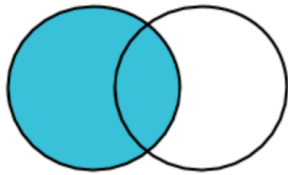


Merging and Joining Data Sets

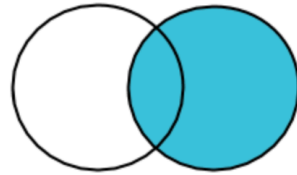
Data Wrangling in R

Joining

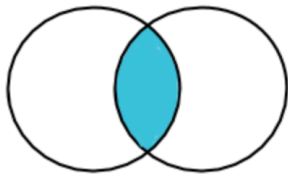
“Combining datasets”



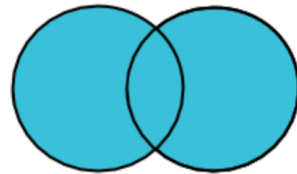
Left Join



Right Join



Inner Join



**Full Outer
Join**

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

data_As

```
# A tibble: 2 × 3
  State    June_vacc_rate May_vacc_rate
  <chr>         <dbl>         <dbl>
1 Alabama      0.516          0.514
2 Alaska       0.627          0.626
```

data_cold

```
# A tibble: 3 × 2
  State    April_vacc_rate
  <chr>         <dbl>
1 Maine      0.795
2 Alaska     0.623
3 Vermont    0.82
```

Inner Join

<https://github.com/gadenbuie/tidyexplain/blob/main/images/inner-join.gif>

`inner_join(x, y)`

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4

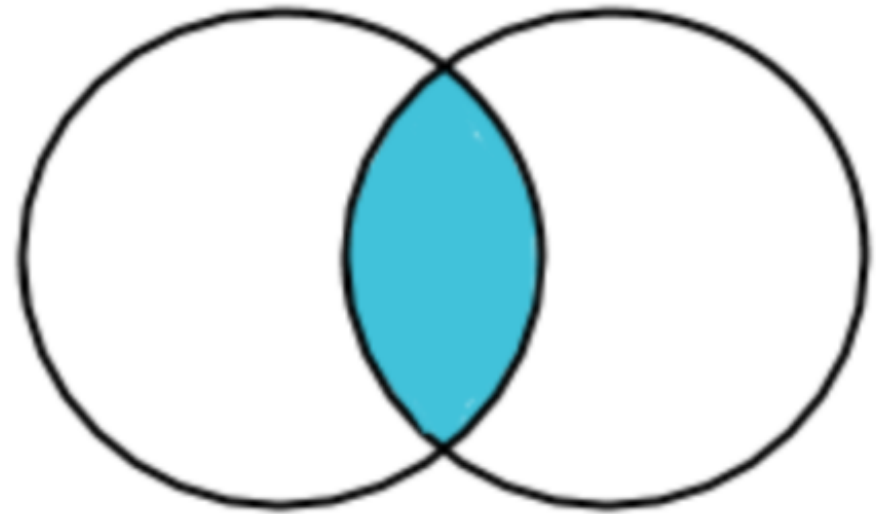
Inner Join

data_As

```
# A tibble: 2 × 3
  State      June_vacc_rate May_vacc_rate
  <chr>          <dbl>         <dbl>
1 Alabama      0.516          0.514
2 Alaska       0.627          0.626
```

data_cold

```
# A tibble: 3 × 2
  State      April_vacc_rate
  <chr>          <dbl>
1 Maine      0.795
2 Alaska     0.623
3 Vermont    0.82
```



Inner Join

Inner Join

```
lj <- inner_join(data_As, data_cold)
```

Joining with `by = join\_by(State)`

```
lj
```

```
# A tibble: 1 × 4  
  State June_vacc_rate May_vacc_rate April_vacc_rate  
  <chr>      <dbl>      <dbl>      <dbl>  
1 Alaska      0.627      0.626      0.623
```

Left Join

<https://raw.githubusercontent.com/gadenbuie/tidyexplain/main/images/left-join.gif>

`left_join(x, y)`

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4

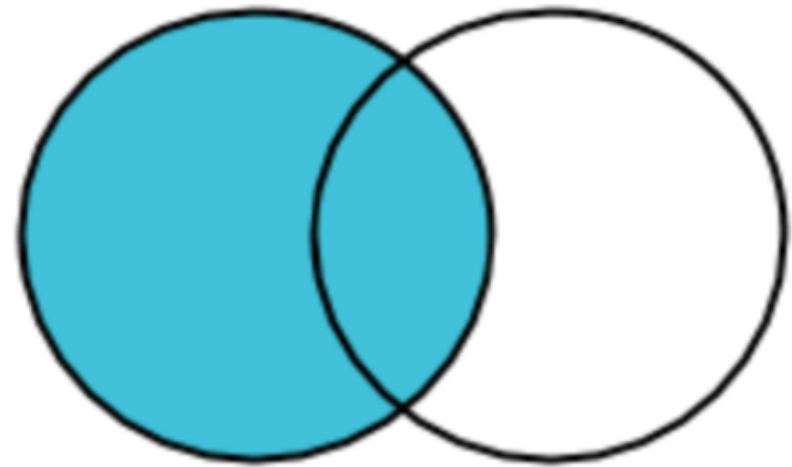
Left Join

data_As

```
# A tibble: 2 × 3
  State      June_vacc_rate May_vacc_rate
  <chr>          <dbl>         <dbl>
1 Alabama      0.516          0.514
2 Alaska       0.627          0.626
```

data_cold

```
# A tibble: 3 × 2
  State      April_vacc_rate
  <chr>          <dbl>
1 Maine      0.795
2 Alaska     0.623
3 Vermont    0.82
```



Left Join

Left Join

```
lj <- left_join(data_As, data_cold)
```

Joining with `by = join\_by(State)`

```
lj
```

```
# A tibble: 2 × 4  
  State    June_vacc_rate May_vacc_rate April_vacc_rate  
  <chr>         <dbl>         <dbl>         <dbl>  
1 Alabama      0.516          0.514          NA  
2 Alaska       0.627          0.626          0.623
```

Install **tidylog** package to log outputs

Numbers in parentheses indicate that these rows are not included in the result.

```
# install.packages("tidylog")
library(tidylog)
left_join(data_As, data_cold)
```

```
Joining with `by = join_by(State)`
left_join: added one column (April_vacc_rate)
> rows only in data_As 1
> rows only in data_cold (2)
> matched rows 1
> ===
> rows total 2
```

```
# A tibble: 2 × 4
  State      June_vacc_rate May_vacc_rate April_vacc_rate
  <chr>          <dbl>         <dbl>         <dbl>
1 Alabama      0.516          0.514          NA
2 Alaska       0.627          0.626          0.623
```

Right Join

<https://raw.githubusercontent.com/gadenbuie/tidyexplain/main/images/right-join.gif>

`right_join(x, y)`

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4

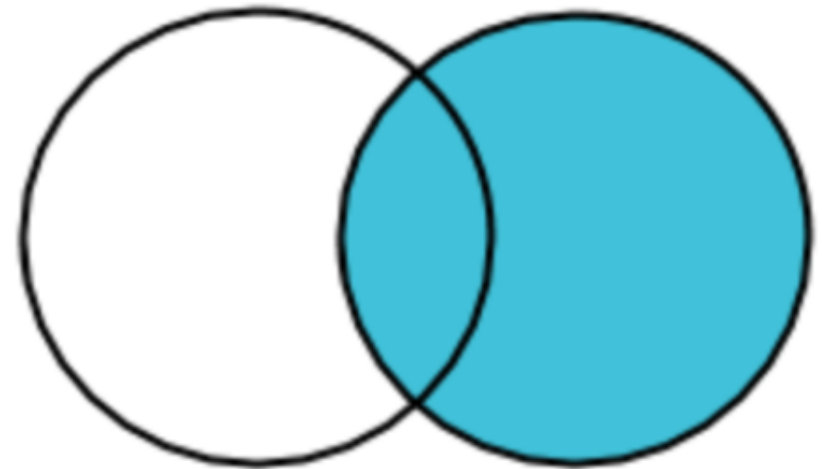
Right Join

data_As

```
# A tibble: 2 × 3
  State    June_vacc_rate May_vacc_rate
  <chr>         <dbl>         <dbl>
1 Alabama      0.516          0.514
2 Alaska       0.627          0.626
```

data_cold

```
# A tibble: 3 × 2
  State    April_vacc_rate
  <chr>         <dbl>
1 Maine      0.795
2 Alaska     0.623
3 Vermont    0.82
```



Right Join

Right Join

```
rj <- right_join(data_As, data_cold)
```

```
Joining with `by = join_by(State)`  
right_join: added one column (April_vacc_rate)  
> rows only in data_As (1)  
> rows only in data_cold 2  
> matched rows 1  
> ===  
> rows total 3
```

```
rj
```

```
# A tibble: 3 × 4  
  State    June_vacc_rate May_vacc_rate April_vacc_rate  
  <chr>         <dbl>         <dbl>         <dbl>  
1 Alaska      0.627          0.626          0.623  
2 Maine       NA             NA             0.795  
3 Vermont     NA             NA             0.82
```

Left Join: Switching arguments

```
lj2 <- left_join(data_cold, data_As)
```

Joining with `by = join\_by(State)`

left_join: added 2 columns (June_vacc_rate, May_vacc_rate)

> rows only in data_cold 2

> rows only in data_As (1)

> matched rows 1

> ===

> rows total 3

```
lj2
```

A tibble: 3 × 4

	State <chr>	April_vacc_rate <dbl>	June_vacc_rate <dbl>	May_vacc_rate <dbl>
1	Maine	0.795	NA	NA
2	Alaska	0.623	0.627	0.626
3	Vermont	0.82	NA	NA

Full Join

<https://raw.githubusercontent.com/gadenbuie/tidyexplain/main/images/full-join.gif>

`full_join(x, y)`

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4

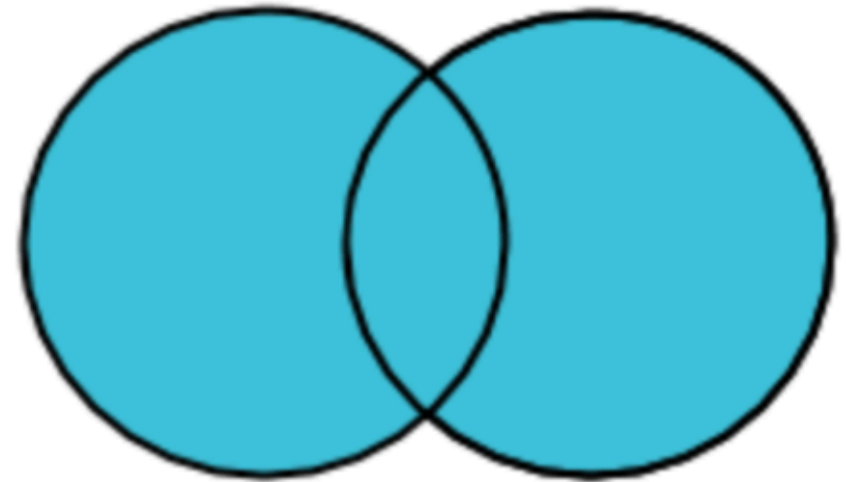
Full Join

data_As

```
# A tibble: 2 × 3
  State      June_vacc_rate May_vacc_rate
  <chr>          <dbl>         <dbl>
1 Alabama      0.516          0.514
2 Alaska       0.627          0.626
```

data_cold

```
# A tibble: 3 × 2
  State      April_vacc_rate
  <chr>          <dbl>
1 Maine      0.795
2 Alaska     0.623
3 Vermont    0.82
```



Full Outer Join

Full Join

```
fj <- full_join(data_As, data_cold)
```

```
Joining with `by = join_by(State)`  
full_join: added one column (April_vacc_rate)  
> rows only in data_As 1  
> rows only in data_cold 2  
> matched rows 1  
> ===  
> rows total 4
```

```
fj
```

```
# A tibble: 4 × 4  
  State    June_vacc_rate May_vacc_rate April_vacc_rate  
  <chr>         <dbl>         <dbl>         <dbl>  
1 Alabama      0.516          0.514          NA  
2 Alaska       0.627          0.626          0.623  
3 Maine        NA            NA            0.795  
4 Vermont      NA            NA            0.82
```

Watch out for “includes duplicates”

<https://github.com/gadenbuie/tidyexplain/blob/main/images/left-join-extra.gif>

`left_join(x, y)`

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4
		2	y5

Stop tidylog

```
unloadNamespace("tidylog")
```

GUT CHECK!

Why use `join` functions?

A. Combine different data sources

B. Connect Rmd to other files

C. Using one data source is too easy and we want our analysis ~ fancy ~

Using the **by** argument

By default joins use the intersection of column names. If **by** is specified, it uses that.

```
full_join(data_As, data_cold, by = "State")
```

```
# A tibble: 4 × 4
  State      June_vacc_rate May_vacc_rate April_vacc_rate
  <chr>      <dbl>         <dbl>         <dbl>
1 Alabama    0.516          0.514          NA
2 Alaska     0.627          0.626          0.623
3 Maine      NA             NA             0.795
4 Vermont    NA             NA             0.82
```

Using the **by** argument

You can join based on multiple columns by using something like `by = c(col1, col2)`.

If the datasets have two different names for the same data, use:

```
full_join(x, y, by = c("a" = "b"))
```

Using “`setdiff`” (base)

We might want to determine what indexes ARE in the first dataset that AREN'T in the second:

```
data_As
```

```
# A tibble: 2 × 3
  State      June_vacc_rate May_vacc_rate
  <chr>          <dbl>         <dbl>
1 Alabama      0.516          0.514
2 Alaska       0.627          0.626
```

```
data_cold
```

```
# A tibble: 3 × 2
  State      April_vacc_rate
  <chr>          <dbl>
1 Maine      0.795
2 Alaska     0.623
3 Vermont    0.82
```


Using “**setdiff**” (base)

Use `setdiff` to determine what indexes ARE in the first dataset that AREN'T in the second:

```
A_states <- data_As %>% pull(State)
cold_states <- data_cold %>% pull(State)
```

```
setdiff(A_states, cold_states)
```

```
[1] "Alabama"
```

```
setdiff(cold_states, A_states)
```

```
[1] "Maine"    "Vermont"
```

Using `bind_rows()` (`dplyr`)

Rows are stacked on top of each other. Works like `rbind()` from base R, but is “smarter” and looks for matching column names.

```
rbind(data_As, data_cold)
```

```
Error in rbind(deparse.level, ...): numbers of columns of arguments do not match
```

```
bind_rows(data_As, data_cold)
```

```
# A tibble: 5 × 4
```

	State	June_vacc_rate	May_vacc_rate	April_vacc_rate
	<chr>	<dbl>	<dbl>	<dbl>
1	Alabama	0.516	0.514	NA
2	Alaska	0.627	0.626	NA
3	Maine	NA	NA	0.795
4	Alaska	NA	NA	0.623
5	Vermont	NA	NA	0.82

bind_rows vs full join

```
full_join(data_As, data_cold)
```

Joining with `by = join\_by(State)`

```
# A tibble: 4 × 4
```

	State <chr>	June_vacc_rate <dbl>	May_vacc_rate <dbl>	April_vacc_rate <dbl>
1	Alabama	0.516	0.514	NA
2	Alaska	0.627	0.626	0.623
3	Maine	NA	NA	0.795
4	Vermont	NA	NA	0.82

Other stuff: `anti_join` (dplyr)

data_As

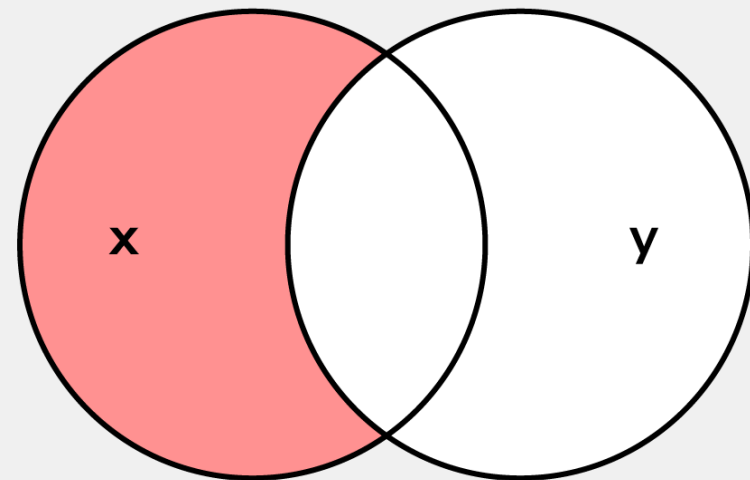
```
# A tibble: 2 × 3
  State      June_vacc_rate May_vacc_rate
<chr>      <dbl>         <dbl>
1 Alabama    0.516           0.514
2 Alaska     0.627           0.626
```

data_cold

```
# A tibble: 3 × 2
  State      April_vacc_rate
<chr>      <dbl>
1 Maine    0.795
2 Alaska   0.623
3 Vermont  0.82
```

[source](#)

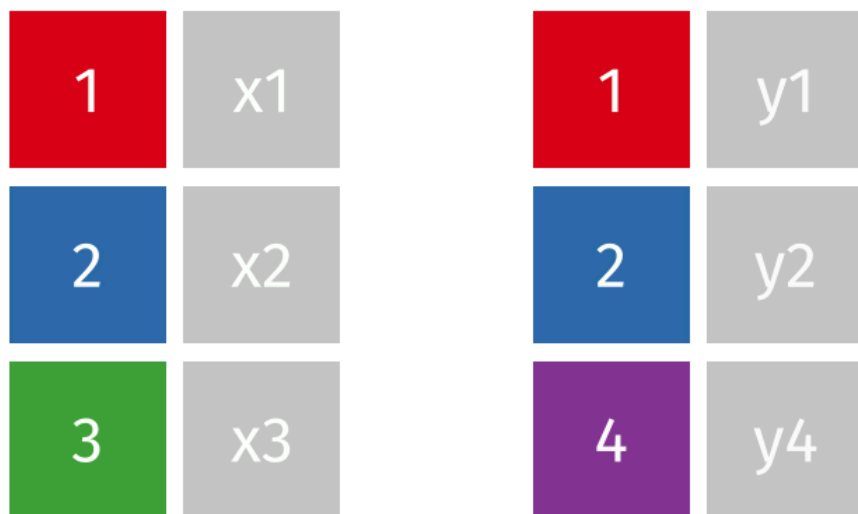
Anti Join in R



anti_join()

<https://raw.githubusercontent.com/gadenbuie/tidyexplain/main/images/anti-join.gif>

`anti_join(x, y)`



Other stuff: `anti_join(dplyr)`

```
anti_join(data_As, data_cold)
```

Joining with `by = join\_by(State)`

```
# A tibble: 1 × 3  
  State    June_vacc_rate May_vacc_rate  
  <chr>         <dbl>         <dbl>  
1 Alabama      0.516          0.514
```

Other stuff: `anti_join(dplyr)`

```
anti_join(data_cold, data_As)
```

Joining with ``by = join_by(State)``

```
# A tibble: 2 × 2
  State    April_vacc_rate
  <chr>         <dbl>
1 Maine         0.795
2 Vermont        0.82
```

Summary

- Merging/joining data sets together - assumes all column names that overlap
 - use the `by = c("a" = "b")` if they differ
- `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`
- Use the `tidylog` package for a detailed summary
- `setdiff(x, y)` shows what in `x` is missing from `y`
- `bind_rows(x, y)` appends datasets

https://sisbid.github.io/Data-Wrangling/12_Data_Merging/lab/merging-lab.Rmd

Extra slides

Learn more

There are lots of options for more complicated merges. Check out the documentation if you have a special use case:

<https://dplyr.tidyverse.org/reference/mutate-joins.html>

Other stuff: `cross_join(dplyr)`

Cross joins match each row in `x` to every row in `y`, resulting in a data frame with `nrow(x) * nrow(y)` rows.

```
cross_join(data_As, data_cold)
```

```
# A tibble: 6 × 5
  State.x June_vacc_rate May_vacc_rate State.y April_vacc_rate
  <chr>      <dbl>         <dbl> <chr>      <dbl>
1 Alabama  0.516           0.514 Maine      0.795
2 Alabama  0.516           0.514 Alaska    0.623
3 Alabama  0.516           0.514 Vermont    0.82
4 Alaska   0.627           0.626 Maine      0.795
5 Alaska   0.627           0.626 Alaska    0.623
6 Alaska   0.627           0.626 Vermont    0.82
```

Other stuff: `nest_join` (`dplyr`)

A nest join leaves `x` almost unchanged, except that it adds a new column for the `y` dataset. Matched values are stored inside the “cell” as a tibble.

```
nj <- nest_join(data_As, data_cold)
```

Joining with ``by = join_by(State)``

```
nj
```

```
# A tibble: 2 × 4  
  State    June_vacc_rate May_vacc_rate data_cold  
  <chr>          <dbl>         <dbl> <list>  
1 Alabama      0.516         0.514 <tibble [0 × 1]>  
2 Alaska      0.627         0.626 <tibble [1 × 1]>
```

Other stuff: `nest_join` (dplyr)

```
nj %>% pull(data_cold)
```

```
[[1]]  
# A tibble: 0 × 1  
# i 1 variable: April_vacc_rate <dbl>
```

```
[[2]]  
# A tibble: 1 × 1  
  April_vacc_rate  
    <dbl>  
1           0.623
```

“Includes duplicates” with both datasets duplicated:

```
data_As <- tibble(State = c("Alabama", "Alaska", "Alaska"),  
                  state_bird = c("wild turkey", "willow ptarmigan", "puffin"))  
data_cold <- tibble(State = c("Maine", "Alaska", "Alaska"),  
                    vacc_rate = c("32.4%", "41.7%", "46.2%"),  
                    month = c("April", "April", "May"))
```

“Includes duplicates” with both datasets duplicated:

```
full_join(data_As, data_cold)
```

```
Joining with `by = join_by(State)`
```

```
Warning in full_join(data_As, data_cold): Detected an unexpected many-to-many relationship between `x` and `y`.
```

```
i Row 2 of `x` matches multiple rows in `y`.
```

```
i Row 2 of `y` matches multiple rows in `x`.
```

```
i If a many-to-many relationship is expected, set `relationship = "many-to-many"` to silence this warning.
```

```
# A tibble: 6 × 4
```

	State	state_bird	vacc_rate	month
	<chr>	<chr>	<chr>	<chr>
1	Alabama	wild turkey	<NA>	<NA>
2	Alaska	willow ptarmigan	41.7%	April
3	Alaska	willow ptarmigan	46.2%	May
4	Alaska	puffin	41.7%	April
5	Alaska	puffin	46.2%	May
6	Maine	<NA>	32.4%	April