Learning to Handle Missing Data: A Practical Guide to the COALESCE Function in SAS

Authored by

Mohammed looti

October 31, 2025

In the realm of data analysis and statistical programming, dealing with incomplete information is an inevitable challenge. The presence of missing values can severely compromise the integrity and reliability of any subsequent analysis, requiring robust strategies for data cleaning and preparation. Fortunately, the **COALESCE** function in SAS offers a highly efficient and elegant mechanism to manage these gaps. This powerful function is designed to scan a list of expressions or variables row-by-row and return the first value it finds that is not considered missing, thus streamlining the process of data imputation and consolidation.

This comprehensive tutorial aims to explore the functionality and practical applications of the **COALESCE** function within the SAS environment. We will break down its fundamental syntax, demonstrate its usage with concrete examples involving real-world data scenarios, and discuss key considerations--such as data type restrictions--that ensure its effective deployment. By mastering **COALESCE**, you can significantly enhance the efficiency of your SAS programs, leading to cleaner, more complete datasets and more trustworthy analytical results.

Understanding the COALESCE Function in SAS

The COALESCE function is fundamentally a conditional selection tool built into the SAS Data Step. Its core purpose is remarkably simple yet highly effective: to evaluate a sequence of arguments and identify the very first one that holds a non-missing value. The function processes the input arguments strictly from left to right, stopping as soon as a valid data point is encountered, and returning that value immediately. This sequential prioritization is what makes **COALESCE** indispensable for hierarchical data consolidation.

If, after checking every argument supplied to the function, all values are found to be missing, the **COALESCE** function itself logically returns a missing value. Understanding this behavior is critical, as it informs how you structure your data cleaning logic. For instance, if you are attempting to create a unified score from three potential sources (Source A, Source B, Source C), **COALESCE(A, B, C)** ensures that A is always preferred if available, followed by B, and finally C. If none are present, the output confirms the absence of data across all sources.

The standard syntax for implementing this function is highly intuitive: `COALESCE(argument-1, argument-2, ..., argument-n)`. Each argument can represent a variable name, a constant (like a default replacement value), or even a complex mathematical expression. This flexibility allows analysts to define sophisticated imputation rules within a single line of code. This methodology is paramount in data preparation, especially when multiple related columns exist, and the goal is to synthesize the most complete and robust single variable possible by systematically filling data gaps using available alternatives.

Setting Up the Scenario: Dealing with Incomplete Sports Data

To demonstrate the practical power of the COALESCE function, let us examine a typical scenario faced in statistical analysis: consolidating performance metrics in a sports dataset. Imagine we are tracking basketball team statistics, focusing on three key offensive metrics: `points`, `rebounds`, and `assists`. Due to recording errors, data submission latency, or various external factors, some entries inevitably contain missing values, denoted by the standard SAS period (`.`).

Our primary analytical goal is to derive a single, consolidated performance metric for each team. This metric should prioritize the highest quality and most crucial statistic first (points), falling back sequentially to secondary metrics (rebounds, then assists) only when the primary data is unavailable. This new variable will serve as a generalized measure of offensive output, ensuring that we leverage all available information even when the preferred data point is missing. The following code block illustrates the creation of our initial dataset, named `original_data`, showcasing the inherent missingness we need to address:

```
/*create dataset*/
data original_data;
input team $ points rebounds assists;
datalines;
Warriors 25 8 7
Wizards . 12 6
Rockets . . 5
Celtics 24 . 5
Thunder . 14 5
Spurs 33 19 .
Nets . . .
Mavericks . 8 10
Kings . . 9
Pelicans . 23 6
;
run;

/*view dataset*/
proc print data=original_data;
```

In this setup, we utilize the SAS `DATA STEP` and the `DATALINES` statement to input the sample data directly. The `points`, `rebounds`, and `assists` variables are all numeric variables, indicated by the numerical entries or the period for missingness. The subsequent `PROC PRINT` command serves to display the raw structure of the `original_data`, allowing us to confirm the input and identify the

scope of the missingness problem before applying the solution.

Obs	team	points	rebounds	assists
1	Warriors	25	8	7
2	Wizards	.	12	6
3	Rockets	.	.	5
4	Celtics	24	.	5
5	Thunder	.	14	5
6	Spurs	33	19	.
7	Nets	.	.	.
8	Maverick	.	8	10
9	Kings	.	.	9
10	Pelicans	.	23	6

Implementing COALESCE for Data Consolidation

Leveraging the existing `original_data`, we can now incorporate the COALESCE function within a new DATA STEP to generate our desired consolidated variable. We will call this new variable `first_non_missing`. Crucially, the order in which we list the variables inside the **COALESCE** function dictates the search hierarchy: we prioritize `points`, then `rebounds`, and finally `assists`. This ensures that the system always attempts to pull the score from the most important metric first.

The following SAS code block demonstrates this implementation, creating a new dataset named `new_data` that contains the results of our imputation logic:

```
/*create new dataset*/
data new_data;
set original_data;
first_non_missing = coalesce(points, rebounds, assists);
run;

/*view new dataset*/
proc print data=new_data;
```

Within the `DATA STEP`, the `SET` statement reads the source data row by row. For every observation, the assignment statement `first_non_missing = coalesce(points, rebounds, assists);` executes the core logic. SAS first checks `points`; if available, it assigns that value and moves to

the next observation. If `points` is missing, it proceeds to check `rebounds`. Only if both are missing does it look at `assists`. This systematic, cascading check guarantees that for any given team, the derived score represents the highest-priority non-missing statistic. Finally, the `PROC PRINT` command displays the resulting `new_data` dataset, allowing us to visually verify the successful data consolidation.

Obs	team	points	rebounds	assists	first_non_missing
1	Warriors	25	8	7	25
2	Wizards	.	12	6	12
3	Rockets	.	.	5	5
4	Celtics	24	.	5	24
5	Thunder	.	14	5	14
6	Spurs	33	19	.	33
7	Nets	.	.	.	.
8	Maverick	.	8	10	8
9	Kings	.	.	9	9
10	Pelicans	.	23	6	23

Interpreting the COALESCE Output

A review of the `new_data` dataset confirms the precise functionality of the COALESCE function in prioritizing and retrieving the first non-missing value. The results highlight several distinct behaviors based on the completeness of the source data:

Complete Data (e.g., Warriors, Spurs): For teams like the Warriors, where `points` (**25**) is present, **COALESCE** stops immediately and returns **25**. The subsequent values (rebounds and assists) are never evaluated. Similarly, the Spurs' entry returns **33** from `points`, even though `assists` is missing.

Partial Missingness (e.g., Wizards, Rockets): The Wizards entry shows missing `points` (`.`). The function checks `rebounds` (**12**), finds it available, and returns **12**. For the Rockets, both `points` and `rebounds` are missing, forcing the function to check `assists` (**5**), which is then returned. This demonstrates the critical sequential fall-back mechanism in action.

Total Missingness (e.g., Nets): The Nets entry is the definitive case for understanding the function's boundary conditions. Since `points`, `rebounds`, and `assists` are all missing (`.`), the COALESCE function, having exhausted all arguments without finding a valid entry, returns a missing value itself (`.`) for `first_non_missing`.

This systematic, row-by-row evaluation ensures that the derived variable `first_non_missing` is populated with the highest-priority available piece of information for every observation. If the final coalesced value is still missing, it signals that all specified source variables for that particular observation contained missing values, which is crucial metadata for subsequent advanced imputation or exclusion steps.

Essential Considerations: Data Types and Alternatives

While the **COALESCE** function provides a powerful solution for data manipulation, its effective use hinges on understanding specific restrictions and alternatives, particularly concerning data types. Failure to adhere to these rules can lead to runtime errors or incorrect results, undermining the integrity of your SAS program.

The most important rule is that the **COALESCE** function is strictly designed for numeric variables. This category includes standard integers, decimal numbers, and critically, all SAS date and time variables, which are stored internally as numeric counts. If your variables are numeric, **COALESCE** correctly identifies the missing values (`.`) and performs the selection as intended. Attempting to pass character variables (text strings) to **COALESCE** will result in errors because the function is not equipped to handle character missingness (which is typically represented by blank spaces).

If your data cleaning task involves consolidating data from character variables, you must utilize the dedicated character counterpart: the **COALESCEC** function. The COALESCEC function operates with the exact same prioritization logic as **COALESCE**, but it is specifically tailored to search for the first non-blank or non-empty string in the provided list of arguments. Using the correct function based on the data type ensures that your imputation and data consolidation efforts are robust and error-free, maintaining high data quality throughout the analysis pipeline.

Conclusion and Final Recommendations

The **COALESCE** function is an indispensable, productivity-boosting tool for any analyst working in the SAS environment. It offers a concise, readable, and highly efficient method for addressing missing values by systematically prioritizing and selecting the first available non-missing entry from a specified list of numeric variables. As demonstrated with the sports statistics example, **COALESCE** simplifies what would otherwise require complex conditional `IF-THEN-ELSE` logic, significantly improving code maintainability and efficiency.

To ensure the accuracy of your data preparation efforts, always remember the distinction between **COALESCE** (for numeric data) and its character-specific twin, the COALESCEC function. Integrating these functions into your standard programming repertoire will empower you to manage data incompleteness with precision, leading to more reliable modeling and statistical inference.

For those eager to deepen their SAS expertise and explore other common data manipulation techniques, we recommend consulting the following resources: