

Merging Data Sets

Data Wrangling in R

The **merge** function exists, don't use
it

Joining in `dplyr`

- Merging/joining data sets together - usually on key variables, usually “id”
- `?join` - see different types of joining for `dplyr`
- Let's look at <https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf>
- `inner_join(x, y)` - only rows that match for `x` and `y` are kept
- `full_join(x, y)` - all rows of `x` and `y` are kept
- `left_join(x, y)` - all rows of `x` are kept even if not merged with `y`
- `right_join(x, y)` - all rows of `y` are kept even if not merged with `x`
- `anti_join(x, y)` - all rows from `x` not in `y` keeping just columns from `x`.

Merging: Simple Data

base has baseline data for ids 1 to 10 and Age

```
base <- tibble(id = 1:10, Age = seq(55, 60, length=10))  
head(base, 2)
```

```
# A tibble: 2 x 2  
  id   Age  
  <int> <dbl>  
1     1  55  
2     2 55.6
```

visits has ids 1 to 8, then 11 (new id), and 3 visits and outcome

```
visits <- tibble(id = c(rep(1:8, 3), 11), visit= c(rep(1:3, 8), 3),  
                 Outcome = seq(10, 50, length=25))  
tail(visits, 2)
```

```
# A tibble: 2 x 3  
  id visit Outcome  
  <dbl> <dbl>   <dbl>  
1     8     3    48.3  
2    11     3    50
```

Inner Join

```
ij = inner_join(base, visits)
```

```
Joining, by = "id"
```

```
dim(ij)
```

```
[1] 24  4
```

```
tail(ij)
```

```
# A tibble: 6 x 4
   id    Age visit Outcome
  <dbl> <dbl> <dbl>   <dbl>
1     7  58.3     1     20
2     7  58.3     3    33.3
3     7  58.3     2    46.7
4     8  58.9     2    21.7
5     8  58.9     1     35
6     8  58.9     3    48.3
```

Left Join

```
lj = left_join(base, visits)
```

```
Joining, by = "id"
```

```
dim(lj)
```

```
[1] 26  4
```

```
tail(lj)
```

```
# A tibble: 6 x 4
   id   Age visit Outcome
  <dbl> <dbl> <dbl>   <dbl>
1     7  58.3     2    46.7
2     8  58.9     2    21.7
3     8  58.9     1     35
4     8  58.9     3    48.3
5     9  59.4    NA     NA
6    10  60     NA     NA
```

Right Join

```
rj = right_join(base, visits)
```

```
Joining, by = "id"
```

```
tail(rj, 3)
```

```
# A tibble: 3 x 4  
   id    Age visit Outcome  
  <dbl> <dbl> <dbl>   <dbl>  
1     8  58.9     1     35  
2     8  58.9     3    48.3  
3    11   NA     3     50
```

Right Join: Switching arguments

```
rj2 = right_join(visits, base)
```

```
Joining, by = "id"
```

```
tail(rj2, 3)
```

```
# A tibble: 3 x 4  
   id visit Outcome Age  
  <dbl> <dbl>   <dbl> <dbl>  
1     8     3    48.3  58.9  
2     9    NA     NA   59.4  
3    10    NA     NA   60
```

```
identical(rj2, lj) ## after some rearranging
```

```
[1] TRUE
```


Full Join

```
fj = full_join(base, visits)
```

```
Joining, by = "id"
```

```
tail(fj, 4)
```

```
# A tibble: 4 x 4
   id    Age visit Outcome
  <dbl> <dbl> <dbl>   <dbl>
1     8  58.9     3    48.3
2     9  59.4    NA     NA
3    10   60    NA     NA
4    11   NA     3     50
```

Logging the joins

The `tidylog` package can show you log outputs from `dplyr` (newly added). You will need to install to use.

```
library(tidylog)
left_join(base, visits)
```

```
Joining, by = "id"
```

```
left_join: added 2 columns (visit, Outcome)
```

```
> rows only in x      2
```

```
> rows only in y    ( 1)
```

```
> matched rows      24      (includes duplicates)
```

```
>                      =====
```

```
> rows total        26
```

```
# A tibble: 26 x 4
```

	id	Age	visit	Outcome
	<dbl>	<dbl>	<dbl>	<dbl>
1	1	55	1	10
2	1	55	3	23.3
3	1	55	2	36.7

Using the `by` argument

By default - uses intersection of column names. If `by` specified, then uses that, but if other columns with same name, adds `suffix`.

```
base = base %>% mutate(x = 5)
visits = visits %>% mutate(x = 4)
head(full_join(base, visits))
```

Joining, `by = c("id", "x")`

```
# A tibble: 6 x 5
   id    Age    x visit Outcome
<dbl> <dbl> <dbl> <dbl>   <dbl>
1     1    55     5     NA      NA
2     2   55.6     5     NA      NA
3     3   56.1     5     NA      NA
4     4   56.7     5     NA      NA
5     5   57.2     5     NA      NA
6     6   57.8     5     NA      NA
```

Using the **by** argument

```
head(full_join(base, visits, by = "id"))
```

```
# A tibble: 6 x 6
```

	id	Age	x.x	visit	Outcome	x.y
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	55	5	1	10	4
2	1	55	5	3	23.3	4
3	1	55	5	2	36.7	4
4	2	55.6	5	2	11.7	4
5	2	55.6	5	1	25	4
6	2	55.6	5	3	38.3	4

```
head(full_join(base, visits, by = "id", suffix = c("_base", "_visit")))
```

```
# A tibble: 6 x 6
```

	id	Age	x_base	visit	Outcome	x_visit
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	55	5	1	10	4
2	1	55	5	3	23.3	4
3	1	55	5	2	36.7	4
4	2	55.6	5	2	11.7	4
5	2	55.6	5	1	25	4
6	2	55.6	5	3	38.3	4

Using the **by** argument if column names different

```
base = base %>%  
  select(-x) %>%  
  mutate(myvar = 4)  
visits = visits %>%  
  select(-x) %>%  
  mutate(MyVar = 4)  
full_join(base, visits, by = c("id", "myvar" = "MyVar"))
```

```
# A tibble: 27 x 5  
   id    Age myvar visit Outcome  
  <dbl> <dbl> <dbl> <dbl>   <dbl>  
1     1    55     4     1     10  
2     1    55     4     3    23.3  
3     1    55     4     2    36.7  
4     2   55.6     4     2    11.7  
5     2   55.6     4     1     25  
6     2   55.6     4     3    38.3  
7     3   56.1     4     3    13.3  
8     3   56.1     4     2    26.7  
9     3   56.1     4     1     40  
10    4   56.7     4     1     15  
# ... with 17 more rows
```