How to Easily Create a Three-Way Table in R Using the expss Package

Authored by

stats writer

November 20, 2025

Introduction to Three-Way Tables in R

The ability to effectively analyze and visualize relationships between multiple factors is fundamental in statistical analysis. When dealing with categorical variables, the primary tool for summarizing these relationships is the contingency table. While two-way tables are common, analyzing the interaction among three variables requires the construction of a three-way table. This structure allows researchers to examine how the frequency distribution of two variables changes across the levels of a third variable, providing a much deeper insight into multivariate data structures. Generating these complex summaries efficiently is easily accomplished within the R statistical environment, leveraging powerful functions built directly into its core.

Although specialized packages like **`expss`** offer sophisticated functions for handling complex survey data and generating publication-ready tables, the most straightforward and resource-efficient method for generating three-way frequency tables relies on tools available in base R. The key functions we will focus on are **`xtabs()`** for initial tabulation and **`ftable()`** for producing a compact, readable output. Mastering these functions ensures that analysts can reliably produce accurate frequency counts without needing to install external dependencies, making the code highly portable and reproducible. We will explore the syntax, application, and detailed interpretation of the results generated by these intrinsic R commands.

Generating a three-way table is not merely about counting occurrences; it is about structuring the data in a manner that facilitates formal statistical testing, such as tests for conditional independence or exploring interaction effects. By clearly defining the relationships among three dimensions--such as demographic group, treatment type, and outcome measure--the resulting table serves as the crucial preliminary step before applying more complex modeling techniques. The examples provided here are designed to illustrate the practical steps required to transform raw dataset variables into structured, statistically meaningful frequency distributions, setting the stage for advanced data exploration.

Understanding the Nature of Three-Way Contingency Tables

A three-way table is technically a multi-dimensional array that cross-classifies observations based on the levels of three separate categorical variables. Unlike a two-way table, which provides a simple row-by-column matrix, a three-way table introduces a third dimension, often visualized as layers or slices. Each layer represents a specific level of the third variable (the stratifying variable), and within that layer, the standard two-way cross-classification of the first two variables is displayed. This stratification is essential because marginal associations observed in a two-way table can sometimes be misleading or entirely disappear when a third variable is controlled for, a phenomenon famously known as **Simpson's Paradox**.

The structure of the three-way table allows for the analysis of conditional relationships. For

instance, if we analyze the relationship between Variable A and Variable B, the third dimension, Variable C, allows us to examine if that A-B relationship holds true separately when C is at Level 1, Level 2, or Level 3. This depth of analysis is critical in fields like epidemiology, social sciences, and market research, where confounding factors often influence observed relationships. In R, this three-dimensional structure is often represented internally as an object of class `"table"` or `"array"`, enabling powerful indexing and manipulation.

When working with these tables, it is important to clearly designate which variable serves as the primary stratification factor, as this choice influences the clarity and ease of interpretation of the output. While the `xtabs()` function in base R generates the full three-dimensional array, the raw output can sometimes be cumbersome to read, especially with many levels. Therefore, understanding the underlying mathematical structure--the frequency counts for every possible combination of the three factor levels--is paramount before moving on to presentation tools like `ftable()`.

Utilizing the Base R Function: The Power of `xtabs()`

The primary workhorse for creating frequency tables in R is the `xtabs()` function. The name stands for "cross tabulation," and it efficiently constructs contingency tables using a formula interface, which is highly intuitive for R users. The basic syntax mirrors that used for statistical models, specifying the variables to be tabulated on the right side of a tilde (**~**), separated by plus signs (**+**), and referencing the data frame that contains these variables.

For a three-way table involving variables `var1`, `var2`, and `var3` stored in a data frame named `df`, the command structure is remarkably simple yet powerful. The general form is shown below. Note that the order in which the variables are listed in the formula determines the structure of the resulting array, specifically which variable will define the 'slices' or layers of the output, thus affecting readability and subsequent analysis.

three_way <- xtabs(~ var1 + var2 + var3, data=df)

The `xtabs()` function is built into base R, meaning no special library installations are required. It returns an object of class `table`. When this object is printed, R defaults to showing the layers sequentially. For a three-way table, this means R will first fix the level of the third variable (e.g., `var3 = Level A`) and then display the two-way table for `var1` and `var2` for that fixed level. It then repeats this process for `var3 = Level B`, and so on. While mathematically sound, this layered structure is often less compact than researchers prefer for publication or quick inspection, which leads us to the utility of the `ftable()` function.

Flattening Complex Output with `ftable()`

While the output generated by **`xtabs()`** accurately represents the three-dimensional frequency array, it can be spread across multiple sections on the console, making direct comparison difficult. To address this, base R provides the **`ftable()`** function, which stands for "flat table." This function takes the multi-dimensional table object produced by **`xtabs()`** and transforms it into a two-dimensional matrix display, significantly improving readability and compactness.

The **`ftable()`** function smartly organizes the variables, placing some in the rows and others in the columns to minimize the required screen space. By default, it moves the variables that typically have fewer unique levels into the columns, allowing the variables with more levels to structure the rows. This arrangement is highly efficient for visual comparison. Using our previous example, applying **`ftable()`** to the **`three_way`** object yields the following general structure:

three_way_ftable <- ftable(three_way)

A crucial aspect of **`ftable()`** is that it maintains the integrity of the underlying frequency counts; it simply reformats the presentation. It typically places the row variables stacked on the left, separated by spaces or grouping lines, and the column variables stretched horizontally across the top. This flattened structure makes it ideal for easily exporting results to other programs or including them directly in reports, as it provides a single, cohesive view of the entire cross-classification. Remember that both **`xtabs()`** and **`ftable()`** are foundational functions inherent to base R, ensuring stability and accessibility.

Practical Implementation: Creating the Basketball Player Dataset

To demonstrate the practical application of these functions, we will work with a sample dataset containing information about basketball players. This example will allow us to investigate the relationship between three categorical variables: the player's **team**, their **position** (Guard or Forward), and whether they are a **starter** (Yes or No). We will first define and populate the data frame in R, which is the necessary prerequisite for using the cross-tabulation functions.

The dataset includes ten records, capturing four variables: **`team`** (A or B), **`position`** (G or F), **`starter`** (Yes or No), and **`points`** (a quantitative variable, though only the categorical variables will be used for the three-way table). It is always beneficial to explicitly define the dataset and verify its structure before proceeding with statistical analysis. The code block below shows the initialization and display of our sample data frame, **`df`**:

#create data frame
df <- data.frame(team=c('A', 'A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'B'),

```
position=c('G', 'G', 'G', 'F', 'F', 'G', 'G', 'F', 'F', 'F'),
starter=c('Yes', 'No', 'No', 'Yes', 'No',
'Yes', 'No', 'Yes', 'Yes', 'No'),
points=c(30, 28, 24, 24, 28, 14, 16, 20, 34, 29))

#view data frame
df
```

team position starter points
1 A G Yes 30
2 A G No 28
3 A G No 24
4 A F Yes 24
5 A F No 28
6 B G Yes 14
7 B G No 16
8 B F Yes 20
9 B F Yes 34
10 B F No 29

Our objective is to create a frequency table that summarizes the counts of players based on the combinations of `team`, `position`, and `starter` status. This involves cross-tabulating these three variables simultaneously. This specific application is common in sports analytics where coaches or analysts might want to see how starting roles are distributed across different teams and positions. By organizing the data this way, we can quickly identify imbalances or common patterns in player allocation, moving beyond simple two-way comparisons.

Generating and Interpreting the Three-Way Table Output

With the data frame ready, we now apply the `xtabs()` function. We specify the formula `~ team + position + starter`, which instructs R to count the occurrences for every combination of the levels of these three variables. In this formulation, `starter` will naturally become the variable defining the slices of the three-dimensional array in the default print output.

The following code block executes the cross-tabulation and displays the resulting three-way array object:

```
#create three-way table
three_way <- xtabs(~ team + position + starter, data=df)

#view three-way table
```

three_way

, , starter = No

position
team F G
A 1 2
B 1 1

, , starter = Yes

position
team F G
A 1 1
B 2 1

The output is presented in two distinct two-way slices, separated by the levels of the `starter` variable. The first slice summarizes the players who are **Not Starters** (`starter = No`). Within this slice, we observe that Team A has one Forward (F) and two Guards (G) who are reserves, while Team B has one reserve Forward and one reserve Guard. The second slice, corresponding to `starter = Yes`, shows the frequency of starters. Here, Team A has one starting Forward and one starting Guard, and Team B has two starting Forwards and one starting Guard. This layered view is essential for understanding conditional frequencies.

While the layered output is detailed, the flattened structure provided by `ftable()` often offers a clearer summary, placing all frequencies side-by-side. By applying `ftable()` to the `three_way` object, we achieve this consolidated display, where `team` and `position` define the rows, and `starter` defines the columns:

#convert table to ftable
three_way_ftable <- ftable(three_way)

#view ftable
three_way_ftable

starter No Yes
team position
A F 1 1
G 2 1
B F 1 2
G 1 1

Interpreting this flat table is straightforward. For example, the row labeled "A F" (Team A, Forward) shows counts of 1 under "No" and 1 under "Yes". This means there was **1** player on team A who was a Forward and was not a starter, and **1** player on team A who was a Forward and was a starter. Similarly, the row "B F" (Team B, Forward) indicates **1** non-starter and **2** starters. The flat table greatly simplifies the visualization of all combinations, facilitating rapid interpretation of the data distribution across all three dimensions.

The combination of Team A, Position F, and Non-starter status yielded a count of **1** player.

The combination of Team A, Position G, and Non-starter status yielded a count of **2** players.

The combination of Team B, Position F, and Starter status yielded a count of **2** players.

The total number of Guards on Team B, regardless of starter status, is 1 (No) + 1 (Yes) = **2** players.

Advanced Considerations and Alternative Approaches

While `xtabs()` and `ftable()` provide the foundational methods for creating three-way tables, analysts often require additional customization, such as displaying cell percentages, row percentages, or column percentages instead of raw counts. Although base R allows percentage calculation using the `prop.table()` function applied to the `xtabs()` output, incorporating custom labels and complex formatting often necessitates external packages.

For highly customized or publication-ready tables, the R ecosystem offers several powerful alternatives. The `tidyverse` ecosystem, particularly the `dplyr` package combined with `tidyr`, allows for creating tables through a series of chained operations (using `group_by()`, `summarise()`, and `pivot_wider()`). This approach offers extreme flexibility in defining how variables are aggregated and displayed, though it requires familiarity with tidyverse syntax. Another notable package is `gmodels`, which provides the `CrossTable()` function, capable of generating extensive contingency tables that automatically include cell counts, expected values, chi-square statistics, and various percentage calculations in a single output, streamlining the exploratory data analysis process for categorical variables.

Furthermore, the three-way table itself is often the precursor to visualization. Effective visualization of multi-dimensional categorical data can be achieved using packages like `vcd` (Visualizing Categorical Data), which specializes in graphical representations such as mosaic plots or association plots. These visual tools translate the numerical frequencies found in the three-way table into intuitive graphical displays, making complex interactions easier to grasp than inspecting large numerical tables alone. Ultimately, the choice between base R methods and specialized packages depends on the required level of complexity, formatting, and the need for accompanying

statistical tests.