How to Easily Remove Rows with Missing Values in SAS

Authored by

stats writer

November 20, 2025

In data analysis, handling missing values is a fundamental requirement for maintaining data integrity and ensuring the reliability of statistical models. In SAS, one of the most efficient methods to manage observations containing incomplete data is by systematically identifying and removing the associated rows. This process typically utilizes conditional logic within a DATA step to evaluate the presence of missing elements and execute the necessary corrective action.

The core philosophy behind this approach is precision: rather than imputing potentially erroneous values, we opt to remove observations that lack critical data points. This technique is particularly valuable when the proportion of missing data is small, or when listwise deletion does not significantly impact statistical power. We employ a combination of the `IF` statement and specialized functions, such as the CMISS function, to achieve clean, analysis-ready datasets.

Fundamentals of Conditional Data Removal in SAS

When aiming to remove rows afflicted by missing values, we leverage the power of the DATA step in conjunction with the DELETE statement. The DATA step serves as the foundational environment for creating, modifying, and managing SAS datasets. Within this step, we can apply conditional logic that checks specific criteria for each observation; if the criteria are met--in this case, the presence of any missing data--the observation is immediately dropped from the output dataset.

The most straightforward and powerful syntax uses the CMISS function, which stands for Count Missing values. This function efficiently scans a list of variables for a given observation and returns the total count of missing numeric or character values found. By checking if this count is greater than zero, we can isolate and remove any observation that fails the completeness check.

You can use the following basic syntax to remove rows with missing values from a dataset in SAS:

```
data new_data;
set my_data;
if cmiss(of _all_) then delete;
run;
```

This particular example initiates a DATA step that creates a new dataset called **new_data**. It reads observations sequentially from the existing source dataset, **my_data**. The conditional statement `if cmiss(of _all_) then delete;` instructs SAS to check every variable in the observation (using the _ALL_ keyword) for a missing value. If the count of missing values returned by CMISS is greater than zero, the DELETE statement prevents that row from being written to **new_data**.

Understanding the CMISS Function and Variable Lists

The effectiveness of this technique relies heavily on the correct implementation of the CMISS function. Unlike simply checking one variable (e.g., `IF variable IS MISSING`), CMISS allows for a simultaneous check across multiple columns, making the syntax highly compact and efficient, especially when dealing with wide datasets. This function is crucial because it accounts for both numeric missing values (represented by a period, `.`) and character missing values (represented by a blank space).

The argument `(of _all_)` within CMISS is a key component, utilizing a specialized _ALL_ variable list. This convenient list shortcut tells SAS to look across every single column available in the current observation buffer defined by the `SET` statement. This ensures a comprehensive listwise deletion, removing any row that contains incompleteness regardless of which variable is affected.

While _ALL_ is highly useful for blanket removal, it is important to note that SAS also supports other variable list specifications, such as `_NUMERIC_` (to check only numeric variables) or `_CHARACTER_` (to check only character variables). Analysts often choose these alternatives when they intentionally allow missingness in certain descriptive or non-critical columns. However, for a strict listwise deletion requiring full completion, using `of _all_` is the appropriate choice.

Practical Example: Identifying Incomplete Observations

To demonstrate the application of this method, let us consider a sample dataset. Suppose we have the following dataset in SAS that contains information about various basketball teams, including their scores and assists. We must ensure that our statistical analysis only uses complete records, meaning no team should be included if their `points` or `assists` data are missing.

The following code uses the standard `DATA` and `DATALINES` statements to construct our raw dataset, intentionally embedding several missing values represented by the period (`.`) for numeric variables:

```
/*create dataset*/
data my_data;
input team $ points assists;
datalines;
Mavs 113 22
Pacers 95 .
Cavs . .
Lakers 114 20
Heat 123 39
Kings . 22
```

```
Raptors 105 11
Hawks 95 25
Magic 103 26
Spurs 119 .
;
run;

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

Upon execution of this code, we can view the initial structure of **my_data** using `PROC PRINT`.

Analyzing the Initial Dataset State

The output below clearly illustrates the presence of incomplete records. Several teams, such as the Pacers, Cavs, Kings, and Spurs, have one or more data points designated as missing. The goal of our data cleansing operation is to eliminate these rows entirely, leaving only fully populated observations for downstream analysis.

Obs	team	points	assists
1	Mavs	113	22
2	Pacers	95	.
3	Cavs	.	.
4	Lakers	114	20
5	Heat	123	39
6	Kings	.	22
7	Raptors	105	11
8	Hawks	95	25
9	Magic	103	26
10	Spurs	119	.

Notice that there are several rows with missing values. Specifically, the Pacers observation is missing `assists`, the Cavs observation is missing both `points` and `assists`, the Kings observation is missing `points`, and the Spurs observation is missing `assists`. For robust analysis, these observations must be treated as incomplete and removed from the dataset.

Applying the Row Deletion Logic

We can now implement the conditional logic within a new DATA step to create a filtered dataset. This implementation uses the `IF` condition to trigger the DELETE statement whenever CMISS returns a count greater than zero for the specified variables. Since we are using _ALL_, the check is applied to both the numeric columns (`points`, `assists`) and the character column (`team`).

The following code executes this crucial data cleansing step:

```
/*create new dataset that removes rows with missing values from existing dataset*/
data new_data;
set my_data;
if cmiss(of _all_) then delete;
run;

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

When the DATA step processes the row for the 'Pacers', CMISS finds one missing value (`assists = .`), returns 1, the condition is met (1 > 0), and the DELETE statement executes. This iterative process ensures that only rows passing the full data quality check are retained in **new_data**.

Verifying the Cleaned Output

After running the cleansing code, we use `PROC PRINT` again, this time on the newly created dataset, **new_data**, to confirm the result. We expect to see a reduced number of observations, corresponding exactly to those rows that had complete data in the original set.

Obs	team	points	assists
1	Mavs	113	22
2	Lakers	114	20
3	Heat	123	39
4	Raptors	105	11
5	Hawks	95	25
6	Magic	103	26

We can see that all rows with missing values (Pacers, Cavs, Kings, Spurs) have been removed from the dataset. The resulting dataset, **new_data**, now contains only six observations, each fully

populated across all variables, satisfying the requirement for listwise deletion. This process guarantees that subsequent analytical procedures are based solely on complete and reliable information.

Key Considerations for Using CMISS(OF _ALL_)

It is vital for SAS programmers to fully grasp the nuances of the CMISS function and its arguments to avoid unintended data loss.

Note #1: The argument _**all**_ within the CMISS function specifies that SAS should look for missing values in *all* columns for each row. When using the _ALL_ variable list, this check applies to both numeric and character variables, ensuring comprehensive identification of incompleteness. If you only wanted to check numeric columns, you would use `cmiss(of _numeric_)`.

Note #2: If you need to retain rows where missingness is acceptable in only specific columns (e.g., `optional_field`), you must replace `_all_` with an explicit list of critical variables (e.g., `cmiss(of points assists)`). Only critical fields should be included in the CMISS check to prevent over-deleting valid observations.

Note #3: You can find the complete documentation for the CMISS function on the official SAS support website, which provides detailed technical specifications and examples for various use cases.

Conclusion and Related Data Management Techniques

The conditional `IF CMISS(OF _ALL_) THEN DELETE;` statement offers the cleanest and most efficient way to perform listwise deletion of incomplete observations in SAS. This method is fundamental to initial data preparation and cleaning workflows. Mastering the use of the DATA step, combined with system functions like CMISS, empowers analysts to ensure the highest standard of data quality before proceeding to complex modeling.

For those interested in exploring alternative data management tasks, the following tutorials explain how to perform other common operations in SAS: