How to create a contingency table in R

Authored by

stats writer

December 12, 2025

RECOMMENDED CITATION

stats writer (2025). *How to create a contingency table in R*. PSYCHOLOGICAL SCALES. Retrieved from https://scales.arabpsychology.com/?p=107299

The analysis of categorical variables is a fundamental requirement in numerous fields, ranging from social sciences to market research and epidemiology. When dealing with observations categorized by distinct groups, researchers often require a clear, structured method to visualize the joint distribution of these groups. This necessity makes the concept of contingency tables, also known as crosstabulations, an indispensable tool for data summarization and statistical testing. The R programming environment, renowned for its statistical capabilities, provides highly efficient and intuitive built-in tools for generating these summaries, simplifying complex data manipulation tasks into single function calls through the use of the `table()` function.

A contingency table serves as a matrix representation of the counts of observations classified according to two or more classification variables. Unlike simple frequency counts that address marginal distributions (the total counts for individual categories), the contingency table reveals the joint frequency of occurrences, allowing statisticians to explore potential associations or dependencies between the variables in question. This ability to cross-tabulate data is essential for foundational hypothesis testing, particularly when investigating relationships between nominal or ordinal variables where observations are classified into mutually exclusive cells. Understanding the precise implementation of the `table()` function in R is the key to unlocking this powerful analytical technique.

This comprehensive guide details the process of creating a clean, valid contingency table in R, focusing exclusively on base R functions to ensure maximum accessibility and reproducibility. We will walk through the necessary steps: data preparation, the application of the primary `table()` function, the enhancement of the output using `addmargins()` for complete summaries, and a detailed interpretation of the resulting statistical structure. Mastering the creation and interpretation of contingency tables is a core skill for any data analyst utilizing R, enabling deeper insights into the structure and relationships present within their data.

Understanding the Structure of Contingency Tables

A contingency table is fundamentally a statistical tabulation that organizes data by displaying the frequency distribution of two or more categorical variables simultaneously. Its architectural design is meant to summarize the joint relationship between these variables, providing a concise count of how often specific combinations of categories occur within a given dataset. The traditional structure utilizes the categories of one variable to define the rows, and the categories of the second variable to define the columns. This matrix layout ensures every possible pairing of categories has a dedicated cell displaying its corresponding frequency.

The crucial distinction between a contingency table and a simple frequency distribution lies in the data presented within the cells. The values inside the resulting cells represent the **joint frequencies**, meaning the actual count of observations that simultaneously possess both the row

characteristic and the column characteristic. Conversely, the totals found along the edges of the table, known as marginal frequencies, represent the total counts for each category independently of the other variable. For instance, the total number of 'TV' sales, regardless of country, is a marginal frequency, whereas the number of 'TV' sales specifically in 'Country A' is a joint frequency. Accurate statistical inference relies heavily on distinguishing between these two types of frequencies.

While the most common and easily interpretable form is the two-way table (a simple matrix involving two variables), the methodology can be extended to handle multi-way contingency tables involving three or more variables. Though technically possible through R's array structure, higher-dimensional tables become increasingly difficult to visualize and interpret directly without specialized techniques or collapsing dimensions. For foundational analysis and practical business reporting, the two-way contingency table remains the foundational tool for efficiently assessing patterns and potential dependencies between two factors.

Executing the Tabulation with the table() Function

The primary functionality for generating these cross-classified summaries in R is provided by the highly effective base function, `table()`. This function is designed to build a frequency table by counting the number of occurrences of the unique combinations of the input arguments. It processes vectors containing categorical data--which should ideally be factor variables or character strings--and transforms them into a structured matrix output that forms the body of the contingency table. The syntax is minimal, requiring only the variables intended for cross-tabulation.

For generating a two-way contingency table, the `table()` function must be supplied with two distinct vectors of identical length. These vectors correspond directly to the columns of the dataset that contain the categorical data points. R automatically handles the categorization process: it identifies all unique levels within each vector and proceeds to tally the joint frequencies for every permutation of these levels. It is essential to remember that the order in which the variables are specified determines the orientation of the output; the first argument typically defines the rows, and the second defines the columns.

The output object resulting from the `table()` call is an array structure in R, often displayed as a matrix, where the row and column names are automatically assigned based on the unique categories present in the input vectors. This initial output provides all the joint frequencies necessary for analysis but generally excludes the marginal sums, meaning the totals for rows and columns are not immediately present. This is why a subsequent enhancement step, detailed later, is often required to create a publication-ready or statistically complete summary of the contingency analysis.

Setting Up the Example Dataset in R

To illustrate the practical application of these concepts, we will construct a synthetic dataset simulating product orders. This example requires a clear dataset structure that includes two categorical variables: the product type purchased and the country of origin. Our goal is to determine the exact frequency distribution of product sales cross-classified by country, thereby requiring a two-way contingency table for structured visualization.

We establish a dataframe named `df` consisting of 20 simulated product orders. The variables include `order_num`, `product` (categorized as 'TV', 'Radio', or 'Computer'), and `country` (categorized as 'A', 'B', 'C', or 'D'). The construction of this dataframe utilizes base R functions like `data.frame()` and the highly useful `rep()` function, which repeats vector elements a specified number of times. The `rep()` function, paired with a vector of counts in the `times` argument, ensures that our simulated data accurately reflects a predetermined distribution--for example, allocating 9 TVs, 6 Radios, and 5 Computers across the 20 orders, and distributing the 20 orders equally among the four countries (5 orders per country).

This careful preparation of the data is crucial because the integrity of the contingency table rests entirely on the quality and structure of the input vectors. By using the following code, we create and then immediately display the initial dataframe, confirming the correct categorization and distribution of the 20 observations before proceeding to the tabulation step. This initial inspection is a vital part of the data workflow, ensuring that the variables are correctly aligned and ready for cross-tabulation.

The exact R commands used to construct and display the data are presented in the code block below:

Example: Contingency Table in R

Suppose we have the following dataset that shows information for 20 different product orders, including the type of product purchased along with the country that the product was purchased in:

#create data
df <- data.frame(order_num = 1:20,
product=rep(c('TV', 'Radio', 'Computer'), times=c(9, 6, 5)),
country=rep(c('A', 'B', 'C', 'D'), times=5))

#view data
df

order_num product country

1 1 TV A
2 2 TV B
3 3 TV C
4 4 TV D
5 5 TV A
6 6 TV B
7 7 TV C
8 8 TV D
9 9 TV A
10 10 Radio B
11 11 Radio C
12 12 Radio D
13 13 Radio A
14 14 Radio B
15 15 Radio C
16 16 Computer D
17 17 Computer A
18 18 Computer B
19 19 Computer C
20 20 Computer D

Generating the Cross-Tabulation Output

Once the sample data frame `df` is prepared and verified, the primary task of generating the contingency table requires only a single, efficient call to the `table()` function. We specify the two categorical columns of interest--`product` and `country`--accessing them through the standard R syntax `df$column_name`. The function processes these two vectors, counting the joint occurrences, and stores the resulting frequency matrix in a new variable, which we name `table`.

The command is structured as `table <- table(df$product, df$country)`. As noted previously, the arrangement of variables in the argument list dictates the table orientation. Since `df$product` is the first argument, product types ('Computer', 'Radio', 'TV') form the rows of the matrix. Conversely, `df$country` forms the columns ('A', 'B', 'C', 'D'). This convention is vital for consistent and accurate reading of the final output, ensuring that the analyst knows precisely which variable corresponds to the marginal row totals and which corresponds to the column totals.

The resulting table object now contains the core joint frequencies. For instance, examining the cell at the intersection of 'Computer' and 'A' reveals that only 1 computer order originated from Country A, while the cell at 'TV' and 'A' shows 3 orders. This raw tabulation immediately highlights the

distribution patterns and any preliminary differences in sales preferences across the countries. While mathematically complete for joint frequency analysis, this raw output often requires the inclusion of summary statistics--the marginal totals--to be statistically functional and easily interpretable, which leads us to the next step.

The calculation and resulting output of the core table are shown below:

#create contingency table
table <- table(df$product, df$country)

#view contingency table
table

A B C D
Computer 1 1 1 2
Radio 1 2 2 1
TV 3 2 2 2

Enhancing Output with Row and Column Margins

A statistically complete contingency table must include marginal totals, which are the row and column sums that provide the overall frequency distribution for each variable independently. These marginal sums are indispensable not only for calculating proportions but also for verifying the total observation count and serving as the foundation for statistical tests like the chi-squared test. R provides the `addmargins()` function specifically for this purpose, designed to operate directly on table and array objects generated by `table()`.

The `addmargins()` function requires the previously calculated table object (`table`) as its sole argument and automatically performs the summation across both dimensions. It appends a final row labeled 'Sum' containing the column totals and a final column also labeled 'Sum' containing the row totals. The most critical summation is found at the intersection of these two marginal additions--the bottom-right cell--which represents the **grand total** number of observations in the dataset. This grand total should always match the total number of rows (20) in the original dataframe, serving as a robust internal consistency check.

The resulting table object, which we name `table_w_margins`, is now significantly more informative and suitable for presentation. Analysts can instantly determine the total sales volume for each product type (row sums) and the overall order traffic originating from each country (column sums), alongside the detailed joint counts. This enhanced structure not only simplifies interpretation but also prepares the table for subsequent analysis, such as calculating conditional probabilities or standardized residuals.

The R command and the final, comprehensive contingency table with margins are provided below:

#add margins to contingency table
table_w_margins <- addmargins(table)

#view contingency table
table_w_margins

A B C D Sum
Computer 1 1 1 2 5
Radio 1 2 2 1 6
TV 3 2 2 2 9
Sum 5 5 5 5 20

Detailed Interpretation of the Final Results

The interpretation phase is where the summarized data translates into actionable insights regarding the joint distribution of the categorical variables. This requires systematically analyzing the three distinct components of the margin-inclusive table: the grand total, the marginal totals (row and column sums), and the joint frequencies (the inner cell values).

First, the **grand total**, located in the bottom right corner (the intersection of the two 'Sum' totals), confirms the total number of observations analyzed. In this example, the value of 20 ensures that all product orders were correctly processed and included in the tabulation. This total serves as the denominator when calculating overall proportions for the entire dataset.

Second, the **marginal totals** provide crucial univariate summaries. The row sums (the rightmost column) indicate the total frequency for each product type across all countries: 5 Computers, 6 Radios, and 9 TVs were ordered. Similarly, the column sums (the bottom row) show the total order volume originating from each country: 5 products were ordered from country A, 5 from country B, 5 from country C, and 5 from country D. These totals confirm that while overall product demand varies (TVs are most popular), the total number of orders is perfectly balanced across the four countries.

Third, the **joint frequencies**, located in the internal cells, provide the most granular details by describing the specific co-occurrence of categories. For example, the cell corresponding to 'TV' and 'A' shows 3 orders, indicating that Country A demonstrated the highest affinity for TV purchases relative to other countries, and relative to other products within that country. By comparing these joint counts, analysts can begin to infer patterns or potential relationships, such as noticing that Country D has a higher frequency of Computer sales (2) compared to the other countries (all 1), suggesting a non-uniform distribution of product preference across regions.

The value in the bottom right corner shows the **total number of products ordered**: 20.

The values along the right side show the **row sums**: A total of 5 computers were ordered, 6 radios were ordered, and 9 TV's were ordered.

The values along the bottom of the table show the **column sums**: A total of 5 products were ordered from country A, 5 from country B, 5 from country C, and 5 from country D.

The values inside the table show the **number of specific products ordered from each country**: 1 computer from country A, 1 radio from country A, 3 TV's from country A, etc.

Advanced Applications and Proportional Analysis

While the frequency table itself is highly informative, its primary value often lies in its utility as the necessary input for advanced statistical analysis. The most common immediate follow-up to generating a contingency table is to formally test for independence between the two categorical variables using the chi-squared test, which is easily executed in R using the `chisq.test()` function applied directly to the table object. This test quantifies whether the observed joint frequencies deviate significantly from what would be expected if the product type and the country of origin were completely independent of one another.

Furthermore, raw frequency counts can sometimes be misleading, especially when comparing samples of different sizes. Therefore, converting the contingency table into a table of proportions or percentages is frequently required for robust comparative analysis. R's `prop.table()` function facilitates this conversion, offering flexibility in how the proportions are calculated. By default, it uses the grand total as the denominator. However, by specifying the second argument (e.g., `margin = 1` for row proportions or `margin = 2` for column proportions), one can calculate conditional probabilities--for instance, the percentage of orders from Country A that were TVs (column proportion) or the percentage of all TV orders that originated in Country A (row proportion).

Finally, once the statistical insights are generated, clear communication is essential. Although R's console output is statistically sound, visualization packages, notably `ggplot2` or specialized tools within environments like RStudio, can transform the tabular data into compelling graphical forms. Mosaic plots, grouped bar charts, or specialized heatmaps are effective visualization methods that powerfully communicate the patterns of association discovered through the cross-tabulation process, completing the data analysis workflow from raw data to statistical conclusion and final presentation.