Handling Duplicates in Data

Bita Taheri (Matriculation:400889819)

Hochschule Fresenius - University of Applied Science

Author Note

The authors have no conflicts of interest to disclose.

Correspondence concerning this article should be addressed to Bita Taheri (Matriculation:400889819), Email: taheri.bita@stud.hs-fresenius.de

Abstract

Duplicate data entries are a common issue in data analysis and can significantly impact the accuracy and reliability of analytical results. This report discusses the nature of duplicates, their different types, and the challenges they pose, such as skewed statistics, biased models, and inefficient resource usage. It highlights best practices for identifying and handling duplicates using both base R and the tidyverse (especially the dplyr package). Practical examples using the built-in iris dataset demonstrate how to detect, inspect, and remove duplicate rows. By applying tools like duplicated(), unique(), and distinct(), analysts can ensure clean, trustworthy data that leads to valid insights. The report emphasizes that handling duplicates is a critical step in the data cleaning process, essential for high-quality and reproducible analysis.

Keywords: Data Cleaning, Duplicate Records, Data Quality, dplyr, Tidyverse, Data Preprocessing, Data Analysis, Data Integrity, nBase R, Data Visualization

Handling Duplicates in Data

Table of contents

Introduction 4

introduction 4

Detecting and Handling Duplicates in R 6

A numeric vector with some repeats 6

Identify which elements are duplicates 6

Extract the unique values (remove duplicates) 7

Remove duplicate rows based on a single column, e.g., Sepal.Width 7

Affidavit 11

Handling Duplicates in Data

introduction

In data analysis, duplicate entries refer to records that appear more than once in a dataset. These can arise unintentionally due to data entry errors, merging datasets, or system glitches, and if left unchecked they can distort analytical results . Duplicates give undue weight to certain observations, potentially skewing statistics and leading to misleading conclusions. . For example, duplicate customer records might inflate sales totals or duplicate experimental measurements could bias averages. Beyond statistical distortion, duplicates also waste storage space and computational resources, and can complicate data management. Therefore, identifying and handling duplicates is a critical step in data cleaning to ensure the integrity of any analysis.

This report explores what duplicates are and why they are problematic, the different types of duplicates, the challenges they pose in analysis, and demonstrates how to detect and resolve duplicates in R. We will also walk through a real-world example in RStudio, showing step-by-step how to identify and address duplicates, and conclude with best practices. The guidance and code examples are aimed at readers with basic R knowledge and make use of authoritative resources such as R for Data Science and official R documentation for reference.

Types of Duplicates in Data

Not all duplicates are the same – it's important to distinguish their types to handle them appropriately. Common categories include:

Not all duplicates are the same, and understanding their types is key to handling them correctly. The main categories include:

- Exact Duplicates: Rows that are completely identical across all columns. These usually result from data entry errors or merging datasets. They are typically unintentional and should be removed.

- Partial (Near) Duplicates: Records that share key fields (like ID or name) but differ slightly in other values, such as formatting or timestamps. These are harder to detect and may require custom rules or fuzzy matching.

- Intentional vs. Unintentional Duplicates: Some duplicates are valid, like repeated measurements in longitudinal studies or sales logs. These should not be removed but analyzed properly (e.g. aggregated or paired).

In contrast, unintentional duplicates—such as repeated entries due to copy-paste errors—should usually be eliminated.

Even intentional duplicates can cause problems if not handled carefully. Always evaluate duplicates in context to decide whether to keep, combine, or drop them.

Why Duplicates Are Problematic: Analytical Challenges

Duplicate records in a dataset can seriously affect the quality and accuracy of data analysis. Key problems include:

- Skewed Statistics: Duplicates inflate metrics like totals, means, and standard deviations, leading to inaccurate results.
- Misleading Visuals: Charts and graphs may appear distorted due to repeated values, making the data look skewed or clustered when it's not.
- Model Bias: In predictive modeling, duplicates can cause overfitting by giving too much weight to certain patterns, which reduces model reliability.
- False Significance: Duplicates can exaggerate correlations and affect hypothesis testing by violating the assumption of independent observations.
- Wasted Resources: Extra data increases storage needs and slows down processing, especially in large datasets.
- Data Quality Issues: Unexpected duplicates often signal deeper problems like flawed data entry or merging errors.

In short, duplicates must be identified and carefully handled to ensure valid, efficient, and trustworthy analysis.

Detecting and Handling Duplicates in R

R provides robust tools for identifying and removing duplicates, both in base R and in the tidyverse collection of packages. This section details how to use these tools with code examples.

We will cover base R functions like duplicated(), unique(), and anyDuplicated(), as well as tidyverse approaches with dplyr (especially the distinct()function).

Conceptual illustration of identifying and removing duplicate rows in a dataset (blue rows indicate duplicates). In R, base functions like duplicated()/unique() and the dplyr function distinct() are commonly used to address duplicates.

Base R Techniques for Duplicates

Base R has built-in functions to detect duplicates and extract unique values. The primary function is duplicated(), which returns a logical vector indicating which elements or rows are duplicates of a previous occurrencerdrr.io. Each element of the vector is TRUE if that row/element has appeared before and is thus a duplicate, and FALSE if it is the first occurrence or unique. A related base function, unique(), returns the unique elements or rows of a vector/data frame, effectively removing duplicates. Let's start with a simple example on a vector to illustrate duplicated():

A numeric vector with some repeats

```
x <- c(1, 1, 4, 5, 4, 6)
```

Identify which elements are duplicates

```
duplicated(x) #> [1] FALSE TRUE FALSE FALSE TRUE FALSE
```

Here, the output shows TRUE for positions 2 and 5, meaning x[2] and x[5] are duplicates of earlier values in the vector (indeed, x[2] = 1 is a repeat of x[1] = 1, and x[5] = 4 repeats x[3] = 4). The first occurrence of a value is not marked as a duplicate – by definition, duplicated() only flags the subsequent occurrences. We can use this result to extract or remove duplicates: # Extract the duplicate values themselves x[duplicated(x)] #> [1] 1 4

Extract the unique values (remove duplicates)

x[!duplicated(x)] #> [1] 1 4 5 6

In the above, x[duplicated(x)] returns the values that had appeared before (1 and 4 in this case). Conversely, prefixing with ! (logical NOT) gives the values that are not duplicates, effectively the unique set {1, 4, 5, 6}. Using !duplicated()on a data frame will filter to unique rows. In fact, the base R unique() function achieves the same: unique(x) would yield c(1, 4, 5, 6) in this example, and unique(my_data_frame) removes duplicate rows from a data frame. For data frames, duplicated() operates row-wise. For example, if we have a data frame df, duplicated(df) will return a logical vector of length equal to the number of rows, with TRUE for each row that is identical to a previous row. We can use this to identify and remove duplicate rows. Consider a data frame df and the pattern:

dup_rows <- df[duplicated(df),] # subset of all duplicate rows (beyond first occurrences) df_no_dups <- df[!duplicated(df),] # dataframe with duplicates removed

The first line collects all rows that are duplicates (this will exclude the first instance of each set of duplicates). The second line keeps only the rows that are not flagged as duplicates, which is equivalent to unique(df)

Partial duplicates by column: Sometimes we want to detect duplicates based on a subset of columns (for instance, find repeated IDs regardless of other differences). In base R, one can use the duplicated() function on a specific column or a combination of columns. For example, duplicated(df$ID) would flag duplicate IDs in the data frame. We can also use duplicated(df[, c("col1","col2")]) to check for duplicates with respect to a combination of col1 and col2. Rows where both col1 and col2 match an earlier row will be marked. As an example, using R's built-in iris dataset (which has columns for flower measurements and species):

Remove duplicate rows based on a single column, e.g., Sepal.Width

iris_unique_width <- iris[!duplicated(iris$Sepal.Width),]

This keeps only the first occurrence of each Sepal.Width value and drops any later rows that repeat a Sepal.Width seen before. After this operation, iris_unique_width would have one row per unique sepal width measurement (in iris, that yields 23 rows out of 150, as there were many repeated widths). Similarly, one could use multiple columns by constructing a key, e.g., !duplicated(iris[, c("Sepal.Length","Species")]) to remove rows where both length and species repeat. R also provides the helper function anyDuplicated(), which is essentially a faster check to see if any duplicates exist in an object. It returns the index of the first duplicate found (or 0 if no duplicates). This is useful for a quick confirmation. For instance:

anyDuplicated(iris) #> [1] 143

The result "143" (hypothetical output) indicates that the 143rd row of iris is a duplicate of an earlier row (meaning at least one duplicate exists). If the result were 0, it would mean no duplicates. Using which(duplicated(iris)) would list all indices of duplicate rows, not just the first. In summary, base R's approach to duplicates typically involves using duplicated() to create a logical index, then subsetting the data. While effective, it can be a bit clunky for complex operations (like deciding which duplicates to keep based on some criteria). That's where the tidyverse tools often offer more convenience.

Of course! Here's a **longer and more detailed summary** of the section on **Tidyverse Techniques (dplyr)** for handling duplicates, without being too lengthy:

Tidyverse Techniques (dplyr) – Summary

In the **tidyverse**, the `dplyr` package offers a clear and powerful way to detect and remove duplicates using the `distinct()` function. It works similarly to base R's `unique()`, but is often more efficient and user-friendly—especially when working with data frames.

How `distinct()` Works:

- When you run `distinct(df)`, it returns a new data frame where **duplicate rows are removed**, keeping **only the first occurrence** of each unique row.
- The **original order of rows is preserved**, and only the later repeated ones are dropped.

Focusing on Specific Columns:

You don't have to consider the whole dataset. You can apply `distinct()` to specific columns to check for uniqueness in part of the data:

```
distinct(df, column1, column2, .keep_all = TRUE)
```

- This keeps **one row per unique combination** of the selected columns.
- The option `.keep_all = TRUE` tells R to keep the entire row (not just the selected columns).
- If you leave `.keep_all = FALSE` (the default), only the columns you list will be returned.

Example Using `iris` Dataset:

The iris dataset has 150 rows. Here's how you can check and remove duplicates:

```
library(dplyr)


iris_unique <- iris %>% distinct()
nrow(iris_unique)
# Output: 149
```

This confirms that there is **one duplicate row**, and it has been removed. The `distinct()` function simplifies this task with just one line of code.

Using `distinct()` on Subsets of Data:

You can find unique combinations based on just a few columns. For example:

```
iris_species_lengths <- iris %>%
  distinct(Species, Petal.Length, .keep_all = FALSE)
```

This returns only the `Species` and `Petal.Length` columns with unique combinations, ignoring other variables.

Counting Duplicates:

Another useful approach is to **count how many times each row or combination appears**, then filter only those with duplicates:

```
iris %>%
  count(Species, Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, sort = TRUE) %>%
  filter(n > 1)
```

This groups by all key columns and returns rows that appear more than once, showing exactly how many times each duplicated row is repeated.

If you just want the **number** of duplicated rows:

```
# Base R
sum(duplicated(iris))


# Or with dplyr
nrow(iris) - nrow(distinct(iris))
```

Both approaches will return `1` for the iris dataset, because it contains one duplicate.

Other Useful Tools:

- **`janitor::get_dupes()`**: A handy function from the `janitor` package that lists duplicate rows and how often each occurs.
- **`data.table` package**: Offers very fast tools for handling duplicates in large datasets.

Summary of Benefits:

- `distinct()` is easy to read and integrate into data cleaning pipelines using `%>%`.
- It works for full data frames or selected columns.
- It's more efficient and consistent than some base R alternatives.

- Combined with `count()`, it helps **inspect**, not just remove, duplicates.

Refer to these sources to get more information on data duplication. (Datanovia, 2020)

(Sanderson, 2024)

(Social Science Computing Cooperative, 2020)

Affidavit

I hereby affirm that this submitted paper was authored unaided and solely by me. Additionally, no other sources than those in the reference list were used. Parts of this paper, including tables and figures, that have been taken either verbatim or analogously from other works have in each case been properly cited with regard to their origin and authorship. This paper either in parts or in its entirety, be it in the same or similar form, has not been submitted to any other examination board and has not been published.

I acknowledge that the university may use plagiarism detection software to check my thesis. I agree to cooperate with any investigation of suspected plagiarism and to provide any additional information or evidence requested by the university.

Checklist:

- ☐ The handout contains 3-5 pages of text.
- ☐ The submission contains the Quarto file of the handout.
- ☐ The submission contains the Quarto file of the presentation.
- ☐ The submission contains the HTML file of the handout.
- ☐ The submission contains the HTML file of the presentation.
- ☐ The submission contains the PDF file of the handout.
- ☐ The submission contains the PDF file of the presentation.
- ☐ The title page of the presentation and the handout contain personal details (name, email, matriculation number).
- ☐ The handout contains a abstract.

- ☐ The presentation and the handout contain a bibliography, created using BibTeX with APA citation style.
- ☐ Either the handout or the presentation contains R code that proof the expertise in coding.
- ☐ The handout includes an introduction to guide the reader and a conclusion summarizing the work and discussing potential further investigations and readings, respectively.
- ☐ All significant resources used in the report and R code development.
- ☐ The filled out Affidavit.
- ☐ A concise description of the successful use of Git and GitHub, as detailed here: https://github.com/hubchev/make_a_pull_request.
- ☐ The link to the presentation and the handout published on GitHub.

[Bita Taheri,] [06/04/2025,] [Koln]

Datanovia. (2020). *Identify and remove duplicate data in r*. https://www.datanovia.com/en/lessons/identify-and-remove-duplicate-data-in-r/. https://www.datanovia.com/en/lessons/identify-and-remove-duplicate-data-in-r/

Sanderson, S. (2024). *How to use the duplicated function in base r*. https://www.r-bloggers.com/2024/01/how-to-use-the-duplicated-function-in-base-r/. https://www.r-bloggers.com/2024/01/how-to-use-the-duplicated-function-in-base-r/

Social Science Computing Cooperative. (2020). *Data wrangling essentials: Section 4.9 - duplicate observations*. https://sscc.wisc.edu/sscc/pubs/dwr/duplicates.htm. https://sscc.wisc.edu/sscc/pubs/dwr/duplicates.htm