How to aggregate multiple columns in R?

Authored by

stats writer

December 11, 2025

Aggregation is a fundamental operation in R programming for effective data manipulation and analysis. It involves boiling down extensive datasets into meaningful summary statistics based on specified grouping factors. While contemporary packages often simplify this process, the base R environment provides the highly robust and versatile aggregate() function. Mastering this function is essential for any serious R user, as it offers a powerful, vectorized method to apply functions across various subsets of a data frame.

The core utility of the aggregate() function lies in its ability to handle complex grouping scenarios, making the transition from raw data to insightful summaries seamless. Unlike functions focused on single-column application, `aggregate()` is specifically designed to manage multiple variables for both grouping and summarization simultaneously. This capability mirrors the efficiency of pivot tables in spreadsheet software or the crucial GROUP BY clause in SQL databases, proving indispensable for tasks ranging from initial exploratory data analysis to generating sophisticated final reports.

To use this powerful tool, three essential arguments are required: the data object to be processed (typically a data frame), a formula that strictly defines which columns are to be aggregated and which columns define the groups, and finally, the statistical function to be applied (e.g., mean, sum, median). This structure provides a highly flexible framework for calculating metrics across various granularities of the dataset.

Understanding the Base R Aggregate Function

We utilize the **aggregate()** function in R to efficiently produce summary statistics for one or more variables within a supplied data frame. This function is particularly effective because it uses the standard R formula interface, which provides exceptional clarity regarding the relationship between the dependent (summarized) and independent (grouping) variables.

The basic structure of the function call employs the syntax shown below, which clearly distinguishes between the outcome variable(s) being summarized and the predictor variable(s) used for grouping:

aggregate(sum_var ~ group_var, data = df, FUN = mean)

The structure of the formula must always place the variables to be summarized on the left side of the tilde (`~`) operator, while the variables defining the groups are placed on the right. This arrangement is foundational to how R interprets the aggregation request. The arguments defining the function's behavior are detailed as follows:

sum_var: This parameter specifies the quantitative variable, or variables (using `cbind()`), that you wish to summarize or calculate metrics upon.

group_var: This crucial parameter defines the categorical variable(s) by which the data frame will be partitioned. The summary calculation is performed independently within each unique combination of these grouping variables.
data: This is simply the name of the input data frame or object containing the variables specified in the formula.
FUN: This argument requires the summary statistic function to compute (e.g., `mean`, `sum`, `sd`, `max`). The output of the aggregation is determined entirely by this function.

Throughout this tutorial, we will explore several practical examples demonstrating how to leverage this function to aggregate data based on varying complexities--from summarizing a single column based on one group to summarizing multiple columns across multiple grouping factors. We will use the following example data frame, representing sports statistics, for all demonstrations:

```
#create data frame
df <- data.frame(team=c('A', 'A', 'A', 'B', 'B', 'B', 'C', 'C'),
conf=c('E', 'E', 'W', 'W', 'W', 'W', 'W', 'W'),
points=c(1, 3, 3, 4, 5, 7, 7, 9),
rebounds=c(7, 7, 8, 3, 2, 7, 14, 13))
```

#view data frame
df

team conf points rebounds
1 A E 1 7
2 A E 3 7
3 A W 3 8
4 B W 4 3
5 B W 5 2
6 B W 7 7
7 C W 7 14
8 C W 9 13

Example 1: Summarize One Variable & Group by One Variable

The simplest and most common application of the aggregate() function involves calculating a summary metric for a single dependent variable, segmented by a single grouping variable. This scenario is ideal when you need a quick overview of performance metrics across distinct categories, such as finding the average points scored by each team in our example data set.

In this specific case, the variable to summarize is `points`, and the grouping factor is `team`. The

formula syntax `points ~ team` clearly dictates this relationship. We instruct R to apply the `mean` function to the `points` column for every unique value found in the `team` column. Additionally, the argument `na.rm = TRUE` is included as a best practice, ensuring that any missing values (NA) in the `points` column are ignored during the calculation of the average.

The following code executes this calculation and produces a new, aggregated data frame where each row corresponds to a unique team and the calculated mean points:

#find mean points scored, grouped by team
aggregate(points ~ team, data = df, FUN = mean, na.rm = TRUE)

team points
1 A 2.333333
2 B 5.333333
3 C 8.000000

The resulting output shows that Team C has the highest mean points score (8.00), while Team A has the lowest (2.33). This basic aggregation provides essential insights into team performance differences almost instantaneously.

Example 2: Summarize One Variable & Group by Multiple Variables

Data often requires segmentation across multiple dimensions to uncover more nuanced patterns. The aggregate() function easily accommodates multiple grouping variables by separating them with the addition operator (`+`) on the right side of the tilde in the R formula.

In this example, we aim to calculate the mean points scored, but this time, the aggregation must be segmented by both `team` and `conf` (conference). The formula becomes `points ~ team + conf`. R will generate a summary statistic for every unique combination of team and conference present in the source data frame. This level of granularity helps analysts determine if a team's performance varies significantly between different conferences.

We maintain the same summary function (`mean`) and the same data source (`df`), modifying only the complexity of the grouping variables within the formula. This demonstrates the function's flexibility without requiring major structural changes to the command:

#find mean points scored, grouped by team and conference
aggregate(points ~ team + conf, data = df, FUN = mean, na.rm = TRUE)

team conf points
1 A E 2.000000

2 A W 3.000000
3 B W 5.333333
4 C W 8.000000

The result clearly distinguishes the performance of Team A when playing in Conference E (2.00 points) versus Conference W (3.00 points), a distinction that was masked in the previous example where only `team` was used for grouping.

Example 3: Summarize Multiple Variables & Group by One Variable

A key advantage of the base R **aggregate()** function over some specialized grouping functions is its native support for summarizing multiple dependent variables simultaneously. To achieve this, we must use the `cbind()` function (column bind) on the left side of the tilde operator. The `cbind()` function effectively tells R to treat the specified columns as a composite unit for the purpose of applying the summary statistic.

In this scenario, we wish to calculate both the mean `points` and the mean `rebounds`, segmented only by the `team` variable. We construct the formula as `cbind(points, rebounds) ~ team`. The output will include separate columns for the aggregated results of both points and rebounds for each team. This avoids the necessity of running two separate aggregation commands, significantly streamlining the code and improving execution efficiency.

The following code demonstrates how to find the mean points and the mean rebounds, grouped solely by team:

#find mean points scored, grouped by team and conference
aggregate(cbind(points,rebounds) ~ team, data = df, FUN = mean, na.rm = TRUE)

team points rebounds
1 A 2.333333 7.333333
2 B 5.333333 4.000000
3 C 8.000000 13.500000

The resulting aggregated data frame now contains three columns: `team`, `points` (mean), and `rebounds` (mean). This single operation provides a comprehensive view of team performance across two crucial statistical dimensions.

Example 4: Summarize Multiple Variables & Group by Multiple Variables

The most complex and powerful application of the **aggregate()** function involves combining the

techniques from Examples 2 and 3. Here, we summarize multiple dependent variables (using `cbind()`) while partitioning the dataset based on multiple independent grouping variables (using the addition operator `+` in the R formula). This structure allows for the deepest possible level of breakdown using a single line of code.

For our final example, we will calculate the mean `points` and mean `rebounds`, grouped simultaneously by `team` and `conf`. The full formula is written as `cbind(points, rebounds) ~ team + conf`. This command instructs R to identify every unique combination of team and conference, and then calculate the mean for both points and rebounds for the records falling into that specific group.

This highly granular aggregation is critical for uncovering specific performance efficiencies or weaknesses that might be obscured when grouping by only one factor. The result is a highly detailed summary table:

#find mean points scored, grouped by team and conference
aggregate(cbind(points,rebounds) ~ team + conf, data = df, FUN = mean, na.rm = TRUE)

team conf points rebounds
1 A E 2.000000 7.0
2 A W 3.000000 8.0
3 B W 5.333333 4.0
4 C W 8.000000 13.5

The output confirms that Team A performs better in Conference W in terms of both points and rebounds, highlighting the value of multi-dimensional aggregation for accurate performance assessment.

Advanced Considerations and Efficiency

While the aggregate() function is powerful, users often inquire about its performance compared to newer packages. For extremely large datasets (millions of rows), functions provided by packages like `data.table` or `dplyr` often offer superior computational speed due to underlying C++ optimizations. However, for moderate datasets and for maintaining compatibility within the base R environment, `aggregate()` remains an extremely readable and robust choice.

One important consideration when using any aggregation function is the handling of missing data. As demonstrated in all examples, including the `na.rm = TRUE` argument ensures that null values are correctly ignored during the calculation of the summary statistics. Failing to include this parameter often results in an output of NA for groups containing any missing data points, potentially obscuring valid results.

In summary, the **aggregate()** function provides a foundational method for data summarization in R. By mastering the formula interface--particularly the use of `cbind()` for multiple outcome variables and the addition operator (`+`) for multiple grouping factors--analysts gain immediate access to comprehensive descriptive statistics essential for advanced data exploration.