Learning to Impute Missing Data with the fill() Function in R

Authored by

Mohammed looti

November 13, 2025

Introduction to Handling Missing Data in R

In the field of **R** programming and data analysis, analysts frequently encounter datasets afflicted by incomplete or **missing values**. These missing entries, often represented as `NA` (Not Available) within an **R data frame**, pose significant challenges to statistical modeling and accurate data interpretation. Addressing these gaps is a critical step in the data cleaning pipeline, often requiring methods to impute or propagate existing values to fill the voids. One common and highly efficient requirement is propagating the last observed value (LOCF - Last Observation Carried Forward) or the next observed value (NOCB - Next Observation Carried Backward) to maintain data continuity, especially in time-series or sequential data structures.

While various complex **imputation** techniques exist, the simplest and most robust approach for propagating adjacent values is provided by the modern **tidyr** package, a cornerstone of the Tidyverse ecosystem. This article focuses specifically on mastering the `fill()` function, a powerful tool designed precisely for carrying forward or backward non-missing observations within specified columns of a **data frame**.

Introducing the fill() Function and the tidyr Package

The `fill()` function is specifically engineered to handle sequential missing data propagation. It operates on the principle of replacing an `NA` with the closest non-missing value in a specified direction. This functionality is crucial when dealing with datasets where the missing observation is likely to be identical to the one immediately preceding or following it, such as when data collection equipment temporarily fails or when categorical identifiers are only recorded once for a sequence of rows.

To access this powerful functionality, users must ensure they have installed and loaded the **tidyr** package. If the package is not yet available in your local **R** library, the installation process is straightforward, requiring a simple command execution:

install.packages('tidyr')

Once the **tidyr** package is successfully installed and loaded using `library(tidyr)`, the `fill()` function becomes available. The default behavior of the **fill()** function is to fill in missing values in a `"down"` manner, meaning the value that appears directly above the missing value will be used to fill in the missing value. The simplicity and efficiency of this function make it a preferred method for preliminary data cleaning and preparation.

Understanding the fill() Syntax and Parameters

The formal syntax of the `fill()` function is designed to be highly intuitive, adhering to the standard Tidyverse philosophy of piping and clear argument specification. Understanding its parameters is key to leveraging its full power, particularly the directional argument that dictates the imputation method.

The basic structure is as follows:

fill(data, ..., .direction = c("down", "up", "downup", "updown"))

The function requires the following essential arguments:

data: The name of a **data frame** or tibble object that contains the **missing values** that need to be addressed.
...: This is where the user identifies the specific columns within the data frame that should be targeted for filling. You can select one column or multiple columns simultaneously.
.direction: This crucial argument determines the method of propagation. By default, it is set to `"down"`.

The `.direction` parameter offers four distinct options for data propagation:

`"down"` (Default): Performs a Last Observation Carried Forward (LOCF). Missing values are replaced by the most recent non-missing value appearing immediately above them in the same column.
`"up"`: Performs a Next Observation Carried Backward (NOCB). Missing values are replaced by the next non-missing value appearing immediately below them.
`"downup"`: First applies the `"down"` propagation, and then, if any `NA` values still remain (e.g., they were at the beginning of the column), it applies the `"up"` propagation.
`"updown"`: First applies the `"up"` propagation, followed by the `"down"` propagation for any remaining missing entries.

The default behavior, `"down"`, is the most frequently used method, as it ensures that the missing entry inherits the property of the observation that chronologically preceded it. However, the flexibility offered by the other directional arguments allows the function to adapt to diverse data structures and analytical needs.

Practical Example: Setting up the Data Frame with Missing Data

To illustrate the practical application of the `fill()` function, let us first construct a sample **data frame** in **R**. This dataset, which simulates basketball player statistics, intentionally includes several `NA` entries to demonstrate how the function handles gaps in sequential data:

#create data frame

```
df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'),
points=c(99, 68, 86, 88, 95, 74, 78, 93),
assists=c(22, NA, 31, 35, 34, NA, 28, 31),
rebounds=c(30, NA, NA, 24, 30, 36, 30, 29))

#view data frame
df
```

team points assists rebounds
1 A 99 22 30
2 A 68 NA NA
3 A 86 31 NA
4 A 88 35 24
5 B 95 34 30
6 B 74 NA 36
7 B 78 28 30
8 B 93 31 29

Notice that several of the columns in the data frame have **missing values** (NA) in various locations. The columns represent:

team: The name of the team each player belongs on.
points: The total points scored by the player.
assists: The total assists made by the player (our target for imputation).
rebounds: The total rebounds by the player.

Suppose that we would like to fill in each of the missing values in the **assists** column of the data frame with the most recently available value above it in each column. This scenario perfectly aligns with the default operation of the `fill()` function.

Implementing Directional Filling (The "down" Default)

The most straightforward and common method for handling sequential **missing values** is the Last Observation Carried Forward (LOCF), achieved by using the default `.direction = "down"` setting. This approach assumes that if a value is missing, the preceding valid value in the sequence is the best estimate. We will apply this to the **assists** column of our sample data frame using the pipe operator (`%>%`) from the **tidyr** package.

By executing the following code, we instruct **R** to traverse the **assists** column from top to bottom, replacing each `NA` with the value immediately above it:

library(tidyr)

#fill in missing values in assists columns
df %>% fill(assists)

team points assists rebounds
1 A 99 22 30
2 A 68 22 NA
3 A 86 31 NA
4 A 88 35 24
5 B 95 34 30
6 B 74 34 36
7 B 78 28 30
8 B 93 31 29

Notice that each of the missing values in the **assists** column of the data frame have been filled in with the most recently available value above it. For example, the `NA` in row 2 is now 22, carried down from row 1. This demonstrates a successful LOCF operation, providing a clean and complete column ready for further analysis.

Reversing the Flow: Using the "up" Direction

In contrast to LOCF, sometimes the data context demands the use of a Next Observation Carried Backward (NOCB), achieved by explicitly setting the direction to `"up"`. This strategy is essential when the missing observation should logically inherit the value of the measurement immediately following it. This might be necessary, for example, if data logging starts slightly before the first valid measurement is captured.

We can easily modify our previous command to utilize the `"up"` direction for the **assists** column:

library(tidyr)

#fill in missing values in assists columns
df %>% fill(assists, .direction="up")

team points assists rebounds
1 A 99 22 30
2 A 68 31 NA
3 A 86 31 NA
4 A 88 35 24
5 B 95 34 30

6 B 74 28 36
7 B 78 28 30
8 B 93 31 29

Notice that each of the missing values in the **assists** column of the data frame have now been filled in with the most recently available value below it. For instance, the `NA` in row 2 is replaced by 31 (the value from row 3). The choice between `"up"` and `"down"` must always be informed by the underlying meaning and structure of the data and the specific requirements of the analysis.

Conclusion and Additional Resources

The `fill()` function from the **tidyr** package provides a remarkably efficient and readable solution for propagating values to handle sequential **missing values** in **R**. Its directional arguments--`"down"`, `"up"`, `"downup"`, and `"updown"`--offer precise control over how imputation occurs across columns, making it an indispensable tool for data preparation.

For more complex scenarios where gaps exist both at the beginning and the middle of the column, using hybrid directions like `"downup"` ensures comprehensive imputation. This approach first handles gaps by looking backward (down) and then addresses any remaining initial gaps by looking forward (up), maximizing the utilization of available data. Analysts interested in exploring further advanced functionality of this tool should consult the complete documentation for the **fill()** function from the **tidyr** package. Understanding these capabilities is fundamental for becoming proficient in modern **R** data wrangling.

The following tutorials explain how to perform other common tasks in R:

Additional Resources

The following tutorials explain how to perform other common tasks in R:

<!--

Featured Posts

-->