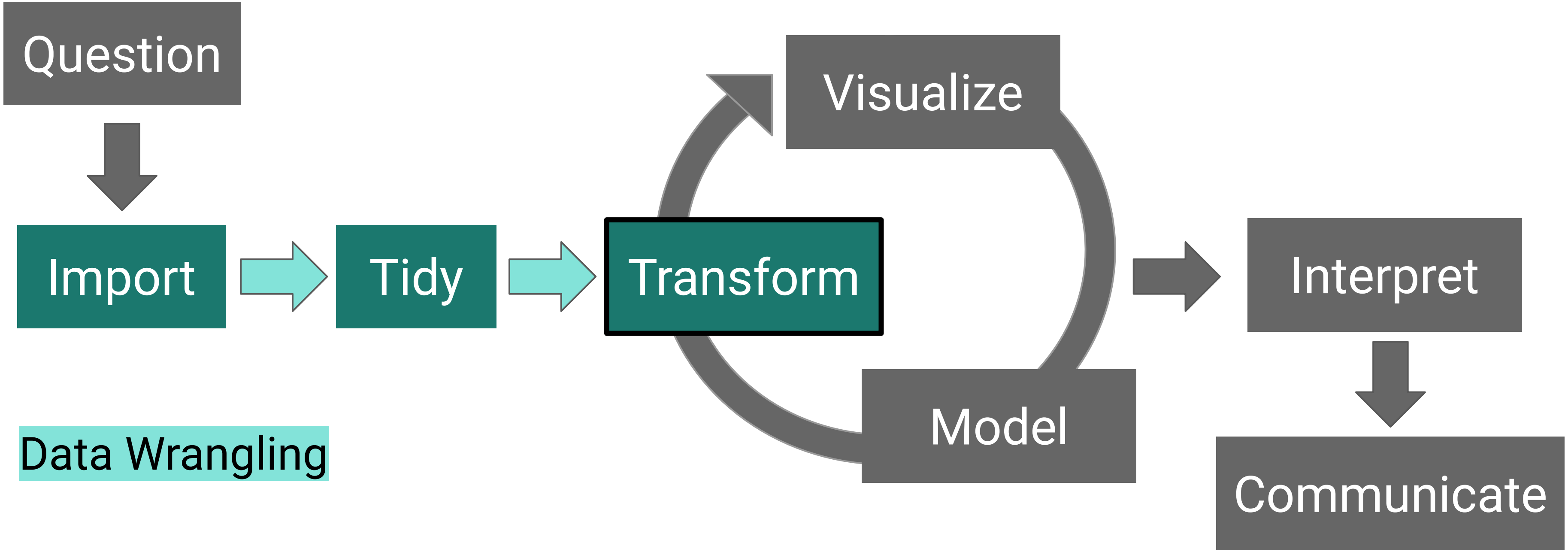
Merging Data

Lecture 6

Objective

- To merge data frames

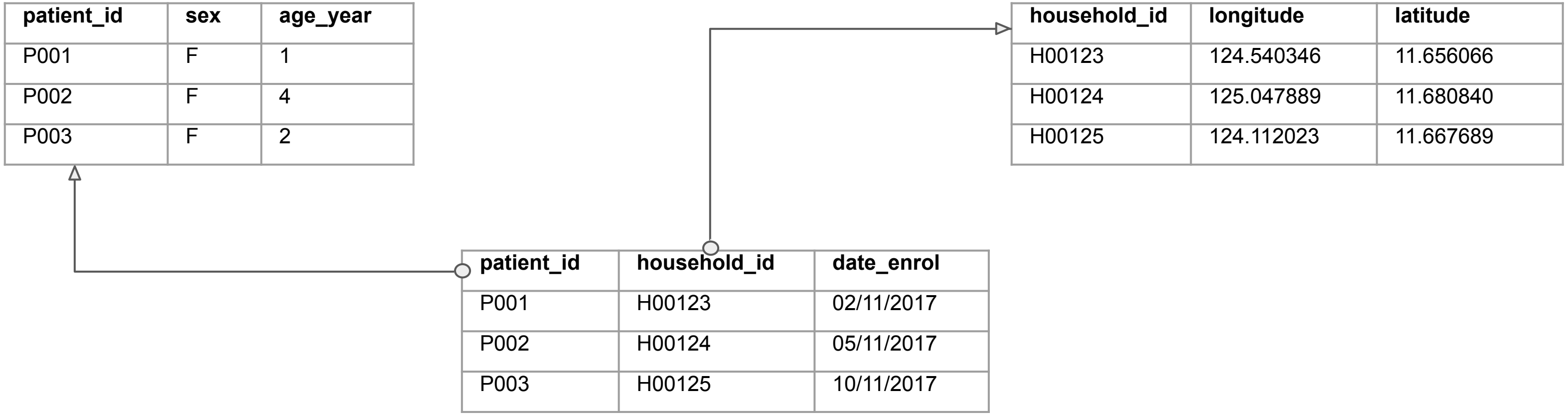
Motivation



Modified from https://r4ds.had.co.nz/explore-intro.html#explore-intro

Motivation

- it is rare that data analysis involves only a single table of data
- usually many tables of data are involved, which must combine to answer the questions of interest
- most data stored in relational database (e.g. SQL database)



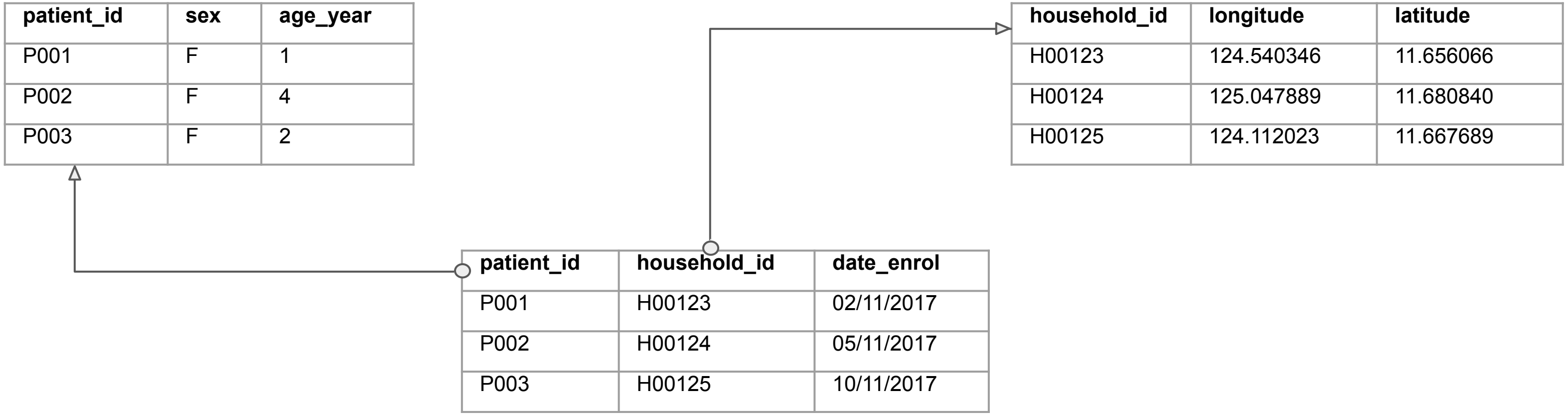
patient_id	sex	age_year
P001	F	1
P002	F	4
P003	F	2

household_id	longitude	latitude
H00123	124.540346	11.656066
H00124	125.047889	11.680840
H00125	124.112023	11.667689

patient_id	household_id	date_enrol
P001	H00123	02/11/2017
P002	H00124	05/11/2017
P003	H00125	10/11/2017

Keys

- variables used to connect each pair of tables
- variables that uniquely identifies an observation



patient_id	sex	age_year
P001	F	1
P002	F	4
P003	F	2

household_id	longitude	latitude
H00123	124.540346	11.656066
H00124	125.047889	11.680840
H00125	124.112023	11.667689

patient_id	household_id	date_enrol
P001	H00123	02/11/2017
P002	H00124	05/11/2017
P003	H00125	10/11/2017

3 Functions for Merging Data Frames

1. **Mutating joins** - add new variables to one data frame that match observations in another
2. **Filtering joins** - filter observations from one data frame that match an observation in the other table
3. **Set operations** - treat observations as elements of a set

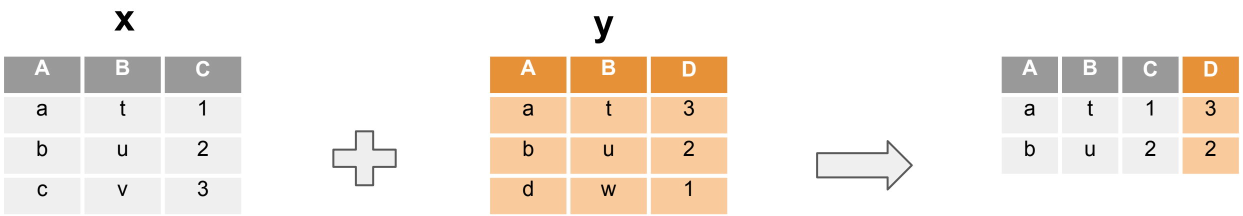
Mutating joins

- mutating join allows you to combine variables from two tables
- first matches observations by their keys, then copies across variables from one table to the other
- inner join and outer join

inner_join( )

- inner join matches pairs of observations whenever their keys are equal
- keeps observations that appear in both tables

```
> inner_join(x, y, by = c("A", "B"))
```



x

A	B	C
a	t	1
b	u	2
c	v	3

y

A	B	D
a	t	3
b	u	2
d	w	1

A	B	C	D
a	t	1	3
b	u	2	2

inner_join( )

```
> data_1
```

```
# A tibble: 5 x 3
patient_ID   sex      age_year
  <chr>      <chr>    <dbl>
1 P001       female   1
2 P002       female   4
3 P003       female   2
4 P004       male     3
5 P005       male     4
```

```
> data_2
```

```
# A tibble: 5 x 3
  patient_id weight_kg height_cm
  <chr>      <dbl>      <dbl>
1 P001       9.1        73
2 P002       16.4       NA
3 P003       10.5       85
4 P004       13.2       95
5 P006       15.9      104
```

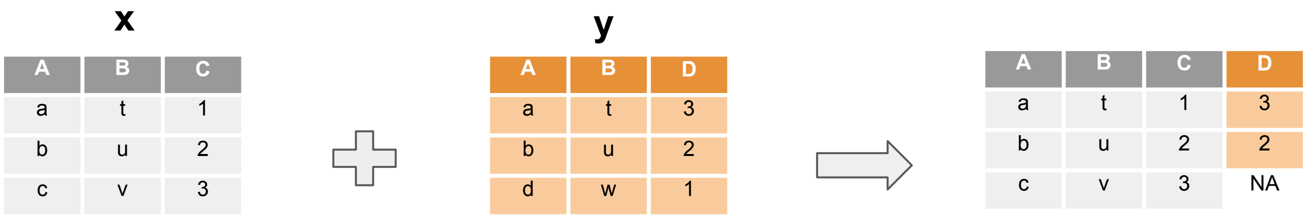
```
> inner_join(data_1, data_2, by = c("patient_ID" = "patient_id"))
```

```
# A tibble: 4 x 5
  patient_ID sex     age_year weight_kg  height_cm
  <chr>      <chr>     <dbl>    <dbl>       <dbl>
1 P001       female      1        9.1        73
2 P002       female      4       16.4        NA
3 P003       female      2       10.5        85
4 P004       male        3       13.2        95
```

left_join( )

- outer join keeps observations that appear in at least one of the tables
- **left join** keeps all observations in x

```
> left_join(x, y, by = c("A", "B")
```



x

A	B	C
a	t	1
b	u	2
c	v	3

y

A	B	D
a	t	3
b	u	2
d	w	1

A	B	C	D
a	t	1	3
b	u	2	2
c	v	3	NA

left_join( )

```
> data_1
```

```
# A tibble: 5 x 3
patient_ID   sex      age_year
  <chr>      <chr>    <dbl>
1 P001       female   1
2 P002       female   4
3 P003       female   2
4 P004       male     3
5 P005       male     4
```

```
> data_2
```

```
# A tibble: 5 x 3
  patient_id weight_kg height_cm
  <chr>      <dbl>     <dbl>
1 P001       9.1       73
2 P002       16.4      NA
3 P003       10.5      85
4 P004       13.2      95
5 P006       15.9     104
```

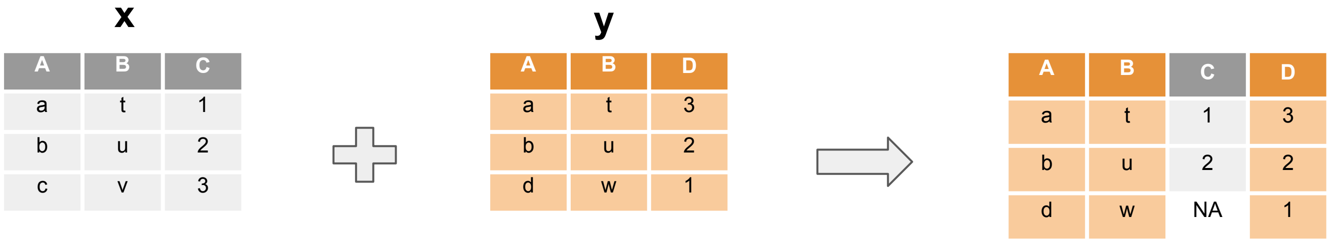
```
> left_join(data_1, data_2, by = c("patient_ID" = "patient_id"))
```

```
# A tibble: 5 x 5
  patient_ID sex     age_year weight_kg  height_cm
  <chr>      <chr>      <dbl>    <dbl>      <dbl>
1 P001       female       1        9.1         73
2 P002       female       4       16.4         NA
3 P003       female       2       10.5         85
4 P004       male         3       13.2         95
5 P005       male         4       NA           NA
```

right_join( )

- outer join keeps observations that appear in at least one of the tables
- **right join** keeps all observations in y

```
> right_join(x, y, by = c("A", "B")
```



x

A	B	C
a	t	1
b	u	2
c	v	3

y

A	B	D
a	t	3
b	u	2
d	w	1

A	B	C	D
a	t	1	3
b	u	2	2
d	w	NA	1

right_join( )

```
> data_1
```

```
# A tibble: 5 x 3
patient_ID     sex          age_year
  <chr>        <chr>        <dbl>
1 P001         female       1
2 P002         female       4
3 P003         female       2
4 P004         male         3
5 P005         male         4
```

```
> data_2
```

```
# A tibble: 5 x 3
  patient_id weight_kg height_cm
  <chr>       <dbl>       <dbl>
1 P001        9.1         73
2 P002        16.4        NA
3 P003        10.5        85
4 P004        13.2        95
5 P006        15.9       104
```

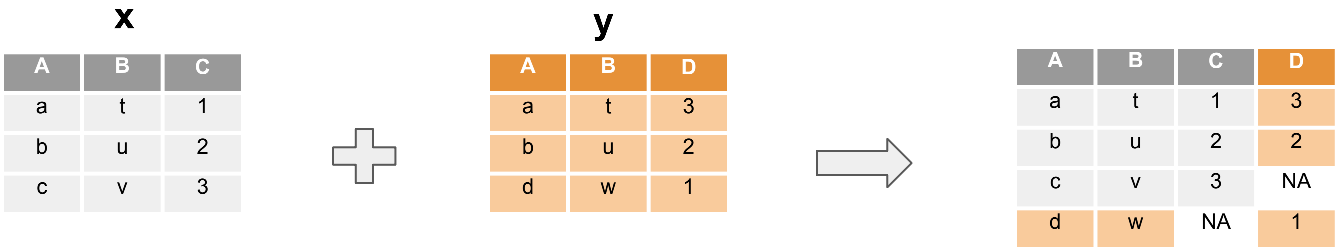
```
> right_join(data_1, data_2, by = c("patient_ID" = "patient_id"))
```

```
# A tibble: 5 x 5
  patient_ID sex      age_year weight_kg height_cm
  <chr>       <chr>    <dbl>     <dbl>      <dbl>
1 P001        female       1       9.1         73
2 P002        female       4      16.4         NA
3 P003        female       2      10.5         85
4 P004        male         3      13.2         95
5 P006        NA           NA     15.9        104
```

full_join( )

- **full join** keeps all observations in x and y

```
> full_join(x, y, by = c("A", "B")
```



x

A	B	C
a	t	1
b	u	2
c	v	3

y

A	B	D
a	t	3
b	u	2
d	w	1

A	B	C	D
a	t	1	3
b	u	2	2
c	v	3	NA
d	w	NA	1

full_join( )

```
> data_1
```

```
# A tibble: 5 x 3
patient_ID     sex            age_year
  <chr>        <chr>          <dbl>
1 P001         female         1
2 P002         female         4
3 P003         female         2
4 P004         male           3
5 P005         male           4
```

```
> data_2
```

```
# A tibble: 5 x 3
  patient_id weight_kg height_cm
  <chr>         <dbl>         <dbl>
1 P001          9.1           73
2 P002          16.4          NA
3 P003          10.5          85
4 P004          13.2          95
5 P006          15.9         104
```

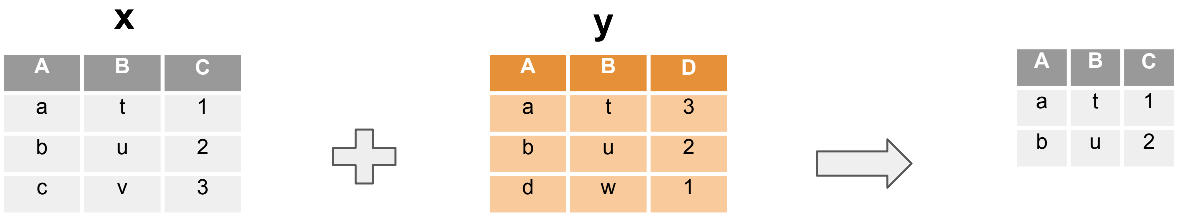
```
> full_join(data_1, data_2, by = c("patient_ID" = "patient_id"))
```

```
  patient_ID sex       age_year weight_kg height_cm
  <chr>         <chr>     <dbl>       <dbl>        <dbl>
1 P001          female        1         9.1           73
2 P002          female        4        16.4           NA
3 P003          female        2        10.5           85
4 P004          male          3        13.2           95
5 P005          male          4          NA           NA
6 P006          NA            NA       15.9          104
```

Filtering joins

- filtering joins match observations in the same way as mutating joins, but affect the observations, instead of the variables
- **semi_join( )** - **keeps** all observations in x that have a match in y

```
> semi_join(x, y)
```



x

A	B	C
a	t	1
b	u	2
c	v	3

y

A	B	D
a	t	3
b	u	2
d	w	1

A	B	C
a	t	1
b	u	2

semi_join( )

```
> data_1
```

```
# A tibble: 5 x 3
patient_ID     sex         age_year
  <chr>        <chr>       <dbl>
1 P001         female      1
2 P002         female      4
3 P003         female      2
4 P004         male        3
5 P005         male        4
```

```
> data_2
```

```
# A tibble: 5 x 3
  patient_id weight_kg height_cm
  <chr>        <dbl>        <dbl>
1 P001         9.1          73
2 P002         16.4         NA
3 P003         10.5         85
4 P004         13.2         95
5 P006         15.9        104
```

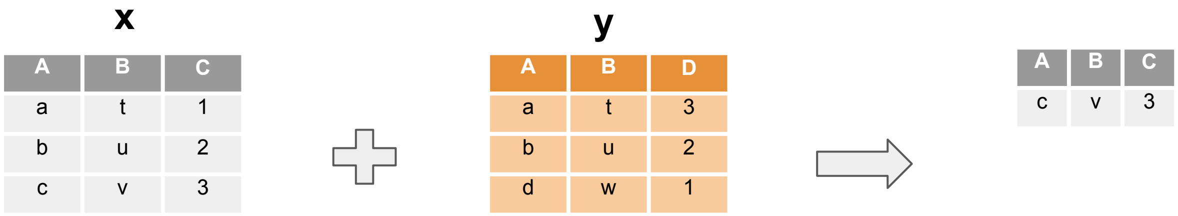
```
> semi_join(data_1, data_2, by = c("patient_ID" = "patient_id"))
```

```
# A tibble: 4 x 3
  patient_ID   sex         age_year
  <chr>        <chr>       <dbl>
1 P001         female      1
2 P002         female      4
3 P003         female      2
4 P004         male        3
```

anti_join( )

- **anti_join( ) - drops** all observations in x that have a match in y
- useful to see observations that are not joined

```
> anti_join(x, y)
```



x

A	B	C
a	t	1
b	u	2
c	v	3

y

A	B	D
a	t	3
b	u	2
d	w	1

A	B	C
c	v	3

anti_join( )

```
> data_1
```

```
# A tibble: 5 x 3
patient_ID     sex          age_year
  <chr>        <chr>        <dbl>
1 P001         female       1
2 P002         female       4
3 P003         female       2
4 P004         male         3
5 P005         male         4
```

```
> data_2
```

```
# A tibble: 5 x 3
  patient_id weight_kg height_cm
  <chr>        <dbl>        <dbl>
1 P001         9.1          73
2 P002         16.4         NA
3 P003         10.5         85
4 P004         13.2         95
5 P006         15.9        104
```

```
> anti_join(data_1, data_2, by = c("patient_ID" = "patient_id"))
```

```
# A tibble: 1 x 3
  patient_ID   sex    age_year
  <chr>        <chr>     <dbl>
1 P005         male          4
```

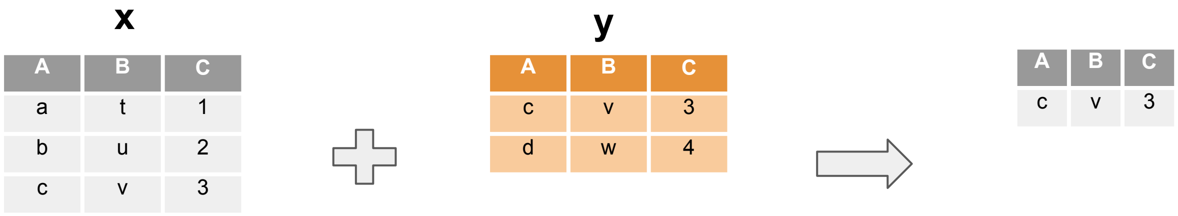
Set operations

- use for combining rows
- these operations work with a complete row, comparing the values of every variable
- expect the x and y inputs to have the same variables, and treat the observations like sets
- **intersect( )** - return only observations in both x and y
- **union( )** - return unique observations in x and y
- **setdiff( )** - return observations in x, but not in y

intersect( )

- returns rows that appear only in both x and y

```
> intersect(x, y)
```



x

A	B	C
a	t	1
b	u	2
c	v	3

y

A	B	C
c	v	3
d	w	4

A	B	C
c	v	3

intersect( )

```
> data_x
```

```
# A tibble: 3 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P002       female        4      16.4        96
3 P003       female        2      10.5        85
```

```
> data_y
```

```
# A tibble: 3 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P003       female        2      10.5        85
2 P004       male          3      13.2        95
3 P005       male          4      15.9       104
```

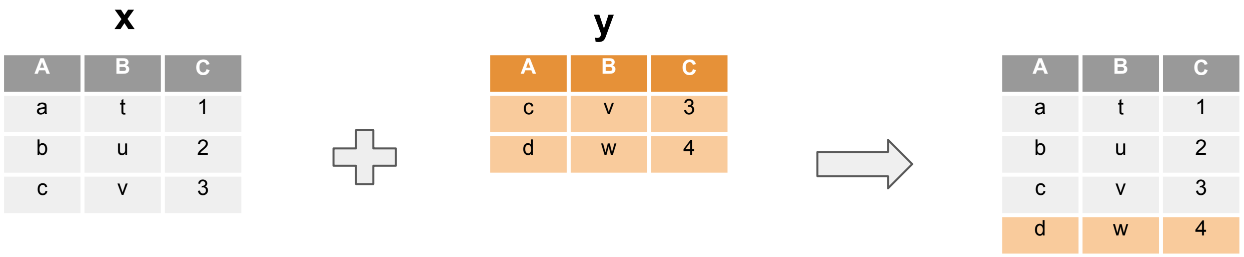
```
> intersect(data_x, data_y)
```

```
# A tibble: 1 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P003       female        2      10.5        85
```

union( )

- returns rows that appear in x or y
- duplicates will be removed

```
> union(x, y)
```



x

A	B	C
a	t	1
b	u	2
c	v	3

y

A	B	C
c	v	3
d	w	4

A	B	C
a	t	1
b	u	2
c	v	3
d	w	4

union( )

```
> data_x
```

```
# A tibble: 3 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P002       female        4      16.4        96
3 P003       female        2      10.5        85
```

```
> data_y
```

```
# A tibble: 3 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P003       female        2      10.5        85
2 P004       male          3      13.2        95
3 P005       male          4      15.9       104
```

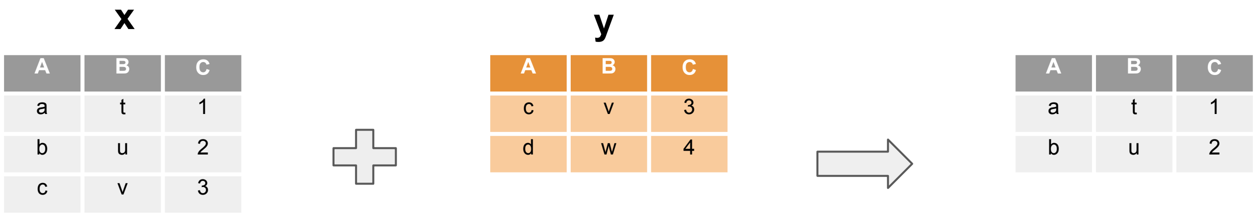
```
> union(data_x, data_y)
```

```
# A tibble: 5 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P002       female        4      16.4        96
3 P003       female        2      10.5        85
4 P004       male          3      13.2        95
5 P005       male          4      15.9       104
```

setdiff( )

- returns rows that appear in x but not in y

```
> setdiff(x, y)
```



x

A	B	C
a	t	1
b	u	2
c	v	3

y

A	B	C
c	v	3
d	w	4

A	B	C
a	t	1
b	u	2

setdiff( )

```
> data_x
```

```
# A tibble: 3 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P002       female        4      16.4        96
3 P003       female        2      10.5        85
```

```
> data_y
```

```
# A tibble: 3 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P003       female        2      10.5        85
2 P004       male          3      13.2        95
3 P005       male          4      15.9       104
```

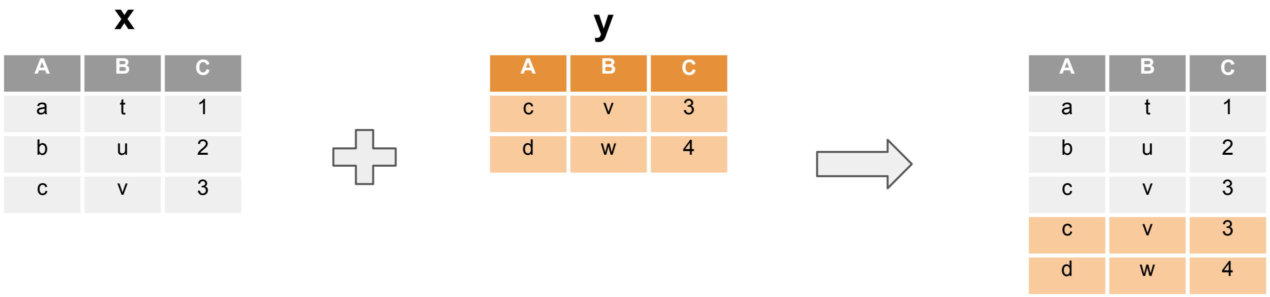
```
> setdiff(data_x, data_y)
```

```
# A tibble: 2 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P002       female        4      16.4        96
```

bind_rows( )

- to stack tables on top of the other as they are

```
> bind_rows(x, y)
```



x

A	B	C
a	t	1
b	u	2
c	v	3

y

A	B	C
c	v	3
d	w	4

A	B	C
a	t	1
b	u	2
c	v	3
c	v	3
d	w	4

bind_rows( )

```
> data_x
```

```
# A tibble: 3 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P002       female        4      16.4        96
3 P003       female        2      10.5        85
```

```
> data_y
```

```
# A tibble: 3 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P003       female        2      10.5        85
2 P004       male          3      13.2        95
3 P005       male          4      15.9       104
```

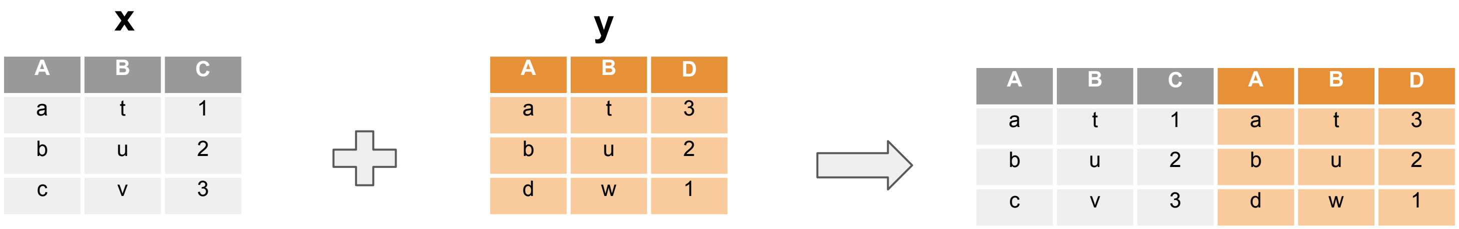
```
> bind_rows(data_x, data_y)
```

```
# A tibble: 6 x 5
  patient_ID sex    age_year weight_kg height_cm
  <chr>      <chr>     <dbl>     <dbl>     <dbl>
1 P001       female        1       9.1        73
2 P002       female        4      16.4        96
3 P003       female        2      10.5        85
4 P003       female        2      10.5        85
5 P004       male          3      13.2        95
6 P005       male          4      15.9       104
```

bind_cols( )

- to paste tables beside each other as they are

```
> bind_cols(x, y)
```



x

A	B	C
a	t	1
b	u	2
c	v	3

y

A	B	D
a	t	3
b	u	2
d	w	1

A	B	C	A	B	D
a	t	1	a	t	3
b	u	2	b	u	2
c	v	3	d	w	1

bind_cols( )

```
> data_1
```

```
# A tibble: 5 x 3
patient_ID     sex           age_year
  <chr>        <chr>         <dbl>
1 P001         female        1
2 P002         female        4
3 P003         female        2
4 P004         male          3
5 P005         male          4
```

```
> data_2
```

```
# A tibble: 5 x 3
  patient_id weight_kg height_cm
  <chr>          <dbl>          <dbl>
1 P001           9.1             73
2 P002           16.4            NA
3 P003           10.5            85
4 P004           13.2            95
5 P006           15.9           104
```

```
> bind_cols(data_1, data_2)
```

```
# A tibble: 5 x 6
  patient_ID sex       age_year patient_id weight_kg     height_cm
  <chr>            <chr>  <dbl>             <chr>         <dbl>         <dbl>
1 P001           female          1         P001          9.1             73
2 P002           female          4         P002          16.4            NA
3 P003           female          2         P003          10.5            85
4 P004           male            3         P004          13.2            95
5 P005           male            4         P006          15.9           104
```

Take-away message

- Knowing how to merge tables is useful since data are not always available in a single table