

# Transform Data with

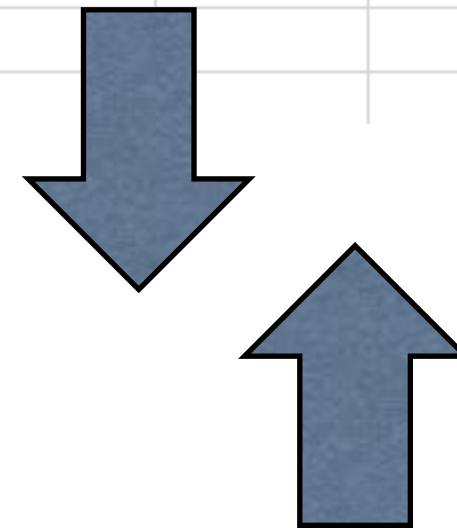


# row/column binding



# rows: `rbind()` or `dplyr::bind_rows()`

StationCode	isSWMP	DateTimeStamp	Historical	Provisiona	CollMethd	REP	F_Record	PO4F	F_PO4F	NH4F	F_NH4F	N
gndbhnut	P	1/13/2016 12:05	0	1	1	1			<-2> [GDM] (CSM)		<-2> [GDM] (CSM)	
gndbhnut	P	1/13/2016 12:06	0	1	1	2			<-2> [GDM] (CSM)		<-2> [GDM] (CSM)	
gndbhnut	P	2/10/2016 8:51	0	1	1	1		0.002	<-4> [SBL]	0.038	<0>	
gndbhnut	P	11/16/2016 12:00	0	1	1	1		0.009	<0>	0.032	<0>	
gndbhnut	P	11/16/2016 12:01	0	1	1	2		0.014	<0>	0.038	<0>	
gndbhnut	P	12/14/2016 8:28	0	1	1	1		0.002	<0>	0.016	<1> [GSM] (CHB)	
gndbhnut	P	12/14/2016 8:29	0	1	1	2		0.002	<0>	0.016	<1> [GSM] (CHB)	



**additional information** as  
**additional rows.**

gndbhnut	P	1/17/2017 11:18	0	1	1	1		0.004	<0>	0.153	<0>	
gndbhnut	P	1/17/2017 11:19	0	1	1	2		0.005	<0>	0.153	<0>	
gndbhnut	P	2/13/2017 10:43	0	1	1	1		0.002	<-4> [SBL]	0.158	<0>	
gndbhnut	P	2/13/2017 10:44	0	1	1	2		0.002	<-4> [SBL]	0.173	<0>	
gndbhnut	P	3/13/2017 10:56	0	1	1	1		0.003	<0>	0.157	<0>	
gndbhnut	P	3/13/2017 10:57	0	1	1	2		0.002	<0>	0.148	<0>	
gndbhnut	P	4/10/2017 16:35	0	1	1	1		0.002	<0>	0.01	<0>	
gndbhnut	P	4/10/2017 16:36	0	1	1	2		0.004	<0>	0.01	<0>	
gndbhnut	P	5/8/2017 17:19	0	1	1	1		0.002	<-4> [SBL]	0.09	<0>	
gndbhnut	P	5/8/2017 17:20	0	1	1	2		0.002	<0>	0.081	<0>	
gndbhnut	P	6/19/2017 16:09	0	1	1	1		0.016	<0>	0.064	<0>	
gndbhnut	P	6/19/2017 16:10	0	1	1	2		0.018	<0>	0.128	<0>	
gndbhnut	P	7/19/2017 16:32	0	1	1	1		0.006	<0>	0.012	<0>	



# columns: `cbind()` or `dplyr::bind_cols()`

	year_sampled	salinity_ppt	water_temp_c
1	2005	8.1	23.3
2	2005	7.9	24.3
3	2005	8.2	24.8
4	2005	8.2	25.3
5	2005	9.0	24.8
6	2005	9.2	28.1
7	2005	8.4	26.2
8	2005	8.9	27.3
9	2005	11.5	27.6
10	2005	10.3	27.8
11	2005	10.4	26.5
12	2005	12.0	28.9
13	2005	10.2	27.6
14	2005	18.5	29.3



	Comp.1	Comp.2	Comp.3	Comp.4
1	-1.469666048	0.50861098	-0.815343241	-1.46335107
2	-1.755692184	0.24370146	-0.836290971	-2.18330031
3	1.080570412	0.35374028	-2.236078623	-1.74776945
4	-1.369822421	0.17215002	-1.135181748	-2.24956333
5	-3.314759863	0.37444935	-0.137357002	-1.45224979
6	-1.986256735	0.49716679	-1.157056717	-1.09732360
7	-3.498663437	0.32469551	-0.265300898	-1.79427409
8	-0.261967574	0.41546947	-1.878794797	-1.38602895
9	-3.517448511	0.00847880	-0.318385366	-1.433961470
10	-2.478236506	0.03892872	-0.867960703	-1.807396915
11	1.766413104	0.23587234	-2.718299079	-1.189025121
12	-1.401074695	0.04024652	-1.431812399	-1.186210314
13	-3.001089532	0.20084810	-0.598864512	-1.447290637
14	-3.459846552	-0.54891989	-0.283326428	-0.239123537

**rows match**  
but  
**no identifying  
information for a join**

- after PCA/nMDS
- after a 'for' loop



# Joining Datasets





# mutating joins



# Motivating Example: Fish data

**additional information**  
as **additional columns**  
based on **matching information**

	A	B	C	D	E
1	site	habitat_type	lat	long	location
2	11	erosional edge	30.37163	-88.4438	Bayou Cumbest
3	14	erosional edge	30.3557	-88.4495	Pt aux Chens Bay
4	2	seagrass	30.38508	-88.4022	Middle Bay
5	3	seagrass	30.36205	-88.3977	Grand Bay
6	6	erosional edge	30.34905	-88.3973	Grand Battures
7	8	seagrass	30.35493	-88.4106	Jose Bay

	A	B	C	D	E	F	G
1	collection_id	site	season	year_sampled	salinity_ppt	do_mgl	water_temp_c
2	NFM08-142	2	Winter	2008	18.4	8.24	14
3	NFM08-143	3	Winter	2008	17.3	7.98	14.5
4	NFM08-146	6	Winter	2008	17.8	8.68	13.9
5	NFM08-148	8	Winter	2008	19.3	8.52	15.2
6	NFM08-151	11	Winter	2008	18.1	7.27	17.1
7	NFM08-154	14	Winter	2008	19.6	9.12	18.3
8	NFM08-156	2	Spring	2008	17.4	6.15	27.4
9	NFM08-157	3	Spring	2008	18.7	5.8	28.2
10	NFM08-160	6	Spring	2008	18.2	6.17	30
11	NFM08-162	8	Spring	2008	18.7	7.16	29.1
12	NFM08-165	11	Spring	2008	12.9	5.92	31.9
13	NFM08-168	14	Spring	2008	16.7	7.72	31.9
14	NFM08-169	2	Summer	2008	18.8	3.42	29.6
15	NFM08-170	3	Summer	2008	19.2	4.12	28.6
16	NFM08-173	6	Summer	2008	19.2	5.78	29.8
17	NFM08-175	8	Summer	2008	20.8	5.07	30.2
18	NFM08-178	11	Summer	2008	10.4	4.29	31.9
19	NFM08-180	14	Summer	2008	20.5	5.81	31.5



# Motivating Example: Fish data

**additional information**  
as **additional columns**  
based on **matching information**

	collection_id	site	habitat_type	lat	long	season	year_sampled	salinity_ppt	do_mgl	water_temp_c
1	NFM08-142	2	seagrass	30.38508	-88.40215	Winter	2008	18.4	8.24	14.00
2	NFM08-143	3	seagrass	30.36205	-88.39772	Winter	2008	17.3	7.98	14.50
3	NFM08-146	6	erosional edge	30.34905	-88.39725	Winter	2008	17.8	8.68	13.90
4	NFM08-148	8	seagrass	30.35493	-88.41063	Winter	2008	19.3	8.52	15.20
5	NFM08-151	11	erosional edge	30.37163	-88.44382	Winter	2008	18.1	7.27	17.10
6	NFM08-154	14	erosional edge	30.35570	-88.44950	Winter	2008	19.6	9.12	18.30
7	NFM08-156	2	seagrass	30.38508	-88.40215	Spring	2008	17.4	6.15	27.40
8	NFM08-157	3	seagrass	30.36205	-88.39772	Spring	2008	18.7	5.80	28.20
9	NFM08-160	6	erosional edge	30.34905	-88.39725	Spring	2008	18.2	6.17	30.00





# common syntax

Each join function returns a data frame / tibble.

```
left_join(x, y, by = NULL, ...)
```

join  
function

data frames  
to join

names of columns  
to join on

# Toy data

```
band <- tribble(
  ~name,    ~band,
  "Mick",    "Stones",
  "John",    "Beatles",
  "Paul",    "Beatles"
)
```

band

name	band
Mick	Stones
John	Beatles
Paul	Beatles

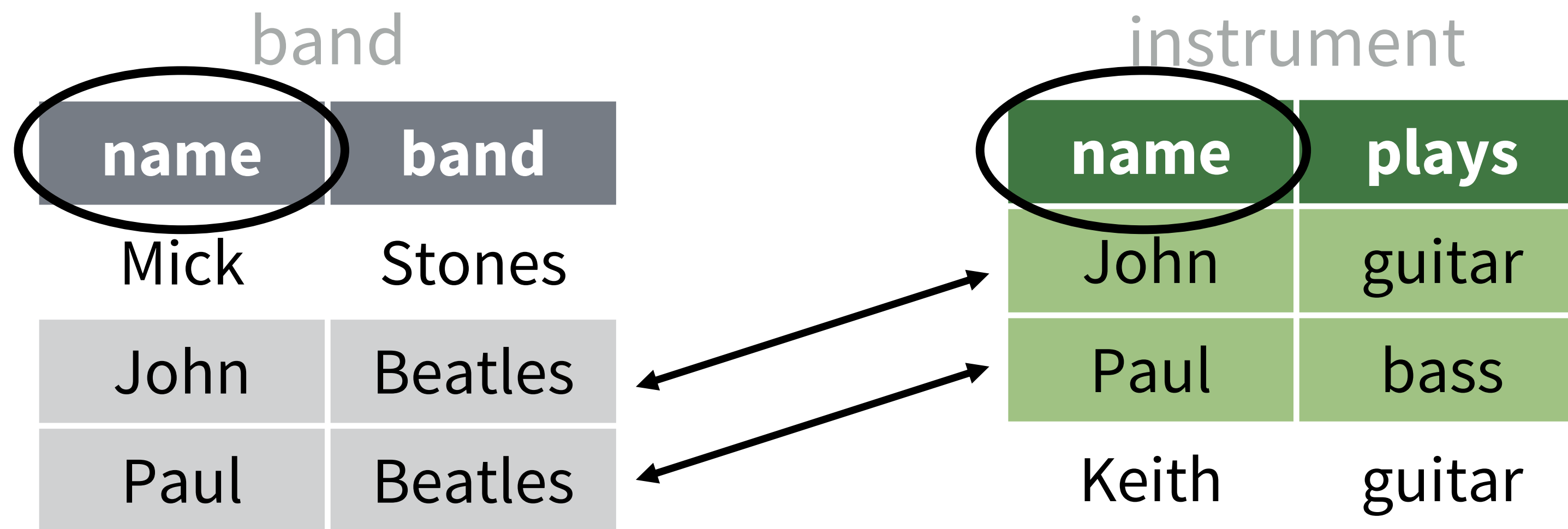
```
instrument <- tribble(
  ~name,    ~plays,
  "John",    "guitar",
  "Paul",    "bass",
  "Keith",   "guitar"
)
```

instrument

name	plays
John	guitar
Paul	bass
Keith	guitar



# Toy data



# left

```
band %>% left_join(instrument, by = "name")
```

band

name	band
Mick	Stones
John	Beatles
Paul	Beatles

+

instrument

name	plays
John	guitar
Paul	bass
Keith	guitar

=

name	band	plays
Mick	Stones	<NA>
John	Beatles	guitar
Paul	Beatles	bass



# right

```
band %>% right_join(instrument, by = "name")
```

band

name	band
Mick	Stones
John	Beatles
Paul	Beatles

+

instrument

name	plays
John	guitar
Paul	bass
Keith	guitar

=

name	band	plays
John	Beatles	guitar
Paul	Beatles	bass
Keith	<NA>	guitar



# full

```
band %>% full_join(instrument, by = "name")
```

band

name	band
Mick	Stones
John	Beatles
Paul	Beatles

+

instrument

name	plays
John	guitar
Paul	bass
Keith	guitar

=

name	band	plays
Mick	Stones	<NA>
John	Beatles	guitar
Paul	Beatles	bass
Keith	<NA>	guitar





# inner

```
band %>% inner_join(instrument, by = "name")
```

band

name	band
Mick	Stones
John	Beatles
Paul	Beatles

+

instrument

name	plays
John	guitar
Paul	bass
Keith	guitar

=

name	band	plays
John	Beatles	guitar
Paul	Beatles	bass



# What if the names do not match?

Use a named vector to match on variables with different names.

```
df1 %>% left_join(df2, by = c("df1_col" = "df2_col"))
```

A named vector

The name of the element = the column name in the first data set

The value of the element = the column name in the second data set



# Toy data

```
band <- tribble(  
  ~name,    ~band,  
  "Mick",   "Stones",  
  "John",   "Beatles",  
  "Paul",   "Beatles"  
)
```

band

name	band
Mick	Stones
John	Beatles
Paul	Beatles

```
instrument2 <- tribble(  
  ~artist,  ~plays,  
  "John",  "guitar",  
  "Paul",  "bass",  
  "Keith", "guitar"  
)
```

instrument2

artist	plays
John	guitar
Paul	bass
Keith	guitar



# nonmatching names

```
band %>% left_join(instrument2, by = c("name" = "artist"))
```

band			instrument2					
<b>name</b>	<b>band</b>		<b>artist</b>	<b>plays</b>		<b>name</b>	<b>band</b>	<b>plays</b>
Mick	Stones	+	John	guitar	=	Mick	Stones	<NA>
John	Beatles		Paul	bass		John	Beatles	guitar
Paul	Beatles		Keith	guitar		Paul	Beatles	bass



# Recap: Two table verbs

 **left\_join()** retains all cases in **left** data set

 **right\_join()** retains all cases in **right** data set

 **full\_join()** retains all cases in **either** data set

 **inner\_join()** retains only cases in **both** data sets



# Two table verbs

### Vectorized Functions

to use with mutate()

mutate() and transmute() apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

vectorized function

Offsets

dplyr::lag() - Offset elements by 1  
dplyr::lead() - Offset elements by -1

Cumulative Aggregates

dplyr::cumall() - Cumulative all()  
dplyr::cumany() - Cumulative any()  
dplyr::cummax() - Cumulative max()  
dplyr::cummean() - Cumulative mean()  
dplyr::cummin() - Cumulative min()  
dplyr::cumprod() - Cumulative prod()  
dplyr::cumsum() - Cumulative sum()

Rankings

dplyr::cume\_dist() - Proportion of all values <=   
dplyr::dense\_rank() - rank with ties = min, no gaps  
dplyr::min\_rank() - rank with ties = min  
dplyr::ntile() - bins into n bins  
dplyr::percent\_rank() - min\_rank scaled to [0,1]  
dplyr::row\_number() - rank with ties = "first"

Math

+, -, \*, /, ^, %/%, %% - arithmetic ops  
log(), log2(), log10() - logs  
<, <=, >, >=, !=, == - logical comparisons

Misc

dplyr::between() - x > right & x < left  
dplyr::case\_when() - multi-case if\_else()  
dplyr::coalesce() - first non-NA values by element across a set of vectors  
if\_else() - element-wise if() + else()  
dplyr::na\_if() - replace specific values with NA  
pmax() - element-wise max()  
pmin() - element-wise min()  
dplyr::recode() - Vectorized switch()  
dplyr::recode\_factor() - Vectorized switch() for factors

### Summary Functions

to use with summarise()

summarise() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

summary function

Counts

dplyr::n() - number of values/rows  
dplyr::n\_distinct() - # of uniques  
sum(is.na()) - # of non-NA's

Location

mean() - mean, also mean(is.na())  
median() - median

Logicals

mean() - Proportion of TRUE's  
sum() - # of TRUE's

Position/Order

dplyr::first() - first value  
dplyr::last() - last value  
dplyr::nth() - value in nth location of vector

Rank

quantile() - nth quantile  
min() - minimum value  
max() - maximum value

Spread

IQR() - Inter-Quartile Range  
mad() - mean absolute deviation  
sd() - standard deviation  
var() - variance

### Row names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.

rownames\_to\_column()

Move row names into col.  
a <- rownames\_to\_column(iris,  
var = "C")

column\_to\_rownames()

Move col in row names.  
column\_to\_rownames(a,  
var = "C")

Also has\_rownames(), remove\_rownames()

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com

Learn more with `browseVignettes(package = "dplyr", "tibble")` • dplyr 0.5.0 • tibble 1.2.0 • Updated: 12/16

## Combine Tables

### Combine Variables

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

Use `bind_cols()` to paste tables beside each other as they are.

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(...)`

Returns tables placed side by side as a single table.  
BE SURE THAT ROWS ALIGN.

Use a "Mutating Join" to join one table to columns from another, matching values with the rows that they correspond to. Each join retains a different combination of values from the tables.

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

`left_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ...)`  
Join matching values from y to x.

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

`right_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ...)`  
Join matching values from x to y.

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

`inner_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ...)`  
Join data. Retain only rows with matches.

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

`full_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ...)`  
Join data. Retain all values, all rows.

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

Use `by = c("col1", "col2")` to specify the column(s) to match on.

`left_join(x, y, by = "A")`

A.x	B.x	C	A.y	B.y
a	t	1	d	w
b	u	2	b	u
c	v	3	a	t

Use a named vector, `by = c("col1" = "col2")`, to match on columns with different names in each data set.  
`left_join(x, y, by = c("C" = "D"))`

A1	B1	C	A2	B2
a	t	1	d	w
b	u	2	b	u
c	v	3	a	t

Use `suffix` to specify suffix to give to duplicate column names.  
`left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))`

### Combine Cases

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

z		
A	B	C
c	v	3
d	w	4

Use `bind_rows()` to paste tables below each other as they are.

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

`bind_rows(..., .id = NULL)`

Returns tables one on top of the other as a single table. Set `.id` to a column name to add a column of the original table names (as pictured)

A	B	C
c	v	3

`intersect(x, y, ...)`

Rows that appear in both x and z.

A	B	C
a	t	1
b	u	2

`setdiff(x, y, ...)`

Rows that appear in both x but not z.

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

`union(x, y, ...)`

Rows that appear in x or z. (Duplicates removed). `union_all()` retains duplicates.

Use `setequal()` to test whether two data sets contain the exact same rows (in any order).

### Extract Rows

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

Use a "Filtering Join" to filter one table against the rows of another.

A	B	C
a	t	1
b	u	2

`semi_join(x, y, by = NULL, ...)`

Return rows of x that have a match in y. USEFUL TO SEE WHAT WILL BE JOINED.

A	B	C
c	v	3

`anti_join(x, y, by = NULL, ...)`

Return rows of x that do not have a match in y. USEFUL TO SEE WHAT WILL NOT BE JOINED.

