

# Transform Data with



# Joining Datasets



# Motivating Example: Guana Fish data

```
names (fish)
```



# Motivating Example: Guana Fish data

```
names (fish)
```

1	anchoa_hepsetus
2	anchoa_mitchilli
3	brevoortia_tyrannus
4	brevoortia_sp
5	chaetodipterus_faber
6	ctenogobius_oleosoma
7	ctenogobius_pseudofasciatus



# Motivating Example: Guana Fish data

```
names (fish)
```

1	anchoa_hepsetus
2	anchoa_mitchilli
3	brevoortia_tyrannus
4	brevoortia_sp
5	chaetodipterus_faber
6	ctenogobius_oleosoma
7	ctenogobius_pseudofasciatus

13	callinectes_sapidus
14	callinectes_similis
15	callinectes_sp
16	farfantepenaeus_aztecus
17	litopenaeus_setiferus
18	palaemonetes_sp
19	loliguncula_brevis



# Motivating Example: `fish` data

Wait a second....  
aren't these  
invertebrates??

```
names (fish)
```

1	<code>anchoa_hepsetus</code>
2	<code>anchoa_mitchilli</code>
3	<code>brevoortia_tyrannus</code>
4	<code>brevoortia_sp</code>
5	<code>chaetodipterus_faber</code>
6	<code>ctenogobius_oleosoma</code>
7	<code>ctenogobius_pseudofasciatus</code>

13	<code>callinectes_sapidus</code>
14	<code>callinectes_similis</code>
15	<code>callinectes_sp</code>
16	<code>farfantepenaeus_aztecus</code>
17	<code>litopenaeus_setiferus</code>
18	<code>palaemonetes_sp</code>
19	<code>loliguncula_brevis</code>

# Motivating Example: Guana Fish data

Maybe we want to make a data frame that looks like this.  
And we need to **bring additional information** into our data frame, as **additional columns**, based on matching.

UNID	species	type	n
1	anchoa_mitchilli	fish	1000
1	brevoortia_patronus	fish	50
1	callinectes_sapidus	invert	8
2	anchoa_mitchilli	fish	200
2	brevoortia_patronus	fish	600
2	callinectes_sapidus	invert	4



# Motivating Example: Guana Fish data

```
View(critters)
```





# Motivating Example

# na Fish data

```
View(critters)
```

	species	type
1	anchoa_hepsetus	fish
2	anchoa_mitchilli	fish
3	brevortia_tyrannus	fish
4	brevortia_sp	fish
5	chaetodipterus_faber	fish
6	ctenogobius_oleosoma	fish
7	ctenogobius_pseudofasciatus	fish
8	ctenogobius_sp	fish
9	cynoscion_nebulosus	fish
10	cynoscion_sp	fish
11	cynoscion_sp_2	fish
12	cyprinodon_variegatus	fish
13	callinectes_sapidus	invert
14	callinectes_similis	invert
15	callinectes_sp	invert
16	farfantepenaeus_aztecus	invert
17	litopenaeus_setiferus	invert
18	palaemonetes_sp	invert
19	loliguncula_brevis	invert



# Motivating Example 2: yearly data files

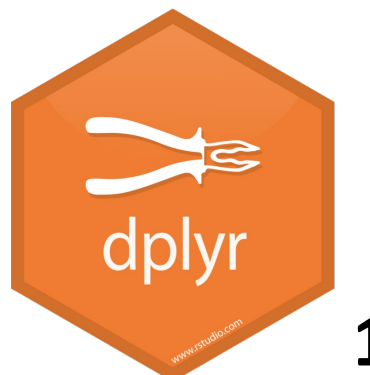
gndbhnut2016.csv - Excel													
Kimber													
File Home Insert Page Layout Formulas Data Review View Developer Help OffCAT Search													
A2 X ✓ fx gndbhnut													
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	StationCode	isSWMP	DateTimeStamp	Historical	Provisiona	CollMethd	REP	F_Record	PO4F	F_PO4F	NH4F	F_NH4F	NH4F
2	gndbhnut	P	1/13/2016 12:05	0	1	1	1			<-2> [GDM] (CSM)		<-2> [GDM] (CSM)	
3	gndbhnut	P	1/13/2016 12:06	0	1	1	2			<-2> [GDM] (CSM)		<-2> [GDM] (CSM)	
4	gndbhnut	P	2/10/2016 8:51	0	1	1	1		0.002	<-4> [SBL]	0.038	<0>	
5	gndbhnut	P	2/10/2016 8:52	0	1	1	2		0.002	<-4> [SBL]	0.038	<0>	
6	gndbhnut	P	3/21/2016 9:36	0	1	1	1		0.012	<0>	0.012	<0>	
7	gndbhnut	P	3/21/2016 9:37	0	1	1	2		0.008	<0>	0.012	<0>	
8	gndbhnut	P	4/19/2016 15:53	0	1	1	1		0.007	<0>	0.017	<0>	
9	gndbhnut	P	4/19/2016 15:54	0	1	1	2		0.008	<0>	0.014	<0>	
10	gndbhnut	P	5/18/2016 16:37	0	1	1	1		0.008	<0>	0.019	<0>	
11	gndbhnut	P	5/18/2016 16:38	0	1	1	2		0.007	<0>	0.017	<0>	
12	gndbhnut	P	6/15/2016 14:36	0	1	1	1		0.002	<0>	0.236	<0>	
13	gndbhnut	P	6/15/2016 14:37	0	1	1	2		0.002	<0>	0.163	<0>	
14	gndbhnut	P	7/13/2016 16:47	0	1	1	1		0.002	<0>	0.004	<0>	
15	gndbhnut	P	7/13/2016 16:48	0	1	1	2		0.003	<0>	0.003	<0>	
16	gndbhnut	P	8/22/2016 9:49	0	1	1	1		0.002	<0>	0.114	<0>	
17	gndbhnut	P	8/22/2016 9:50	0	1	1	2		0.003	<0>	0.111	<0>	
18	gndbhnut	P	9/19/2016 10:03	0	1	1	1		0.015	<0>	0.189	<0>	
19	gndbhnut	P	9/19/2016 10:04	0	1	1	2		0.015	<0>	0.178	<0>	
20	gndbhnut	P	10/19/2016 8:17	0	1	1	1		0.006	<0>	0.059	<0>	
21	gndbhnut	P	10/19/2016 8:18	0	1	1	2		0.008	<0>	0.06	<0>	
22	gndbhnut	P	11/16/2016 12:00	0	1	1	1		0.009	<0>	0.032	<0>	
23	gndbhnut	P	11/16/2016 12:01	0	1	1	2		0.014	<0>	0.038	<0>	
24	gndbhnut	P	12/14/2016 8:28	0	1	1	1		0.002	<0>	0.016	<1> [GSM] (CHB)	
25	gndbhnut	P	12/14/2016 8:29	0	1	1	2		0.002	<0>	0.016	<1> [GSM] (CHB)	
26													
27													
28													
29													



# Motivating Example 2: yearly data files

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)	

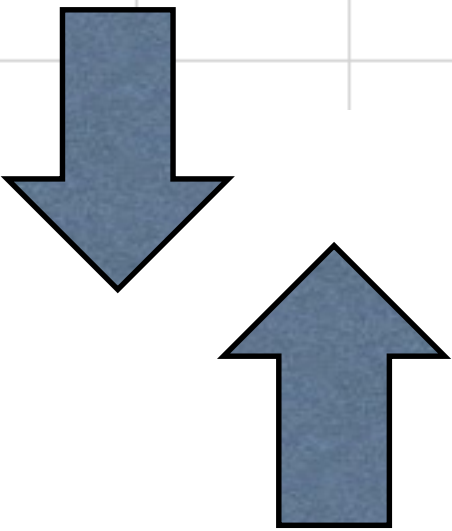
gndbhnut	P	1/17/2017 11:18	0	1	1	1		0.004	<0>	0.153	<0>	
gndbhnut	P	1/17/2017 11:19	0	1	1	2		0.005	<0>	0.153	<0>	
gndbhnut	P	2/13/2017 10:43	0	1	1	1		0.002	<-4> [SBL]	0.158	<0>	
gndbhnut	P	2/13/2017 10:44	0	1	1	2		0.002	<-4> [SBL]	0.173	<0>	
gndbhnut	P	3/13/2017 10:56	0	1	1	1		0.003	<0>	0.157	<0>	
gndbhnut	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>	





# Motivating Example 2: yearly data files

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)	



Again, we need to **bring additional information** into our data frame.  
This time 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>	



# Motivating Example 2: yearly data files

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)	
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>	





# Your Turn 1

## 5 minutes

1. Run the code chunk starting on line 18 to generate toy data.
2. Import the following data frames, and make sure the column names are all lower-case and connected by underscores ( `_` ); no spaces or periods.
  - a. `guana_fish_subset.csv`, as a data frame called `"fish"`
  - b. `critter_types.csv`, as a data frame called `"critters"`
3. Examine all the data frames you now have.



# Your Turn 1

## Answer

```
fish <- read.csv(here::here("data", "guana_fish_subset.csv"),  
                stringsAsFactors = FALSE) %>%  
  janitor::clean_names()  
critters <- read.csv(here::here("data", "critter_types.csv"),  
                    stringsAsFactors = FALSE)
```



# mutating joins





# Motivating Example: Guana Fish data

**additional information**  
as **additional columns**

UNID	species	type	n
1	anchoa_mitchilli	fish	1000
1	brevoortia_patronus	fish	50
1	callinectes_sapidus	invert	8
2	anchoa_mitchilli	fish	200
2	brevoortia_patronus	fish	600
2	callinectes_sapidus	invert	4



# 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

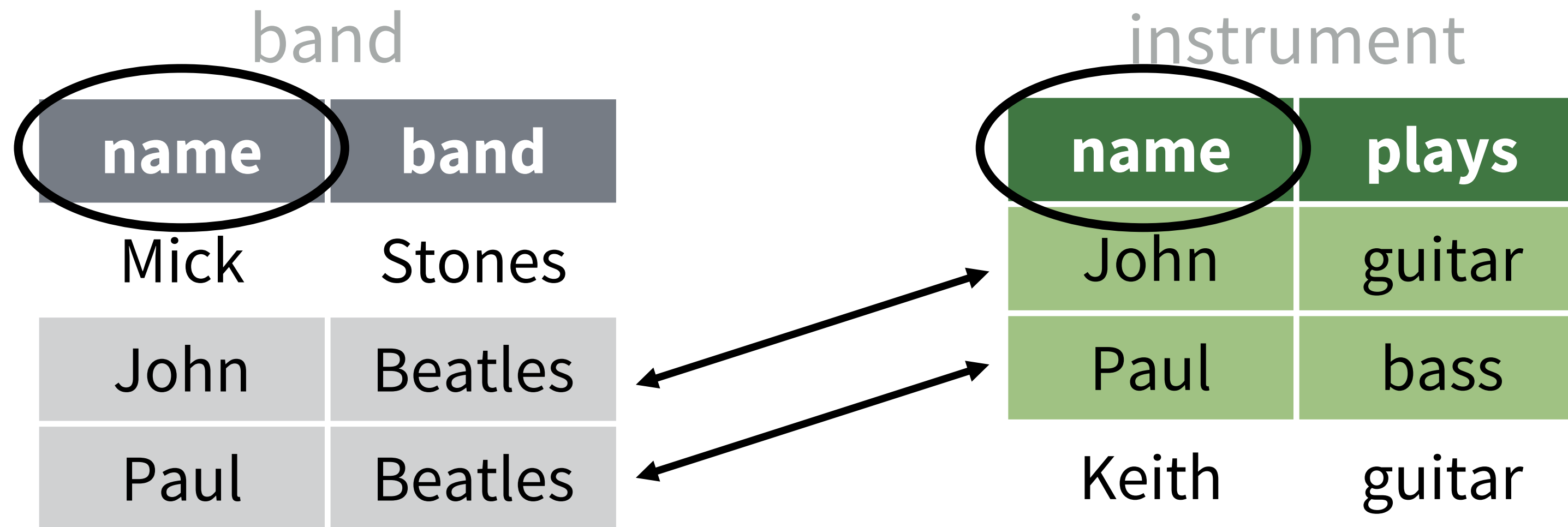
```
instrument <- tribble(  
  ~name,    ~plays,  
  "John",   "guitar",  
  "Paul",   "bass",  
  "Keith",  "guitar"  
)
```

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



# Your Turn 2

## 10 minutes

```
fish %>%
```

```
  pivot_____(_____) %>%
```

```
  _____join(_____) %>%
```

```
  group_by(_____) %>%
```

```
  _____(_____)
```

**1. Pivot fish to a long format, with one row per species per sample**

**2. Join critters to fish**

**3. Compute the total number caught by type (fish/invertebrate), by time of day (day/night)**



# Your Turn 2

## Answer

```
fish %>%  
  pivot_longer(cols = 2:20,  
               names_to = "species",  
               values_to = "count") %>%  
  left_join(critters, by = "species") %>%  
  group_by(type, diel) %>%  
  summarize(total = sum(count))
```



# Your Turn 2

## Answer

```
fish %>%  
  pivot_longer(cols = 2:20,  
               names_to = "species",  
               values_to = "count") %>%  
  left_join(critters, by = "species") %>%  
  group_by(type, diel) %>%  
  summarize(total = sum(count)) %>%  
  pivot_wider(names_from = type,  
              values_from = total)
```



# filtering joins



# filtering joins

**Mutating joins** use information from one data set **to add variables** to another data set (like **mutate()**)

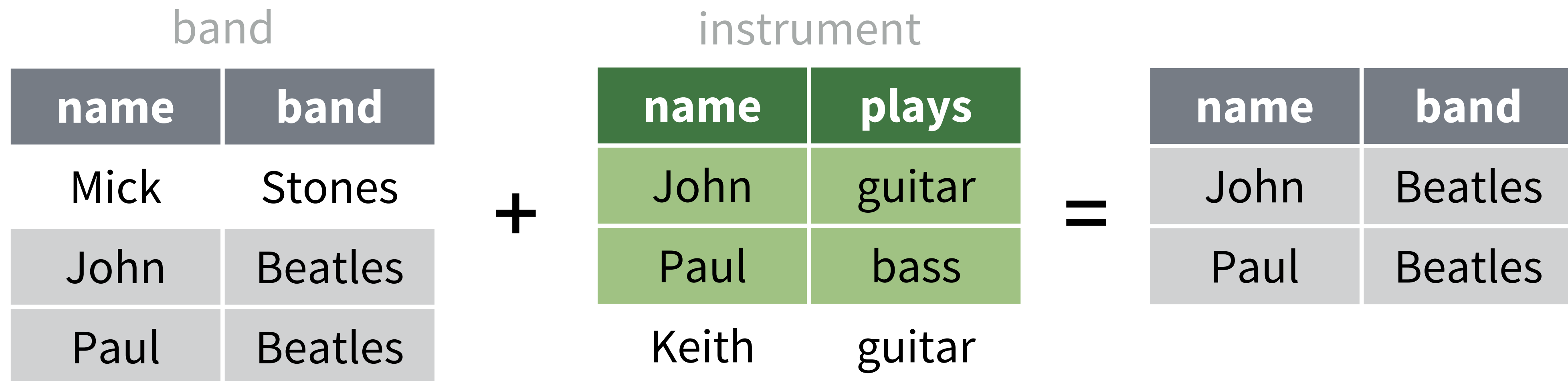
**Filtering joins** use information from one data set **to extract cases** from another data set (like **filter()**)





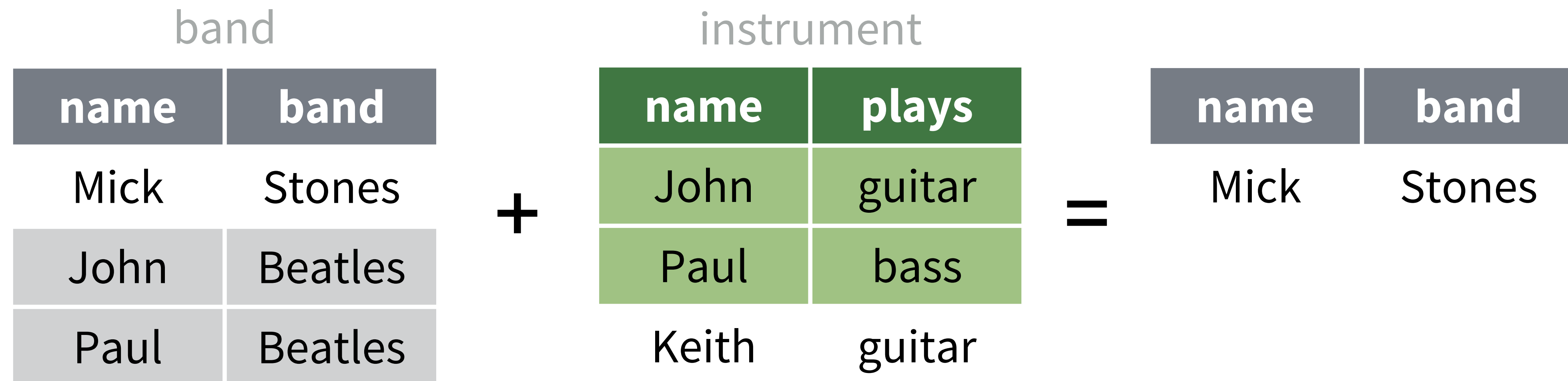
# semi

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



# anti

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



# 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



**semi\_join()** extracts cases that **have a match**



**anti\_join()** extracts cases that **do not have a match**



# 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()

### Combine Variables

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 bind\_cols() to paste tables beside each other as they are.

A	B	C	A	B	C
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.

### Combine Cases

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 bind\_rows() to paste tables below each other as they are.

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

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
c	v	3	NA
d	w	NA	1

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

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

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

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

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

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

left\_join(x, y, by = "A")

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"))

Use suffix to specify suffix to give to duplicate column names.

left\_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))

### 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
a	t	1
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.



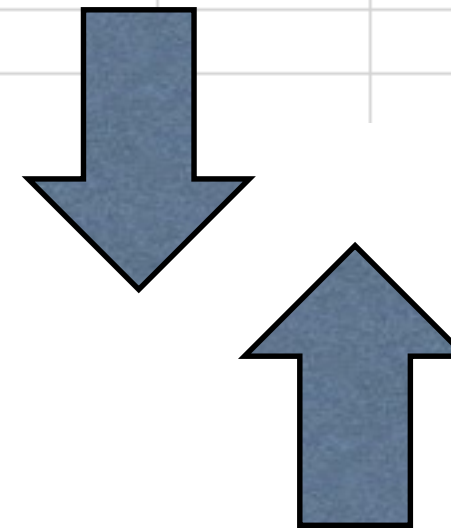
row/column  
binding





# 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()`

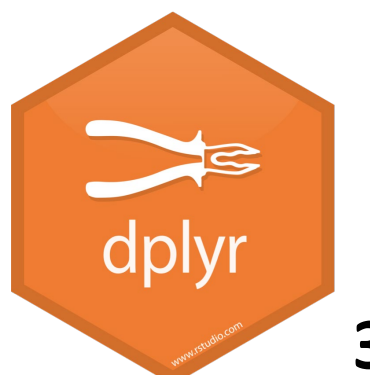
	year_sampled	salinity_ppt	water_temp_c
1	2005	8.1	23.3
2	2005	7.9	24.3
3	2005	8.2	24.8
4	2005	8.2	25.3
5	2005	9.0	24.8
6	2005	9.2	28.1
7	2005	8.4	26.2
8	2005	8.9	27.3
9	2005	11.5	27.6
10	2005	10.3	27.8
11	2005	10.4	26.5
12	2005	12.0	28.9
13	2005	10.2	27.6
14	2005	18.5	29.3



	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





# Your Turn 3

## 3 minutes

Make sure to run the “Toy data for binding” code chunk, starting on line 163 of your .Rmd file.

Then navigate to line 219, “Your Turn 3”. Take a guess at what the next two code chunks will return.

Were you right? Were you surprised? Tell us in the chat box!



# Your Turn 4

## 5 minutes

Run the code chunk starting on line 245.

In the following code chunk, use both `rbind` and `bind_rows()` to join `bh2016` and `bh2017` together. Examine them. Are they the same? Did you expect them to be?

One more code chunk: use both `rbind` and `bind_rows()` to bind all **three** `bh` data frames together. Examine them and answer the same questions.