Joins and Pivots in R

2025-06-09

Part 1: Introduction to Joins

What are joins?

Joins combine data from two datasets based on common columns (keys).

Real-world example: You have student names and IDs in one file, and their test scores in another file. You want to combine them to see which student got which score.

```
library(tidyverse)

# Create example datasets
students <- data.frame(
  student_id = c(1, 2, 3, 4),
  name = c("Alice", "Bob", "Charlie", "Diana"),
  major = c("Math", "English", "Math", "Biology")
)

test_scores <- data.frame(
  student_id = c(1, 2, 3, 5),  # Note: student 4 missing, student 5 extra
  test_score = c(85, 92, 78, 88),
  test_date = c("2023-01-15", "2023-01-15", "2023-01-16", "2023-01-16")
)

print("Students table:")
```

```
## [1] "Students table:"
```

```
students
```

```
##   student_id    name   major
## 1          1   Alice    Math
## 2          2     Bob English
## 3          3 Charlie    Math
## 4          4   Diana Biology
```

```
print("Test scores table:")
```

```
## [1] "Test scores table:"
```

```
test_scores
```

```
##   student_id test_score  test_date
## 1          1         85 2023-01-15
## 2          2         92 2023-01-15
## 3          3         78 2023-01-16
## 4          5         88 2023-01-16
```

Part 2: The Five Main Join Types

1. Inner Join - Keep only matching rows

Rule: Only keep rows that exist in BOTH datasets

```
inner_result <- inner_join(students, test_scores, by = "student_id")
print("Inner Join Result:")
```

```
## [1] "Inner Join Result:"
```

```
inner_result
```

```
##   student_id    name   major test_score  test_date
## 1          1   Alice    Math         85 2023-01-15
## 2          2     Bob English         92 2023-01-15
## 3          3 Charlie    Math         78 2023-01-16
```

What happened?

- Only students 1, 2, 3 appear (they exist in both datasets)

- Student 4 (Diana) disappeared - no test score

- Student 5's score disappeared - no student info

---

2. Left Join - Keep all from left dataset

Rule: Keep ALL rows from the left (first) dataset, add matching data from right

```
left_result <- left_join(students, test_scores, by = "student_id")
print("Left Join Result:")
```

```
## [1] "Left Join Result:"
```

```
left_result
```

```
##   student_id    name   major test_score  test_date
## 1          1   Alice    Math         85 2023-01-15
## 2          2     Bob English         92 2023-01-15
## 3          3 Charlie    Math         78 2023-01-16
## 4          4   Diana Biology         NA       <NA>
```

What happened?

- ALL students kept (1, 2, 3, 4)

- Diana (student 4) has NA for test_score and test_date

- Student 5's score still disappeared

---

3. Right Join - Keep all from right dataset

Rule: Keep ALL rows from the right (second) dataset, add matching data from left

```
right_result <- right_join(students, test_scores, by = "student_id")
print("Right Join Result:")
```

```
## [1] "Right Join Result:"
```

```
right_result
```

```
##   student_id    name   major test_score  test_date
## 1          1   Alice    Math         85 2023-01-15
## 2          2     Bob English         92 2023-01-15
## 3          3 Charlie    Math         78 2023-01-16
## 4          5    <NA>    <NA>         88 2023-01-16
```

What happened?

- ALL test scores kept (students 1, 2, 3, 5)

- Student 5 has NA for name and major

- Diana (student 4) disappeared

---

4. Full Join - Keep everything

Rule: Keep ALL rows from BOTH datasets

```
full_result <- full_join(students, test_scores, by = "student_id")
print("Full Join Result:")
```

```
## [1] "Full Join Result:"
```

```
full_result
```

```
##   student_id    name   major test_score  test_date
## 1          1   Alice    Math         85 2023-01-15
## 2          2     Bob English         92 2023-01-15
## 3          3 Charlie    Math         78 2023-01-16
## 4          4   Diana Biology         NA       <NA>
## 5          5    <NA>    <NA>         88 2023-01-16
```

What happened?

- Everyone appears (students 1, 2, 3, 4, 5)

- Missing data filled with NA

---

5. Anti Join - Keep non-matching rows from left

Rule: Keep rows from the left dataset that do NOT have matches in the right dataset

```
anti_result <- anti_join(students, test_scores, by = "student_id")
print("Anti Join Result:")
```

```
## [1] "Anti Join Result:"
```

```
anti_result
```

```
##   student_id  name   major
## 1          4 Diana Biology
```

What happened?

- Only Diana (student 4) appears

- She's the only student who did NOT take the test

- No columns from test_scores are included

When is this useful?

- Find students who missed the test

- Identify customers who haven't made purchases

- Find data quality issues (unmatched records)

Part 3: Quick Reference - When to Use Which Join

Join Type	When to Use	Example
`inner_join()`	Only want complete cases	"Show only students who took the test"
`left_join()`	Keep all from main dataset	"Show all students, even those who missed the test"
`right_join()`	Keep all from second dataset	"Show all test scores, even mystery students"
`full_join()`	Don't want to lose any data	"Keep everything for investigation"
`anti_join()`	Find what's missing	"Show students who DIDN'T take the test"

Most common: `left_join()` - usually you have a main dataset and want to add information to it.

Part 4: Introduction to Pivots

What are pivots?

Pivots reshape data between "wide" and "long" formats.

Wide format: Each variable is a column **Long format**: Multiple observations per row, with variables in rows

```
# Create example: student grades in different subjects
grades_wide <- data.frame(
  student = c("Alice", "Bob", "Charlie"),
  math = c(85, 92, 78),
  english = c(88, 85, 92),
  science = c(90, 88, 85)
)

print("Wide format:")
```

```
## [1] "Wide format:"
```

```
grades_wide
```

```
##   student math english science
## 1   Alice   85      88      90
## 2     Bob   92      85      88
## 3 Charlie   78      92      85
```

---

pivot_longer() - Wide to Long

Use when: You have multiple columns that represent the same type of measurement

```
# Convert to long format
grades_long <- grades_wide %>%
  pivot_longer(cols = c(math, english, science),  # columns to pivot
               names_to = "subject",              # name for the new column
               values_to = "grade")               # name for the values

print("Long format:")
```

```
## [1] "Long format:"
```

```
grades_long
```

```
## # A tibble: 9 x 3
##   student subject grade
##   <chr>   <chr>   <dbl>
## 1 Alice   math       85
## 2 Alice   english    88
## 3 Alice   science    90
## 4 Bob     math       92
## 5 Bob     english    85
## 6 Bob     science    88
## 7 Charlie math       78
## 8 Charlie english    92
## 9 Charlie science    85
```

Why use long format? - Easier to make plots with ggplot2 - Better for statistical analysis - More flexible for grouping and summarizing

---

pivot_wider() - Long to Wide

Use when: You want to compare values side by side

```
# Convert back to wide format
grades_wide_again <- grades_long %>%
  pivot_wider(names_from = subject,    # column that becomes new column names
              values_from = grade)     # column that provides the values

print("Back to wide format:")
```

```
## [1] "Back to wide format:"
```

```
grades_wide_again
```

```
## # A tibble: 3 x 4
##   student  math english science
##   <chr>   <dbl>   <dbl>   <dbl>
## 1 Alice      85      88      90
## 2 Bob        92      85      88
## 3 Charlie    78      92      85
```

Part 5: Practical Examples

Example 1: Combining Student Data

```
# Real scenario: student info + course enrollments
student_info <- data.frame(
  id = 1:4,
  name = c("Amy", "Ben", "Carl", "Dana"),
  year = c(2, 1, 3, 2)
)

course_grades <- data.frame(
  id = c(1, 1, 2, 3, 3, 4),
  course = c("Math101", "Bio201", "Math101", "Eng150", "Bio201", "Eng150"),
  grade = c("A", "B+", "B", "A-", "B", "C+")
)

# Join to see student names with their grades
student_grades <- left_join(student_info, course_grades, by = "id")
print("Student grades:")
```

```
## [1] "Student grades:"
```

```
student_grades
```

```
##   id name year  course grade
## 1  1  Amy    2 Math101     A
## 2  1  Amy    2  Bio201    B+
## 3  2  Ben    1 Math101     B
## 4  3 Carl    3  Eng150    A-
## 5  3 Carl    3  Bio201     B
## 6  4 Dana    2  Eng150    C+
```

Example 2: Reshape for Analysis

```
# Convert to wide format to compare courses
grades_comparison <- student_grades %>%
  select(name, course, grade) %>%
  pivot_wider(names_from = course, values_from = grade)

print("Grades comparison:")
```

```
## [1] "Grades comparison:"
```

```
grades_comparison
```

```
## # A tibble: 4 x 4
##   name  Math101 Bio201 Eng150
##   <chr> <chr>   <chr>  <chr>
## 1 Amy   A       B+     <NA>
## 2 Ben   B       <NA>   <NA>
## 3 Carl  <NA>    B      A-
## 4 Dana  <NA>    <NA>   C+
```

Summary

Key Points to Remember:

Joins: - Use `inner_join()` when you only want complete matches - Use `left_join()` when you want to keep all rows from your main dataset - Use `anti_join()` when you want to find what's missing or unmatched - Always specify the `by =` argument to be clear about your join key

Pivots: - Use `pivot_longer()` when you have multiple columns of the same type of data - Use `pivot_wider()` when you want to compare values side by side - Long format is usually better for analysis and plotting

Next Steps:

- Practice with your own datasets
- Try combining joins and pivots in the same analysis
- Experiment with multiple join keys: `by = c("col1", "col2")`