post01: Exploring Join Operations

Introduction

In Stat 133 we have reviewed and explored the power within the package 'dplyr'. Operations such as 'filter' , 'select' , 'arrange' , 'mutate' , and 'summarise' have helped simplify and expedite common operations. While we have reviewed (and used) these operations, one set of 'dplyr' operations we have not fully reviewed are its join operations.

Join operations allow the user to bring together multiple separate tables in the form of data frames into a single data frame (df). This may be useful for when:

- The dfs contain the same rows, but different columns.
    - Ex: df1 is players' names, rebounds, 3-pointers, salary and df2 is that same set of players' name, assists, 2-pointers, and steals.
- The dfs contain different rows, but the same columns.
    - Ex: df1 is team stats for the 2016 season and df2 is team stats for the 2015 season.
- One df is a subset of the other.
    - Ex: df1 is team stats for the American League, and df2 is team stats for the entire MLB. An anti-join (which will be introduced later) may be useful to extract stats for the National League.
- And much, much, more! :)

Types of Joins

Before jumping into the syntax and examples of joins, the following graphic helps to summarize the basic idea behind each join. Each circle of the Venn-diagram represents a data frame, with the interesction representing common values (rows) between tables.

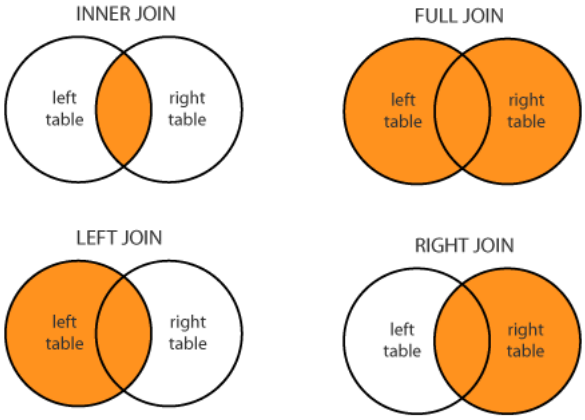
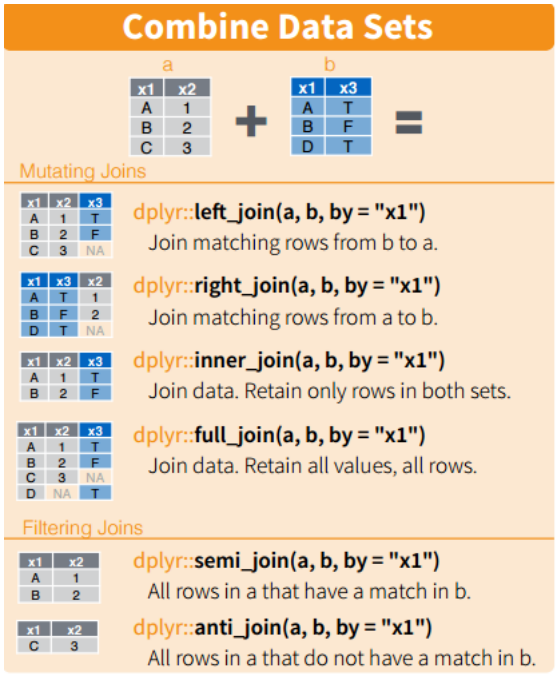


Image Courtesy of DoFactory

The different ways to implement joins in R is well summarized by the following graphic courtesy of the RStudio cheat sheet



Building Intuition

Rather than providing rigorous definitions to each type of join, it is likely more effective to explore joins through examples. Hopefully, the following simple exercises and examples will provide a degree of intuition to each join operations and when it should be used.

Let's begin by constructing 2 simple data frames.

```
names <- c("Michael", "Colin", "Sasheer", "Beck", "Kyle", "Leslie", "Lorne")
points <- c(52 ,81, 22, 121, 120, 69, 81)
rebounds <- c(1,2,1,22,3,4,4)
assists <- c(21,41,22,11,19,18,30)
steals <- c(10,9,8,7,6,9,12)
df1 <- data.frame(names, points, rebounds, stringsAsFactors = FALSE)
df2 <- data.frame(names, assists, steals, stringsAsFactors = FALSE)

head(df1)
```

```
##     names points rebounds
## 1 Michael     52        1
## 2   Colin     81        2
## 3 Sasheer     22        1
## 4    Beck    121       22
## 5    Kyle    120        3
## 6  Leslie     69        4
```

```
head(df2)
```

```
##     names assists steals
## 1 Michael      21     10
## 2   Colin      41      9
## 3 Sasheer      22      8
## 4    Beck      11      7
## 5    Kyle      19      6
## 6  Leslie      18      9
```

Now let's suppose we want to combine df1 and df2 into a single data frame. Let's take the most naive approach possible: try ALL the join operations.

```
lj <- left_join(df1, df2)
```

```
## Joining, by = "names"
```

```
lj
```

```
##     names points rebounds assists steals
## 1 Michael     52        1      21     10
## 2   Colin     81        2      41      9
## 3 Sasheer     22        1      22      8
## 4    Beck    121       22      11      7
## 5    Kyle    120        3      19      6
## 6  Leslie     69        4      18      9
## 7   Lorne     81        4      30     12
```

```
rj <- right_join(df1, df2)
```

```
## Joining, by = "names"
```

```
rj
```

```
##     names points rebounds assists steals
## 1 Michael     52        1      21     10
## 2   Colin     81        2      41      9
## 3 Sasheer     22        1      22      8
## 4    Beck    121       22      11      7
## 5    Kyle    120        3      19      6
## 6  Leslie     69        4      18      9
## 7   Lorne     81        4      30     12
```

```
ij <- inner_join(df1, df2)
```

```
## Joining, by = "names"
```

```
ij
```

```
##     names points rebounds assists steals
## 1 Michael     52        1      21     10
## 2   Colin     81        2      41      9
## 3 Sasheer     22        1      22      8
## 4    Beck    121       22      11      7
## 5    Kyle    120        3      19      6
## 6  Leslie     69        4      18      9
## 7   Lorne     81        4      30     12
```

```
fj <- full_join(df1, df2)
```

```
## Joining, by = "names"
```

```
fj
```

```
##     names points rebounds assists steals
## 1 Michael     52        1      21     10
## 2   Colin     81        2      41      9
## 3 Sasheer     22        1      22      8
## 4    Beck    121       22      11      7
## 5    Kyle    120        3      19      6
## 6  Leslie     69        4      18      9
## 7   Lorne     81        4      30     12
```

```
sj <- semi_join(df1, df2)
```

```
## Joining, by = "names"
```

```
sj
```

```
##     names points rebounds
## 1 Michael     52        1
## 2   Colin     81        2
## 3 Sasheer     22        1
## 4    Beck    121       22
## 5    Kyle    120        3
## 6  Leslie     69        4
## 7   Lorne     81        4
```

```
aj <- anti_join(df1, df2)
```

```
## Joining, by = "names"
```

```
aj
```

```
## [1] names    points   rebounds
## <0 rows> (or 0-length row.names)
```

As we can observe from above, the joins that give us the result we want are the left, right, inner, and full join.

So how, does these joins differ? Let's explore this by having the two data frames having df2 almost the same as last, but this time with some (but not all) the same players as df1.

```
names1 <- c("Michael", "Colin", "Sasheer", "Beck", "Kyle", "Leslie", "Lorne")
names2 <- c("Michael", "Aidy", "Beck", "Kyle", "Fred", "Bill")
points1 <- c(52 ,81, 22, 121, 120, 69, 81)
rebounds1 <- c(1,2,1,22,3,4,4)
assists2 <- c(21,11,11,19,20,20)
steals2 <- c(10,19,7,6,28,29)

df1 <- data.frame(name = names1, points = points1, rebounds = rebounds1, stringsAsFactors = FALSE)
df2 <- data.frame(name = names2, assists = assists2, steals = steals2, stringsAsFactors = FALSE)
```

```
lj <- left_join(df1, df2)
```

```
## Joining, by = "name"
```

```
lj
```

```
##      name points rebounds assists steals
## 1 Michael     52        1      21     10
## 2   Colin     81        2      NA     NA
## 3 Sasheer     22        1      NA     NA
## 4    Beck    121       22      11      7
## 5    Kyle    120        3      19      6
## 6  Leslie     69        4      NA     NA
## 7   Lorne     81        4      NA     NA
```

```
rj <- right_join(df1, df2)
```

```
## Joining, by = "name"
```

```
rj
```

```
##      name points rebounds assists steals
## 1 Michael     52        1      21     10
## 2    Aidy     NA       NA      11     19
## 3    Beck    121       22      11      7
## 4    Kyle    120        3      19      6
## 5    Fred     NA       NA      20     28
## 6    Bill     NA       NA      20     29
```

```
ij <- inner_join(df1, df2)
```

```
## Joining, by = "name"
```

```
ij
```

```
##      name points rebounds assists steals
## 1 Michael     52        1      21     10
## 2    Beck    121       22      11      7
## 3    Kyle    120        3      19      6
```

```
fj <- full_join(df1, df2)
```

```
## Joining, by = "name"
```

```
fj
```

```
##       name points rebounds assists steals
## 1  Michael     52        1      21     10
## 2    Colin     81        2      NA     NA
## 3  Sasheer     22        1      NA     NA
## 4     Beck    121       22      11      7
## 5     Kyle    120        3      19      6
## 6   Leslie     69        4      NA     NA
## 7    Lorne     81        4      NA     NA
## 8     Aidy     NA       NA      11     19
## 9     Fred     NA       NA      20     28
## 10    Bill     NA       NA      20     29
```

```
sj <- semi_join(df1, df2)
```

```
## Joining, by = "name"
```

```
sj
```

```
##      name points rebounds
## 1 Michael     52        1
## 2    Beck    121       22
## 3    Kyle    120        3
```

```
aj <- anti_join(df1, df2)
```

```
## Joining, by = "name"
```

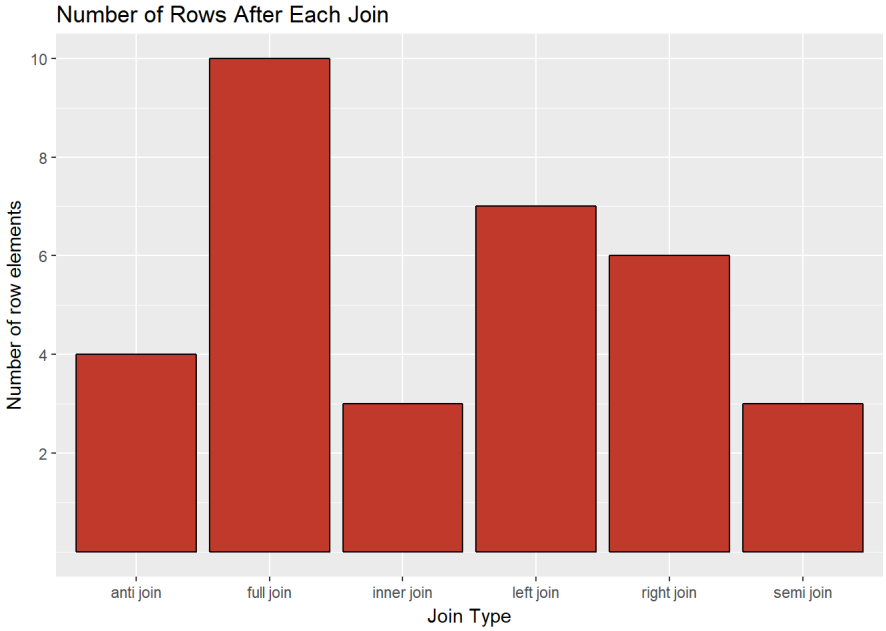
```
aj
```

```
##      name points rebounds
## 1   Colin     81        2
## 2 Sasheer     22        1
## 3  Leslie     69        4
## 4   Lorne     81        4
```

By looking at each join more closely, we can get a better idea of how they differ from one another.

- **left_join**: Within the left join, only the names from the first (left) dataframe are kept. Wherever there are unknown values for a given person, an "NA" is put in as a placeholder.
- **right_join**: Within the right join, only the names from the last (right) dataframe are kept. Wherever there are unknown values for a given person, an "NA" is put in as a placeholder.
- **inner_join**: This join appears the same as a left (or right) join, where only the names within both dataframes appear in the final table.
- **full_join**: This join includes all possible rows from the left and right joins, i.e. all names are included.
- **semi_join**: Returns all rows in df1 with a match in df2.
- **anti_join**: Returns all rows in df1 WITHOUT a match in df2.

```
arr <- c(nrow(lj), nrow(rj), nrow(ij), nrow(fj), nrow(sj), nrow(aj))
names <- c("left join", "right join", "inner join", "full join" , "semi join", "anti join")
df_nrows <- data.frame(names, arr)
p <-ggplot(df_nrows, aes(names,arr))
p +
  labs(x = "Join Type", y = "Number of row elements", title = "Number of Rows After Each Join") +
  geom_bar(stat = "identity", color = "black", fill = "#c0392b") +
  scale_y_continuous(breaks=c(2,4,6,8,10))
```



Based off of the bar graph above, and using what you know about joins, can you characterize what the Venn Diagram looks like for the two tables used? In other words how many rows are there in the left table, the right table, and how many rows do they have in common?

DISCUSSION: Exploring Joins through Examples!

Below we will explore, in much briefer examples, additional operations one could perform using the join operations we have learned above. The less common joins (anti, semi) will not be explored.

For all of the examples we will use the following data frames:

```
names_a <- c("A","B","C")
nums_a <- c(1,2,3)
bool_a <- c(FALSE,FALSE,FALSE)
df_a <- data.frame(names = names_a, nums = nums_a, bool = bool_a, stringsAsFactors = FALSE)
df_a
```

```
##   names nums  bool
## 1     A    1 FALSE
## 2     B    2 FALSE
## 3     C    3 FALSE
```

```
names_b <- c("B","A","D")
nums_b <- c(2,1,4)
bool_b <- c(FALSE,FALSE,FALSE)
df_b = data.frame(names = names_b, nums = nums_b, bool = bool_b, stringsAsFactors = FALSE)
df_b
```

```
##   names nums  bool
## 1     B    2 FALSE
## 2     A    1 FALSE
## 3     D    4 FALSE
```

```
names_c <- c("D","E")
nums_c <- c(4,5)
bool_c <- c(FALSE,TRUE)
df_c <- data.frame(names = names_c, nums = nums_c, bool = bool_c, stringsAsFactors = FALSE)
df_c
```

```
##   names nums  bool
## 1     D    4 FALSE
## 2     E    5  TRUE
```

Using “by =” when joining

When joining two (or more) tables it is useful to specify what you want to join “by.” For example: if you join by just “name,” the join operator will perform the appropriate join where the “name” values are the same. If the two tables are joined by “name, nums, bool,” all of those values must be the same across both tables.

```
# NOTE: df_c has no additional information on any row in df_a. So, perhaps the "best" join would return only df_a
(as df_C has nothing to contribute).

# Only names must be the same. Results in MESSIER join.
df_by1 <- left_join(df_a, df_c, by = c("names"))
df_by1
```

```
##   names nums.x bool.x nums.y bool.y
## 1     A      1  FALSE     NA     NA
## 2     B      2  FALSE     NA     NA
## 3     C      3  FALSE     NA     NA
```

```
# All col values (names, nums, bool) must be the same. Results in CLEANER join.
df_by2 <- left_join(df_a, df_c, by = c("names", "nums", "bool"))
df_by2
```

```
##   names nums  bool
## 1     A    1 FALSE
## 2     B    2 FALSE
## 3     C    3 FALSE
```

Nested Join Functions

Like most other functions, users can implement a function on the result of another funciton, i.e. f o g(x) = f(g(x))

```
# Perform inner join on result of (right join of df_a and df_b) and df_c
df_nested <- right_join((right_join(df_a, df_b)) , df_c)
```

```
## Joining, by = c("names", "nums", "bool")
## Joining, by = c("names", "nums", "bool")
```

Joining Multiple Tables

Below we will try to perform multiple joins. First by right joining df_a, df_b. And then right joining that result and df_c.

It turns out that putting multiple tables into the join function does not work as you might think, and for clarity’s sake (and accuracy’s sake), it is often better to nest the functions.

```
# DON'T DO THIS
df_mult <- right_join(df_a, df_b, df_c, by = c("names", "nums", "bool"))

# Do this:
df_mult <- right_join(df_a, df_b, by = c("names", "nums", "bool")) %>%
  right_join(., df_c, by = c("names", "nums", "bool"))

df_mult
```

```
##   names nums  bool
## 1     D    4 FALSE
## 2     E    5  TRUE
```

Same name, different row?

What happens if one cell of a table row is changed? How will that affect our most common join operations?

```r
# Change the value of A's boolean in the a dataframe to TRUE.
df_a$bool = c(TRUE, FALSE, FALSE)

# This will make an inner join by all cols exclude A's row, as the two A rows are no longer the same.
df_name <- inner_join(df_a, df_b, by = c("names", "nums", "bool"))
df_name
```

```
##   names nums  bool
## 1     B    2 FALSE
```

```r
# However, it will not impact left or right join.
df_name2 <- left_join(df_a, df_b, by = c("names", "nums", "bool"))
df_name2
```

```
##   names nums  bool
## 1     A    1  TRUE
## 2     B    2 FALSE
## 3     C    3 FALSE
```

Conclusion

This blog was intended to give a taste for the ways to use joins and provide an idea of what they are capable of. It is by no means comprehensive, it is simply to provide the basic gist for a novice.

Uncovered Concepts

The Merge Operation:

This is R's built-in join operator. It was not covered, because dplyr's join operation (what was covered), is much faster.

Other dplyr methods for combining data sets:

In this blog we have covered mutating joins (left, right, inner, full) and filtering joins (semi, anti).

There are also, as can be found in the RStudio Cheat Sheet, set operations (intersect, union, setdiff), and binding operations (bind_rows, bind_cols). For more information refer to the RStudio Cheat Sheet.

Additional Resources:

- Using the built-in merge function
- Dplyr Documentation
- Intro Video to dplyr joins

Quick Review

Here is a brief recap of what we learned:

- The basic definition of a join operation.
- The motivation behind join operations.
- The types of join operations offered in the dplyr package.
    - The result of each join on two example data frames.
- EXAMPLES!
    - Using the "by" input in a join
    - Nested Joins
    - Joining Multiple Tables
    - Same name, different row?
        - How are different joins affected by a change in a single cell's value?

Buh-Bye

That's it folks! Thank you for reading.