Deleting missing data, often referred to as **listwise deletion** (dropping rows) or **pairwise deletion** (dropping columns), is a straightforward technique in the cleaning phase. However, it's generally done only under specific circumstances due to the risk of losing valuable information and introducing bias.

## When to Drop Rows (Listwise Deletion) **🗑R◯**

Dropping an entire row (observation) when it contains one or more missing values is appropriate in the following situations:

1. **Missing Completely At Random (MCAR) Data:**
   * **Theory:** If the data is missing MCAR (the missingness is unrelated to any observed or unobserved data), and the number of missing values is very small (<5% of the total dataset), deleting the few incomplete rows is the simplest and safest option. It minimizes bias while keeping the dataset clean.
   * **Reason:** The small loss of data is acceptable and avoids the complexity and potential distortion of imputation.
2. **Critical Data is Missing:**
   * **Theory:** When a row is missing a value in the **target variable** (what you are trying to predict) or a **core feature** that cannot be accurately imputed.
   * **Reason:** You cannot train or evaluate a model on a row where the outcome is unknown or a core predictor is absent.
3. **High-Confidence Data Entry Error:**
   * **Theory:** The missingness is clearly due to a recording or sensor failure that makes the data point entirely unreliable.
   * **Reason:** Keeping a corrupted row, even if imputed, pollutes the dataset.

### Python Example (Dropping Rows)

The pandas function df.dropna() is used to drop rows containing any NaN values.

Python

import pandas as pd  
import numpy as np  
  
# Create sample data with missing values  
data = {'A': [1, 2, np.nan, 4], 'B': [5, np.nan, 7, 8]}  
df = pd.DataFrame(data)  
  
# Drop rows where ANY value is missing  
df\_dropped\_rows = df.dropna(axis=0, how='any')  
  
# Output: Only the first and last rows remain  
# print(df\_dropped\_rows)  
# A B  
# 0 1.0 5  
# 3 4.0 8

## When to Drop Columns (Pairwise Deletion) **❌**

Dropping an entire feature (column) from the dataset is a drastic measure and should only be done when imputation is not feasible.

1. **Excessive Missingness:**
   * **Theory:** A column has a very high proportion of missing values (e.g., >70%).
   * **Reason:** There is not enough actual data to make a reliable statistical estimate (imputation). Dropping the column is better than using a feature that is mostly synthetic (imputed) and likely noise.
2. **Irrelevant/Redundant Feature:**
   * **Theory:** The column is not related to the target variable and is not critical for analysis, even if its missingness is moderate.
   * **Reason:** Removing the column reduces dimensionality and cleaning effort without sacrificing predictive power.

### Python Example (Dropping Columns)

The df.dropna() function with axis=1 is used to drop columns. You can also use the thresh parameter to specify a minimum number of non-missing values required to keep the column.

Python

# Assuming a column 'C' is mostly missing  
df['C'] = [np.nan, np.nan, 9, np.nan]   
# df now has 4 rows. 'C' is missing 75% of its data.  
  
# Drop columns that have more than 50% missing data (needs at least 3 non-NaN values to keep)  
df\_dropped\_cols = df.dropna(axis=1, thresh=len(df) \* 0.5)  
  
# Output: Column 'C' (which only has 1 non-NaN value) would be dropped.  
# df\_dropped\_cols now contains only 'A' and 'B'  
# print(df\_dropped\_cols)

## Caveats (Why You Usually Don't Drop) ⚠️

* **Bias:** If data is **Missing Not At Random (MNAR)** or **Missing At Random (MAR)**, dropping the data can introduce significant bias, as the resulting dataset will no longer be representative of the original population.
* **Information Loss:** Dropping rows can severely reduce your sample size, leading to less powerful statistical tests and reduced model accuracy.
* **Alternative:** **Imputation** is almost always the preferred approach when the percentage of missing data is significant, as it preserves both the sample size and the existing variance in the feature.