# How to Easily Calculate Mode by Group in R: A Step-by-Step Guide

Authored by

**stats writer**

November 27, 2025

Calculating the **mode**, which is defined as the most frequently occurring value within a dataset, often requires specific handling in the **R statistical environment**. While **aggregate()** or **tapply()** functions in base R can technically be adapted for group-wise calculations, modern data manipulation often benefits from the clarity provided by the **dplyr** package. This comprehensive guide details how to implement a custom function in R and apply it efficiently to calculate the mode for distinct groups within a **data frame**, ensuring accurate results even when dealing with multimodal distributions.

Understanding group-wise calculations is essential for complex data analysis. When working with large datasets, analysts frequently need to summarize variables based on categorical identifiers--for instance, finding the most common score achieved by athletes on different teams. Since R does not include a direct, built-in function for calculating the mode, analysts must first define a robust custom function capable of handling various distributional complexities, including cases where a single group might exhibit more than one mode.

The **mode** of a dataset represents the most frequently occurring observation. Unlike the mean or median, the mode is applicable to all levels of measurement, making it a versatile measure of central tendency.

Despite its fundamental nature, the **R statistical software** does not come equipped with a standardized, built-in function to calculate the mode of a vector or dataset. Therefore, to perform this statistical operation effectively, especially when grouping data, we must utilize the following user-defined function:

```
find_mode <- function(x) {
u <- unique(x)
tab <- tabulate(match(x, u))
u
}
```

This custom **function**, named find_mode, is designed to first identify all unique values within the input vector (u <- unique(x)). It then counts the frequency of each unique value (tab <- tabulate(match(x, u))). Finally, it returns the unique value(s) that correspond to the maximum frequency found in the vector (u). This structure is critical because it correctly handles situations where multiple values share the highest frequency.

The following examples demonstrate how to utilize this robust function in conjunction with grouping tools to successfully calculate the **mode** segmented by various categorical variables within a **data frame**. We will focus on the highly efficient grouping capabilities provided by the **dplyr** package, which is widely used in R for data wrangling tasks.

## Understanding Grouped Statistical Summaries

In data analysis, particularly using **R**, calculating summary statistics that are conditional on categories is fundamental. When we calculate the mode by group, we partition the dataset based on one variable (e.g., 'team') and then compute the mode for a second variable (e.g., 'points') independently within each partition. This process ensures that the resulting statistics are specific to the subgroups defined by the categorical variable.

While base R offers functions like aggregate(), the **dplyr** package, part of the Tidyverse, provides a more intuitive and readable syntax for performing these operations using pipes (%>%) and explicit verb functions like group_by() and summarize(). This approach significantly streamlines the workflow for complex data manipulations.

The efficiency of the grouped approach relies on first defining the groups using group_by(). Once the data frame is logically grouped, the summarize() **function** applies the custom find_mode function to the target column within each defined group, returning the mode for that specific subset of data. This methodology is demonstrated clearly in the subsequent examples.

## Example 1: Calculating Mode for Unimodal Groups

Consider a practical scenario involving sports statistics where we track the points scored by players across different basketball teams. In this first example, we utilize a dataset where each group exhibits a single, clear mode (unimodal distribution). This setup illustrates the most common use case for grouped mode calculation.

Suppose we have the following **data frame** in **R** that shows the points scored by basketball players across two different teams, A and B. We aim to find the most frequent point total for each team independently.

**#define data frama**
**df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'),**
**points=c(5, 7, 7, 9, 12, 12, 10, 14))**

#view data frame
df

team points
1 A 5
2 A 7
3 A 7
4 A 9

5 B 12
6 B 12
7 B 10
8 B 14

To determine the **mode** of points for each team, we apply our previously defined find_mode **function** within the **dplyr** workflow. We first load the necessary library, define the function (if not already defined), and then group the data by the team variable before summarizing the points.

```
library(dplyr)

#define function to calculate mode
find_mode <- function(x) {
u <- unique(x)
tab <- tabulate(match(x, u))
u
}

#calculate mode of 'points' by 'team'
df %>%
group_by(team) %>%
summarize(mode_points = find_mode(points))

# A tibble: 2 x 2
team mode_points

1 A 7
2 B 12
```

Analyzing the resulting output table, which is generated as a tibble in the **R** console, provides clear insight into the most frequent scores for each group.

The calculated mode of points for team A is **7**, as this score occurs twice (out of four observations), which is more frequent than scores 5 and 9.
The calculated mode of points for team B is **12**, which also occurs twice, dominating the frequency count compared to scores 10 and 14.

## The Challenge of Multimodal Data

A significant advantage of the custom find_mode function is its inherent ability to correctly identify and return multiple modes, a property critical for handling multimodal datasets. A distribution is

considered **multimodal** if two or more distinct values share the highest frequency count. Unlike calculating the mean or median, which always yields a single value, the mode calculation must be flexible enough to accommodate these complex distributions.

If we were to use a simpler approach that only returned the first value achieving the maximum frequency, we would fail to capture the complete picture of the data's central tendency. This could lead to misleading statistical interpretations, particularly in fields like demographics or quality control where understanding peak frequencies is paramount.

The structure of the find_mode **function** (specifically u) ensures that R returns a vector containing all values that meet the criteria of maximum frequency. When this function is used within the summarize() framework of **dplyr**, the resulting output naturally expands to show all identified modes for the group, as demonstrated in the next example.

## Example 2: Handling Groups with Multiple Modes (Bimodal)

In this second scenario, we modify the original **data frame** slightly to introduce bimodal characteristics into one of the groups. This provides a crucial test case for our custom mode calculation **function**, ensuring it performs accurately under conditions where multiple values occur with the same maximum frequency.

Suppose we have the following updated dataset in **R**. Notice that for Team B, the scores 12 and 10 now both appear twice, making Team B's distribution of scores bimodal.

```
#define data frama
df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'),
points=c(5, 7, 7, 9, 12, 12, 10, 10))

#view data frame
df

team points
1 A 5
2 A 7
3 A 7
4 A 9
5 B 12
6 B 12
7 B 10
8 B 10
```

Just as before, we apply the grouping and summarizing method using the **dplyr** package. Since our find_mode function is vectorized and capable of returning multiple results, the summarize() command will automatically unpack these multiple results, displaying each mode on its own row associated with the corresponding group identifier.

**library(dplyr)**

```
#define function to calculate mode
find_mode <- function(x) {
u <- unique(x)
tab <- tabulate(match(x, u))
u
}

#calculate mode of 'points' by 'team'
df %>%
group_by(team) %>%
summarize(mode_points = find_mode(points))

# A tibble: 3 x 2
# Groups: team
team mode_points

1 A 7
2 B 12
3 B 10
```

The resulting output clearly demonstrates the successful identification of the multiple modes for Team B.

The mode of points for team A remains **7** (a single mode).
The modes of points for team B are correctly identified as both **12** and **10**.

In this specific output structure generated by **R** and **dplyr**, there were two points values that occurred most frequently (10 and 12) for team B. As expected, our custom function returned both, and the summarize() command expanded the result, listing each of these mode values on a separate line corresponding to team B in the final output tibble. This behavior is crucial for accurate multimodal statistical reporting.

## Alternatives to the dplyr Workflow

While the **dplyr** package offers the most modern and readable approach for calculating grouped statistics, it is important to acknowledge that base R provides built-in tools capable of achieving the same result. Analysts familiar with older R syntax or environments where external packages are restricted might prefer using the base R functions aggregate() or tapply().

The **aggregate() function**, for instance, allows applying a function (in our case, find_mode) to a dataset grouped by factors specified in a formula interface. The syntax is slightly less intuitive than group_by() but is equally powerful. Similarly, **tapply()** is excellent for applying a function over ragged arrays, which is essentially what grouping creates, though its output often requires further manipulation to revert into a clean **data frame** structure.

Regardless of the chosen grouping method--be it dplyr::group_by(), aggregate(), or tapply()--the critical component remains the custom find_mode function. Since R treats the mode as a non-standard statistical summary, having a robust function definition is the prerequisite for any accurate calculation, especially when operating on subsets of data.

## Conclusion

Calculating the **mode** by group in **R** requires defining a custom statistical **function** due to the lack of a built-in base R utility. By leveraging the flexibility of this custom function, combined with powerful grouping mechanics provided by packages like **dplyr**, analysts can efficiently summarize the most frequent values for subsets of data.

The examples provided illustrate that this method is highly robust, successfully managing both unimodal and multimodal distributions across different groups within a single **data frame**. Mastering this technique is essential for anyone performing descriptive statistical analysis or data profiling in R, ensuring that reported measures of central tendency are complete and accurate.

The following tutorials explain how to calculate other descriptive statistics in R: