Merging Data Sets

Data Wrangling in R

Joining in `dplyr`

- Merging/joining data sets together - usually on key variables, usually “id”
- `?join` - see different types of joining for `dplyr`
- `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)
```

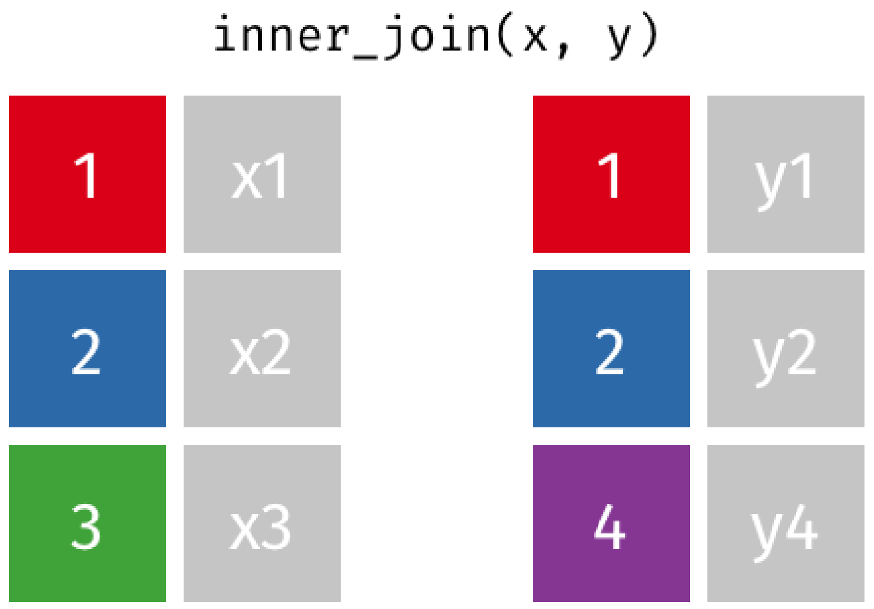
```
# A tibble: 2 x 2
     id   Age
  <int> <dbl>
1     1  55
2     2  55.6
```

`visits` has ids `2` to `11`, 3 different visits, and an outcome

```
visits <- tibble(id = rep(2:11, 3), visit= rep(1:3, 10),
                    Outcome = seq(10,50, length=30))
head(visits, 2)
```

```
# A tibble: 2 x 3
     id visit Outcome
  <int> <int>   <dbl>
1     2     1    10
2     3     2    11.4
```

Inner Join



https://github.com/gadenbuie/tidyexplain/blob/master/images/inner-join.gif

Inner Join

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

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

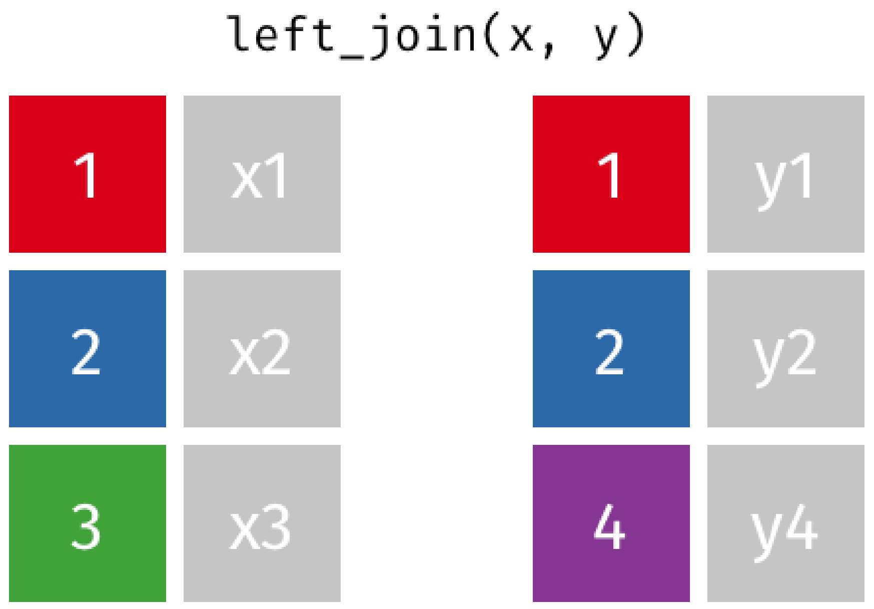
```
dim(ij)
```

```
[1] 27  4
```

```
head(ij)
```

```
# A tibble: 6 x 4
     id   Age visit Outcome
  <int> <dbl> <int>   <dbl>
1     2  55.6     1    10  
2     2  55.6     2    23.8
3     2  55.6     3    37.6
4     3  56.1     2    11.4
5     3  56.1     3    25.2
6     3  56.1     1    39.0
```

Left Join



https://github.com/gadenbuie/tidyexplain/blob/master/images/left-join.gif

Left Join

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

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

```
dim(lj)
```

```
[1] 28  4
```

```
head(lj)
```

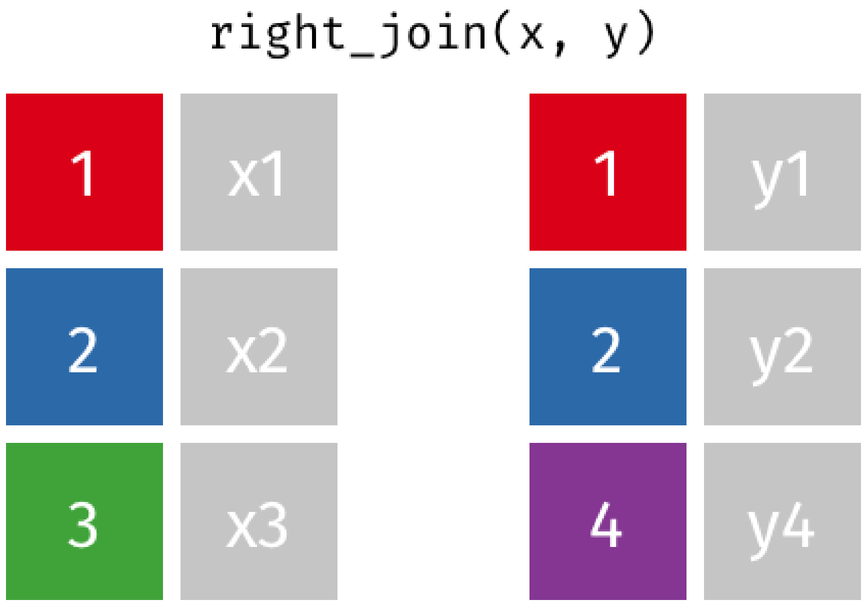
```
# A tibble: 6 x 4
     id   Age visit Outcome
  <int> <dbl> <int>   <dbl>
1     1  55      NA    NA  
2     2  55.6     1    10  
3     2  55.6     2    23.8
4     2  55.6     3    37.6
5     3  56.1     2    11.4
6     3  56.1     3    25.2
```

Install `tidylog` package to log outputs

```r
# install.packages("tidylog")
library(tidylog)
left_join(base, visits)
```

```
Joining, by = "id"

left_join: added 2 columns (visit, Outcome)

           > rows only in x    1

           > rows only in y  ( 3)

           > matched rows     27    (includes duplicates)

           >                 ====

           > rows total       28

# A tibble: 28 x 4
     id   Age visit Outcome
  <int> <dbl> <int>   <dbl>
1     1  55      NA    NA
2     2  55.6     1    10
3     2  55.6     2    23.8
4     2  55.6     3    37.6
5     3  56.1     2    11.4
6     3  56.1     3    25.2
```

Right Join



https://github.com/gadenbuie/tidyexplain/blob/master/images/right-join.gif

Right Join

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

```
Joining, by = "id"
right_join: added 2 columns (visit, Outcome)
            > rows only in x  ( 1)
            > rows only in y    3
            > matched rows     27
            >                 ====
            > rows total       30
```

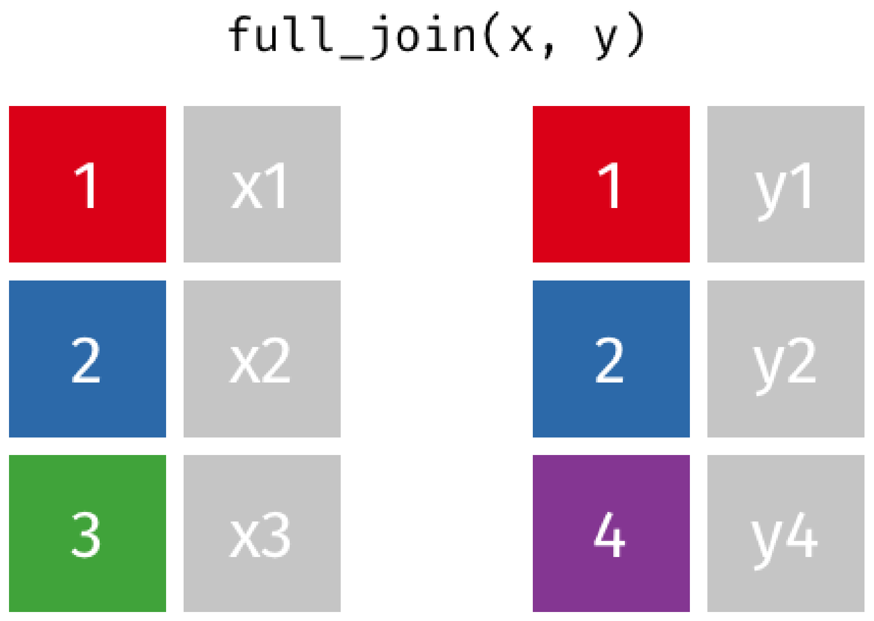
Left Join: Switching arguments

```
lj2 = left_join(visits, base)
```

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

```
left_join: added one column (Age)
           > rows only in x    3
           > rows only in y  ( 1)
           > matched rows     27
           >                ====
           > rows total       30
```

Full Join



https://github.com/gadenbuie/tidyexplain/blob/master/images/full-join.gif

Full Join

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

```
Joining, by = "id"

full_join: added 2 columns (visit, Outcome)

           > rows only in x     1

           > rows only in y     3

           > matched rows      27    (includes duplicates)

           >                 ====

           > rows total        31
```

Full Join

Note what tidylog means by `includes duplicates`. Data from `base` is being duplicated.

```r
# fj = full_join(base, visits)
head(fj, 10)
```

```
# A tibble: 10 x 4
      id   Age visit Outcome
   <int> <dbl> <int>   <dbl>
 1     1  55      NA    NA  
 2     2  55.6     1    10  
 3     2  55.6     2    23.8
 4     2  55.6     3    37.6
 5     3  56.1     2    11.4
 6     3  56.1     3    25.2
 7     3  56.1     1    39.0
 8     4  56.7     3    12.8
 9     4  56.7     1    26.6
10     4  56.7     2    40.3
```

Duplicated

The `duplicated` command can give you indications if there are duplications in a **vector**:

```
duplicated(1:5)
```

```
[1] FALSE FALSE FALSE FALSE FALSE
```

```
duplicated(c(1:5, 1))
```

```
[1] FALSE FALSE FALSE FALSE FALSE  TRUE
```

```
fj %>% mutate(dup_id = duplicated(id))
```

```
# A tibble: 31 x 5
      id   Age visit Outcome dup_id
   <int> <dbl> <int>   <dbl> <lgl>
 1     1  55      NA     NA   FALSE
 2     2  55.6     1     10   FALSE
 3     2  55.6     2     23.8 TRUE
 4     2  55.6     3     37.6 TRUE
 5     3  56.1     2     11.4 FALSE
 6     3  56.1     3     25.2 TRUE
 7     3  56.1     1     39.0 TRUE
 8     4  56.7     3     12.8 FALSE
 9     4  56.7     1     26.6 TRUE
10     4  56.7     2     40.3 TRUE
# … with 21 more rows
```

Using the `by` argument

By default - uses intersection of column names. If `by` specified, then uses that.

```
# for multiple, by = c(col1, col2)
head(full_join(base, visits, by = "id"))
```

```
# A tibble: 6 x 4
     id   Age visit Outcome
  <int> <dbl> <int>   <dbl>
1     1  55      NA     NA
2     2  55.6     1     10
3     2  55.6     2     23.8
4     2  55.6     3     37.6
5     3  56.1     2     11.4
6     3  56.1     3     25.2
```

Using the `by` argument

You can use `by` if the column names don't match exactly.

```
base2 = base %>% rename(patient = id) # rename the column
head(full_join(base2, visits, by = c("patient" = "id")))
```

```
# A tibble: 6 x 4
  patient   Age visit Outcome
    <int> <dbl> <int>   <dbl>
1       1  55      NA    NA  
2       2  55.6     1    10  
3       2  55.6     2    23.8
4       2  55.6     3    37.6
5       3  56.1     2    11.4
6       3  56.1     3    25.2
```