# How to Calculate Correlation in R Despite Missing Values: A Step-by-Step Guide

Authored by

**stats writer**

November 20, 2025

## Introduction: The Challenge of Missing Data in R

When conducting statistical analysis, especially involving relationships between variables, dealing with missing values is a fundamental challenge. The presence of these gaps, typically represented by **NA** (Not Available) in the R programming language, can significantly impede calculations. If not explicitly managed, standard functions designed for complete datasets will often fail or, worse, return misleading results. Specifically, when attempting to compute the correlation between two variables using the base **R** function, the presence of even a single pair where one or both observations are missing will usually cause the function to return **NA** for the entire calculation. This behavior is intentional, forcing the analyst to make a deliberate choice about how to handle the incomplete data.

Fortunately, **R** provides robust mechanisms within its core statistical functions to seamlessly handle missing values without requiring manual data scrubbing beforehand. While one traditional approach involves using functions like **na.omit()** to create a subset of the data entirely free of missing observations, a more direct and often preferred method utilizes specific arguments within the correlation function itself. These arguments determine the computational strategy used when encountering **NA**s, allowing researchers to choose between eliminating cases entirely or using all available information on a pair-by-pair basis, depending on the analytical goals and the nature of the data loss. Understanding these distinct methods is key to generating reliable correlation coefficients in real-world datasets that are invariably imperfect.

This comprehensive guide will detail the two primary, most efficient methods for calculating correlation coefficients and full correlation matrices in **R** when missing values are present. We will focus specifically on the powerful **use** argument within the standard cor() function, exploring the implications of setting it to **'complete.obs'** for simple bivariate correlation and **'pairwise.complete.obs'** for generating complex correlation matrices. By mastering these techniques, you can ensure that your statistical models are both accurate and utilize the maximum available information from your dataset, thereby maintaining the integrity of your subsequent analyses.



## Understanding How R Handles Missing Data (NA)

In the world of statistical computing, the representation of truly unknown or missing data points is standardized using the symbol **NA**. It is essential to recognize that **NA** is not the same as zero or an empty string; it is a dedicated placeholder for "Not Available." When any arithmetic operation or statistical calculation in **R** involves an **NA** value, the result of that operation is typically also **NA**, a principle known as the "NA propagates" rule. This strict propagation rule is designed as a safety feature, preventing the silent generation of potentially biased results by ignoring the fact that information is absent. For instance, if you try to sum a vector containing **NA**, the result will be **NA**

unless you specifically instruct the summing function (like **sum()**) to remove the missing values.

The standard cor() function adheres rigorously to this propagation principle. When calculating the Pearson correlation coefficient, which is the default method, the function requires a complete set of paired observations for every data point involved in the calculation. If, for a specific row, one variable has a numerical value but the corresponding paired variable holds an **NA**, that entire pair is deemed incomplete. If this incompleteness exists anywhere in the vectors used for calculation, the cor() function will stop and return **NA**, indicating that the correlation cannot be computed without a strategy for handling the missing inputs. This is why analysts must explicitly tell **R** how to proceed when confronting **NA**s, which is achieved through the **use** argument.

The choice of missing data handling strategy often depends on the type of analysis being performed and the extent of the missingness. While some highly specialized packages offer sophisticated imputation methods (where missing data are estimated), for standard correlation calculations, **R** provides two simple yet powerful deletion methods: listwise deletion (using **'complete.obs'**) and pairwise deletion (using **'pairwise.complete.obs'**). Listwise deletion is typically safer for subsequent modeling but can drastically reduce sample size, while pairwise deletion maximizes data usage for the correlation calculation itself, though it can lead to a non-positive definite correlation matrix in rare cases. Selecting the correct method begins with understanding the specific parameters available within the cor() function.

## Method 1: Calculating Simple Correlation Using `complete.obs`

When calculating the correlation coefficient between exactly two variables, X and Y, the **use='complete.obs'** argument instructs **R** to employ listwise deletion specific to those two variables. Listwise deletion means that **R** will only consider observations (rows) where values for **both** X and Y are present (i.e., neither is **NA**). If a row contains an **NA** in either X or Y, that entire row is discarded from the calculation of the correlation coefficient. This ensures that the final coefficient is calculated based on a perfectly clean subset of the data, guaranteeing the integrity of the covariance and standard deviation calculations that underpin the Pearson correlation coefficient.

This method is highly recommended when the missingness is relatively small, or when you intend to use the same subset of data for subsequent statistical tests, maintaining consistency across analyses. By ensuring that every data point used in the calculation is complete, **'complete.obs'** avoids the mathematical complications that arise from mixing subsets. While it might reduce the effective sample size (n) used for the correlation, the resulting coefficient is robust and easily interpretable, as it represents the relationship within the shared, non-missing portion of the data.

The syntax for implementing this method is straightforward and directly integrates into the standard cor() function call. You provide the two variables, X and Y, followed by the essential **use**

parameter. The code chunk below summarizes the required structure:

**cor(x, y, use='complete.obs')**

This approach simplifies the process significantly, removing the need for preliminary data cleaning steps like calling **na.omit()** on a temporary data frame. Instead, the cleaning and calculation are performed simultaneously within the function call, leading to cleaner and more efficient statistical programming. The next sections will provide a concrete example demonstrating how the cor() function behaves both when this argument is omitted and when it is correctly applied, illustrating the dramatic difference in output.

## Detailed Walkthrough: Attempting `cor()` without Handling Missing Values

To fully appreciate the necessity of specifying a missing data handling strategy, consider a scenario where we have two vectors, **x** and **y**, containing several missing values (**NA**s). We attempt to calculate the correlation coefficient directly using the standard cor() function without any further arguments. As previously discussed, **R**'s strict propagation rule dictates the outcome in such a situation. The code below sets up the vectors and attempts the initial calculation:

**#create two variables**
**x <- c(70, 78, 90, 87, 84, NA, 91, 74, 83, 85)**
**y <- c(90, NA, 79, 86, 84, 83, 88, 92, 76, 75)**

#attempt to calculate correlation coefficient between x and y
cor(x, y)

NA

As clearly demonstrated by the output, the function returns **NA**. This result is not an error; rather, it is **R** informing the user that the correlation calculation cannot be completed under the default rules because the dataset is incomplete. Specifically, the observation at index 6 in vector **x** is missing (**NA**), and the observation at index 2 in vector **y** is missing (**NA**). Because the standard cor() function requires every single pair of data points to be available for computation, the moment it encounters any **NA** in either vector, the result defaults to **NA**, effectively stopping the analysis.

This initial failure serves as a critical diagnostic check. It confirms that the data contains gaps and that the analyst must implement a strategy to handle them. For many beginners in **R**, seeing this **NA** output can be frustrating, but it is a necessary step that highlights the need for explicitly defining the data handling policy. We cannot simply proceed with subsequent statistical modeling until a valid numerical correlation coefficient is calculated. The solution lies in applying the

**use='complete.obs'** argument to instruct **R** to proceed using only the valid, paired data points.

## Detailed Walkthrough: Successfully Using the `complete.obs` Argument

To successfully calculate the correlation coefficient, we re-run the cor() function on the vectors **x** and **y**, but this time, we include the crucial argument **use='complete.obs'**. This argument triggers listwise deletion relative to the two input vectors, meaning **R** first identifies which rows have values present in both **x** and **y**, temporarily removes the incomplete rows, and then proceeds with the standard calculation on the reduced, complete dataset. This targeted approach allows the correlation to be computed successfully.

The revised code block below demonstrates the successful execution and the resulting numerical output:

```
#create two variables
x <- c(70, 78, 90, 87, 84, NA, 91, 74, 83, 85)
y <- c(90, NA, 79, 86, 84, 83, 88, 92, 76, 75)

#calculate correlation coefficient between x and y
cor(x, y, use='complete.obs')
```

-0.4888749

The result, **-0.4888749**, is the calculated Pearson correlation coefficient for the variables **x** and **y**, based only on the observations where both values were available. This indicates a moderate negative linear relationship between the two variables. It is vital to remember that this coefficient is calculated using a smaller effective sample size than the original data frame, which is an inherent trade-off of using listwise deletion. The primary advantage here is the statistical purity of the calculation; every element contributing to the correlation is based on actual, observed data points, ensuring a mathematically consistent result.

In summary, for simple bivariate correlation between two vectors containing missing values, the **use='complete.obs'** argument provides a reliable and clean method for obtaining a robust correlation coefficient. This technique is especially useful when the research focus is strictly on the relationship between these two specific variables, and the loss of a few observations is deemed acceptable compared to the alternative of performing complex imputation procedures.

## Method 2: Calculating the Correlation Matrix Using `pairwise.complete.obs`

When computing a correlation matrix for multiple variables (e.g., X, Y, Z), relying solely on **'complete.obs'** can be overly restrictive, especially if missingness is scattered across the dataset.

Applying **'complete.obs'** to a data frame means that if even a single row has an **NA** in *any* variable, that row is excluded from calculating *all* correlation pairs, potentially leading to a massive loss of valuable data and shrinking the effective sample size to an unacceptably low level.

To maximize the use of available data when computing a correlation matrix, we must utilize the argument **use='pairwise.complete.obs'**. This method employs pairwise deletion. Instead of discarding an entire row due to one missing value, it calculates each individual correlation coefficient (e.g., correlation(X, Y), correlation(X, Z), correlation(Y, Z)) based only on the subset of data where those specific two variables are present. Thus, the sample size used to calculate correlation(X, Y) might be different from the sample size used to calculate correlation(Y, Z), depending on where the missing values lie.

The key benefit of **'pairwise.complete.obs'** is data maximization; it extracts the maximum amount of information for every possible pair of variables. However, analysts must be aware of the inherent risk: because different correlation coefficients within the matrix are calculated on different subsets of the data, the resulting correlation matrix is not guaranteed to be positive definite. While this is rarely an issue in descriptive statistics, it can cause problems if the matrix is subsequently used in multivariate modeling techniques like factor analysis or structural equation modeling, which often require a positive definite matrix. Nonetheless, for initial exploratory analysis, this method is highly efficient.

## Practical Application: Utilizing `pairwise.complete.obs` for Robust Results

Let us examine the multivariate case using a data frame, **df**, with three variables: **x**, **y**, and **z**. As demonstrated earlier, attempting to calculate the correlation matrix without specifying a handling strategy results in an output matrix full of **NA**s because no single row contains complete data across all three variables. To successfully overcome this limitation and maximize data use, we apply **use='pairwise.complete.obs'** to the data frame. This instructs **R** to calculate each of the three necessary cross-correlations (X-Y, X-Z, Y-Z) using every observation pair available for that specific calculation.

The following code block demonstrates the setup and the successful generation of the fully populated correlation matrix:

```
#create data frame with some missing values
df <- data.frame(x=c(70, 78, 90, 87, 84, NA, 91, 74, 83, 85),
y=c(90, NA, 79, 86, 84, 83, 88, 92, 76, 75),
z=c(57, 57, 58, 59, 60, 78, 81, 83, NA, 90))

#create correlation matrix for variables using only pairwise complete observations
cor(df, use='pairwise.complete.obs')
```

x y z
x 1.0000000 -0.4888749 0.1311651
y -0.4888749 1.0000000 -0.1562371
z 0.1311651 -0.1562371 1.0000000

The resulting matrix now provides clear numerical estimates for all bivariate relationships: X and Y show a moderate negative correlation (-0.489), X and Z show a weak positive correlation (0.131), and Y and Z show a weak negative correlation (-0.156). The coefficients for each pairwise combination of variables in the data frame are successfully shown, confirming the efficacy of the **'pairwise.complete.obs'** argument in sparse data environments. This method is exceptionally useful for exploratory data analysis, where the primary goal is to quickly assess the linear relationships between variables while minimizing the loss of valuable data points.

## Summary and Best Practices for Handling NA in Statistical Analysis

Calculating correlation in R when faced with missing values (**NA**s) requires a deliberate strategy governed by the **use** argument within the cor() function. Failing to specify this argument will almost always lead to an uninformative **NA** result, halting the analytical process. By utilizing either **'complete.obs'** or **'pairwise.complete.obs'**, analysts can successfully generate reliable correlation coefficients based on the available data.

The choice between these two powerful methods depends entirely on the scope of your analysis:

**use='complete.obs':** Best for simple, two-variable (bivariate) correlation calculations. It employs listwise deletion specific to the two variables involved, resulting in a single, robust sample size for that particular coefficient. It is a statistically conservative and safe approach for most initial correlation checks.

**use='pairwise.complete.obs':** Essential for generating a full correlation matrix for three or more variables. This method maximizes data utilization by calculating each correlation pair independently, though it introduces the caveat that the resulting matrix may not be positive definite, potentially impacting subsequent advanced multivariate modeling.

In conclusion, mastering these simple yet critical arguments ensures that your statistical work in **R** remains accurate and robust, even when dealing with the inevitable imperfections of real-world data. Always document the method used (either listwise or pairwise deletion) when reporting your results, as this transparency is a cornerstone of sound statistical practice.

You can use the following standard syntaxes to calculate correlation coefficients in **R** when one or more variables have missing values:

**Method 1: Calculate Correlation Coefficient with Missing Values Present**

**cor(x, y, use='complete.obs')**

**Method 2: Calculate Correlation Matrix with Missing Values Present**

**cor(df, use='pairwise.complete.obs')**

The following examples recap how these methods are utilized in practice within the **R** environment.

## Example 1: Demonstrating Bivariate Correlation with Missing Values

Suppose we attempt to use the **cor()** function to calculate the Pearson correlation coefficient between two variables when missing values are present. If we do not specify the handling method, the function returns **NA**:

**#create two variables**
**x <- c(70, 78, 90, 87, 84, NA, 91, 74, 83, 85)**
**y <- c(90, NA, 79, 86, 84, 83, 88, 92, 76, 75)**

#attempt to calculate correlation coefficient between x and y
cor(x, y)

NA

The **cor()** function returns **NA** since we didn't specify how to handle missing values. To avoid this issue and apply listwise deletion on these two vectors, we use the argument **use='complete.obs'** so that **R** knows to only use pairwise observations where both values are present:

**#create two variables**
**x <- c(70, 78, 90, 87, 84, NA, 91, 74, 83, 85)**
**y <- c(90, NA, 79, 86, 84, 83, 88, 92, 76, 75)**

#calculate correlation coefficient between x and y
cor(x, y, use='complete.obs')

-0.4888749

The correlation coefficient between the two variables is calculated successfully, utilizing only the complete paired observations. Note that the cor() function only used pairwise combinations where both values were present when calculating the correlation coefficient.

## Example 2: Generating a Correlation Matrix with Missing Values

Suppose we attempt to use the **cor()** function to create a correlation matrix for a data frame with three variables when missing values are present. Without specifying a handling method, the matrix returns **NA**s for the cross-correlations:

```
#create data frame with some missing values
df <- data.frame(x=c(70, 78, 90, 87, 84, NA, 91, 74, 83, 85),
y=c(90, NA, 79, 86, 84, 83, 88, 92, 76, 75),
z=c(57, 57, 58, 59, 60, 78, 81, 83, NA, 90))

#attempt to create correlation matrix for variables in data frame
cor(df)

x y z
x 1 NA NA
y NA 1 NA
z NA NA 1
```

To avoid this issue and maximize the use of available data, we employ the argument **use='pairwise.complete.obs'** so that **R** calculates each specific correlation using all available pairs for that calculation:

```
#create data frame with some missing values
df <- data.frame(x=c(70, 78, 90, 87, 84, NA, 91, 74, 83, 85),
y=c(90, NA, 79, 86, 84, 83, 88, 92, 76, 75),
z=c(57, 57, 58, 59, 60, 78, 81, 83, NA, 90))

#create correlation matrix for variables using only pairwise complete observations
cor(df, use='pairwise.complete.obs')

x y z
x 1.0000000 -0.4888749 0.1311651
y -0.4888749 1.0000000 -0.1562371
z 0.1311651 -0.1562371 1.0000000
```

The correlation coefficients for each pairwise combination of variables in the data frame are now shown, providing a complete and actionable correlation matrix for further analysis.