How can I join or merge data frames in R?

Authored by

stats writer

June 24, 2024

Joining or merging data frames in R is a process of combining data from two or more data frames into a single data frame. This can be done by matching common columns or rows between the data frames. The resulting merged data frame will have all the information from the original data frames, allowing for further analysis and manipulation. There are various functions and methods available in R, such as merge() and join(), that allow for different types of joins, including inner, outer, left, and right, depending on the desired outcome. This process is essential for data integration and can help in gaining a better understanding of the data.

By using the merge() function we can perform join operation on data frames in R programming. This merge() function supports all basic SQL Join Types like inner join, left or left outer join, right or right outer join, full outer join, and cross join. If there are more than two data frames to be joined, then you can use `reduce()` method available in `tidyverse` library, this package also supports all other joins in R programming.

To perform Anti and semi-joins use the dplyr package functions `anti_join()` and `semi_join()`. In this article, I will explain multiple approaches for joining data frames using R examples. If you are interested in joining more than two data frames, please refer to the section on joining multiple data frames.

R base uses `merge()join` to perform most of the join by changing the values to parameters `all` and `by`.

Join	R Base	dplyr Package
Inner Join	`all=FALSE`	`inner_join()`
Left or Left Outer Join	`all.x=TRUE`	`left_join()`
Right or Right Outer Join	`all.y=TRUE`	`right_join()`
Full Outer Join	`all=TRUE`	`full_join()`
Cross Join	`by=NULL`, `all=FALSE`	Not Available
Left Anti Join	Not Available	`anti_join()`
Left Semi Join	Not Available	`semi_join()`

R Join Data Frames

1. merge() Syntax

Below is the merge() syntax used to join the data frames.

```
# Syntax of merge() function
merge(x, y, ...)
# S3 method for default
merge(x, y, ...)
# S3 method for data.frame
merge(x, y, by = intersect(names(x), names(y)),
by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,
sort = TRUE, suffixes = c(".x",".y"), no.dups = TRUE,
incomparables = NULL, ...)
```

Remaining tutorial I will explain Join Types using the above syntaxes.

Before we dive into examples of R joins, let's create two data frames, `emp` and `dept`. The `emp` data frame has a unique `emp_id` column, while `dept` has a unique `dept_id` column. Additionally, the `dept_id` in `emp` corresponds to the `dept_id` in `dept`.

```
# Create Emp Data Frame
emp_df=data.frame(
emp_id=c(1,2,3,4,5,6),
name=c("Smith","Rose","Williams","Jones","Brown","Brown"),
superior_emp_id=c(-1,1,1,2,2,2),
dept_id=c(10,20,10,10,40,50)
)
```

```
# Create dept Data Frame
dept_df=data.frame(
dept_id=c(10,20,30,40),
dept_name=c("Finance","Marketing","Sales","IT")
)
```

```
emp_df
dept_df
```

Yields below output.

```
> emp_df
  emp_id     name superior_emp_id dept_id
1      1    Smith               -1      10
2      2     Rose                1      20
3      3 Williams                1      10
4      4    Jones                2      10
5      5    Brown                2      40
6      6    Brown                2      50

> dept_df
  dept_id dept_name
1      10   Finance
2      20 Marketing
3      30     Sales
4      40        IT
```

2. Inner Join

In R, the default join is an inner join, also known as a natural join. It's primarily used to merge data frames based on a specific column. If the column values don't match, the corresponding rows from both data frames (like `emp` and `dept`) are excluded from the result. Here by default, it uses `all = FALSE`. This join is similar to a set intersection.

```
# R Inner Join
df2 <- merge(x = emp_df, y = dept_df, by = "dept_id")
print(df2)
```

When we perform an Inner Join on our datasets, it removes `dept_id50` from the `emp` dataset and `dept_id30` from the `dept` dataset. The resulting dataset after this join is shown below.

```
> df2
  dept_id emp_id     name superior_emp_id dept_name
1      10      1    Smith               -1   Finance
2      10      3 Williams                1   Finance
3      10      4    Jones                2   Finance
4      20      2     Rose                1 Marketing
5      40      5    Brown                2        IT
```

Similarly, use inner_join() from the dplyr package to get the same result.

```
# Import dplyr
library(dplyr)

# inner join using dplyr package
df2 = emp_df %>% inner_join(dept_df, by = "dept_id")
```

3. Left Outer Join

R left join is used to perform left joining on two data frames based on the common columns of single/multiple and return the data frame having all rows from the left data frame and any matching rows from the right data frame. Unmatched rows from the right data frame will have NA values in the columns from the right data frame.

Use all.x=TRUE to perform the left outer join in R.

```
# R left outer join data.frames
df2 <- merge(x = emp_df, y = dept_df,
by = "dept_id", all.x=TRUE)
df2
```

In our dataset, `dept_id` 50 doesn't have a corresponding record in the `dept` dataset, resulting in NA values for the dept_name. Meanwhile, `dept_id` 30 from the `dept` dataset has been removed from the results. The outcome of the Join expression is shown below.

	dept_id	emp_id	name	superior_emp_id	dept_name
1	10	1	Smith	-1	Finance
2	10	3	Williams	1	Finance
3	10	4	Jones	2	Finance
4	20	2	Rose	1	Marketing
5	40	5	Brown	2	IT
6	50	6	Brown	2	<NA>

Similarly, use left_join() from the dplyr package to get the same result.

```
# Import dplyr
library(dplyr)
```

# left join using dplyr package
df2 = emp_df %>% left_join(dept_df, by = "dept_id")

4. Right Outer Join

To perform a right outer join, use `all.y = TRUE`. This approach contrasts with a left join by keeping all rows from the right data frame, even if there's no corresponding match in the left data frame. If there's no match, the resulting dataset will contain `NA` values for the unmatched fields. Additionally, any records from the left data frame without a match in the right data frame will be excluded from the final output.

Use all.y=TRUE to perform the right outer join

```
# R right outer join data.frames
df2 <- merge(x = emp_df, y = dept_df,
by = "dept_id", all.y=TRUE)
df2
```

In our example, the right dataset has a record with `dept_id 30` that isn't present in the left dataset `emp`, resulting in NA (or null) values for the `emp` fields in this row. Additionally, the record with `dept_id 50` is excluded because there's no corresponding match in the right dataset. Below is the result of the above Join expression.

```
> df2
  dept_id emp_id     name superior_emp_id dept_name
1      10      1    Smith              -1   Finance
2      10      3 Williams               1   Finance
3      10      4    Jones               2   Finance
4      20      2     Rose               1 Marketing
5      30     NA     <NA>              NA     Sales
6      40      5    Brown               2        IT
```

Similarly, use right_join() from the dplyr package to get the same result.

```
# Import dplyr
library(dplyr)
```

# right join using dplyr package
df2 = emp_df %>% right_join(dept_df, by = "dept_id")

5. Full Outer Join

Use `all=TRUE` to perform full outer join, it returns all rows from both DataFrames, where join expression doesn't match it returns NA on respective record columns.

```
# R full outer join data.frames
df2 <- merge(x = emp_df, y = dept_df,
by = "dept_id", all=TRUE)
df2
```

From our `emp` dataset's `dept_id` with value 50 doesn't have a record on `dept` hence dept columns have NA and `dept_id` 30 doesn't have a record in `emp` hence you see `NA` on the `emp` columns. Below is the result of the above Join expression.

```
> df2
  dept_id emp_id    name superior_emp_id dept_name
1      10      1   Smith              -1   Finance
2      10      3 Williams               1   Finance
3      10      4   Jones               2   Finance
4      20      2    Rose               1 Marketing
5      30     NA    <NA>              NA     Sales
6      40      5   Brown               2        IT
7      50      6   Brown               2      <NA>
```

Similarly, use full_join() from the dplyr package to get the same result.

```
# Import dplyr
library(dplyr)
```

full join using dplyr package

df2 = emp_df %>% full_join(dept_df, by = "dept_id")

6. Cross Join

Use by=NULL to perform cross-join. A Cross Join, also known as cartesian join, performs every row of one dataframe and is being joined with every other row of another dataframe.

```
# R Cross join data.frames
df2 <- merge(x = emp_df, y = dept_df,
by = NULL)
df2
```

7. Left Semi Join

A semi join in R is like an inner join, but with a key difference, it returns all columns from the left data frame while disregarding any columns from the right data frame. Essentially, this join provides the rows from the left data frame that have a corresponding match in the right data frame based on the specified join condition. Rows in both the left and right data frames that do not meet the join condition are excluded.

You could achieve a similar result by selecting specific columns after an inner join, but a semi join is generally more efficient. In R, you can use the `semi_join()` function from the dplyr package to perform a left semi join.

```
# Import dplyr
library(dplyr)
```

# left semi join
df2 = emp_df %>% semi_join(dept_df, by = "dept_id")
df2

Below is the result of the above join expression.

```
  emp_id     name superior_emp_id dept_id
1      1    Smith               -1      10
2      2     Rose                1      20
3      3 Williams                1      10
4      4    Jones                2      10
5      5    Brown                2      40
```

8. Left Anti Join

An anti-join in R does the reverse of a left semi-join. It retrieves only the columns from the left data frame for rows where there's no corresponding match in the right data frame. To perform left anti-join in R use the `anti_join()` function from the `dplyr` package.

In other words, it selects all rows from the left data frame that are not present in the right data frame (similar to left df - right df).

```
# Import dplyr package
library(dplyr)
```

```
# left anti join data.frames
df2 = emp_df %>% anti_join(dept_df, by = "dept_id")
df2
```

Yields below output

```
  emp_id  name superior_emp_id dept_id
1      6 Brown                2      50
```

9. Source Code | R Example

```
# Create Emp Data Frame
emp_df=data.frame(
```

```
emp_id=c(1,2,3,4,5,6),
name=c("Smith","Rose","Williams","Jones","Brown","Brown"),
superior_emp_id=c(-1,1,1,2,2,2),
dept_id=c(10,20,10,10,40,50)
)

# Create dept Data Frame
dept_df=data.frame(
dept_id=c(10,20,30,40),
dept_name=c("Finance","Marketing","Sales","IT")
)

# inner join data.frames
df2 <- merge(x = emp_df, y = dept_df, by = "dept_id")

# left outer join data.frames
df2 <- merge(x = emp_df, y = dept_df,
by = "dept_id", all.x=TRUE)

# right outer join data.frames
df2 <- merge(x = emp_df, y = dept_df,
by = "dept_id", all.y=TRUE)

# full outer join data.frames
df2 <- merge(x = emp_df, y = dept_df,
by = "dept_id", all=TRUE)

# cross outer join data.frames
df2 <- merge(x = emp_df, y = dept_df,
by = NULL)

# Import dplyr package
library(dplyr)
# left semi join data.frames
df2 = emp_df %>% semi_join(dept_df, by = "dept_id")
df2

# left anti join data.frames
df2 = emp_df %>% anti_join(dept_df, by = "dept_id")
df2
```

The examples explained here are available on the GitHub project for reference.

Conclusion

In this article, I have explained how to perform R join data frames using the merge() function and dplyr package. Also, explained how to perform `inner`, `left outer`, `right outer`, `left anti`, `left semi`, and `cross` joins with examples.

Related Articles

References:

Happy Learning !!