Econ 106: Data Analyis for Economics

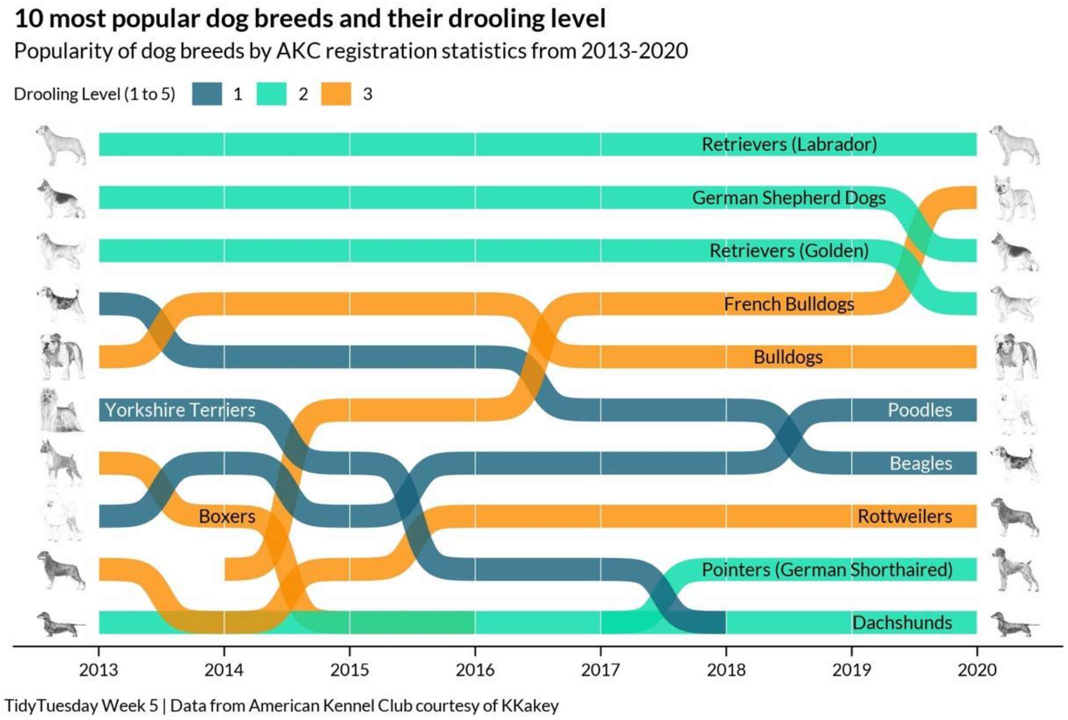
Lecture 12

slides adapted from: https://r4ds.had.co.nz/tidy-data.html

Reminders

- Lab #3 is due Sunday 11:59pm
- MS #2 is posted, will be due the following Sunday (start it early!)
- No lecture next Monday (Veteran’s Day)

#tidytuesday



https://x.com/leeolney3/status/1488387306662813701?s=20

#tidytuesday

- Missing: the ranking of dogs most likely to use a basket at a pillow

Louberto!

Outline

- Joins:
  - left
  - inner
  - anti

Combining Data Tables

- Suppose you wanted to add the year founded column to the superheroes data
- How do we do this?

	name	alignment	gender	publisher
1	Magneto	bad	male	Marvel
2	Storm	good	female	Marvel
3	Mystique	bad	female	Marvel
4	Batman	good	male	DC
5	Joker	bad	male	DC
6	Catwoman	bad	female	DC
7	Hellboy	good	male	Dark Horse Comics

	publisher	yr_founded
1	DC	1934
2	Marvel	1939
3	Image	1992

Combining Data Tables

- First, we need to identify the variables that will link the tables together

	name	alignment	gender	publisher
1	Magneto	bad	male	Marvel
2	Storm	good	female	Marvel
3	Mystique	bad	female	Marvel
4	Batman	good	male	DC
5	Joker	bad	male	DC
6	Catwoman	bad	female	DC
7	Hellboy	good	male	Dark Horse Comics

	publisher	yr_founded
1	DC	1934
2	Marvel	1939
3	Image	1992

Identify the Linking Variable(s)

- publisher is a variable that can tell us how to add yr_founded information to the superhero dataframe
- The variables used to connect two tables are called **keys**

	name	alignment	gender	publisher
1	Magneto	bad	male	Marvel
2	Storm	good	female	Marvel
3	Mystique	bad	female	Marvel
4	Batman	good	male	DC
5	Joker	bad	male	DC
6	Catwoman	bad	female	DC
7	Hellboy	good	male	Dark Horse Comics

	publisher	yr_founded
1	DC	1934
2	Marvel	1939
3	Image	1992

Primary Keys

- publisher uniquely identifies an observation in the publishers table (it is a primary key)
- It shows up in more than one row in the superhero data, but that's ok.

	name	alignment	gender	publisher
1	Magneto	bad	male	Marvel
2	Storm	good	female	Marvel
3	Mystique	bad	female	Marvel
4	Batman	good	male	DC
5	Joker	bad	male	DC
6	Catwoman	bad	female	DC
7	Hellboy	good	male	Dark Horse Comics

	publisher	yr_founded
1	DC	1934
2	Marvel	1939
3	Image	1992

Many-to-one Matching

- It's ok that information on the right table will show up on multiple rows of the left table (it has a clear destination)
- this is called many-to-one matching

	name	alignment	gender	publisher
1	Magneto	bad	male	Marvel
2	Storm	good	female	Marvel
3	Mystique	bad	female	Marvel
4	Batman	good	male	DC
5	Joker	bad	male	DC
6	Catwoman	bad	female	DC
7	Hellboy	good	male	Dark Horse Comics

	publisher	yr_founded
1	DC	1934
2	Marvel	1939
3	Image	1992

Problems: wrong primary key

- In this example, term does not uniquely identify rows in the right data frame or the left data frame (it is not a primary key in either data frame)

Term	Year	GPA
Fall	2022	3.4
Winter	2023	3.3
Spring	2023	3.1
Fall	2023	3.2

Term	Year	Hours worked
Fall	2022	20
Winter	2023	30
Spring	2023	10
Fall	2023	40

Problems: wrong primary key

- If we tried to combine the data based on Term, The fall data rows on the right will match to the same rows on the left (many-to many matching)

Term	Year	GPA
Fall	2022	3.4
Winter	2023	3.3
Spring	2023	3.1
Fall	2023	3.2

Term	Year	Hours worked
Fall	2022	20
Winter	2023	30
Spring	2023	10
Fall	2023	40

Problems: wrong primary key

- As a result, R will add extra rows for every combination of matches to the Term variable

Term	Year	GPA
Fall	2022	3.4
Winter	2023	3.3
Spring	2023	3.1
Fall	2023	3.2

Term	Year	Hours worked
Fall	2022	20
Winter	2023	30
Spring	2023	10
Fall	2023	40

Term	Year	GPA	Hours worked
Fall	2022	3.4	20
Fall	2023	3.4	40
Winter	2023	3.3	30
Spring	2023	3.1	10
Fall	2023	3.2	40
Fall	2022	3.2	20

Solution: Primary key can be multiple variables

- We need to use more than one variable to link these two data frames correctly
- The primary key is year **and** term

Term	Year	GPA
Fall	2022	3.4
Winter	2023	3.3
Spring	2023	3.1
Fall	2023	3.2

Term	Year	Hours worked
Fall	2022	20
Winter	2023	30
Spring	2023	10
Fall	2023	40

https://pollev.com/vsovero

Solution: Primary key can be multiple variables

- We need to use more than one variable to link these two data frames correctly
- The primary key is year **and** term
- there is no ambiguity on which rows should be matched together

Term	Year	GPA
Fall	2022	3.4
Winter	2023	3.3
Spring	2023	3.1
Fall	2023	3.2

Term	Year	Hours worked
Fall	2022	20
Winter	2023	30
Spring	2023	10
Fall	2023	40

Term	Year	GPA	Hours worked
Fall	2022	3.4	20
Winter	2023	3.3	30
Spring	2023	3.1	10
Fall	2023	3.2	40

left_join(x=superheroes,y=publishers,
by="publisher")

Structure of a join

Required arguments:

- x: Data on the left:
- y: Data on the right

Optional, but recommended argument:

- by: primary key

x

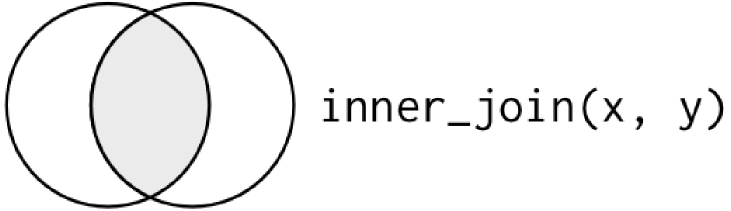
superheroes			
name	alignment	gender	publisher
Magneto	bad	male	Marvel
Storm	good	female	Marvel
Mystique	bad	female	Marvel
Batman	good	male	DC
Joker	bad	male	DC
Catwoman	bad	female	DC
Hellboy	good	male	Dark Horse Comics

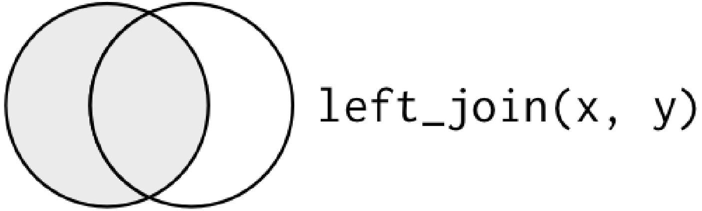
y

publishers	
publisher	yr_founded
DC	1934
Marvel	1939
Image	1992

Types of Joins

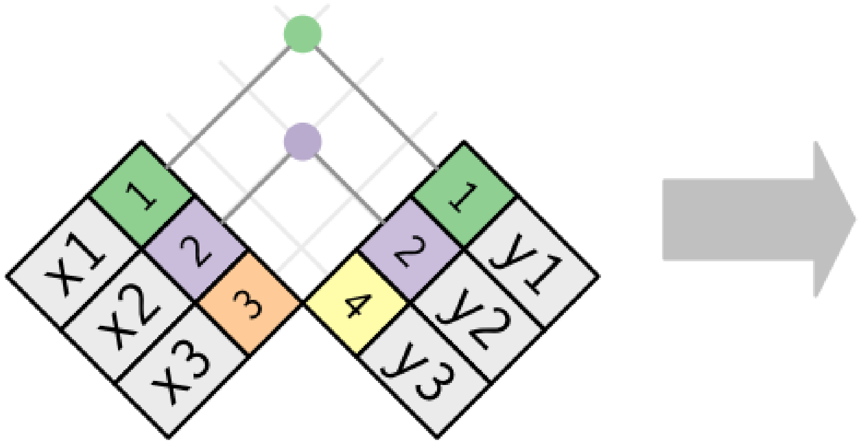
- Left Join: keep all rows in the left dataset, even if they can't be matched
- Inner join: only keep rows in the left dataset that can be matched





Inner Join

- Only keep matches



key	val_x	val_y
1	x1	y1
2	x2	y2

https://pollev.com/vsovero

inner_join(x=superheroes, y=publishers, by="publisher")

- Only keep the observations from x that can be matched to y
- Hellboy is dropped from the final table

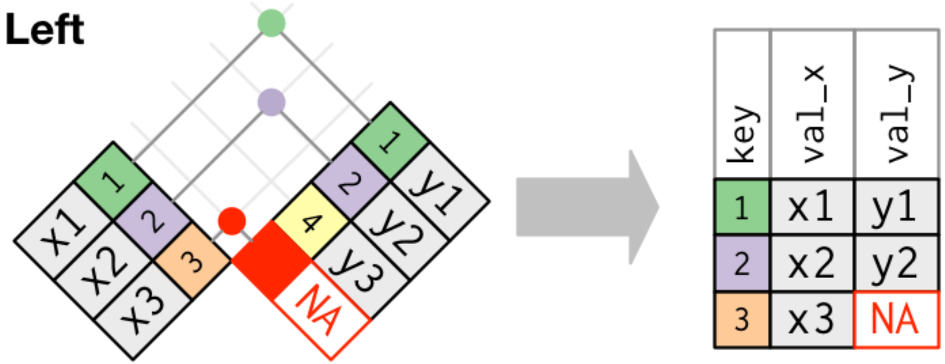
superheroes			
name	alignment	gender	publisher
Magneto	bad	male	Marvel
Storm	good	female	Marvel
Mystique	bad	female	Marvel
Batman	good	male	DC
Joker	bad	male	DC
Catwoman	bad	female	DC
Hellboy	good	male	Dark Horse Comics

publishers	
publisher	yr_founded
DC	1934
Marvel	1939
Image	1992

inner_join(x = superheroes, y = publishers)				
name	alignment	gender	publisher	yr_founded
Magneto	bad	male	Marvel	1939
Storm	good	female	Marvel	1939
Mystique	bad	female	Marvel	1939
Batman	good	male	DC	1934
Joker	bad	male	DC	1934
Catwoman	bad	female	DC	1934

Left Join

- A **left join** keeps all rows in the left data table
- For matched rows, it merges in the values from the right data table
- For unmatched rows, it fills in NA values for the variables coming from the right data table.



https://pollev.com/vsovero

left_join (x=superheroes, y=publishers, by="publisher")

- Keep all the observations from x (table on the **left**), regardless of whether or not they can be matched to y
- Hellboy cannot be matched, so there is no information on yr_founded (NA)

superheroes			
name	alignment	gender	publisher
Magneto	bad	male	Marvel
Storm	good	female	Marvel
Mystique	bad	female	Marvel
Batman	good	male	DC
Joker	bad	male	DC
Catwoman	bad	female	DC
Hellboy	good	male	Dark Horse Comics

publishers	
publisher	yr_founded
DC	1934
Marvel	1939
Image	1992

left_join(x = superheroes, y = publishers)				
name	alignment	gender	publisher	yr_founded
Magneto	bad	male	Marvel	1939
Storm	good	female	Marvel	1939
Mystique	bad	female	Marvel	1939
Batman	good	male	DC	1934
Joker	bad	male	DC	1934
Catwoman	bad	female	DC	1934
Hellboy	good	male	Dark Horse Comics	NA

Be Careful: Order Matters

left_join(x=publishers,y=superheroes,
by="publisher")

x

publishers	
publisher	yr_founded
DC	1934
Marvel	1939
Image	1992

y

superheroes			
name	alignment	gender	publisher
Magneto	bad	male	Marvel
Storm	good	female	Marvel
Mystique	bad	female	Marvel
Batman	good	male	DC
Joker	bad	male	DC
Catwoman	bad	female	DC
Hellboy	good	male	Dark Horse Comics

What happens if we swap the order of the datasets?

1. Data on the left: publishers
2. Data on the right: superheroes
3. Linking variable(s): publisher

left_join (x=publishers, y=superheroes, by="publisher")

- Keep all the observations from x (table on the **left**), regardless of whether or not they can be matched to y
- Image cannot be matched, so there is no information on variables in the superheroes data table(set to NA)

publishers

publisher	yr_founded
DC	1934
Marvel	1939
Image	1992

superheroes

name	alignment	gender	publisher
Magneto	bad	male	Marvel
Storm	good	female	Marvel
Mystique	bad	female	Marvel
Batman	good	male	DC
Joker	bad	male	DC
Catwoman	bad	female	DC
Hellboy	good	male	Dark Horse Comics

left_join(x = publishers, y = superheroes)

publisher	yr_founded	name	alignment	gender
DC	1934	Batman	good	male
DC	1934	Joker	bad	male
DC	1934	Catwoman	bad	female
Marvel	1939	Magneto	bad	male
Marvel	1939	Storm	good	female
Marvel	1939	Mystique	bad	female
Image	1992	NA	NA	NA

Best Practices for Joins

- Define your “main” dataset as x (the data on the left)
- If you want to add variable to your main dataset, left joins are safer
- Left joins preserve all the original observations in your main dataset

The Recommended Join

left_join (x=superheroes, y=publishers, by="publisher")

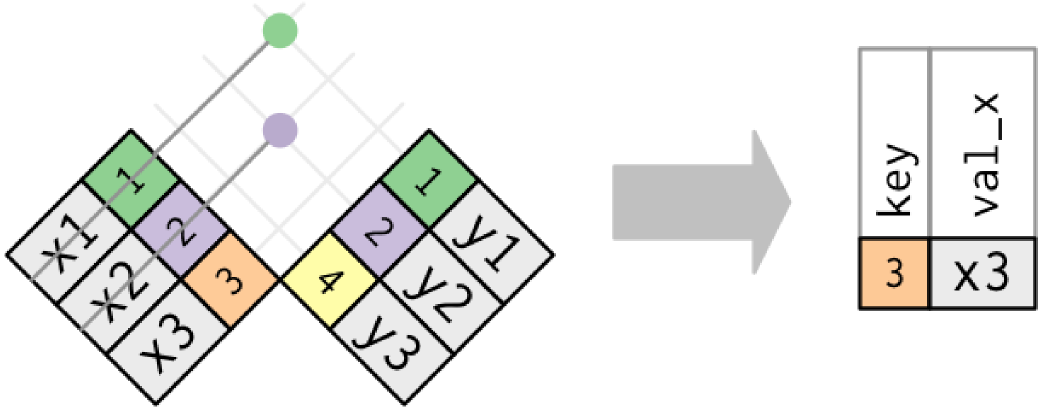
superheroes			
name	alignment	gender	publisher
Magneto	bad	male	Marvel
Storm	good	female	Marvel
Mystique	bad	female	Marvel
Batman	good	male	DC
Joker	bad	male	DC
Catwoman	bad	female	DC
Hellboy	good	male	Dark Horse Comics

publishers	
publisher	yr_founded
DC	1934
Marvel	1939
Image	1992

left_join(x = superheroes, y = publishers)				
name	alignment	gender	publisher	yr_founded
Magneto	bad	male	Marvel	1939
Storm	good	female	Marvel	1939
Mystique	bad	female	Marvel	1939
Batman	good	male	DC	1934
Joker	bad	male	DC	1934
Catwoman	bad	female	DC	1934
Hellboy	good	male	Dark Horse Comics	NA

Anti Join

- An anti-join keeps the rows from the left data table that *don't* have a match
- Anti-joins are useful for checking what didn't match and figuring out why



anti_join(x=superheroes, y=publishers, by="publisher")

- Keep the observations from x that have **no match** to y
- Hellboy cannot be matched, so this is the only observation kept
- There is no yr_founded because by definition there is no match to the publishers data

superheroes			
name	alignment	gender	publisher
Magneto	bad	male	Marvel
Storm	good	female	Marvel
Mystique	bad	female	Marvel
Batman	good	male	DC
Joker	bad	male	DC
Catwoman	bad	female	DC
Hellboy	good	male	Dark Horse Comics

publishers	
publisher	yr_founded
DC	1934
Marvel	1939
Image	1992

anti_join(x = superheroes, y = publishers)			
name	alignment	gender	publisher
Hellboy	good	male	Dark Horse Comics

Example: NYC Flights

- Datasets containing information on flights in and out of the New York area in 2013

```r
library(nycflights13)
data(flights)
data(weather)
```

Flights and Weather

- Suppose you wanted to add weather data to the flights data table
- Specifically, you want to add information on the weather at the time of the scheduled departure
- Look at all the variables in each data table. Which are needed to link these tables?

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_d
1	2013	1	1	517	515	2	830	819	
2	2013	1	1	533	529	4	850	830	
3	2013	1	1	542	540	2	923	850	
4	2013	1	1	544	545	-1	1004	1022	
5	2013	1	1	554	600	-6	812	837	
6	2013	1	1	554	558	-4	740	728	
7	2013	1	1	555	600	-5	913	854	
8	2013	1	1	557	600	-3	709	723	
9	2013	1	1	557	600	-3	838	846	
10	2013	1	1	558	600	-2	753	745	
11	2013	1	1	558	600	-2	849	851	
12	2013	1	1	558	600	-2	853	856	
13	2013	1	1	558	600	-2	924	917	
14	2013	1	1	558	600	-2	923	937	
15	2013	1	1	559	600	-1	941	910	

	origin	year	month	day	hour	temp	dewp
1	EWR	2013	1	1	1	39.02	26.06
2	EWR	2013	1	1	2	39.02	26.96
3	EWR	2013	1	1	3	39.02	28.04
4	EWR	2013	1	1	4	39.92	28.04
5	EWR	2013	1	1	5	39.02	28.04
6	EWR	2013	1	1	6	37.94	28.04
7	EWR	2013	1	1	7	39.02	28.04
8	EWR	2013	1	1	8	39.92	28.04
9	EWR	2013	1	1	9	39.92	28.04
10	EWR	2013	1	1	10	41.00	28.04

Flights with Weather Added

- Flights is the “main” dataset
- we want to keep all flights, even if there isn’t weather data available

```
flights_with_weather<-
left_join(x=flights,y=weather,

by=c("origin", "year","month", "day",
"hour","time_hour"))
```

What if we forget variables?

- let's suppose we forgot to include time_hour as a linking variable
- We still have enough information to make the join happen correctly
- data table will contain two new variables:
  - time_hour.x
  - time_hour.y

```
flights_with_weather<-
left_join(x=flights, y=weather,
by=c("origin", "year","month", "day", "hour"))
```

What if we forget variables?

```
flights_with_weather<-left_join(x=flights,y=weather,
                                by=c("origin", "year","month", "day", "hour"))
```

minute	time_hour.x	temp	dewp	humid	wind_dir	wind_speed	wind_gust	precip	pressure	visib	time_hour.y
15	2013-01-01 05:00:00	39.02	28.04	64.43	260	12.65858	NA	0	1011.9	10	2013-01-01 05:00:00
29	2013-01-01 05:00:00	39.92	24.98	54.81	250	14.96014	21.86482	0	1011.4	10	2013-01-01 05:00:00
40	2013-01-01 05:00:00	39.02	26.96	61.63	260	14.96014	NA	0	1012.1	10	2013-01-01 05:00:00
45	2013-01-01 05:00:00	39.02	26.96	61.63	260	14.96014	NA	0	1012.1	10	2013-01-01 05:00:00
0	2013-01-01 06:00:00	39.92	24.98	54.81	260	16.11092	23.01560	0	1011.7	10	2013-01-01 06:00:00
58	2013-01-01 05:00:00	39.02	28.04	64.43	260	12.65858	NA	0	1011.9	10	2013-01-01 05:00:00
0	2013-01-01 06:00:00	37.94	28.04	67.21	240	11.50780	NA	0	1012.4	10	2013-01-01 06:00:00

Bigger Problem

- The join will not work correctly if we don't provide the primary key needed to match the flight data to the weather data

```
flights_with_weather<-left_join(x=flights,y=weather,
                                by=c("origin", "year","month", "day"))
```

```
  Detected an unexpected many-to-many relationship between `x` and `y`.
i Row 1 of `x` matches multiple rows in `y`.
i Row 8704 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.
```

Solution: Have R identify the primary key

- If you don’t include the by argument, R will “guess” which variables should be used to link two data tables
- It will assume any variables in common are linking variables
- The selections will be listed in the console

flights_with_weather<-
left_join(x=flights,y=weather)

```
> flights_with_weather<-left_join(x=flights,y=weather)
Joining with `by = join_by(year, month, day, origin, hour, time_hour)`
>
> |
```

Health Coverage (tidy version)

- Each observation represents enrollment numbers by region, and year, and insurance type

	Location	year	type	tot_coverage
1	United States	2013	Employer	155696900
2	United States	2013	Non-Group	13816000
3	United States	2013	Medicaid	54919100
4	United States	2013	Medicare	40876300
5	United States	2013	Other Public	6295400
6	United States	2013	Uninsured	41795100
7	United States	2013	Total	313401200
8	United States	2014	Employer	154347500
9	United States	2014	Non-Group	19313000
10	United States	2014	Medicaid	61650400
11	United States	2014	Medicare	41896500
12	United States	2014	Other Public	5985000
13	United States	2014	Uninsured	32967500
14	United States	2014	Total	316159900

Expenditure Data

- We will try to add healthcare expenditure information to the coverage data table
- First, we need to load it and make it tidy

Step 1

- Create a new data frame called spending_long that looks like the table on the right

	Location	year	tot_spending
1	United States	1991__Total Health Spending	675896
2	United States	1992__Total Health Spending	731455
3	United States	1993__Total Health Spending	778684
4	United States	1994__Total Health Spending	820172
5	United States	1995__Total Health Spending	869578
6	United States	1996__Total Health Spending	917540
7	United States	1997__Total Health Spending	969531
8	United States	1998__Total Health Spending	1026103
9	United States	1999__Total Health Spending	1086280
10	United States	2000__Total Health Spending	1162035
11	United States	2001__Total Health Spending	1261944
12	United States	2002__Total Health Spending	1367628
13	United States	2003__Total Health Spending	1477697
14	United States	2004__Total Health Spending	1587994
15	United States	2005__Total Health Spending	1696222
16	United States	2006__Total Health Spending	1804672
17	United States	2007__Total Health Spending	1918820

Step 2

- Create a new data frame called spending_sep that looks like the table on the right

	Location	year	tot_spending
1	United States	1991	675896
2	United States	1992	731455
3	United States	1993	778684
4	United States	1994	820172
5	United States	1995	869578
6	United States	1996	917540
7	United States	1997	969531
8	United States	1998	1026103
9	United States	1999	1086280
10	United States	2000	1162035
11	United States	2001	1261944
12	United States	2002	1367628
13	United States	2003	1477697
14	United States	2004	1587994
15	United States	2005	1696222
16	United States	2006	1804672
17	United States	2007	1918820

Step 3

- Left join coverage_sep and spending_sep, call it coverage_left

Step 4

- Create a data frame of all the observations in the coverage data that cannot be matched to the spending data
- Why do you think they couldn’t be matched?

Step 5

- Create a data frame of all the observations in the spending data that cannot be matched to the coverage data
- Why do you think they couldn't be matched?

https://pollev.com/vsovero