Data Wrangling II:

Groups and Joins

INFO 201

Joel Ross

Winter 2017

https://slides.com/joelross/info201w17-dplyr-ii/live

Today's Objectives

By the end of class, you should be able to

- Feel comfortable using **dplyr**'s *data manipulation grammar* to ask questions
- **Group** observations for computing summary information
- Effectively **join** multiple data frames together

Grammar for Data Manipulation

Words (***verbs***) used to describe ways to manipulate data:

- **Select** the columns of interest
- **Filter** out irrelevant data to keep rows of interest
- **Mutate** a data set by adding more columns
- **Arrange** the rows in a data set
- **Summarize** the data (e.g., calculate the *mean*, *median*, *maximum*, etc).

Why is it helpful to use **a grammar** for data manipulation?

Practice: Today's Data Set

nycflights13: Flights departing NYC in 2013

```
# install library of data sets
install.packages("nycflights13")

# load library
library("nycflights13")

# inspect the `flights` data frame
View(head(flights))  #first 6 rows
```

https://cran.r-project.org/web/packages/nycflights13/index.html

Practice: Today's Data Set

nycflights13: Flights departing NYC in 2013

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier	flight	tailnum	origin	dest	air_time	distance	hour	minute
1	2013	1	1	517	515	2	830	819	11	UA	1545	N14228	EWR	IAH	227	1400	5	15
2	2013	1	1	533	529	4	850	830	20	UA	1714	N24211	LGA	IAH	227	1416	5	29
3	2013	1	1	542	540	2	923	850	33	AA	1141	N619AA	JFK	MIA	160	1089	5	40
4	2013	1	1	544	545	-1	1004	1022	-18	B6	725	N804JB	JFK	BQN	183	1576	5	45
5	2013	1	1	554	600	-6	812	837	-25	DL	461	N668DN	LGA	ATL	116	762	6	0
6	2013	1	1	554	558	-4	740	728	12	UA	1696	N39463	EWR	ORD	150	719	5	58
7	2013	1	1	555	600	-5	913	854	19	B6	507	N516JB	EWR	FLL	158	1065	6	0
8	2013	1	1	557	600	-3	709	723	-14	EV	5708	N829AS	LGA	IAD	53	229	6	0
9	2013	1	1	557	600	-3	838	846	-8	B6	79	N593JB	JFK	MCO	140	944	6	0
10	2013	1	1	558	600	-2	753	745	8	AA	301	N3ALAA	LGA	ORD	138	733	6	0
11	2013	1	1	558	600	-2	849	851	-2	B6	49	N793JB	JFK	PBI	149	1028	6	0
12	2013	1	1	558	600	-2	853	856	-3	B6	71	N657JB	JFK	TPA	158	1005	6	0
13	2013	1	1	558	600	-2	924	917	7	UA	194	N29129	JFK	LAX	345	2475	6	0
14	2013	1	1	558	600	-2	923	937	-14	UA	1124	N53441	EWR	SFO	361	2565	6	0
15	2013	1	1	559	600	-1	941	910	31	AA	707	N3DUAA	LGA	DFW	257	1389	6	0
16	2013	1	1	559	559	0	702	706	-4	B6	1806	N708JB	JFK	BOS	44	187	5	59
17	2013	1	1	559	600	-1	854	902	-8	UA	1187	N76515	EWR	LAS	337	2227	6	0
18	2013	1	1	600	600	0	851	858	-7	B6	371	N595JB	LGA	FLL	152	1076	6	0

Module 10 exercise-4

Updated today; copy and paste new instructions from **class** repo!

What are reasons you would want to calculate summary information for **groups of observations**?

What are reasons you would **not** want to calculate summary information for **groups of observations**?

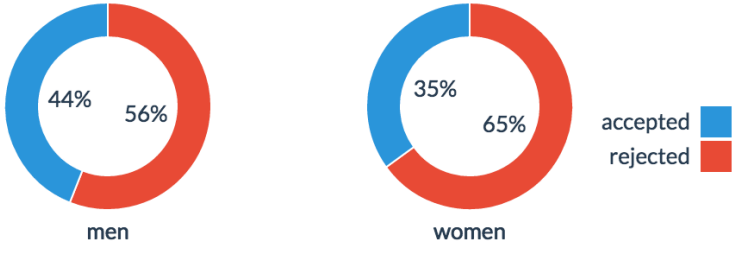
Aside: Simpson's Paradox

A paradox in which a trend appears in different groups but **disappears** when those groups are **combined**.

In 1973, the University of California-Berkeley was sued for sex discrimination. The numbers looked pretty incriminating: the graduate schools had just accepted 44% of male applicants but only 35% of female applicants. When researchers looked at the evidence, though, they uncovered something surprising:

> If the data are properly pooled...there is a small but statistically significant bias in favor of women.
>
> — (p. 403)

It was a textbook case of **Simpson's paradox**.



http://vudlab.com/simpsons/

summarize()

Generate a data frame that has an aggregation of a particular **column**.

```
# Make a data.frame
students <- data.frame(
  name = c('Mason', 'Tabi', 'Bryce', 'Ada', 'Bob','Filipe'),
  section = c('a','a','a','b','b','b'),
  math_exam1 = c(91, 82, 93, 100, 78, 91),
  math_exam2 = c(88, 79, 77, 99, 88, 93),
  spanish_exam1 = c(79, 88, 92, 83, 87, 77),
  spanish_exam2 = c(99, 92, 92, 82, 85, 95)
)

# Calculate summary stats
summarize(students,
          mean_math1 = mean(math_exam1),
          mean_math2 = mean(math_exam2),
          mean_math_scores=mean((math_exam1 + math_exam2) / 2)
)
```

group_by()

Break a into **groups of rows** for future processing.

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14

London	large	22
London	small	16

Beijing	large	121
Beijing	small	56

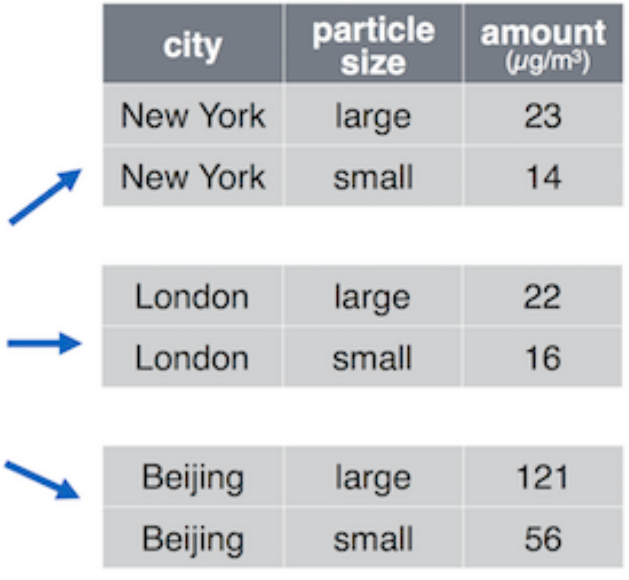
```
# Group data by city
city.grouped <- group_by(pollution, city)
```

Grouped data is like a factor!

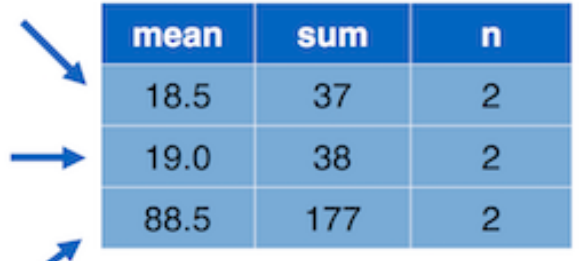
group_by()

Break a into **groups of rows** for future processing.

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56



city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56



mean	sum	n
18.5	37	2
19.0	38	2
88.5	177	2

```
# Get summary statistics BY CITY
city.grouped <- group_by(pollution, city) %>%
  summarize(  # first argument from pipe
    mean = mean(amount), sum = sum(amount), n = n()
  )
```

Module 10 exercise-5

Typo fixes today; copy and paste new instructions from class repo!

Airport Data

Why is this airport information stored in a separate data frame (`airports`)?

	faa	name	lat	lon	alt	tz	dst	tzone
1	04G	Lansdowne Airport	41.13047	-80.61958	1044	-5	A	America/New_York
2	06A	Moton Field Municipal Airport	32.46057	-85.68003	264	-6	A	America/Chicago
3	06C	Schaumburg Regional	41.98934	-88.10124	801	-6	A	America/Chicago
4	06N	Randall Airport	41.43191	-74.39156	523	-5	A	America/New_York
5	09J	Jekyll Island Airport	31.07447	-81.42778	11	-5	A	America/New_York
6	0A9	Elizabethton Municipal Airport	36.37122	-82.17342	1593	-5	A	America/New_York
7	0G6	Williams County Airport	41.46731	-84.50678	730	-5	A	America/New_York
8	0G7	Finger Lakes Regional Airport	42.88356	-76.78123	492	-5	A	America/New_York
9	0P2	Shoestring Aviation Airfield	39.79482	-76.64719	1000	-5	U	America/New_York
10	0S9	Jefferson County Intl	48.05381	-122.81064	108	-8	A	America/Los_Angeles
11	0W3	Harford County Airport	39.56684	-76.20240	409	-5	A	America/New_York
12	10C	Galt Field Airport	42.40289	-88.37511	875	-6	U	America/Chicago
13	17G	Port Bucyrus-Crawford County Airport	40.78156	-82.97481	1003	-5	A	America/New_York
14	19A	Jackson County Airport	34.17586	-83.56160	951	-5	U	America/New_York
15	1A3	Martin Campbell Field Airport	35.01581	-84.34683	1789	-5	A	America/New_York

Multiple Tables

Store data in multiple tables to keep data organized and avoid redundancies.

Can then **join** tables (columns) together into a single data frame to ask questions.

left_join()

Combines two tables into a data frame with the **columns** of both. `left_join()` adds **columns** from the ***right*** table to the **rows** of the ***left*** table.

songs

song	name
Across the Universe	John
Come Together	John
Hello, Goodbye	Paul
Peggy Sue	Buddy

+

artists

name	plays
George	sitar
John	guitar
Paul	bass
Ringo	drums

=

song	name	plays
Across the Universe	John	guitar
Come Together	John	guitar
Hello, Goodbye	Paul	bass
Peggy Sue	Buddy	<NA>

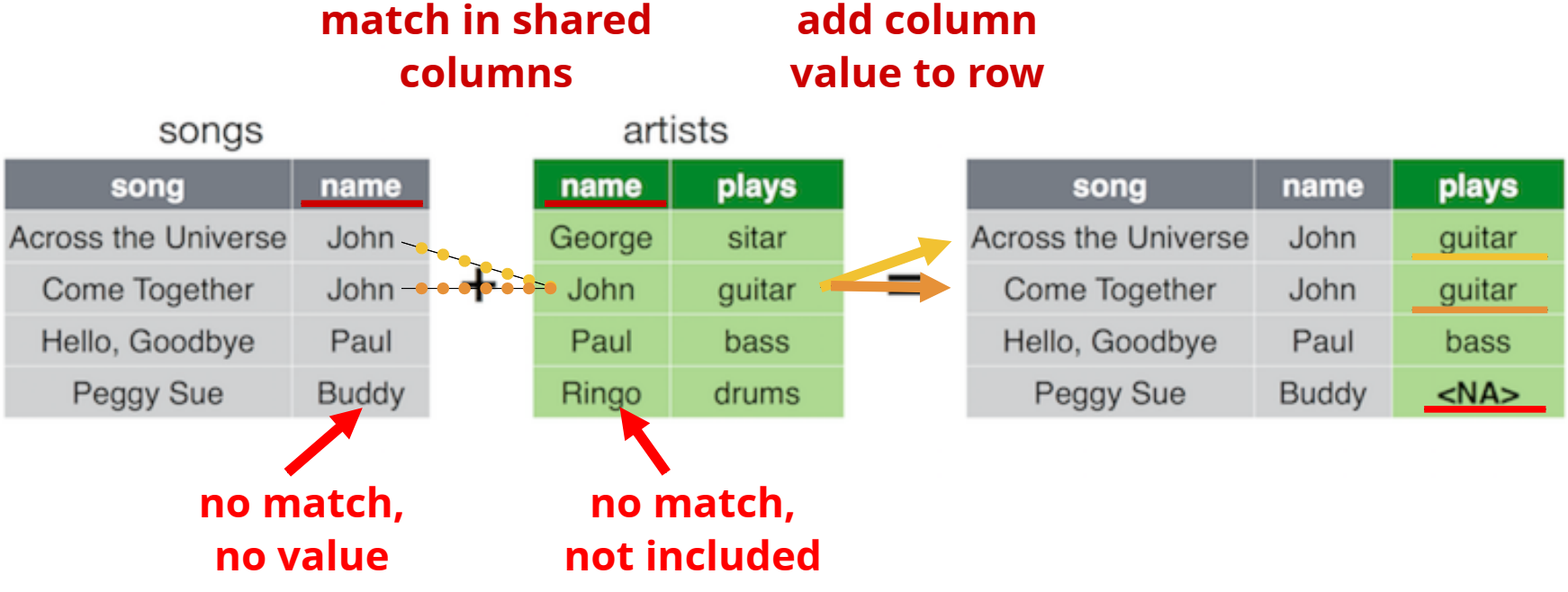
```
# Combine (join) songs and artists data frames
left_join(songs, artists)
```

"left" table

"right" table

Row Identifiers

Joins determine which ***data*** (column values) go in which rows by looking for **shared columns** and treating those as identifiers for "matching" rows. Note the resulting data frame contains **all columns** from both arguments.



Join Example

```r
# Table of contact information
student.contact <- data.frame(
  student.id = c(1, 2, 3, 4),  # id numbers
  email = c("mason@uw.edu", "tabi@uw.edu", "bryce@uw.edu", "ada@uw.edu")
)

# Table of information about majors
student.majors <- data.frame(
  student.id = c(1, 5, 4, 2),  # id numbers
  major = c('sociology', 'biology', 'math','informatics')
)

left_join(student.contact, student.majors)
```

	student.id	email
1	1	mason@uw.edu
2	2	tabi@uw.edu
3	3	bryce@uw.edu
4	4	ada@uw.edu

+

	student.id	major
1	1	sociology
2	5	biology
3	4	math
4	2	informatics

=

	student.id	email	major
1	1	mason@uw.edu	sociology
2	2	tabi@uw.edu	informatics
3	3	bryce@uw.edu	*NA*
4	4	ada@uw.edu	math

by

```r
# Table of contact information
student.contact <- data.frame(
  student.id = c(1, 2, 3, 4),  # id numbers
  email = c("mason@uw.edu", "tabi@uw.edu", "bryce@uw.edu", "ada@uw.edu")
)

# Table of information about majors
student.majors <- data.frame(
  id = c(1, 5, 4, 2),  # id numbers
  major = c('sociology', 'biology', 'math','informatics')
)

left_join(student.contact, student.majors, by = c("student.id" = "id"))
```

	student.id	email
1	1	mason@uw.edu
2	2	tabi@uw.edu
3	3	bryce@uw.edu
4	4	ada@uw.edu

+

	id	major
1	1	sociology
2	5	biology
3	4	math
4	2	informatics

=

	student.id	email	major
1	1	mason@uw.edu	sociology
2	2	tabi@uw.edu	informatics
3	3	bryce@uw.edu	*NA*
4	4	ada@uw.edu	math

Other Joins

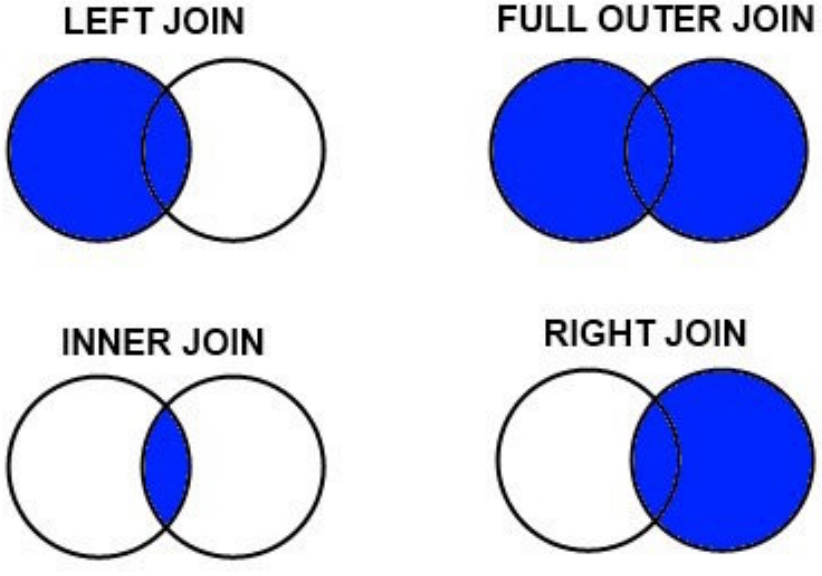
Use other joining functions to determine which **rows** should be in the resulting table (which contains *all columns*)

```
# rows from songs, cols from artists
left_join(songs, artists)

# rows from artists, cols from songs
right_join(songs, artists)

# rows matched in songs AND artists,
# columns from both
inner_join(songs, artists)

# all rows (songs OR artists),
# columns from both
full_join(songs, artists)
```



rows without a value for a column are given the value NA

Module 10 exercise-6

Action Items!

- Be comfortable with **module 9**
- Assignment 4 due ***Tuesday before class***

Tuesday: Downloading data with APIs
(modules available soon...)