very large datasets, relying on the drag-and-fill method can sometimes be inefficient, particularly if the data source is constantly growing or being updated. Google Sheets offers more dynamic, single-cell solutions, primarily through the use of ArrayFormula and the powerful QUERY function, which can process an entire range of cells using one initial formula.

The **ArrayFormula** allows the output of a standard function, like **YEAR**, to spill across multiple rows or columns without needing to manually copy the formula. For example, to extract the year from the entire range A2:A100, you would enter `=ARRAYFORMULA(YEAR(A2:A100))` into cell **B2**. This single formula instantly populates all necessary cells in Column B. This method is highly recommended for maintainability and scalability, as any new dates added to the A column (within the specified range) will automatically trigger the year extraction.

A more sophisticated alternative involves the **QUERY function**, which uses a syntax similar to SQL to select, filter, and manipulate data. While initially complex, **QUERY** provides immense flexibility. To extract the year, you would treat the date column as a groupable entity. You could write a query like `=QUERY(A:A, "SELECT YEAR(A)")`. This method not only extracts the year but can also simultaneously perform aggregations, such as counting how many entries occurred in each specific year, making it ideal for reporting dashboards.

Troubleshooting Common Date Extraction Errors

Although the YEAR function is robust, it relies fundamentally on the input being recognized as a valid date serial number. The most frequent errors encountered during date extraction stem from incorrect date formatting or data type mismatch, leading to unexpected results or error codes like `#VALUE!` or `#NUM!`. Addressing these issues often requires cleaning the source data before

applying the extraction function.

One common issue is that the date is stored as a text string rather than a numerical date value. This often happens when data is imported from external sources or manually entered in a format that Google Sheets does not automatically parse (e.g., "2023 October 27"). If the **YEAR function** returns `#VALUE!`, try wrapping the cell reference within the **DATEVALUE function**, such as `=YEAR(DATEVALUE(A1))`, which forces Google Sheets to attempt interpreting the text string as a date. If the raw date is stored consistently, this conversion is usually successful.

Another important consideration involves regional settings. Dates like "05/06/2023" might be interpreted as May 6th in US format (Month/Day/Year) or as June 5th in European format (Day/Month/Year). If your extracted year values appear incorrect, double-check that the regional settings of your spreadsheet match the format of your source data. Consistency in data entry and verifying the recognized format ensures that functions like **YEAR** and TEXT function operate on the correct underlying numerical date.

Conclusion: Enhancing Data Analysis Capabilities

The ability to accurately and efficiently extract the year from a full date is a cornerstone of effective data analysis in Google Sheets. Whether utilizing the straightforward **YEAR function** for numerical aggregation or the versatile TEXT function for formatted reporting, these tools provide analysts with the necessary precision to categorize and summarize temporal data. The simple implementation of `=YEAR(A1)` serves as the gateway to deeper time-series insights.

By integrating the methods discussed--from basic cell referencing and formula filling to advanced array formulas--users can handle datasets of any size with minimal operational overhead. Furthermore, understanding the interplay between numerical date values and text representations, as facilitated by the **YEAR** and **TEXT** functions, prevents common errors and ensures the reliability of derived data points.

Ultimately, extracting temporal components moves raw data closer to actionable intelligence. With the techniques outlined here, users can confidently prepare their data for advanced operations like creating time-based filters, generating pivot table summaries grouped by year, and building comprehensive annual reports, significantly enhancing their overall spreadsheet proficiency and analytical output.