Learn How to Reorder Factor Levels in R with fct_relevel()

Authored by

Mohammed looti

November 13, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *Learn How to Reorder Factor Levels in R with fct_relevel()*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=24073

In the realm of statistical computing and data analysis, particularly when utilizing the R programming language, managing categorical data is a fundamental requirement. This data is typically stored and manipulated using **factor variables**. Factors are essential tools in R, allowing users to efficiently handle data that falls into distinct groups or levels, such as genders, geographic regions, or--as we will explore--sports teams.

While factors are incredibly powerful, their default behavior often involves ordering levels alphabetically or based on the order of appearance in the dataset. However, in practical data visualization or modeling scenarios, this default order is rarely optimal. Analysts frequently need to specify a custom order for these factor levels--perhaps to set a specific baseline category for regression analysis or to ensure logical presentation in a chart.

Addressing this need for precise factor manipulation, the forcats package (a core component of the tidyverse ecosystem) provides a suite of specialized functions. One of the most frequently used functions for rearranging factor level order is **fct_relevel()**. This function is specifically designed to perform targeted reordering, ensuring your data presentation and statistical models behave exactly as intended.

Understanding Factor Variables and Level Order in R

Before diving into the mechanics of reordering, it is vital to grasp how factor variables function within R. Unlike character vectors, which treat categories as simple text strings, factors treat categories as predefined levels. These levels have an underlying integer representation, and crucially, they possess a defined order. This order impacts how the data is processed, especially in advanced statistical procedures like ANOVA or linear regression, where the first level often serves as the reference category.

The importance of explicit level ordering cannot be overstated. Consider a scenario involving survey data where responses are "Poor," "Average," and "Excellent." If R orders these alphabetically ("Average," "Excellent," "Poor"), any subsequent visualization will be misleading, and any statistical test relying on ordinal relationships will be flawed. Manually setting the order to "Poor," "Average," "Excellent" ensures both the visual output and the statistical interpretation are accurate and meaningful. This is precisely where functions like **fct_relevel()** become indispensable tools for data scientists.

When working with large datasets, factors protect against data entry errors by restricting values to only the predefined levels. If an analyst attempts to introduce a value not listed in the levels (e.g., 'Averag'), R will typically treat this as missing data (NA) or reject the assignment, depending on the context. This structural integrity makes factors the preferred way to handle true categorical variables in R, distinguishing them from simple character strings which lack this inherent structure and ordering mechanism.

The Syntax and Setup of fct_relevel()

To begin using this powerful functionality, we must first ensure the necessary package is available. The **fct_relevel()** function is housed within the forcats package. If you are not already using the tidyverse (which includes forcats), you must install and load this package explicitly.

The basic syntax for the **fct_relevel()** function is straightforward, yet flexible, allowing for precise control over level placement:

fct_relevel(.f, ..., after = 0L)

Understanding the arguments helps maximize its utility:

.f: This is the primary argument, representing the factor variable whose levels you intend to modify.
...: This accepts one or more character strings corresponding to the specific level names you wish to move. These levels will be placed in the order they are listed here.
after: This optional integer argument specifies the position (index) after which the listed levels (in **...**) should be inserted. By default, **after = 0L**, which means the levels are placed at the very beginning (position 1). If **after = 1L**, the levels are placed after the current first level, and so on. This argument provides granularity in placement.

Before executing any factor manipulation, it is important to verify that the **forcats** package is installed. If it is not, the installation is simple, typically requiring only one command, as shown below. Once installed, it must be loaded into the R session using the `library()` function to make **fct_relevel()** accessible for reordering factor levels without error.

Note: Before using the **fct_relevel()** function you may need to first install the **forcats** package by using the following syntax:

install.packages('forcats')

Once the **forcats** package is installed, you can use the **fct_relevel()** function to reorder factor levels without encountering any errors, significantly streamlining your data preparation workflow in R programming language.

Example 1: Moving a Specific Factor Level to the Front

One of the most common tasks in factor manipulation is promoting a specific category to the first position. This is often necessary when setting a baseline for statistical comparisons or highlighting a primary category in a visual output. Since the default value for the `after` argument in **fct_relevel()** is `0L`, we can achieve this objective by simply listing the desired level in the `...`

argument.

Suppose we create a factor variable named `my_teams` that contains the names of various basketball teams. We define the levels explicitly, establishing an initial order: Mavs, Nets, Heat, Kings.

```
#create factor variable
my_teams <- factor(c('Mavs', 'Nets', 'Nets', 'Kings', 'Mavs', 'Heat'),
levels=c('Mavs', 'Nets', 'Heat', 'Kings'))

#view factor variable and its initial levels
my_teams

Mavs Nets Nets Kings Mavs Heat
Levels: Mavs Nets Heat Kings
```

As observed in the output, the `my_teams` factor variable currently maintains the following order of levels:

Mavs Nets Heat Kings

For our analysis, we determine that the "Heat" level should be prioritized, perhaps because it represents the control group or the most recent champion. We need to move "Heat" to the first position of the factor levels, effectively changing the reference point for subsequent statistical modeling.

We can use the fct_relevel() function with the following syntax to execute this reordering. Note that we call the package using `library(forcats)` and then specify the factor variable (`my_teams`) and the level to be moved (`'Heat'`). Since we omit the `after` argument, it defaults to `0L`, placing 'Heat' at the beginning.

```
library(forcats)

#create factor variable (re-run for clarity)
my_teams <- factor(c('Mavs', 'Nets', 'Nets', 'Kings', 'Mavs', 'Heat'),
levels=c('Mavs', 'Nets', 'Heat', 'Kings'))

#move 'Heat' to front of level order
my_teams <- fct_relevel(my_teams, 'Heat')

#view updated factor variable
my_teams
```

Mavs Nets Nets Kings Mavs Heat
Levels: Heat Mavs Nets Kings

Upon reviewing the output, we successfully observe that **Heat** has been moved to the first position of the factor levels. Crucially, the relative order of all other levels--Mavs, Nets, and Kings--has remained unchanged. This demonstrates the targeted precision of the fct_relevel() function, which only affects the placement of the specified level(s) while preserving the hierarchy of the remaining categories.

Example 2: Moving a Factor Level After a Specific Position

While placing a level at the front is common, often we require a level to be placed specifically within the existing order. This is achieved using the `after` argument, which provides fine-grained control over the resulting factor structure. The `after` argument takes an integer specifying the zero-indexed position *after* which the new level(s) should be inserted.

Let us return to our factor variable `my_teams` in its original state, observing the initial level structure:

#create factor variable
my_teams <- factor(c('Mavs', 'Nets', 'Nets', 'Kings', 'Mavs', 'Heat'),
levels=c('Mavs', 'Nets', 'Heat', 'Kings'))

#view factor variable
my_teams

Mavs Nets Nets Kings Mavs Heat
Levels: Mavs Nets Heat Kings

In this scenario, we decide that **Heat** should not be the first level, but rather the second. To achieve this, we need to move "Heat" to the position directly after the current first level, which is "Mavs." Since "Mavs" is at index position 1, we must set the `after` argument to `1L`. This instructs R to place the specified level ('Heat') immediately following the first position.

We utilize the **fct_relevel()** function again, explicitly setting the `after` parameter. If we wanted to place 'Heat' after 'Nets' (which is position 2), we would set `after = 2L`. The value of `after` corresponds to the index position in the original factor levels list.

library(forcats)

#create factor variable (re-run for clarity)

```
my_teams <- factor(c('Mavs', 'Nets', 'Nets', 'Kings', 'Mavs', 'Heat'),
levels=c('Mavs', 'Nets', 'Heat', 'Kings'))

#move 'Heat' to after position 1 (after 'Mavs')
my_teams <- fct_relevel(my_teams, 'Heat', after = 1)

#view updated factor variable
my_teams
```

```
Mavs Nets Nets Kings Mavs Heat
Levels: Mavs Heat Nets Kings
```

The resulting factor levels now show **Heat** moved to the second position, directly following **Mavs**. The initial order of the remaining levels (Nets and Kings) is maintained sequentially after the newly placed level. This demonstrates how the `after` argument provides dynamic control, enabling the user to slot any level into a precise location within the factor hierarchy, a functionality crucial for generating highly customized reports and statistical outputs.

Advanced Applications and Further Resources

The utility of **fct_relevel()** extends beyond moving a single level. It can be used to reorder multiple levels simultaneously. If you list several levels in the `...` argument, they will be moved as a block and placed in the order they were specified, relative to the `after` position. For instance, `fct_relevel(my_teams, 'Kings', 'Heat', after = 0)` would place 'Kings' first, 'Heat' second, and then the rest of the levels would follow.

While **fct_relevel()** focuses on moving levels relative to their current position, the forcats package offers other powerful functions for complete factor control. For instance, `fct_reorder()` allows reordering based on the values of another variable (e.g., ordering teams based on their average score), which is often more useful for visualization. Similarly, `fct_relevel()` is just one piece of a complete factor manipulation toolkit designed to simplify common data wrangling challenges in R programming language.

Mastering factor manipulation is a key skill for any intermediate R user, ensuring that data is not only correctly stored but also correctly interpreted by statistical models and presented clearly in visualizations. The precision offered by functions like fct_relevel() function allows analysts to dictate the narrative of their data from initial preparation through final output.

For those interested in exploring the complete depth and breadth of factor handling available in R, including more complex reordering rules and merging techniques, consulting the official documentation is highly recommended.

Note: You can find the complete documentation for the **fct_relevel()** function from the **forcats** package here.

Additional Resources for R Data Manipulation

The following tutorials explain how to perform other common tasks in R, building upon the foundational knowledge of factor manipulation:

How to combine multiple factor levels into one category.
Using `fct_recode()` to rename factor levels easily.
Techniques for dealing with unused factor levels in datasets.

<!--

Featured Posts

-->