# How to Sum Every Nth Row in Google Sheets: A Step-by-Step Guide

Authored by

**stats writer**

November 30, 2025

ARABPSYCHOLOGY.COM

RECOMMENDED CITATION

stats writer (2025). *How to Sum Every Nth Row in Google Sheets: A Step-by-Step Guide*. PSYCHOLOGICAL SCALES. Retrieved from https://scales.arabpsychology.com/?p=102466

## Introduction to Summing Nth Rows

While Google Sheets offers a vast array of powerful functions, calculating the sum of values only in every Nth row requires a clever combination of specific array and logical functions. Traditional methods using simple ranges often fall short when dealing with non-contiguous data sets defined by a specific interval. This guide will demonstrate how to efficiently calculate a total for every Nth row using a specialized formula structure involving the SUMIF, ArrayFormula, MOD, and ROW functions. Mastering this combination allows users to handle complex analytical requirements, such as summarizing monthly reports from daily data or analyzing periodic samples within a larger dataset.

The need to aggregate data based on regular intervals--such as summing up every third entry, or every sixth entry--is common in statistical analysis and financial modeling within spreadsheets. Unlike simple continuous sums, targeting specific, spaced rows requires a mechanism to identify the row number's position relative to the desired interval (N). While older spreadsheet software might have relied on complex setups involving the **OFFSET** function, modern array processing in Google Sheets provides a cleaner, more robust solution that dynamically adapts to changes in your data range without requiring manual adjustments to helper columns. This powerful technique leverages array calculation capabilities to perform the selection and summation in a single, elegant formula.

Before diving into the mechanics, it is important to understand the goal: we are creating a conditional summation. We need to evaluate every row in the specified range, assign a logical identifier (True/False or 0/Non-zero) based on whether it is the Nth row, and then only sum the values corresponding to the "True" identifiers. This conditional logic is made possible through the interplay between the **MOD** function, which determines periodicity, and the **ROW** function, which supplies the necessary position index for calculation. Furthermore, utilizing the ArrayFormula guarantees that the logic is applied across the entire range simultaneously, eliminating the need to drag the formula down or rely on external calculation steps.

## Understanding the Core Functions Used

To successfully execute the Nth row summation, we rely on four fundamental Google Sheets functions, each serving a distinct and critical purpose in the overall calculation. Understanding their individual roles is key to modifying the formula for different starting points or different intervals (N). These functions transform raw row indices into a usable logical array for conditional summing, ensuring accuracy across the entire dataset.

**The ROW Function:** This function is the foundation of the calculation, as it returns the numerical index of the current row. When applied to a range (e.g., `ROW(A1:A)`), it returns an array of row numbers (1, 2, 3, 4, ...). We use this array to establish the positional reference for our periodicity

ARABPSYCHOLOGY.COM

check, regardless of where the data resides on the sheet.

**The MOD Function (Modulo):** The Modulo function is the computational core of the Nth row selection. It returns the remainder after division. By calculating `MOD(row_index, N)`, where N is the desired interval, we generate a repeating pattern of remainders. For example, if N=3, the remainders will cycle: 1, 2, 0, 1, 2, 0, etc. The rows where the remainder is 0 are precisely the rows we want to sum (after adjusting for the starting point).

**The ArrayFormula Function:** This wrapper function is essential because it forces the functions nested within it, particularly **ROW** and **MOD**, to process entire ranges rather than just single cells. Without ArrayFormula, the internal operations would only apply to the first cell reference (A1), failing to generate the necessary array of periodic criteria for the rest of the column.

**The SUMIF Function:** The SUMIF function performs the final conditional summation. It uses the array of remainders generated by the **MOD** function as its criterion range. It checks for a specific condition (a remainder of 0, indicating the Nth row) and then sums the corresponding values from the source data column specified as the sum range.

## The Universal Formula for Nth Row Summation

The syntax required to sum the values in every Nth row in Google Sheets is concise yet powerful. This generalized formula allows for easy modification of the interval (N) and the starting cell, making it highly adaptable for various data analysis scenarios. The general structure relies on generating an array of 0s and non-zero values, where 0 represents a row that satisfies the Nth row criterion and should be included in the sum.

The core formula structure is presented below, configured to sum every 3rd row starting from the first row of data. We recommend using this as a template, simply replacing the interval (3) and the starting references (A1) as needed for your specific dataset:

You can use the following syntax to sum the values in every nth row in Google Sheets:

**=sumif(ArrayFormula(mod((row(A1:A)-row(A1)+1),3)),0,A1:A)**

This particular formula sums every **3rd** value in column A, starting at cell **A1**.

In this specific instance, the formula is configured to sum every **3rd** value in column A, explicitly starting the count from cell **A1** and continuing down the column. Understanding the specific components within the **MOD** function is critical for customizing the starting point of the summation, which we will explore in detail shortly. The beauty of this array approach is that it is self-contained and avoids the complexity associated with linking multiple cells or creating extraneous data

ARABPSYCHOLOGY.COM

columns.

## Deconstructing the Formula Components: Periodicity and Offset

The most intricate part of the formula is the index calculation provided as the first argument to the MOD function: `(row(A1:A) - row(A1) + 1)`. This segment handles the normalization and offset required to ensure that the Nth count starts correctly relative to the beginning of the data range, rather than relative to the absolute row number of the sheet. This normalization is what provides the formula its flexibility.

Let's break down this complex inner calculation step-by-step to understand the row indexing. First, `ROW(A1:A)` generates the absolute row numbers (e.g., 1, 2, 3, 4...). Next, `ROW(A1)` returns the starting row number (e.g., 1). By performing the subtraction, `(ROW(A1:A) - ROW(A1))`, we effectively reset the counting index to start at 0 (0, 1, 2, 3...). Finally, adding `+ 1` shifts the index back to a standard 1-based counting sequence (1, 2, 3, 4...). This normalized index is essential because the MOD function needs a 1-based index to correctly identify the first, N+1th, 2N+1th, etc., elements based on the remainder.

Once the index is normalized, the structure `MOD(normalized_index, N)` determines the remainder. If we set N=3, the remainders will be 1, 2, 0, 1, 2, 0, and so forth. Since the count is 1-based, the values that produce a remainder of 0 are those that are exactly divisible by 3 (the 3rd, 6th, 9th rows, etc., relative to the start cell). The outer SUMIF function then looks for this remainder of **0** as its criterion, successfully isolating and summing only those values corresponding to the specified interval (N). This technique effectively creates a Boolean mask across the entire column.

The following examples show how to use this syntax in practice with the following column of values in Google Sheets:

## Example 1: Sum Every Third Row

Our first practical demonstration illustrates the most common use case: summing data from the very beginning of the dataset at a fixed, regular interval. For this example, we aim to calculate the total summation of every third row, initiating the calculation from cell **A1**. This setup requires the periodicity parameter (N) within the MOD function to be set to 3.

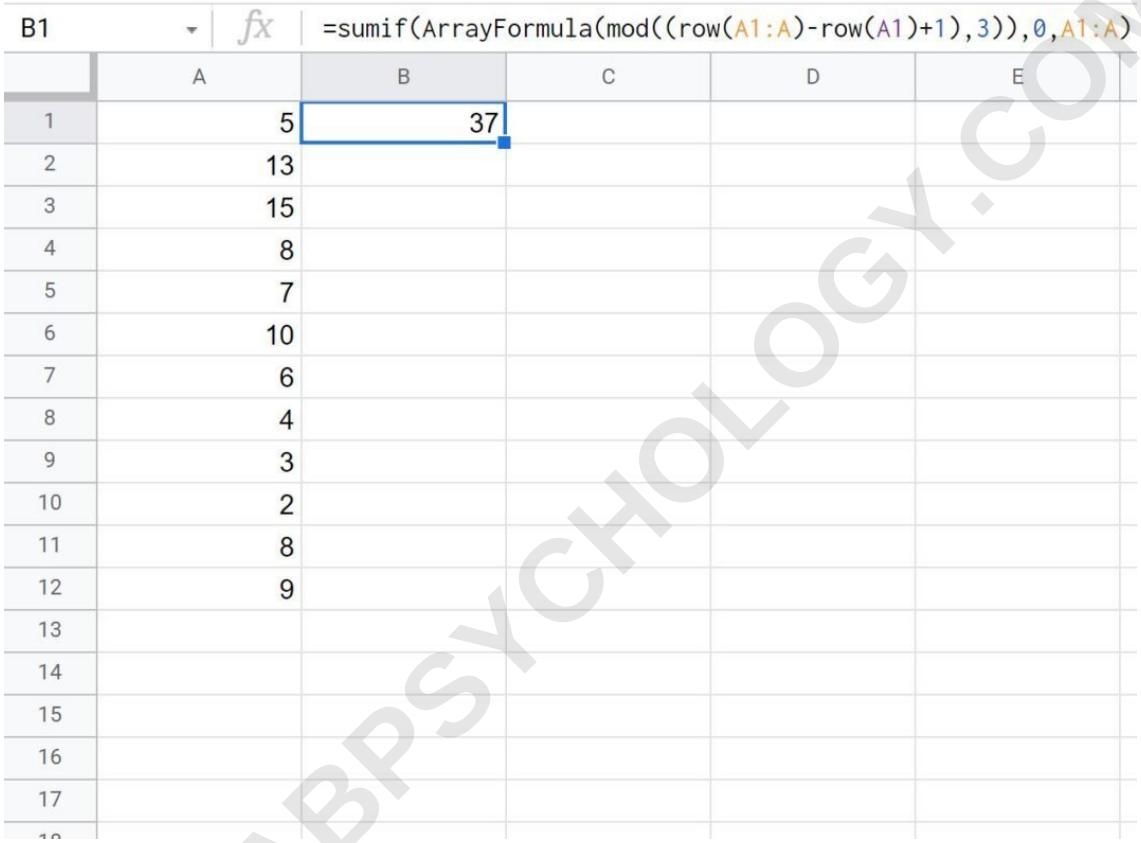
We ensure that the entire range references within the formula (the data range for the index calculation, the starting cell reference, and the final summation range) all begin at **A1**. This configuration guarantees that our 1-based index starts counting from the very first row of the data, ensuring that the 3rd, 6th, 9th, and subsequent rows are correctly targeted by the logic `MOD(index, 3) = 0`.

ARABPSYCHOLOGY.COM

We can use the following formula to sum the values in every third row (starting in cell **A1**) of column A:

**=sumif(ArrayFormula(mod((row(A1:A)-row(A1)+1),3)),0,A1:A)**

The following screenshot visually confirms the implementation and resulting output of this calculation:




ARABPSYCHOLOGY.COM

Upon execution, the summation of every third row results in a total of **37**. This result can be confirmed manually, reinforcing the accuracy of the array calculation. By analyzing the data in column A, we select the values corresponding to the 3rd, 6th, 9th, and 12th positions, relative to the starting cell A1. The formula successfully filters out all non-periodic values.

We can manually verify this by taking the summation of every third row:

Sum: 15 + 10 + 3 + 9 = **37**.

## Example 2: Sum Every Sixth Row

Extending the principle demonstrated in Example 1, this scenario shows how easily the formula

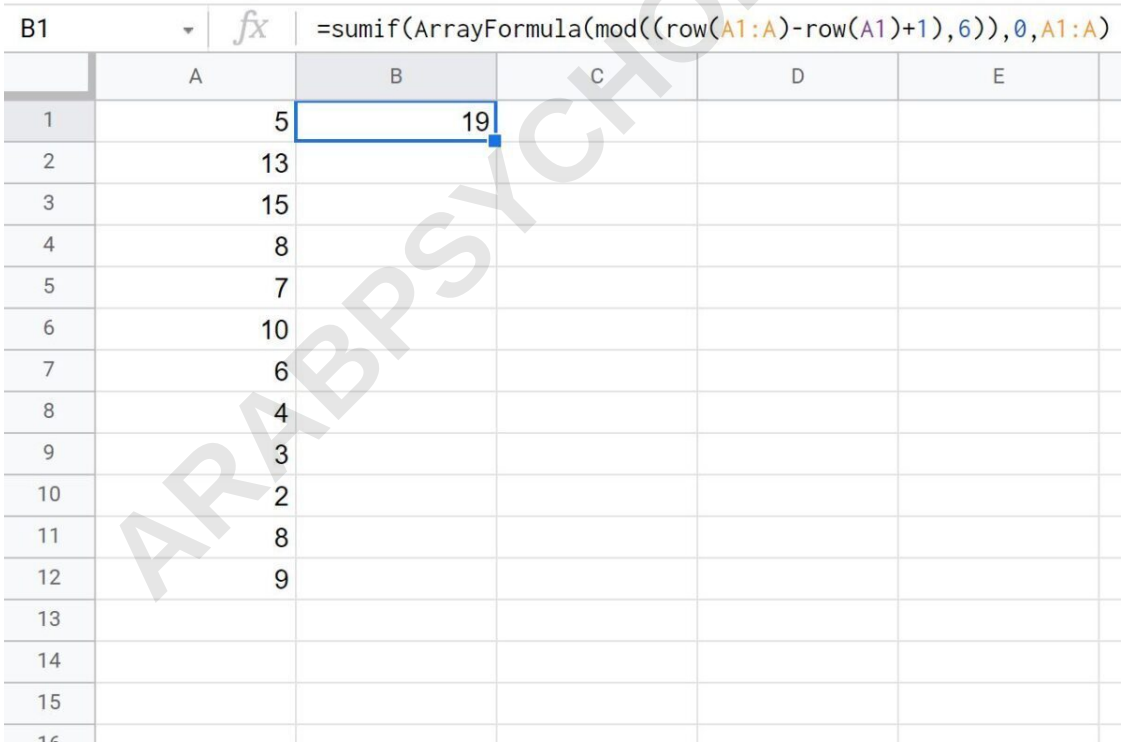
can be adapted to handle a larger interval. Suppose we have a very dense dataset and only need summary metrics captured every six rows--perhaps a weekly measurement derived from daily data recorded over six business days. This adjustment requires only a single, simple change to the core formula structure, demonstrating the formula's scalability.

In this case, the interval parameter (N) must be modified from 3 to 6. All other components, including the starting cell reference (A1) and the range references (A1:A), remain identical because we are still starting the count from the top of the dataset. This minimal required change highlights the robustness of using the MOD function for defining periodicity without disturbing the indexing logic.

We can use the following formula to sum the values in every sixth row (starting in cell **A1**) of column A:

**=sumif(ArrayFormula(mod((row(A1:A)-row(A1)+1),6)),0,A1:A)**

The following screenshot shows how to use this syntax in practice:




Given the extended interval of six rows, fewer data points will satisfy the criteria in our short dataset. In our sample data, only two values meet the condition: the value in the 6th row and the value in the 12th row. The resulting total is significantly smaller, emphasizing the formula's

ARABPSYCHOLOGY.COM

precision in isolating specific periodic data points within the spreadsheet environment, regardless of the interval size.

We can manually verify this by taking the summation of every sixth row:

Sum: 10 + 9 = **19**.

## Example 3: Sum Every Third Row Starting at Row 2 (Applying an Offset)

One of the most valuable features of this array approach is the ability to introduce an arbitrary starting offset. Often, the header row or initial metadata might occupy the first row, requiring the summation count to begin from the second or third row. In this example, we need to sum every third row, but the counting must commence from cell **A2**, meaning the first value summed will be the third row relative to A2 (i.e., A4, A7, A10, and so on).

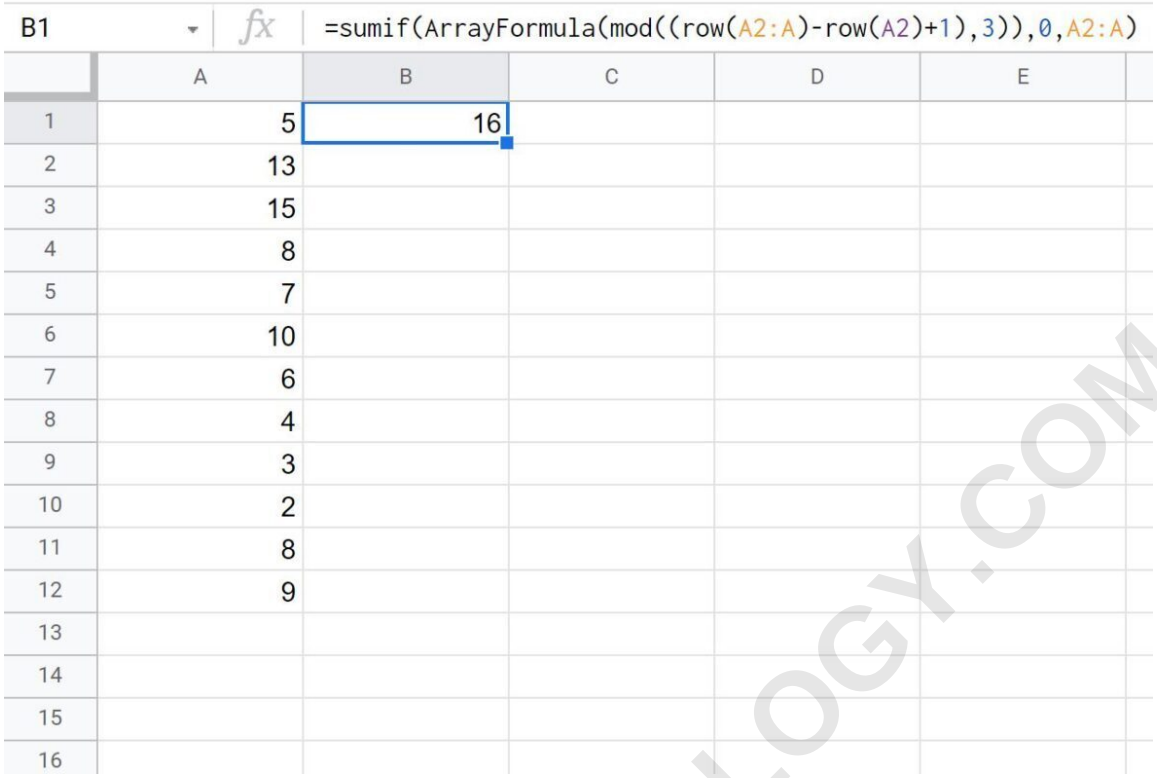
To implement this offset, we must adjust two key components of the formula. First, the range references must start at **A2** (i.e., `A2:A`). Second, and most critically for the index normalization, the starting row reference within the **MOD** function must also change to `ROW(A2)`. This ensures that the calculation `(ROW(A2:A) - ROW(A2) + 1)` correctly resets the 1-based index calculation to begin at A2, accurately capturing the desired periodicity from the new starting point.

We can use the following formula to sum the values in every third row (starting in cell **A2**) of column A:

**=sumif(ArrayFormula(mod((row(A2:A)-row(A2)+1),3)),0,A2:A)**

The following screenshot shows how to use this syntax in practice:

ARABPSYCHOLOGY.COM




The sum of every third row (starting in cell **A2**) turns out to be **16**. This offset calculation effectively shifts the selection criteria down by one row compared to Example 1, demonstrating precise control over data aggregation regardless of the physical starting location within the spreadsheet. The rows targeted are A4, A7, and A10.

We can manually verify this by taking the summation of every third row, starting at cell **A2**:

Sum: 8 + 6 + 2 = **16**.

## Advanced Applications and Considerations

While the primary application discussed here is simple summation, this methodology using SUMIF combined with the periodic indexing array can be extended for other conditional calculations. For instance, instead of summing, you could use a similar array structure with **COUNTIF** to count how many Nth rows contain non-zero values, or use **AVERAGEIF** to calculate the mean of the periodic entries. The key takeaway is that the **ArrayFormula(MOD(...))** section serves as a universally applicable mask or filter for selecting periodic entries, applicable across various conditional statistical functions.

One important consideration is handling blank or non-numeric cells, particularly when using full column references (A:A). Since the formula applies the periodic check across the entire range, it will generate the remainder array for every row. Although SUMIF is generally robust against blank

ARABPSYCHOLOGY.COM

cells within the summation range, if your dataset contains true non-numeric headers or footers, ensure your defined range is constrained to only the relevant numerical data (e.g., A2:A100), or use nested functions like **IFERROR** or **FILTER** to pre-process the data before applying the Nth row calculation.

For extremely large datasets in Google Sheets, especially those exceeding tens of thousands of rows, relying on full column references (like A:A) within complex array formulas can occasionally introduce minor performance delays during recalculation. Although Google Sheets is highly optimized, best practice suggests limiting the range reference (e.g., A1:A5000) if you know the maximum extent of your data, thus reducing the processing overhead required by the ArrayFormula. However, for most common business applications, the full column reference remains the most convenient and adaptable option.

## Summary of Best Practices and Key Takeaways

Mastering the Nth row summation technique provides a significant advantage when analyzing time-series or sequential data in Google Sheets. This method eliminates the need for manual row counting or helper columns, consolidating complex logic into a single, maintainable cell formula.

To ensure successful implementation and easy modification of the formula, keep the following best practices in mind:

**Synchronization of References:** Always ensure that the range used in `ROW(StartCell:EndCell)` and the starting cell used for the offset calculation `ROW(StartCell)` are identical to maintain accurate indexing. Similarly, the data range for the summation part of the SUMIF must match.

**Determining N:** The number N, which defines the interval (e.g., 3 or 6), is the parameter immediately following the normalized index within the MOD function. This is the single variable you will change to adjust the periodicity.

**Using ArrayFormula:** Do not forget the ArrayFormula wrapper. It is essential for ensuring that the **ROW** and **MOD** functions operate on the entire column array rather than just the first cell.

By combining the positional awareness of the **ROW** function with the logical filtration provided by the **MOD** function, analysts gain precise control over periodic data aggregation, enhancing the analytical depth available within Google Sheets.

ARABPSYCHOLOGY.COM