Merging Datasets using R

Assil Noureddine (400924750)

2025-06-11

noureddine.assil@stud.hs-fresenius.de

Abstract

We use fictional employee and salary datasets to demonstrate `left_join()` and `inner_join()`, and visualize the results.

In data science and statistical analysis, combining data from multiple sources is a foundational skill. Real-world data is often distributed across different files, databases, or formats, and meaningful insights usually require integration of these sources. This process—known as merging datasets—involves aligning and joining tables using shared identifiers or keys.

In this handout, we focus on the R programming language, specifically the tidyverse ecosystem, to demonstrate how merging is conducted in a tidy and reproducible manner. We explore the theory behind data joins, types of merges, syntax using dplyr, and common challenges analysts face. By the end, readers will understand how to use joins in R to clean, enrich, and structure data for analysis in real-world projects.

Introduction

In the age of big data and interconnected systems, it's rare that a single dataset contains all the variables required to answer a research question. For example, a healthcare analyst might receive one dataset with patient demographics and another with medical test results. Similarly, a business analyst might receive sales transactions in one file and marketing campaign responses in another.

These fragmented datasets are pieces of a puzzle, and merging them is what brings the full picture into view (Wickham et al., 2023).

The process of merging, also called joining or combining, involves connecting datasets by matching keys (like IDs, dates, or names).

What is a Dataset

In R, a dataset is usually stored as a data frame or tibble: a two-dimensional table where each column is a variable and each row is an observation.

You might have multiple datasets:

- One contains customer info
- Another contains sales
- A third contains survey responses To analyze customer behavior, you must merge these datasets using a common column such as customer_id (Wickham & Grolemund, 2016).

Types of datasets

You might need to merge:

- Relational data: tables linked by IDs (like in a relational database)
- Time-based data: e.g., log files or timestamped data
- Hierarchical data: e.g., schools within districts, patients within hospitals
- Multi-source data: files from different systems or surveys

Regardless of the source, merging depends on shared keys and consistent data structures (Wickham, 2014).

Why merging is useful?

- Combines information from multiple sources into one dataset, giving a more complete view.
- Reveals relationships and patterns that are hidden when data is separated.
- Supports deeper analysis, such as linking customer details with purchase behavior.
- Improves research quality by joining variables like survey results and demographics.
- Reduces duplication and data redundancy by unifying related records.
- Enables better decision-making with more comprehensive and contextual data.
- Essential in real-world projects, where important information is rarely in just one file (Wickham & Grolemund, 2016).

Tools & Packages in R for Merging Datasets

R offers several methods to merge data, but the modern, readable, and flexible approach is using the tidyverse. Here are the primary tools:

1. dplyr (from tidyverse): Provides join functions: left_join(), inner_join(), etc.
2. tidyr: Helps reshape and tidy data before/after merging (e.g. pivot_longer())
3. Base R (merge()): Older method but still works; less readable and less intuitive than dplyr
4. data.table: Very fast for large data; uses different syntax

For teaching, collaboration, and clarity, dplyr is the gold standard in merging tidy datasets (Peng, 2016).

How to use dplyr

Using dplyr, we match rows from two datasets by one or more key columns using these main join functions (Wickham et al., 2023):

Function	What it does
left_join()	Keeps all rows from the first table
right_join()	Keeps all rows from the second table
inner_join()	Keeps only rows that match in both tables
full_join()	Keeps all rows from both tables
anti_join()	Keeps rows in the first table **without** a match
semi_join()	Keeps rows in the first table **with** a match, no extra columns

Code example

Suppose you have two CSV files:

- employees.csv:

employee_id	name	department
1	Alice	HR
2	Bob	IT
3	Charlie	Finance
4	David	marketing

- salaries.csv:

employee_id	salary
2	60000
4	75000
5	50000

Step 1: Load the CSV files into R

```
employees <- read.csv("Data/employees.csv")
salaries <- read.csv("Data/salaries.csv")
```

Step 2: Perform Different types of joins using dplyr

Make sure to load dplyr first:

```
library(dplyr)
```

Inner Join — Employees with salaries available

```
inner_merged <- inner_join(employees, salaries, by = "employee_id")
print(inner_merged)
```

```
  employee_id   name department salary
1           2    Bob         IT  60000
2           4 David  Marketing   75000
```

Left Join — All employees, with salaries where available

```r
left_merged <- left_join(employees, salaries, by = "employee_id")
print(left_merged)
```

```
  employee_id    name department salary
1           1   Alice         HR     NA
2           2     Bob         IT  60000
3           3 Charlie    Finance     NA
4           4  David  Marketing   75000
```

Full Join — All employees and all salary records combined

```r
# Full join to merge all rows from both dataframes
library(dplyr)

full_merged <- full_join(employees, salaries, by = "employee_id")
full_merged
```

```
  employee_id    name department salary
1           1   Alice         HR     NA
2           2     Bob         IT  60000
3           3 Charlie    Finance     NA
4           4  David  Marketing   75000
5           5    <NA>       <NA>  50000
```

Pitfalls and Common Mistakes

- Duplicate keys: If either dataset has duplicate IDs, merges may produce unexpected duplicates.
- Mismatched types: Merging fails if key columns are not of the same type (e.g., one numeric, one character).
- Missing values: NA keys won't match anything—be cautious.
- Wrong join direction: Choose the correct "left" and "right" dataset based on which one you want to keep completely.
- Name collisions: If both datasets have a column with the same name (but different content), R will append .x and .y—which can be confusing (Wickham & Grolemund, 2016).

Discussion

- Merging is not just a technical task. It helps:
- Find relationships between datasets
- Spot missing information Combine different perspectives (e.g., personal data + performance)

In big data or research, merging allows you to build a full picture by joining datasets from surveys, sensors, websites, or government data (Wickham, 2014),(Wickham & Grolemund, 2016).

Conclusion

Merging datasets is a powerful and necessary operation in data science workflows. It allows you to combine data sources, enrich your analysis, and create complex data pipelines. Mastering joins in dplyr is essential for handling real-world projects where data rarely lives in a single file.

The key is to understand your data structure, clean your keys, and apply the appropriate join based on the goal of your analysis. With tidyverse tools, merging is efficient, readable, and scalable (DataCamp, n.d.),(Müller & Wickham, 2023)

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:

[X]The handout contains 3-5 pages of text.

[X]The submission contains the Quarto file of the handout.

[X]The submission contains the Quarto file of the presentation.

[X]The submission contains the HTML file of the handout.

[X]The submission contains the HTML file of the presentation.

[X]The submission contains the PDF file of the handout.

[X]The submission contains the PDF file of the presentation.

[X]The title page of the presentation and the handout contain personal details (name, email, matriculation number).

[X]The handout contains a abstract.

[X]The presentation and the handout contain a bibliography, created using BibTeX with APA citation style.

[X]Either the handout or the presentation contains R code that proof the expertise in coding.

[X]The handout includes an introduction to guide the reader and a conclusion summarizing the work and discussing potential further investigations and readings, respectively.

[X]All significant resources used in the report and R code development.

[X]The filled out Affidavit.

[X]A concise description of the successful use of Git and GitHub, as detailed here: https://github.com/hubchev/make_a_pull_request.

[X]The link to the presentation and the handout published on GitHub.

[Assil Noureddine,] [11 june 2025,] [Kôln]

References

DataCamp. (n.d.). *Joining data in r with dplyr*. https://www.datacamp.com/courses/joining-data-in-r-with-dplyr

Müller, K., & Wickham, H. (2023). *Tibble: Simple data frames*. https://CRAN.R-project.org/package=tibble

Peng, R. D. (2016). *R programming for data science*. Leanpub. https://leanpub.com/rprogramming

Wickham, H. (2014). Tidy data. *Journal of Statistical Software*, *59*(10), 1–23. https://doi.org/10.18637/jss.v059.i10

Wickham, H., François, R., Henry, L., & Müller, K. (2023). *Dplyr: A grammar of data manipulation*. https://CRAN.R-project.org/package=dplyr

Wickham, H., & Grolemund, G. (2016). *R for data science*. O'Reilly Media. https://r4ds.hadley.nz/