Learning to Create Contingency Tables in R for Data Analysis

Authored by

Mohammed looti

November 5, 2025

A **two-way table**, often formally recognized as a contingency table, stands as a cornerstone of statistical analysis. Its primary purpose is to visually and numerically display the joint distribution and **joint frequencies** of observations across two distinct **categorical variables**.

These specialized tables are indispensable tools for statisticians and data scientists seeking to deeply understand the relationship, or association, between two different factors within a dataset. By systematically summarizing the cross-tabulated data, we can efficiently identify underlying patterns, measure dependencies, and visualize distributions that would otherwise remain obscured within large, raw datasets. This structured presentation facilitates rapid interpretation and informs subsequent inferential statistical tests.

To illustrate this concept, consider a common scenario: the following two-way table summarizes the results of a survey administered to 100 individuals, cross-referencing their gender with their preferred sport (Baseball, Basketball, or Football).

In this classic structure, the rows delineate the respondent's gender, while the columns enumerate the specific sport chosen. This organization is vital because it allows for the immediate comparison of preferences and behaviors across different demographic groups, providing insights into conditional probabilities and potential biases in preference distribution:

	Baseball	Basketball	Football	Total
Male	13	15	20	48
Female	23	16	13	52
Total	36	31	33	100

This comprehensive tutorial offers expert guidance and practical, executable examples detailing the methods required to efficiently create, manipulate, and analyze two-way tables within the powerful **R programming environment**.

Example 1: Building a Two-Way Table from a Matrix

One highly effective method for constructing a two-way table, particularly when the data is already pre-aggregated or exists as a summary of counts, involves leveraging the R **matrix** structure. This approach is superior when importing results from external statistical software or when the row and column sums are known prior to computation.

The core process involves two critical steps. First, we must define the matrix itself, ensuring that the numerical elements correctly represent the frequency counts for each cell (the joint

frequencies). Secondly, we utilize the specific R function `as.table()` to formally convert this structured matrix into a true table object. This conversion is crucial because the table object possesses attributes necessary for subsequent statistical functions, such as chi-squared tests or proportion calculations.

It is absolutely essential to define meaningful row and column names within the matrix structure using the `rownames()` and `colnames()` functions before the conversion step. These descriptive names will ultimately serve as the labels for the **categorical variables** in the final, readable two-way table, significantly improving clarity and interpretability.

#create matrix containing pre-aggregated frequency counts
data <- matrix(c(13, 23, 15, 16, 20, 13), ncol=3)

#specify row and column names for clarity
rownames(data) <- c('Male', 'Female')
colnames(data) <- c('Baseball', 'Basketball', 'Football')

#convert the matrix object into a formal table object
data <- as.table(data)

#display the resulting two-way table
data

Baseball Basketball Football
Male 13 15 20
Female 23 16 13

Example 2: Generating Tables Directly from a Data Frame

In the vast majority of real-world data science applications, two-way tables are not built from pre-existing matrices but are rather generated directly from raw, unaggregated observation data stored in a **data frame**. The data frame represents the standard, most versatile structure for storing tabular datasets in R, where observations typically occupy rows and variables are stored in columns.

To successfully create a two-way contingency table from this raw structure, we rely on the fundamental R function, `table()`. This powerful function is designed for cross-tabulation: it accepts two factor variables (columns) from the data frame and efficiently traverses the entire dataset to automatically calculate and summarize the necessary joint frequencies, producing the desired contingency table output.

The following example first demonstrates the construction of a small, representative data frame containing five individual survey responses. Subsequently, we utilize the `table()` function, referencing the data frame's specific columns (`df$gender` and `df$sport`), to perform the cross-tabulation and yield the resultant frequency counts, showcasing the speed and simplicity of this method.

```
#create a sample data frame with raw observations
df <- data.frame(sport=c('Base', 'Base', 'Bask', 'Foot', 'Foot'),
gender=c('Male', 'Female', 'Male', 'Male', 'Female'))

#view the raw data frame content
df

#create two way table by cross-tabulating the 'gender' and 'sport' columns
data <- table(df$gender, df$sport)

#display the resulting two way table
data
```

Base Bask Foot
Female 1 0 1
Male 1 1 1

Example 3: Calculating Marginal Frequencies

While the joint frequencies residing in the body of the table reveal the combined counts for pairs of categories, **marginal frequencies** (also known as marginal sums) are equally essential for statistical completeness. These sums provide vital information regarding the overall distribution of each variable independently, without conditioning on the other variable.

These marginal totals are derived by simply summing the counts across either the rows or the columns. In R, the dedicated function `margin.table()` is the most efficient and standard tool for calculating these sums from a contingency table object. The key to using this function effectively lies in correctly specifying the `margin` argument, which dictates the axis along which the aggregation should occur.

Specifically, setting `margin=1` instructs R to calculate the row totals. This output provides the total count for each category defined by the row variable (e.g., the total number of Male and Female respondents). Conversely, setting `margin=2` directs the function to sum across the columns, yielding the total count for each category defined by the column variable (e.g., the total count for each specific sport, regardless of gender).

#re-create the sample frequency data matrix
data <- matrix(c(13, 15, 20, 23, 16, 13), ncol=3)
rownames(data) <- c('Male', 'Female')
colnames(data) <- c('Baseball', 'Basketball', 'Football')

#find the marginal sums for the row variable (Gender) using margin=1
margin.table(data, margin=1)

Male Female
49 51

#find the marginal sums for the column variable (Sport) using margin=2

margin.table(data, margin=2)

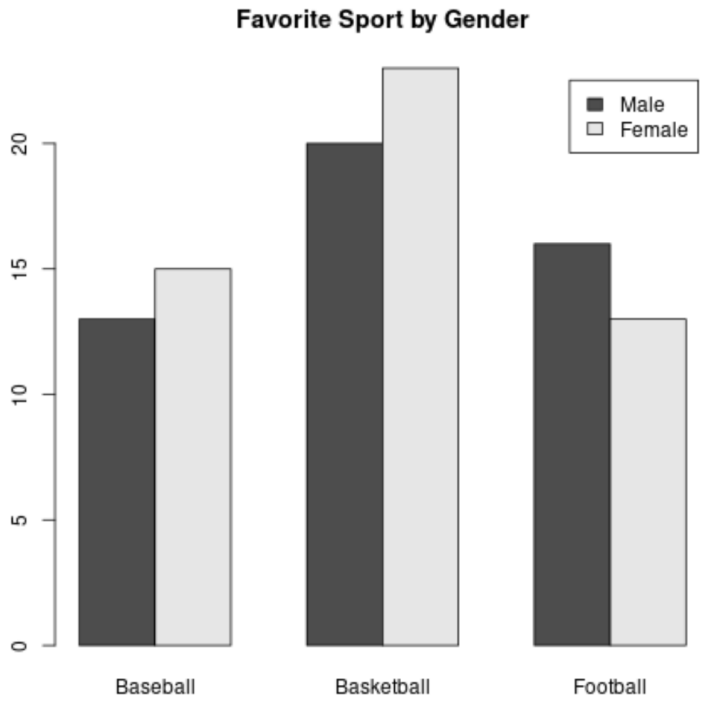
Baseball Basketball Football
28 43 29

Example 4: Visualizing Two-Way Frequencies

While precise numerical tables are fundamental for calculation, visualization is indispensable for communicating results and immediately recognizing underlying data patterns. The **R programming environment** offers robust tools for graphically representing the frequency data contained within a two-way table, primarily through the use of the barplot and the mosaic plot.

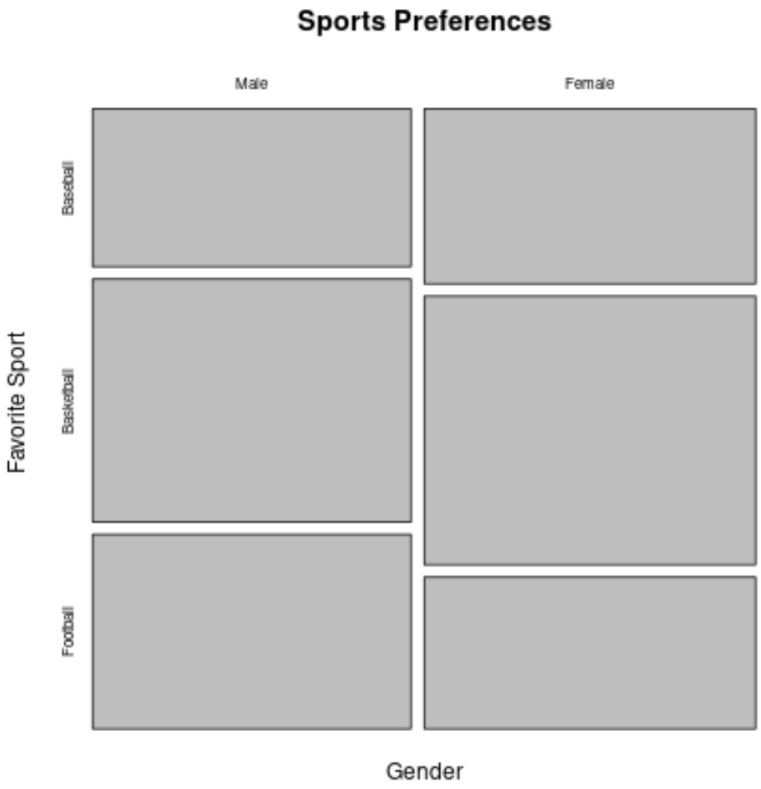
The **barplot** is particularly suited for comparing the conditional distributions of one variable across the distinct categories of the second variable. By supplying the two-way table object directly to the `barplot()` function and setting the argument `beside=TRUE`, we ensure that the bars corresponding to different row categories (e.g., Male vs. Female) are displayed side-by-side for each column category (e.g., each sport). This configuration provides a clear visual comparison of preferences or outcomes within specific groups.

barplot(data, legend=True, beside=True, main='Favorite Sport by Gender')



Alternatively, the **mosaic plot** offers a powerful, proportional visualization of the contingency table. In a mosaic plot, the area of each rectangular tile is designed to be directly proportional to the frequency count found in the corresponding cell of the two-way table. This visual weighting makes the mosaic plot exceptionally effective for quickly assessing potential dependencies or independence between the two **categorical variables**, as deviations from expected independence are visually magnified by the relative size of the tiles.

mosaicplot(data, main='Sports Preferences', xlab='Gender', ylab='Favorite Sport')



Mastering the efficient creation and precise interpretation of **two-way tables** is a foundational skill necessary for performing robust statistical analysis and effective data exploration in R. These detailed examples provide a solid, repeatable framework for handling both pre-aggregated count data and raw, unaggregated inputs, ensuring accuracy in subsequent analytical modeling.