

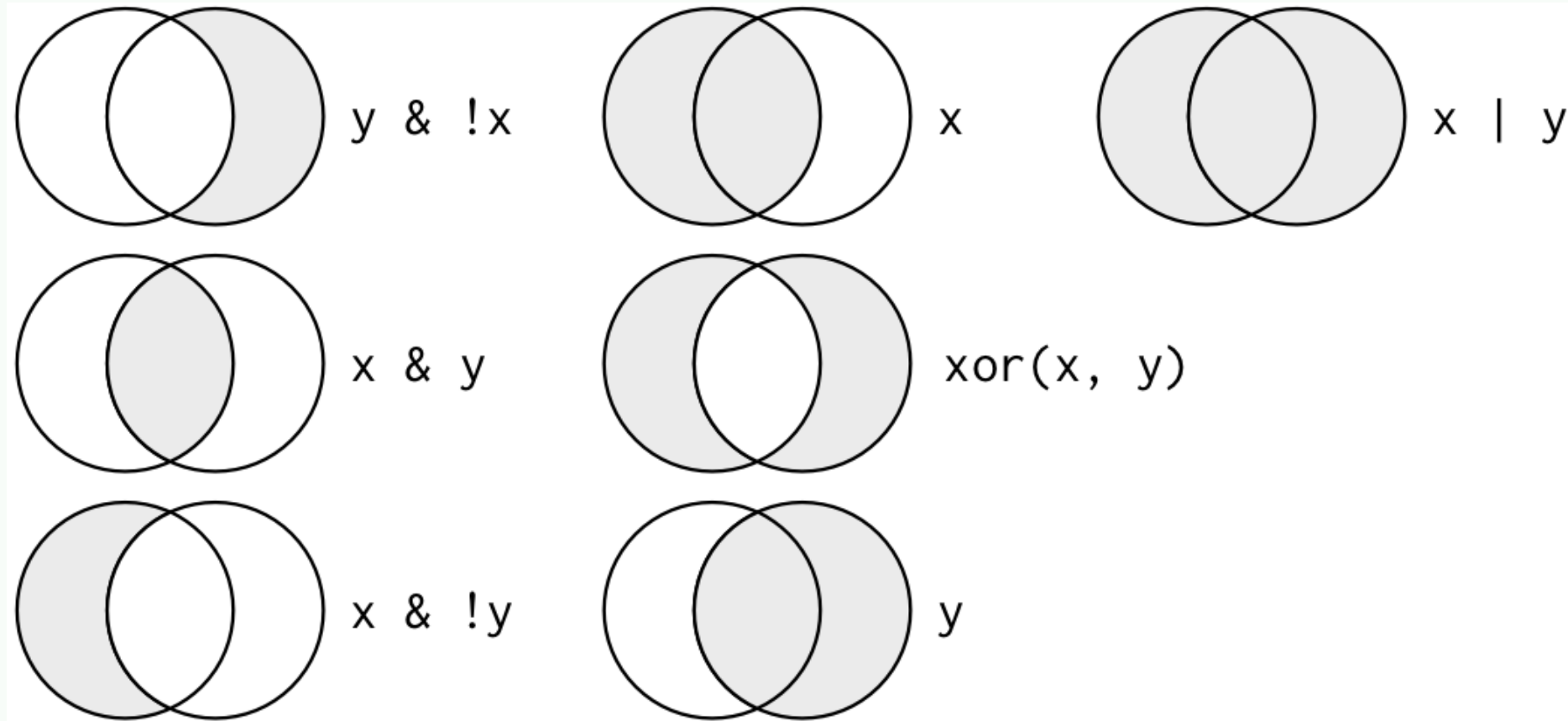
# More Data Wrangling

Spring 2023

April 09 2023

# Boolean operators

For help, [?base::Logic](#)



## rename()

```
polluted_cities <- tribble(
  ~city, ~size, ~amount,
  "New York", "large", 23.4,
  "London", "large", 22.2,
  "Doha", "small", 12.4,
  "Hongkong", "small", 15.6,
  "Beijing", "large", 121.0,
)
```

```
polluted_cities <- tibble(
  city = c("New York", "London", "Doha",
           "Hongkong", "Beijing"),
  size = c("large", "large", "small",
           "small", "large"),
  amount = c(23.4, 22.2, 12.4,
             15.6, 121.0)
)
```

```
polluted_cities %>% rename(quantity = amount)
# A tibble: 5 × 3
```

	city	size	quantity
	<chr>	<chr>	<dbl>
1	New York	large	23.4
2	London	large	22.2
3	Doha	small	12.4
4	Hongkong	small	15.6
5	Beijing	large	121

```
polluted_cities %>% rename(pollution = amount)
# A tibble: 5 × 3
```

	city	size	pollution
	<chr>	<chr>	<dbl>
1	New York	large	23.4
2	London	large	22.2
3	Doha	small	12.4
4	Hongkong	small	15.6
5	Beijing	large	121

## Mutating multiple columns at once: `mutate_*`

- variants of `mutate()` that are useful for mutating multiple columns at once
  - `mutate_at()`, `mutate_if()`, `mutate_all()`, etc.
- which columns get mutated depends on a predicate, can be:
  - a function that returns TRUE/FALSE like `is.numeric()`
  - variable names through `vars()`

```
polluted_cities %>%  
  mutate_at(vars(city:amount), toupper)  
# A tibble: 5 × 3  
  city      size amount  
  <chr>    <chr> <chr>  
1 NEW YORK LARGE 23.4  
2 LONDON   LARGE 22.2  
3 DOHA     SMALL 12.4  
4 HONGKONG SMALL 15.6  
5 BEIJING  LARGE 121
```

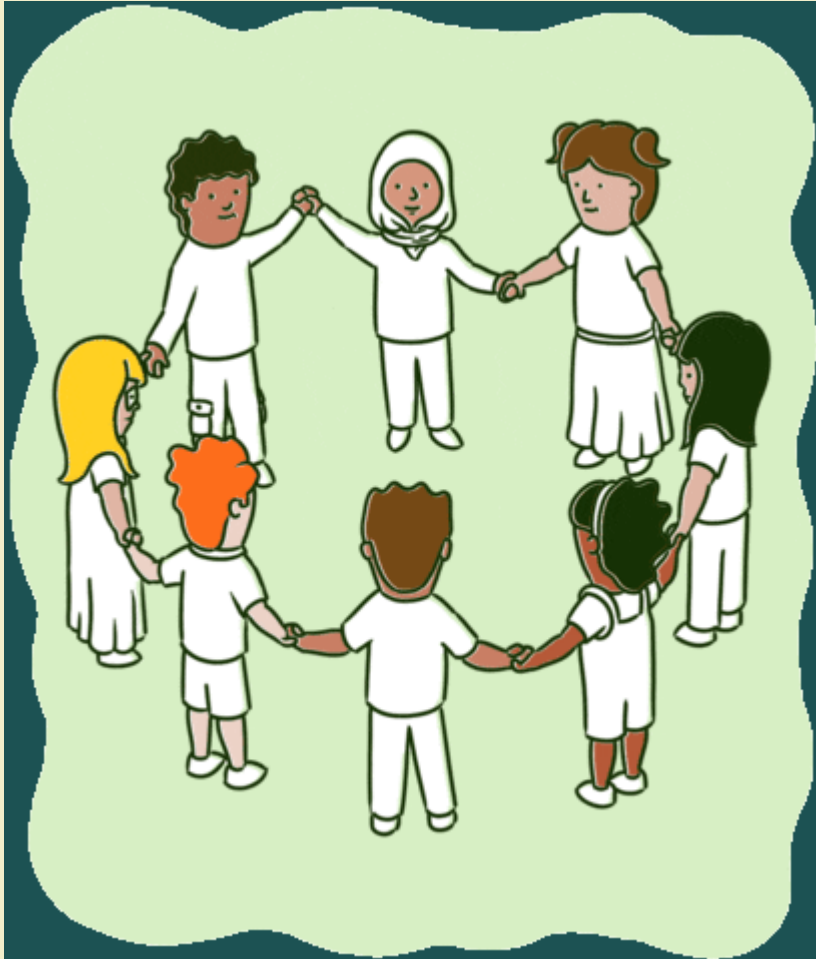
```
polluted_cities %>%  
  mutate_if(is.double, round, digits = 0)  
# A tibble: 5 × 3  
  city      size amount  
  <chr>    <chr> <dbl>  
1 New York large    23  
2 London   large    22  
3 Doha     small    12  
4 Hongkong small    16  
5 Beijing  large   121
```

## Selecting & renaming multiple columns

- `select_*`() & `rename_*`() are variants of `select()` and `rename()`
- use like `mutate_*`() options on previous slide

# ✍ GROUP ACTIVITY 1

10:00

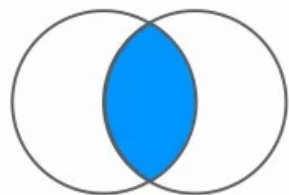


- *Let's go over to [moodle](#)*
- *Get the class activity 7.Rmd file*
- *Work on problem 1*

## Mutating

## Filtering

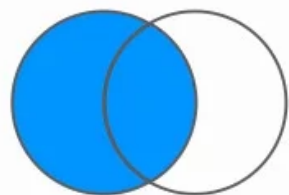
## Set



`inner_join()`

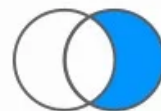
`semi_join()`

`intersect()`

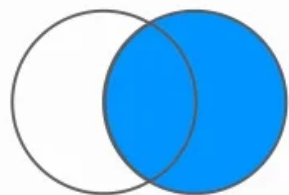
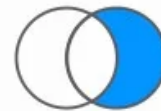


`left_join()`

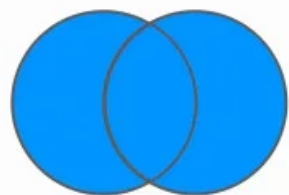
`anti_join()`



`setdiff()`



`right_join()`



`full_join()`

`union()`

## Two-table verbs

- *`inner_join()` - Merge two datasets. Exclude all unmatched rows.*
- *`full_join()` - Merge two datasets. Keep all observations.*
- *`left_join()` - Merge two datasets. Keep all observations from the origin table.*
- *`right_join()` - Merge two datasets. Keep all observations from the destination table.*
- *`anti_join()` - Drops all observations in origin that have a match in destination table.*



## Mutating Joins

Differ in their behavior when a match is not found

- *left\_join()*
- *right\_join()*
- *inner\_join()*
- *full\_join()*

# left\_join()

```
df1 <- tibble(x = c(1, 2), y = 2:1)  
df2 <- tibble(x = c(3, 1), a = 10, b = "a")
```

Table: df1

x	y
1	2
2	1

Table: df2

x	a	b
3	10	a
1	10	a

```
df1 %>% left_join(df2)
```

x	y	a	b
1	2	10	a
2	1	NA	NA

```
df2 %>% left_join(df1)
```

x	a	b	y
3	10	a	NA
1	10	a	2

## *right\_join()*

```
df1 <- tibble(x = c(1, 2), y = 2:1)  
df2 <- tibble(x = c(3, 1), a = 10, b = "a")
```

Table: df1

x	y
1	2
2	1

Table: df2

x	a	b
3	10	a
1	10	a

```
df1 %>% right_join(df2)
```

x	y	a	b
1	2	10	a
3	NA	10	a

```
df2 %>% right_join(df1)
```

x	a	b	y
1	10	a	2
2	NA	NA	1

## *inner\_join()*

```
df1 <- tibble(x = c(1, 2), y = 2:1)  
df2 <- tibble(x = c(3, 1), a = 10, b = "a")
```

Table: df1

x	y
1	2
2	1

Table: df2

x	a	b
3	10	a
1	10	a

```
df1 %>% inner_join(df2)
```

x	y	a	b
1	2	10	a

## Flights data

```
library(nycflights13)
flights2 <- flights %>%
  select(year:day, hour, origin, dest, tailnum, carrier)
```

```
head(flights2)
# A tibble: 6 × 8
   year month   day   hour origin dest tailnum carrier
<int> <int> <int> <dbl> <chr>  <chr> <chr>  <chr>
1  2013     1     1     5 EWR    IAH  N14228  UA
2  2013     1     1     5 LGA    IAH  N24211  UA
3  2013     1     1     5 JFK    MIA  N619AA  AA
4  2013     1     1     5 JFK    BQN  N804JB  B6
5  2013     1     1     6 LGA    ATL  N668DN  DL
6  2013     1     1     5 EWR    ORD  N39463  UA
```

## Airline information

```
head(airlines)
# A tibble: 6 × 2
  carrier name
  <chr>    <chr>
1 9E      Endeavor Air Inc.
2 AA      American Airlines Inc.
3 AS      Alaska Airlines Inc.
4 B6      JetBlue Airways
5 DL      Delta Air Lines Inc.
6 EV      ExpressJet Airlines Inc.
```

## left\_join()

```
flights2 %>%  
  left_join(airlines)  
# A tibble: 336,776 × 9  
   year month   day hour origin dest tailnum carrier name  
   <int> <int> <int> <dbl> <chr> <chr> <chr> <chr> <chr>  
1  2013     1     1     5 EWR   IAH  N14228  UA    United Air Lines Inc.  
2  2013     1     1     5 LGA   IAH  N24211  UA    United Air Lines Inc.  
3  2013     1     1     5 JFK   MIA  N619AA  AA    American Airlines Inc.  
4  2013     1     1     5 JFK   BQN  N804JB  B6    JetBlue Airways  
5  2013     1     1     6 LGA   ATL  N668DN  DL    Delta Air Lines Inc.  
6  2013     1     1     5 EWR   ORD  N39463  UA    United Air Lines Inc.  
7  2013     1     1     6 EWR   FLL  N516JB  B6    JetBlue Airways  
8  2013     1     1     6 LGA   IAD  N829AS  EV    ExpressJet Airlines Inc.  
9  2013     1     1     6 JFK   MCO  N593JB  B6    JetBlue Airways  
10 2013     1     1     6 LGA   ORD  N3ALAA  AA    American Airlines Inc.  
# ... with 336,766 more rows
```

# Planes information

```
head(planes)
# A tibble: 6 × 9
  tailnum year type      manuf...1 model engines seats speed engine
  <chr>   <int> <chr>      <chr>      <chr>   <int> <int> <int> <chr>
1 N10156  2004 Fixed wing multi engine EMBRAER EMB-...     2    55    NA Turbo...
2 N102UW  1998 Fixed wing multi engine AIRBUS... A320...     2   182    NA Turbo...
3 N103US  1999 Fixed wing multi engine AIRBUS... A320...     2   182    NA Turbo...
4 N104UW  1999 Fixed wing multi engine AIRBUS... A320...     2   182    NA Turbo...
5 N10575  2002 Fixed wing multi engine EMBRAER EMB-...     2    55    NA Turbo...
6 N105UW  1999 Fixed wing multi engine AIRBUS... A320...     2   182    NA Turbo...
# ... with abbreviated variable name 1manufacturer
```



# Keys: controlling how the tables are matched

```
flights2 %>% left_join(planes, by = "tailnum")
# A tibble: 336,776 × 16
```

	year.x	month	day	hour	origin	dest	tailnum	carrier	year.y	type	manuf... <sup>1</sup>
	<int>	<int>	<int>	<dbl>	<chr>	<chr>	<chr>	<chr>	<int>	<chr>	<chr>
1	2013	1	1	5	EWR	IAH	N14228	UA	1999	Fixed w...	BOEING
2	2013	1	1	5	LGA	IAH	N24211	UA	1998	Fixed w...	BOEING
3	2013	1	1	5	JFK	MIA	N619AA	AA	1990	Fixed w...	BOEING
4	2013	1	1	5	JFK	BQN	N804JB	B6	2012	Fixed w...	AIRBUS
5	2013	1	1	6	LGA	ATL	N668DN	DL	1991	Fixed w...	BOEING
6	2013	1	1	5	EWR	ORD	N39463	UA	2012	Fixed w...	BOEING
7	2013	1	1	6	EWR	FLL	N516JB	B6	2000	Fixed w...	AIRBUS...
8	2013	1	1	6	LGA	IAD	N829AS	EV	1998	Fixed w...	CANADA...
9	2013	1	1	6	JFK	MCO	N593JB	B6	2004	Fixed w...	AIRBUS
10	2013	1	1	6	LGA	ORD	N3ALAA	AA	NA	<NA>	<NA>

```
# ... with 336,766 more rows, 5 more variables: model <chr>, engines <int>,
# seats <int>, speed <int>, engine <chr>, and abbreviated variable name
#   1manufacturer
```

# Matching keys

```
flights2 %>% left_join(airports, by = c("origin" = "faa"))
# A tibble: 336,776 × 15
  year month   day hour origin dest tailnum carrier name   lat lon alt
  <int> <int> <int> <dbl> <chr> <chr> <chr> <chr> <chr> <dbl> <dbl> <dbl>
1  2013     1     1     5 EWR   IAH  N14228  UA   Newar... 40.7 -74.2  18
2  2013     1     1     5 LGA   IAH  N24211  UA   La Gu... 40.8 -73.9  22
3  2013     1     1     5 JFK   MIA  N619AA  AA   John ... 40.6 -73.8  13
4  2013     1     1     5 JFK   BQN  N804JB  B6   John ... 40.6 -73.8  13
5  2013     1     1     6 LGA   ATL  N668DN  DL   La Gu... 40.8 -73.9  22
6  2013     1     1     5 EWR   ORD  N39463  UA   Newar... 40.7 -74.2  18
7  2013     1     1     6 EWR   FLL  N516JB  B6   Newar... 40.7 -74.2  18
8  2013     1     1     6 LGA   IAD  N829AS  EV   La Gu... 40.8 -73.9  22
9  2013     1     1     6 JFK   MCO  N593JB  B6   John ... 40.6 -73.8  13
10 2013     1     1     6 LGA   ORD  N3ALAA  AA   La Gu... 40.8 -73.9  22
# ... with 336,766 more rows, and 3 more variables: tz <dbl>, dst <chr>,
#   tzone <chr>
```

## Filtering joins

most useful for diagnosing join mismatches

*Filtering joins return a copy of the dataset that has been filtered, not augmented (as with mutating joins)*

- ***semi\_join(x,y)*** : keeps all observations in x that have a match in y.
- ***anti\_join(x,y)*** : drops all observations in x that have a match in y.

## Another example: *semi\_join()*

Table: df1

x	y
1	1
1	2
3	3
4	4

Table: df2

x	z
1	a
1	b
2	a

```
df1 %>% semi_join(df2, by = "x")
```

x	y
1	1
1	2

```
df2 %>% semi_join(df1, by = "x")
```

x	z
1	a
1	b

## Another example: *anti\_join()*

Table: df1

x	y
1	1
1	2
3	3
4	4

Table: df2

x	z
1	a
1	b
2	a

```
df1 %>% anti_join(df2, by = "x")
```

x	y
3	3
4	4

```
df2 %>% anti_join(df1, by = "x")
```

x	z
2	a

# Set Operations

These expect the x and y inputs to have the same variables, and treat the observations like sets:

- **`intersect(x,y)`**
  - *will return only the rows that appear in both datasets*
- **`union(x,y)`**
  - *return every row that appears in one or more of the datasets*
  - *If a row appears multiple times union will only return it once*
- **`setdiff(x,y)`**
  - *will return the rows that appear in the first dataset but not the second*

## Set operations

Table: df1

x	y
1	1
2	1

```
intersect(df1, df2))
```

x	y
1	1

```
setdiff(df1, df2))
```

x	y
2	1

Table: df2

x	y
1	1
2	2

```
union(df1, df2))
```

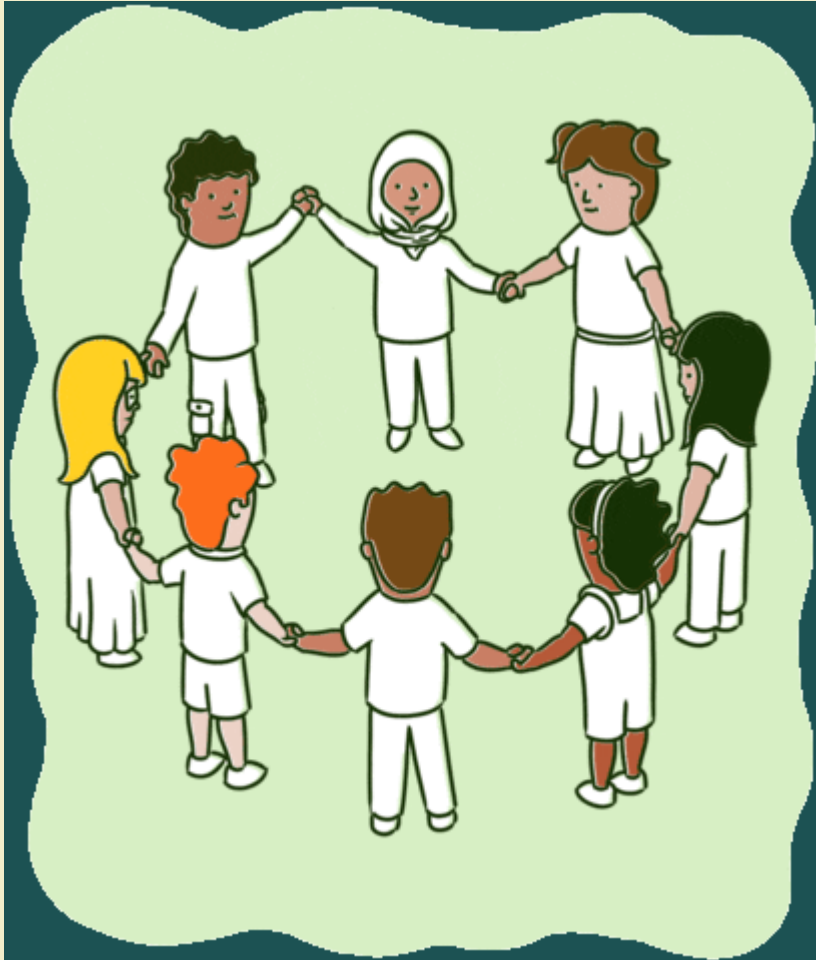
x	y
1	1
2	1
2	2

```
setdiff(df2, df1))
```

x	y
2	2

# ✏ GROUP ACTIVITY 2

10:00



- *Work on problems 2-3*
- *Ask me questions*