《NA 与 NaN：数据缺失值的比较与处理》

\*在数据处理领域，**NA**（Not Available）和**NaN**（Not a Number）虽然都用于表示缺失值，但它们来自不同的语境，具体区别如下：

### **1. 定义与来源**

|  |  |
| --- | --- |
| **NA** | **NaN** |
| 通常作为文本标记，表示 “不可用” 或 “缺失”。常见于 CSV 文件、数据库导出数据等。 | 是 IEEE 754 浮点数标准中定义的特殊值，用于表示无法表示的数值结果（如 0/0）。属于编程语言（如 Python、R）中的数值类型。 |

### **2. 在不同场景中的表现**

#### **（1）CSV 文件中的文本表示**

* **NA**：直接以字符串形式存在，例如：

|  |
| --- |
| NumRooms,Alley,Price  NA,Pave,127500  2,NA,106000 |

* **NaN**：通常不会直接出现在 CSV 中，而是在数据加载后由程序解析为缺失值。

#### **（2）Python 中的处理**

* **NA**：加载为字符串'NA'，需手动转换为缺失值：

|  |
| --- |
| import pandas as pd  df = pd.read\_csv('data.csv') # 'NA'会被视为普通字符串  df = df.replace('NA', pd.NA) # 手动替换为Pandas的缺失值标记 |

* **NaN**：由numpy提供，是浮点数类型的缺失值：

|  |
| --- |
| import numpy as np  df = pd.DataFrame({'A': [1, np.nan, 3]}) # 直接使用NaN |

#### **（3）Pandas 中的缺失值**

* **pd.NA**（Pandas 1.0+）：统一的缺失值标记，支持整数、布尔等更多数据类型。
* **np.nan**：传统的缺失值，仅适用于浮点数和对象类型。

|  |
| --- |
| # pd.NA（新） vs np.nan（旧）  s1 = pd.Series([1, pd.NA, 3], dtype="Int64") # 整数类型支持pd.NA  s2 = pd.Series([1, np.nan, 3]) # 自动转换为浮点类型 |

### **3. 核心差异总结**

|  |  |  |
| --- | --- | --- |
| **维度** | **NA** | **NaN** |
| **数据类型** | 字符串或通用缺失标记 | 浮点数（IEEE 754 标准） |
| **应用场景** | CSV、数据库导出数据 | 数值计算、统计分析 |
| **编程语言** | 需手动处理（如替换为pd.NA） | 内置支持（如numpy.nan） |
| **计算行为** | 参与计算时通常保持为NA | 传播缺失值（任何与NaN的运算结果均为NaN） |

### **4. 实际处理建议**

1. **读取 CSV 时自动识别 NA**：

|  |
| --- |
| df = pd.read\_csv('data.csv', na\_values=['NA']) # 将'NA'自动解析为缺失值 |

1. **处理 NaN**：

|  |
| --- |
| df.fillna(0) # 用0填充NaN  df.dropna() # 删除包含NaN的行 |

1. **选择合适的缺失值类型**：
   * 对于数值计算，优先使用np.nan。
   * 对于整数或布尔类型，使用pd.NA（需 Pandas 1.0+）。

### **示例对比**

|  |  |  |
| --- | --- | --- |
| 操作 | 含'NA'（字符串） | 含np.nan |
| df.isna() | 全 False（视为普通字符串） | 正确识别缺失值 |
| df.mean() | 报错（字符串无法计算均值） | 忽略 NaN 计算均值 |

### **总结**

* **NA** 是通用的文本标记，用于表示数据缺失，需手动处理。
* **NaN** 是编程语言中的数值缺失值，专为计算设计，支持自动传播。

根据数据来源和处理需求，选择合适的缺失值表示方式是数据清洗的关键步骤。