PRINCIPLES OF DATA SCIENCE

JOHN P DICKERSON

Lecture #4 – 9/19/2018

CMSC641
Wednesdays
7pm – 9:30pm

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

ANNOUNCEMENTS

Project 1 is out!

- Announced on ELMS and Piazza
- https://github.com/JohnDickerson/cmsc641-fall2018/tree/master/project1
- Due date is October 3rd

Reminder: Weekly quizzes, due on Wednesdays at noon

"HOW DOES IMPORT WORK"

Python code is stored in module – simply put, a file full of Python code

A package is a directory (tree) full of modules that also contains a file called `__init.py__`

- Packages let you structure Python's module namespace
- E.g., `X.Y` is a submodule `Y` in a package named `X`

For one module to gain access to code in another module, it must import it

EXAMPLE

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

```
# Load (sub)module sound.effects.echo
import sound.effects.echo
# Must use full name to reference echo functions
sound.effects.echo.echofilter(input, output, delay=0.7)
```

https://docs.python.org/2/tutorial/modules.html

EXAMPLE

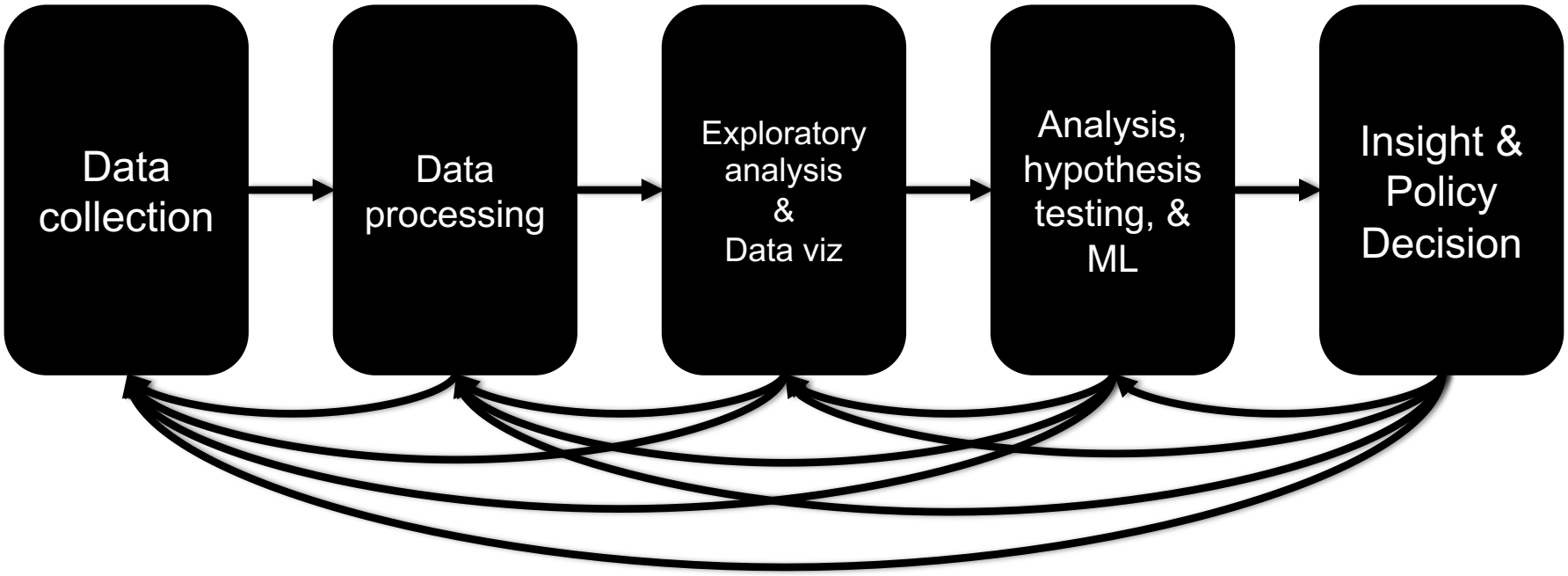
```
# Load (sub)module sound.effects.echo
import sound.effects.echo
# Must use full name to reference echo functions
sound.effects.echo.echofilter(input, output, delay=0.7)
```

```
# Load (sub)module sound.effects.echo
from sound.effects import echo
# No longer need the package prefix for functions in echo
echo.echofilter(input, output, delay=0.7)
```

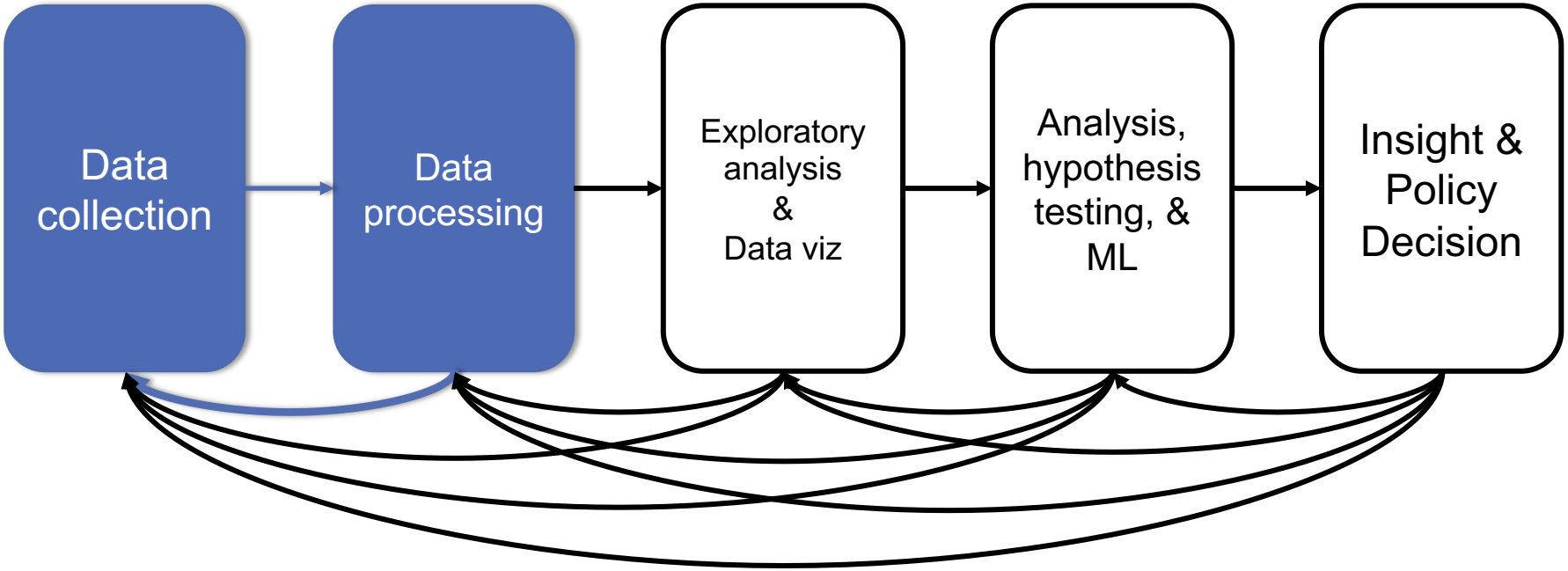
```
# Load a specific function directly
from sound.effects.echo import echofilter
# Can now use that function with no prefix
echofilter(input, output, delay=0.7)
```

https://docs.python.org/2/tutorial/modules.html

THE DATA LIFECYCLE



THE DATA LIFECYCLE



TODAY/NEXT CLASS

- **Tables**
  - Abstraction
  - Operations
- **Pandas**
- **Tidy Data**
- **SQL**

RECAP: TABLES

Special Column, called “Index”, or “ID”, or “Key”
Usually, no duplicates Allowed

Variables
(also called Attributes, or Columns, or Labels)

Observations, Rows, or Tuples

ID	age	wgt_kg	hgt_cm
1	12.2	42.3	145.1
2	11.0	40.8	143.8
3	15.6	65.3	165.3
4	35.1	84.2	185.8

1. SELECT/SLICING

Select only some of the rows, or some of the columns, or a combination

ID	age	wgt_kg	hgt_cm
1	12.2	42.3	145.1
2	11.0	40.8	143.8
3	15.6	65.3	165.3
4	35.1	84.2	185.8

Only columns ID and Age

ID	age
1	12.2
2	11.0
3	15.6
4	35.1

Only rows with wgt > 41

ID	age	wgt_kg	hgt_cm
1	12.2	42.3	145.1
3	15.6	65.3	165.3
4	35.1	84.2	185.8

Both

ID	age
1	12.2
3	15.6
4	35.1

2. AGGREGATE/REDUCE

Combine values across a column into a single value

ID	age	wgt_kg	hgt_cm
1	12.2	42.3	145.1
2	11.0	40.8	143.8
3	15.6	65.3	165.3
4	35.1	84.2	185.8

SUM → **73.9 232.6 640.0**

MAX → **35.1 84.2 185.8**

SUM(wgt_kg^2 - hgt_cm) → **14167.66**

What about ID/Index column?

- Usually not meaningful to aggregate across it
- May need to explicitly add an ID column

3. MAP

Apply a function to every row, possibly creating more or fewer columns

ID	Address
1	College Park, MD, 20742
2	Washington, DC, 20001
3	Silver Spring, MD 20901

ID	City	State	Zipcode
1	College Park	MD	20742
2	Washington	DC	20001
3	Silver Spring	MD	20901

Variations that allow one row to generate multiple rows in the output (sometimes called “flatmap”)

4. GROUP BY

Group tuples together by column/dimension

ID	A	B	C
1	foo	3	6.6
2	bar	2	4.7
3	foo	4	3.1
4	foo	3	8.0
5	bar	1	1.2
6	bar	2	2.5
7	foo	4	2.3
8	foo	3	8.0

A = foo

ID	B	C
1	3	6.6
3	4	3.1
4	3	8.0
7	4	2.3
8	3	8.0

A = bar

ID	B	C
2	2	4.7
5	1	1.2
6	2	2.5

4. GROUP BY

Group tuples together by column/dimension

ID	A	B	C
1	foo	3	6.6
2	bar	2	4.7
3	foo	4	3.1
4	foo	3	8.0
5	bar	1	1.2
6	bar	2	2.5
7	foo	4	2.3
8	foo	3	8.0

B = 1

ID	A	C
5	bar	1.2

B = 2

ID	A	C
2	bar	4.7
6	bar	2.5

B = 3

ID	A	C
1	foo	6.6
4	foo	8.0
8	foo	8.0

B = 4

ID	A	C
3	foo	3.1
7	foo	2.3

4. GROUP BY

Group tuples together by column/dimension

ID	A	B	C
1	foo	3	6.6
2	bar	2	4.7
3	foo	4	3.1
4	foo	3	8.0
5	bar	1	1.2
6	bar	2	2.5
7	foo	4	2.3
8	foo	3	8.0

A = bar, B = 1

ID	C
5	1.2

A = bar, B = 2

ID	C
2	4.7
6	2.5

A = foo, B = 3

ID	C
1	6.6
4	8.0
8	8.0

A = foo, B = 4

ID	C
3	3.1
7	2.3

5. GROUP BY AGGREGATE

Compute one aggregate

Per group

ID	A	B	C
1	foo	3	6.6
2	bar	2	4.7
3	foo	4	3.1
4	foo	3	8.0
5	bar	1	1.2
6	bar	2	2.5
7	foo	4	2.3
8	foo	3	8.0

Group by ‘B’
Sum on C

B = 1

ID	A	C
5	bar	1.2

B = 1

Sum (C)
1.2

B = 2

ID	A	C
2	bar	4.7
6	bar	2.5

B = 2

Sum (C)
7.2

B = 3

ID	A	C
1	foo	6.6
4	foo	8.0
8	foo	8.0

B = 3

Sum (C)
22.6

B = 4

ID	A	C
3	foo	3.1
7	foo	2.3

B = 4

Sum (C)
5.4

5. GROUP BY AGGREGATE

Final result usually seen

As a table

ID	A	B	C
1	foo	3	6.6
2	bar	2	4.7
3	foo	4	3.1
4	foo	3	8.0
5	bar	1	1.2
6	bar	2	2.5
7	foo	4	2.3
8	foo	3	8.0

Group by 'B'
Sum on C

B = 1

Sum (C)
1.2

B = 2

Sum (C)
7.2

B = 3

Sum (C)
22.6

B = 4

Sum (C)
5.4

B	SUM(C)
1	1.2
2	7.2
3	22.6
4	5.4

6. UNION/INTERSECTION/DIFFERENCE

Set operations – only if the two tables have identical attributes/columns

ID	A	B	C
1	foo	3	6.6
2	bar	2	4.7
3	foo	4	3.1
4	foo	3	8.0

U

ID	A	B	C
5	bar	1	1.2
6	bar	2	2.5
7	foo	4	2.3
8	foo	3	8.0

→

ID	A	B	C
1	foo	3	6.6
2	bar	2	4.7
3	foo	4	3.1
4	foo	3	8.0
5	bar	1	1.2
6	bar	2	2.5
7	foo	4	2.3
8	foo	3	8.0

Similarly Intersection and Set Difference manipulate tables as Sets

IDs may be treated in different ways, resulting in somewhat different behaviors

7. MERGE OR JOIN

Combine rows/tuples across two tables if they have the same key

ID	A	B
1	foo	3
2	bar	2
3	foo	4
4	foo	3

⋈

ID	C
1	1.2
2	2.5
3	2.3
5	8.0

ID	A	B	C
1	foo	3	1.2
2	bar	2	2.5
3	foo	4	2.3

What about IDs not present in both tables?

- **Often need to keep them around**
- **Can "pad" with NaN**

7. MERGE OR JOIN

Combine rows/tuples across two tables if they have the same key

Outer joins can be used to ”pad” IDs that don’t appear in both tables

Three variants: LEFT, RIGHT, FULL

SQL Terminology -- Pandas has these operations as well

ID	A	B
1	foo	3
2	bar	2
3	foo	4
4	foo	3

ID	C
1	1.2
2	2.5
3	2.3
5	8.0

ID	A	B	C
1	foo	3	1.2
2	bar	2	2.5
3	foo	4	2.3
4	foo	3	NaN
5	NaN	NaN	8.0

SUMMARY

- **Tables: A simple, common abstraction**
  - Subsumes a set of “strings” – a common input
- **Operations**
  - Select, Map, Aggregate, Reduce, Join/Merge, Union/Concat, Group By
- **In a given system/language, the operations may be named differently**
  - E.g., SQL uses “join”, whereas Pandas uses “merge”
- **Subtle variations in the definitions, especially for more complex operations**

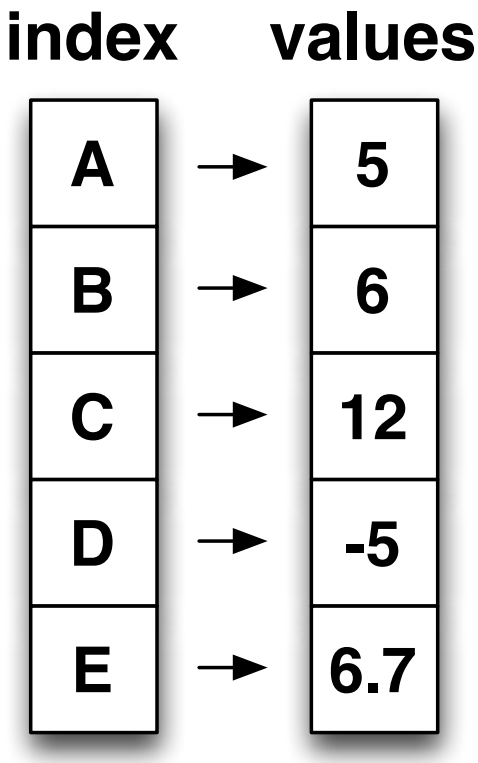
TODAY/NEXT CLASS

- **Tables**
  - Abstraction
  - Operations
- **Pandas**
- **Tidy Data**
- **SQL and Relational Databases**

PANDAS: HISTORY

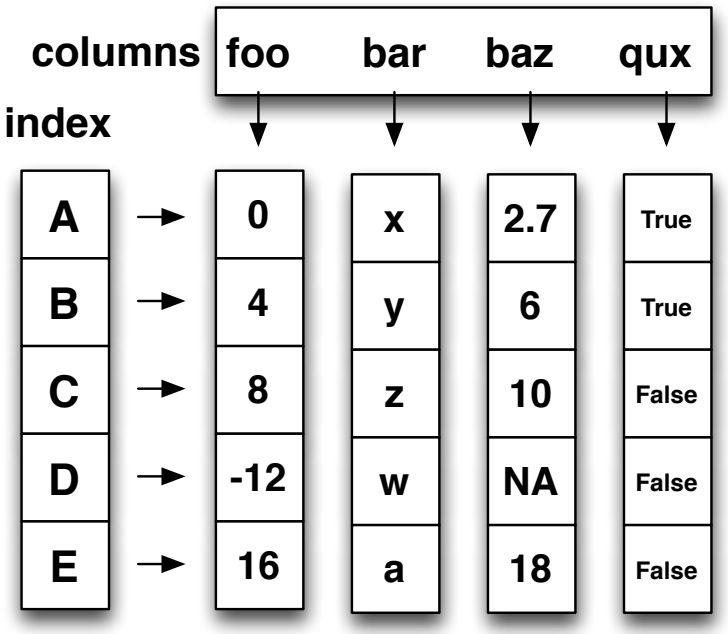
- **Written by: Wes McKinney**
  - Started in 2008 to get a high-performance, flexible tool to perform quantitative analysis on financial data
- **Highly optimized for performance, with critical code paths written in Cython or C**
- **Key constructs:**
  - Series (like a NumPy Array)
  - DataFrame (like a Table or Relation, or R data.frame)
- **Foundation for Data Wrangling and Analysis in Python**

PANDAS: SERIES



- Subclass of numpy.ndarray
- Data: any type
- Index labels need not be ordered
- Duplicates possible but result in reduced functionality

PANDAS: DATAFRAME



- **Each column can have a different type**
- **Row and Column index**
- **Mutable size: insert and delete columns**

- **Note the use of word "index" for what we called "key"**
  - Relational databases use "index" to mean something else
- **Non-unique index values allowed**
  - May raise an exception for some operations

HIERARCHICAL INDEXES

Sometimes more intuitive organization of the data

Makes it easier to understand and analyze higher-dimensional data

e.g., instead of 3-D array, may only need a 2-D array

day		Fri	Sat	Sun	Thur
sex	smoker				
Female	No	3.125	2.725	3.329	2.460
	Yes	2.683	2.869	3.500	2.990
Male	No	2.500	3.257	3.115	2.942
	Yes	2.741	2.879	3.521	3.058

```
first  second
bar    one       0.469112
       two      -0.282863
baz    one      -1.509059
       two      -1.135632
foo    one       1.212112
       two      -0.173215
qux    one       0.119209
       two      -1.044236
dtype: float64
```

ESSENTIAL FUNCTIONALITY

Reindexing to change the index associated with a DataFrame

- Common usage to interpolate, fill in missing values

```
In [84]: obj3 = Series(['blue', 'purple', 'yellow'], index=[0, 2, 4])

In [85]: obj3.reindex(range(6), method='ffill')
Out[85]:
0      blue
1      blue
2    purple
3    purple
4    yellow
5    yellow
```

From: Python for Data Analysis; Wes McKinney

ESSENTIAL FUNCTIONALITY

“drop” to delete entire rows or columns

Indexing, Selection, Filtering: very similar to NumPy

Arithmetic Operations

- Result index union of the two input indexes
- Options to do “fill” while doing these operations

```
In [128]: s1
Out[128]:
a    7.3
c   -2.5
d    3.4
e    1.5
```

```
In [129]: s2
Out[129]:
a   -2.1
c    3.6
e   -1.5
f    4.0
g    3.1
```

```
In [130]: s1 + s2
Out[130]:
a    5.2
c    1.1
d    NaN
e    0.0
f    NaN
g    NaN
```

FUNCTION APPLICATION AND MAPPING

```
In [158]: frame = DataFrame(np.random.randn(4, 3), columns=list('bde'),
   .....:                   index=['Utah', 'Ohio', 'Texas', 'Oregon'])
```

```
In [159]: frame
Out[159]:
               b         d         e
Utah   -0.204708  0.478943 -0.519439
Ohio   -0.555730  1.965781  1.393406
Texas   0.092908  0.281746  0.769023
Oregon  1.246435  1.007189 -1.296221
```

```
In [160]: np.abs(frame)
Out[160]:
               b         d         e
Utah    0.204708  0.478943  0.519439
Ohio    0.555730  1.965781  1.393406
Texas   0.092908  0.281746  0.769023
Oregon  1.246435  1.007189  1.296221
```

```
In [161]: f = lambda x: x.max() - x.min()
```

```
In [162]: frame.apply(f)
Out[162]:
b    1.802165
d    1.684034
e    2.689627
```

```
In [163]: frame.apply(f, axis=1)
Out[163]:
Utah      0.998382
Ohio      2.521511
Texas     0.676115
Oregon    2.542656
```

From: Python for Data Analysis; Wes McKinney

SORTING AND RANKING

```
In [169]: obj = Series(range(4), index=['d', 'a', 'b', 'c'])

In [170]: obj.sort_index()
Out[170]:
a    1
b    2
c    3
d    0
```

```
In [187]: frame = DataFrame({'b': [4.3, 7, -3, 2], 'a': [0, 1, 0, 1],
   .....:                    'c': [-2, 5, 8, -2.5]})

In [188]: frame
Out[188]:
   a    b    c
0  0  4.3 -2.0
1  1  7.0  5.0
2  0 -3.0  8.0
3  1  2.0 -2.5
```

```
In [189]: frame.rank(axis=1)
Out[189]:
   a  b  c
0  2  3  1
1  1  3  2
2  2  1  3
3  2  3  1
```

From: Python for Data Analysis; Wes McKinney

DESCRIPTIVE AND SUMMARY STATISTICS

Table 5-10. Descriptive and summary statistics

Method	Description
`count`	Number of non-NA values
`describe`	Compute set of summary statistics for Series or each DataFrame column
`min, max`	Compute minimum and maximum values
`argmin, argmax`	Compute index locations (integers) at which minimum or maximum value obtained, respectively
`idxmin, idxmax`	Compute index values at which minimum or maximum value obtained, respectively
`quantile`	Compute sample quantile ranging from 0 to 1
`sum`	Sum of values
`mean`	Mean of values
`median`	Arithmetic median (50% quantile) of values
`mad`	Mean absolute deviation from mean value
`var`	Sample variance of values
`std`	Sample standard deviation of values
`skew`	Sample skewness (3rd moment) of values
`kurt`	Sample kurtosis (4th moment) of values
`cumsum`	Cumulative sum of values
`cummin, cummax`	Cumulative minimum or maximum of values, respectively
`cumprod`	Cumulative product of values
`diff`	Compute 1st arithmetic difference (useful for time series)
`pct_change`	Compute percent changes

From: Python for Data Analysis; Wes McKinney

CREATING DATAFRAMES

Directly from Dict or Series

From a Comma-Separated File – CSV file

- pandas.read_csv()
- Can infer headers/column names if present, otherwise may want to reindex

From an Excel File

- pandas.read_excel()

From a Database using SQL (see the reading for an example)

From Clipboard, URL, Google Analytics, …

…

From: Python for Data Analysis; Wes McKinney

MORE…

Unique values, Value counts

Correlation and Covariance

Functions for handling missing data – in a few classes

- dropna(), fillna()

Broadcasting

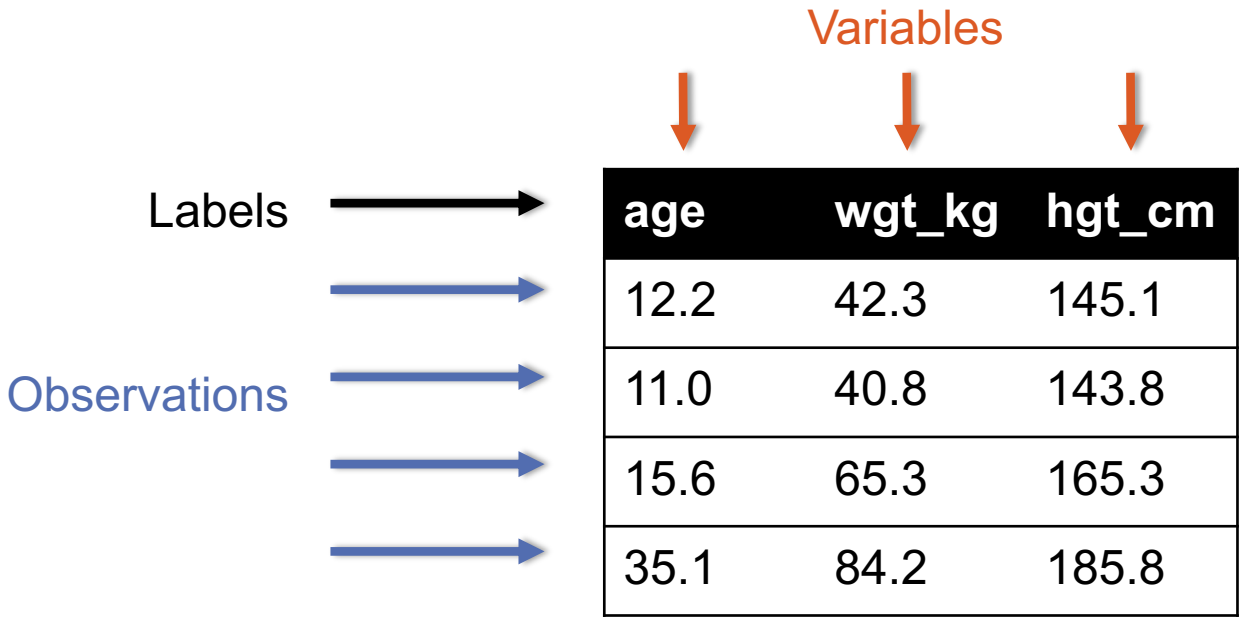
Pivoting

We will see some of these as we discuss data wrangling, cleaning, etc.

TODAY/NEXT CLASS

- **Tables**
  - Abstraction
  - Operations
- **Pandas**
- **Tidy Data**
- **SQL and Relational Databases**

TIDY DATA



age	wgt_kg	hgt_cm
12.2	42.3	145.1
11.0	40.8	143.8
15.6	65.3	165.3
35.1	84.2	185.8

But also:

- **Names of files/DataFrames = description of one dataset**
- **Enforce one data type per dataset (ish)**

EXAMPLE

Variable: measure or attribute:

- age, weight, height, sex

Value: measurement of attribute:

- 12.2, 42.3kg, 145.1cm, M/F

Observation: all measurements for an object

- A specific person is [12.2, 42.3, 145.1, F]

TIDYING DATA I

Name	Treatment A	Treatment B
John Smith	-	2
Jane Doe	16	11
Mary Johnson	3	1

??????????????

Name	Treatment A	Treatment B	Treatment C	Treatment D
John Smith	-	2	-	-
Jane Doe	16	11	4	1
Mary Johnson	3	1	-	2

??????????????

Thanks to http://jeannicholashould.com/tidy-data-in-python.html

TIDYING DATA II

Next class

Name	Treatment	Result
John Smith	A	-
John Smith	B	2
John Smith	C	-
John Smith	D	-
Jane Doe	A	16
Jane Doe	B	11
Jane Doe	C	4
Jane Doe	D	1
Mary Johnson	A	3
Mary Johnson	B	1
Mary Johnson	C	-
Mary Johnson	D	2

MELTING DATA I

religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Dont know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovahs Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

??????????????

MELTING DATA II

```
f_df = pd.melt(df,
               ["religion"],
               var_name="income",
               value_name="freq")
f_df = f_df.sort_values(by=["religion"])
f_df.head(10)
```

religion	income	freq
Agnostic	<$10k	27
Agnostic	$30-40k	81
Agnostic	$40-50k	76
Agnostic	$50-75k	137
Agnostic	$10-20k	34
Agnostic	$20-30k	60
Atheist	$40-50k	35
Atheist	$20-30k	37
Atheist	$10-20k	27
Atheist	$30-40k	52

MORE COMPLICATED EXAMPLE

Billboard Top 100 data for songs, covering their position on the Top 100 for 75 weeks, with two "messy" bits:

- Column headers for each of the 75 weeks
- If a song didn't last 75 weeks, those columns have are null

year	artist.inverted	track	time	genre	date.entered	date.peaked	x1st.week	x2nd.week	...
2000	Destiny's Child	Independent Women Part I	3:38	Rock	2000-09-23	2000-11-18	78	63.0	...
2000	Santana	Maria, Maria	4:18	Rock	2000-02-12	2000-04-08	15	8.0	...
2000	Savage Garden	I Knew I Loved You	4:07	Rock	1999-10-23	2000-01-29	71	48.0	...
2000	Madonna	Music	3:45	Rock	2000-08-12	2000-09-16	41	23.0	...
2000	Aguilera, Christina	Come On Over Baby	3:38	Rock	2000-08-05	2000-10-14	57	47.0	...
2000	Janet	Doesn't Really Matter	4:17	Rock	2000-06-17	2000-08-26	59	52.0	...

Messy columns!

Thanks to http://jeannicholashould.com/tidy-data-in-python.html

MORE COMPLICATED EXAMPLE

```
# Keep identifier variables
id_vars = ["year",
           "artist.inverted",
           "track",
           "time",
           "genre",
           "date.entered",
           "date.peaked"]

# Melt the rest into week and rank columns
df = pd.melt(frame=df,
             id_vars=id_vars,
             var_name="week",
             value_name="rank")
```

Creates one row per week, per record, with its rank

MORE COMPLICATED EXAMPLE

```
# Formatting
df["week"] = df['week'].str.extract('(\d+)',
                                    expand=False).astype(int)
df["rank"] = df["rank"].astype(int)
```

[…, "x2nd.week", 63.0] → […, 2, 63]

```
# Cleaning out unnecessary rows
df = df.dropna()

# Create "date" columns
df['date'] = pd.to_datetime(
                df['date.entered']) +
                pd.to_timedelta(df['week'], unit='w') –
                pd.DateOffset(weeks=1)
```

MORE COMPLICATED EXAMPLE

```
# Ignore now-redundant, messy columns
df = df[["year",
         "artist.inverted",
         "track",
         "time",
         "genre",
         "week",
         "rank",
         "date"]]

df = df.sort_values(ascending=True,
 by=["year","artist.inverted","track","week","rank"])

# Keep tidy dataset for future usage
billboard = df

df.head(10)
```

MORE COMPLICATED EXAMPLE

year	artist.inverted	track	time	genre	week	rank	date
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	1	87	2000-02-26
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	2	82	2000-03-04
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	3	72	2000-03-11
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	4	77	2000-03-18
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	5	87	2000-03-25
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	6	94	2000-04-01
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	7	99	2000-04-08
2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	1	91	2000-09-02
2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	2	87	2000-09-09
2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	3	92	2000-09-16

??????????????

MORE TO DO?

Column headers are values, not variable names?

- Good to go!

Multiple variables are stored in one column?

- Maybe (depends on if genre text in raw data was multiple)

Variables are stored in both rows and columns?

- Good to go!

Multiple types of observational units in the same table?

- Good to go! One row per song's week on the Top 100.

A single observational unit is stored in multiple tables?

- Don't do this!

Repetition of data?

- Lots! Artist and song title's text names. Which leads us to …

TODAY/NEXT CLASS

- **Tables**
  - Abstraction
  - Operations
- **Pandas**
- **Tidy Data**
- **SQL and Relational Databases**

CONTINUING TODAY'S LECTURE

Relational data:

- What is a relation, and how do they interact?

Querying databases:

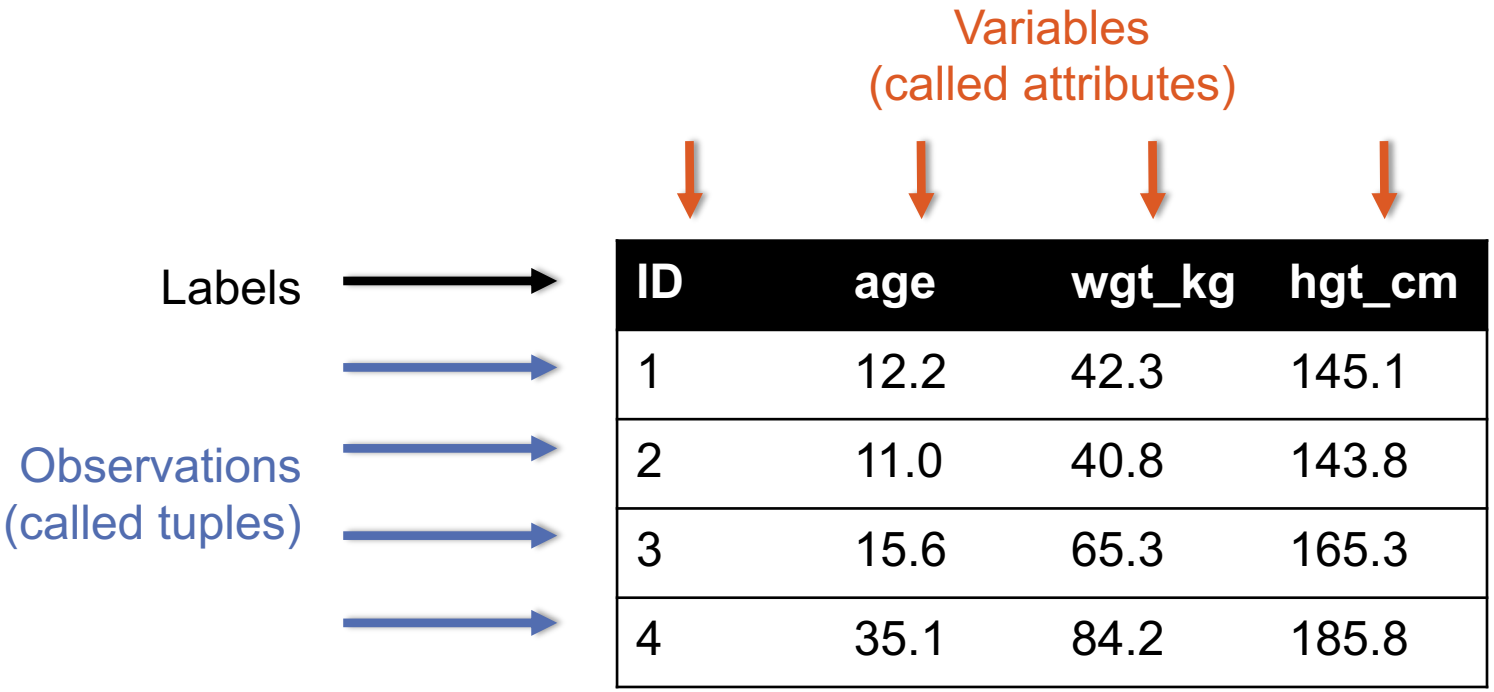
- SQL
- SQLite
- How does this relate to `pandas`?

Joins

Thanks to Zico Kolter for some structure for this lecture!

RELATION

Simplest relation: a table aka tabular data full of unique tuples



ID	age	wgt_kg	hgt_cm
1	12.2	42.3	145.1
2	11.0	40.8	143.8
3	15.6	65.3	165.3
4	35.1	84.2	185.8

PRIMARY KEYS

ID	age	wgt_kg	hgt_cm	nat_id
1	12.2	42.3	145.1	1
2	11.0	40.8	143.8	1
3	15.6	65.3	165.3	2
4	35.1	84.2	185.8	1
5	18.1	62.2	176.2	3
6	19.6	82.1	180.1	1

ID	Nationality
1	USA
2	Canada
3	Mexico

The primary key is a unique identifier for every tuple in a relation

- **Each tuple has exactly one primary key**

AREN'T THESE CALLED "INDEXES"?

Yes, in Pandas; but not in the database world

For most databases, an "index" is a data structure used to speed up retrieval of specific tuples

For example, to find all tuples with nat_id = 2:

- We can either scan the table – O(N)
- Or use an "index" (e.g., binary tree) – O(log N)

FOREIGN KEYS

ID	age	wgt_kg	hgt_cm	nat_id
1	12.2	42.3	145.1	1
2	11.0	40.8	143.8	1
3	15.6	65.3	165.3	2
4	35.1	84.2	185.8	1
5	18.1	62.2	176.2	3
6	19.6	82.1	180.1	1

ID	Nationality
1	USA
2	Canada
3	Mexico

Foreign keys are attributes (columns) that point to a different table's primary key

- **A table can have multiple foreign keys**

RELATION SCHEMA

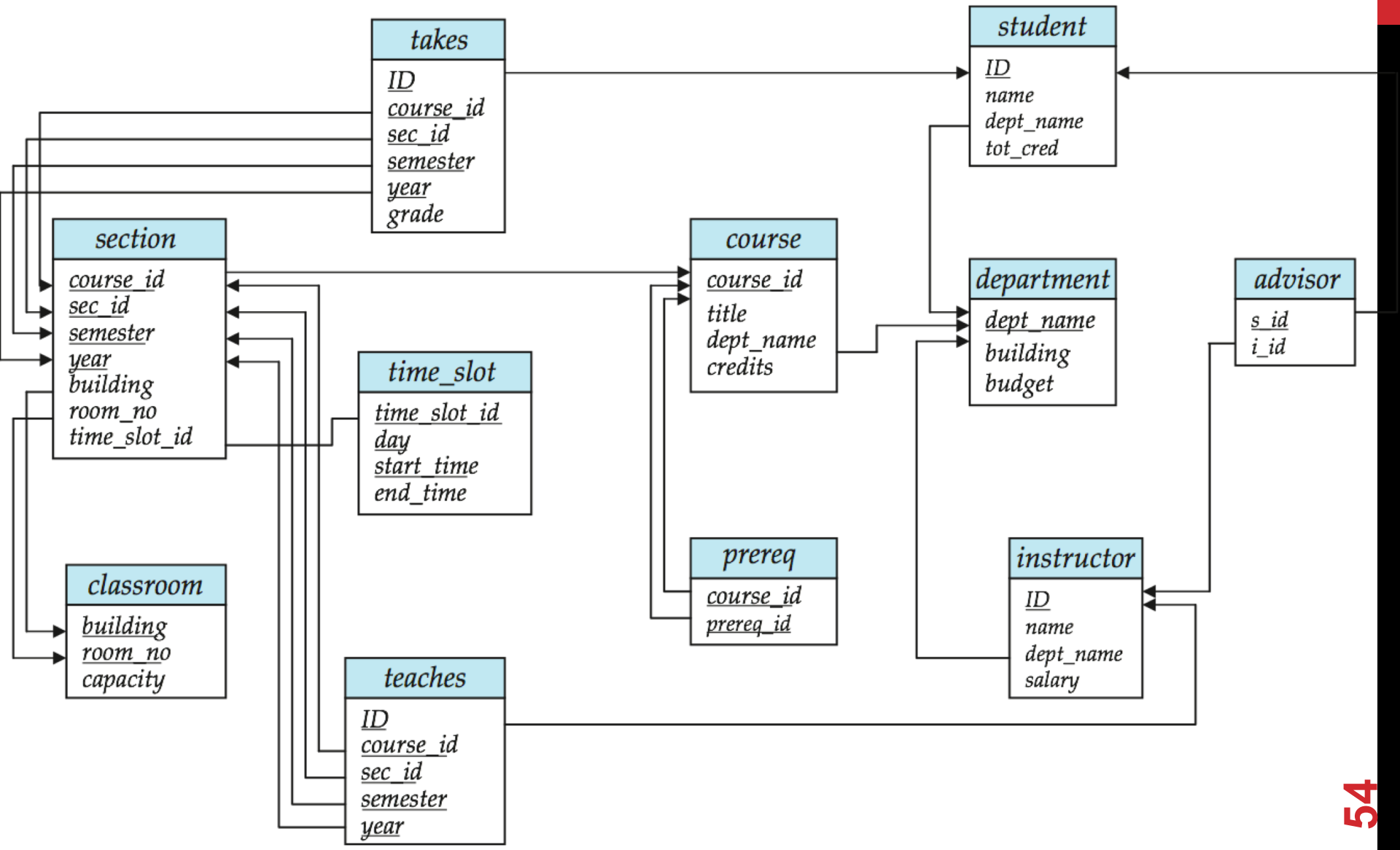
A list of all the attribute names, and their *domains*

```
create table department
    (dept_name varchar(20),
     building varchar(15),
     budget numeric(12,2) check (budget > 0),
     primary key (dept_name)
    );
```

SQL Statements To create Tables

```
create table instructor (
     ID         char(5),
     name       varchar(20) not null,
     dept_name  varchar(20),
     salary     numeric(8,2),
     primary key (ID),
     foreign key (dept_name) references department
)
```

SCHEMA DIAGRAMS



SEARCHING FOR ELEMENTS

Find all people with nationality Canada (nat_id = 2):

??????????????

ID	age	wgt_kg	hgt_cm	nat_id
1	12.2	42.3	145.1	1
2	11.0	40.8	143.8	1
3	15.6	65.3	165.3	2
4	35.1	84.2	185.8	1
5	18.1	62.2	176.2	3
6	19.6	82.1	180.1	1

O(n) 

(DATABASE) INDEXES

Like a hidden sorted map of references to a specific attribute (column) in a table; allows O(log n) lookup instead of O(n)

loc	ID	age	wgt_kg	hgt_cm	nat_id
0	1	12.2	42.3	145.1	1
128	2	11.0	40.8	143.8	2
256	3	15.6	65.3	165.3	2
384	4	35.1	84.2	185.8	1
512	5	18.1	62.2	176.2	3
640	6	19.6	82.1	180.1	1

nat_id	locs
1	0, 384, 640
2	128, 256
3	512

(DATABASE) INDEXES

Actually implemented with data structures like B-trees

- (Take courses like CMSC642, or CMSC424)

But: indexes are not free

- Takes memory to store
- Takes time to build
- Takes time to update (add/delete a row, update the column)

But, but: one index is (mostly) free

- Index will be built automatically on the primary key

Think before you build/maintain an index on other attributes!

RELATIONSHIPS

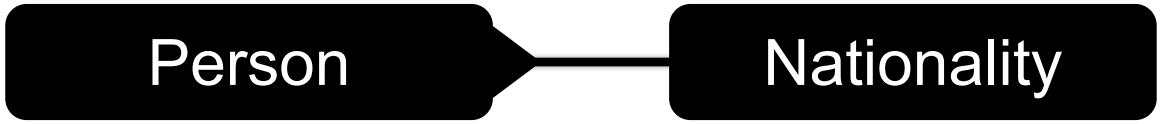
Primary keys and foreign keys define interactions between different tables aka entities. Four types:

- One-to-one
- One-to-one-or-none
- One-to-many and many-to-one
- Many-to-many

Connects (one, many) of the rows in one table to (one, many) of the rows in another table

ONE-TO-MANY & MANY-TO-ONE

One person can have one nationality in this example, but one nationality can include many people.



ID	age	wgt_kg	hgt_cm	nat_id
1	12.2	42.3	145.1	1
2	11.0	40.8	143.8	1
3	15.6	65.3	165.3	2
4	35.1	84.2	185.8	1
5	18.1	62.2	176.2	3
6	19.6	82.1	180.1	1

ID	Nationality
1	USA
2	Canada
3	Mexico

ONE-TO-ONE

Two tables have a one-to-one relationship if every tuple in the first tables corresponds to exactly one entry in the other

In general, you won't be using these (why not just merge the rows into one table?) unless:

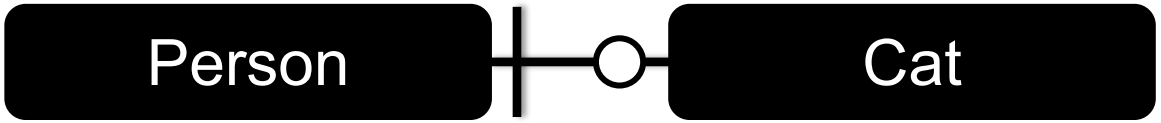
- Split a big row between SSD and HDD or distributed
- Restrict access to part of a row (some DBMSs allow column-level access control, but not all)
- Caching, partitioning, & serious stuff: take CMSC642, or 424

ONE-TO-ONE-OR-NONE

Say we want to keep track of people's cats:

Person ID	Cat1	Cat2
1	Chairman Meow	Fuzz Aldrin
4	Anderson Pooper	Meowly Cyrus
5	Gigabyte	Megabyte

People with IDs 2 and 3 do not own cats*, and are not in the table. Each person has at most one entry in the table.



Is this data tidy?

*nor do they have hearts, apparently.

MANY-TO-MANY

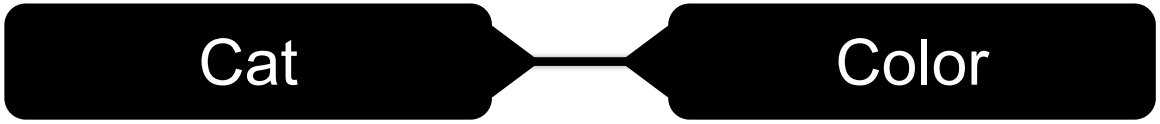
Say we want to keep track of people's cats' colorings:

ID	Name
1	Megabyte
2	Meowly Cyrus
3	Fuzz Aldrin
4	Chairman Meow
5	Anderson Pooper
6	Gigabyte

Cat ID	Color ID	Amount
1	1	50
1	2	50
2	2	20
2	4	40
2	5	40
3	1	100

One column per color, too many columns, too many nulls

Each cat can have many colors, and each color many cats



ASSOCIATIVE TABLES

Cats

ID	Name
1	Megabyte
2	Meowly Cyrus
3	Fuzz Aldrin
4	Chairman Meow
5	Anderson Pooper
6	Gigabyte

Cat ID	Color ID	Amount
1	1	50
1	2	50
2	2	20
2	4	40
2	5	40
3	1	100

Colors

ID	Name
1	Black
2	Brown
3	White
4	Orange
5	Neon Green
6	Invisible

Primary key ??????????

- [Cat ID, Color ID] (+ [Color ID, Cat ID], case-dependent)

Foreign key(s) ??????????

- Cat ID and Color ID

ASIDE: PANDAS

So, this kinda feels like pandas …

- And pandas kinda feels like a relational data system …

Pandas is not strictly a relational data system:

- No notion of primary / foreign keys

It does have indexes (and multi-column indexes):

- pandas.Index: ordered, sliceable set storing axis labels
- pandas.MultiIndex: hierarchical index

Rule of thumb: do heavy, rough lifting at the relational DB level, then fine-grained slicing and dicing and viz with pandas

SQLITE

On-disk relational database management system (RDMS)

- Applications connect directly to a file

Most RDMSs have applications connect to a server:

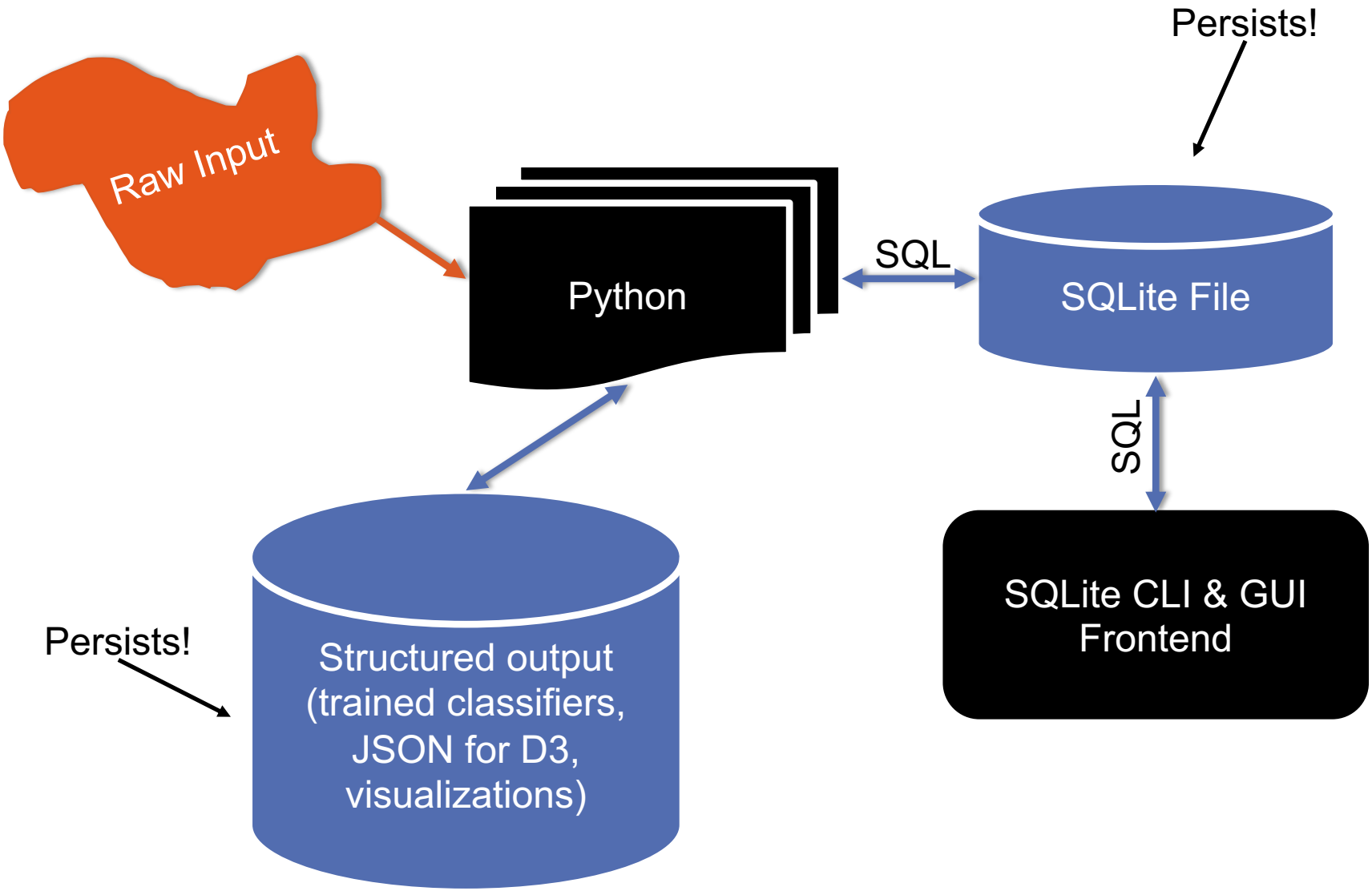
- Advantages include greater concurrency, less restrictive locking
- Disadvantages include, for this class, setup time ☺

Installation:

- `conda install -c anaconda sqlite`
- (Should come preinstalled, I think?)

All interactions use Structured Query Language (SQL)

HOW A RELATIONAL DB FITS INTO YOUR WORKFLOW



CRASH COURSE IN SQL (IN PYTHON)

```
import sqlite3

# Create a database and connect to it
conn = sqlite3.connect("cmsc641.db")
cursor = conn.cursor()

# do cool stuff
conn.close()
```

Cursor: temporary work area in system memory for manipulating SQL statements and return values

If you do not close the connection (`conn.close()`), any outstanding transaction is rolled back

- (More on this in a bit.)

CRASH COURSE IN SQL (IN PYTHON)

```
# Make a table
cursor.execute("""
CREATE TABLE cats (
    id INTEGER PRIMARY KEY,
    name TEXT
)""")
```

?????????

id	name

cats

Capitalization doesn't matter for SQL reserved words

- **SELECT = select = SeLeCt**

Rule of thumb: capitalize keywords for readability

CRASH COURSE IN SQL (IN PYTHON)

```
# Insert into the table
cursor.execute("INSERT INTO cats VALUE (1, 'Megabyte')")
cursor.execute("INSERT INTO cats VALUE (2, 'Meowly Cyrus')")
cursor.execute("INSERT INTO cats VALUE (3, 'Fuzz Aldrin')")
conn.commit()
```

id	name
1	Megabyte
2	Meowly Cyrus
3	Fuzz Aldrin

```
# Delete row(s) from the table
cursor.execute("DELETE FROM cats WHERE id == 2");
conn.commit()
```

id	name
1	Megabyte
3	Fuzz Aldrin

CRASH COURSE IN SQL (IN PYTHON)

```
# Read all rows from a table
for row in cursor.execute("SELECT * FROM cats"):
    print(row)
```

```
# Read all rows into pandas DataFrame
pd.read_sql_query("SELECT * FROM cats", conn, index_col="id")
```

id	name
1	Megabyte
3	Fuzz Aldrin

index_col="id": treat column with label "id" as an index

index_col=1: treat column #1 (i.e., "name") as an index

(Can also do multi-indexing.)

JOINING DATA

A join operation merges two or more tables into a single relation. Different ways of doing this:

- Inner
- Left
- Right
- Full Outer

Join operations are done on columns that explicitly link the tables together

INNER JOINS

id	name
1	Megabyte
2	Meowly Cyrus
3	Fuzz Aldrin
4	Chairman Meow
5	Anderson Pooper
6	Gigabyte

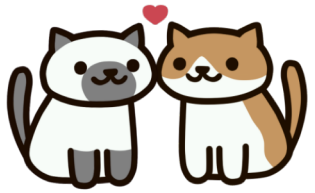
`cats`

cat_id	last_visit
1	02-16-2017
2	02-14-2017
5	02-03-2017

`visits`

Inner join returns merged rows that share the same value in the column they are being joined on (`id` and `cat_id`).

id	name	last_visit
1	Megabyte	02-16-2017
2	Meowly Cyrus	02-14-2017
5	Anderson Pooper	02-03-2017



INNER JOINS

```
# Inner join in pandas
df_cats = pd.read_sql_query("SELECT * from cats", conn)
df_visits = pd.read_sql_query("SELECT * from visits", conn)
df_cats.merge(df_visits, how = "inner",
              left_on = "id", right_on = "cat_id")
```

```
# Inner join in SQL / SQLite via Python
cursor.execute("""
                SELECT
                    *
                FROM
                    cats, visits
                WHERE
                    cats.id == visits.cat_id
                """)
```

LEFT JOINS

Inner joins are the most common type of joins (get results that appear in both tables)

Left joins: all the results from the left table, only some matching results from the right table

Left join (`cats`, `visits`) on (`id`, `cat_id`) ??????????

id	name	last_visit
1	Megabyte	02-16-2017
2	Meowly Cyrus	02-14-2017
3	Fuzz Aldrin	NULL
4	Chairman Meow	NULL
5	Anderson Pooper	02-03-2017
6	Gigabyte	NULL

RIGHT JOINS

Take a guess!

Right join
(`cats`, `visits`)
on
(`id`, `cat_id`)
`??????????`

id	name
1	Megabyte
2	Meowly Cyrus
3	Fuzz Aldrin
4	Chairman Meow
5	Anderson Pooper
6	Gigabyte

`cats`

cat_id	last_visit
1	02-16-2017
2	02-14-2017
5	02-03-2017
7	02-19-2017
12	02-21-2017

`visits`

id	name	last_visit
1	Megabyte	02-16-2017
2	Meowly Cyrus	02-14-2017
5	Anderson Pooper	02-03-2017
7	NULL	02-19-2017
12	NULL	02-21-2017

LEFT/RIGHT JOINS

```
# Left join in pandas
df_cats.merge(df_visits, how = “left”,
              left_on = “id”, right_on = ”cat_id”)
```

```
# Left join in SQL / SQLite via Python
cursor.execute(“SELECT * FROM cats LEFT JOIN visits ON
                    cats.id == visits.cat_id”)
```

```
# Right join in pandas
df_cats.merge(df_visits, how = “right”,
              left_on = “id”, right_on = ”cat_id”)
```

```
# Right join in SQL / SQLite via Python
☹
```

FULL OUTER JOIN

Combines the left and the right join ??????????

id	name	last_visit
1	Megabyte	02-16-2017
2	Meowly Cyrus	02-14-2017
3	Fuzz Aldrin	NULL
4	Chairman Meow	NULL
5	Anderson Pooper	02-03-2017
6	Gigabyte	NULL
7	NULL	02-19-2017
12	NULL	02-21-2017

```
# Outer join in pandas
df_cats.merge(df_visits, how = "outer",
              left_on = "id", right_on = "cat_id")
```

GOOGLE IMAGE SEARCH ONE SLIDE SQL JOIN VISUAL

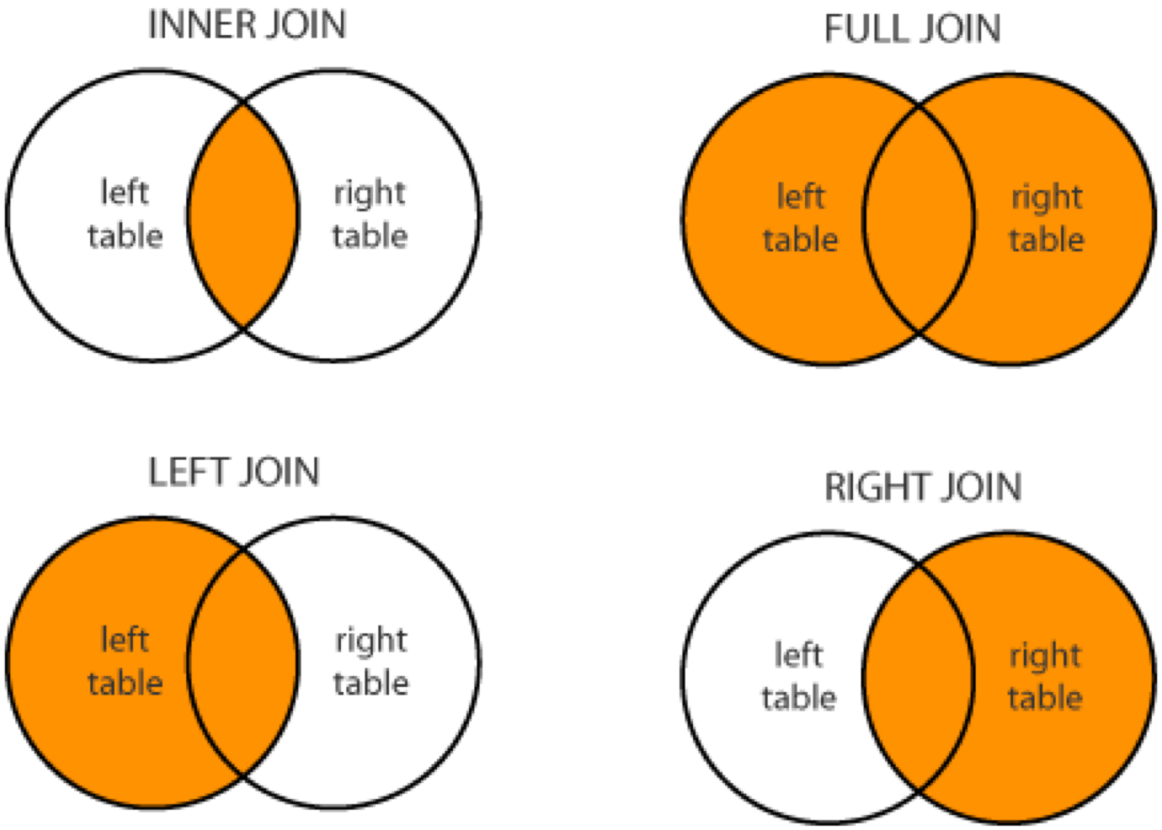


Image credit: http://www.dofactory.com/sql/join

GROUP BY AGGREGATES

```
SELECT nat_id, AVG(age) as average_age
FROM persons GROUP BY nat_id
```

ID	age	wgt_kg	hgt_cm	nat_id
1	12.2	42.3	145.1	1
2	11.0	40.8	143.8	1
3	15.6	65.3	165.3	2
4	35.1	84.2	185.8	1
5	18.1	62.2	176.2	3
6	19.6	82.1	180.1	1

nat_id	average_age
1	19.48
2	15.6
3	18.1

RAW SQL IN PANDAS

If you "think in SQL" already, you'll be fine with pandas:

- `conda install -c anaconda pandasql`
- Info: http://pandas.pydata.org/pandas-docs/stable/comparison_with_sql.html

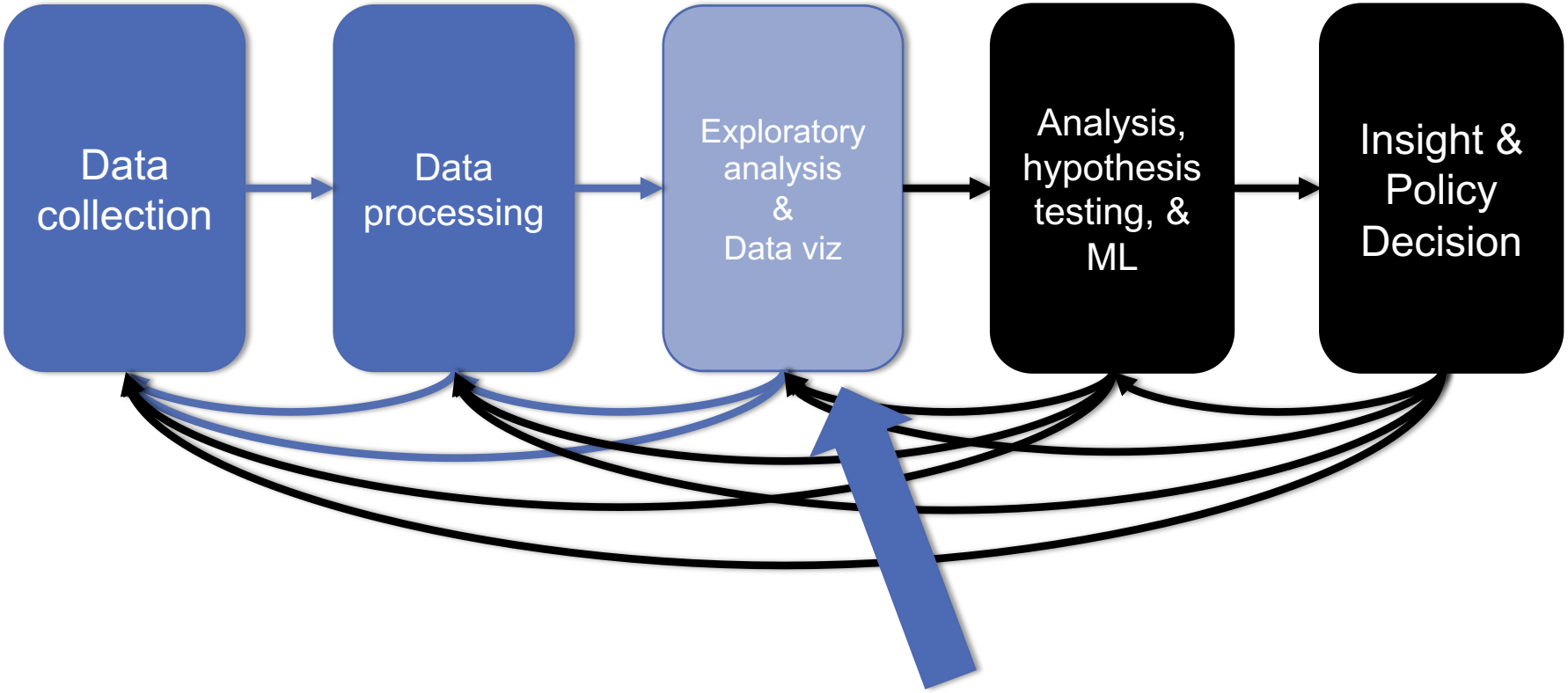
```
# Write the query text
q = """
    SELECT
        *
    FROM
        cats
    LIMIT 10;"""

# Store in a DataFrame
df = sqldf(q, locals())
```

NEXT CLASS:

EXPLORATORY ANALYSIS

TODAY'S LECTURE



TODAY'S LECTURE

Missing Data …

- What is it?
- Simple methods for imputation

… with a tiny taste of Stats/ML lecturers to come.

Thanks to John Atwood and Wenjiang Fu

MISSING DATA

Missing data is information that we want to know, but don't

It can come in many forms, e.g.:

- People not answering questions on surveys
- Inaccurate recordings of the height of plants that need to be discarded
- Canceled runs in a driving experiment due to rain

Could also consider missing columns (no collection at all) to be missing data …

[JA]

KEY QUESTION

Why is the data missing?

- What mechanism is it that contributes to, or is associated with, the probability of a data point being absent?
- Can it be explained by our observed data or not?

The answers drastically affect what we can ultimately do to compensate for the missing-ness

[JA]

COMPLETE CASE ANALYSIS

Delete all tuples with any missing values at all, so you are left only with observations with all variables observed

```
# Clean out rows with nil values
df = df.dropna()
```

Default behavior for libraries for analysis (e.g., regression)

- We'll talk about this much more during the Stats/ML lectures

This is the simplest way to handle missing data. In some cases, will work fine; in others, ??????????????:

- Loss of sample will lead to variance larger than reflected by the size of your data
- May bias your sample

[JA]

EXAMPLE

Dataset: Body fat percentage in men, and the circumference of various body parts [Penrose et al., 1985]

Question: Does the circumference of certain body parts predict body fat percentage?

Given complete data, how would you answer this ?????????

One way to answer is regression analysis:

- One or more independent variables ("predictors")
- One dependent variables ("outcome")

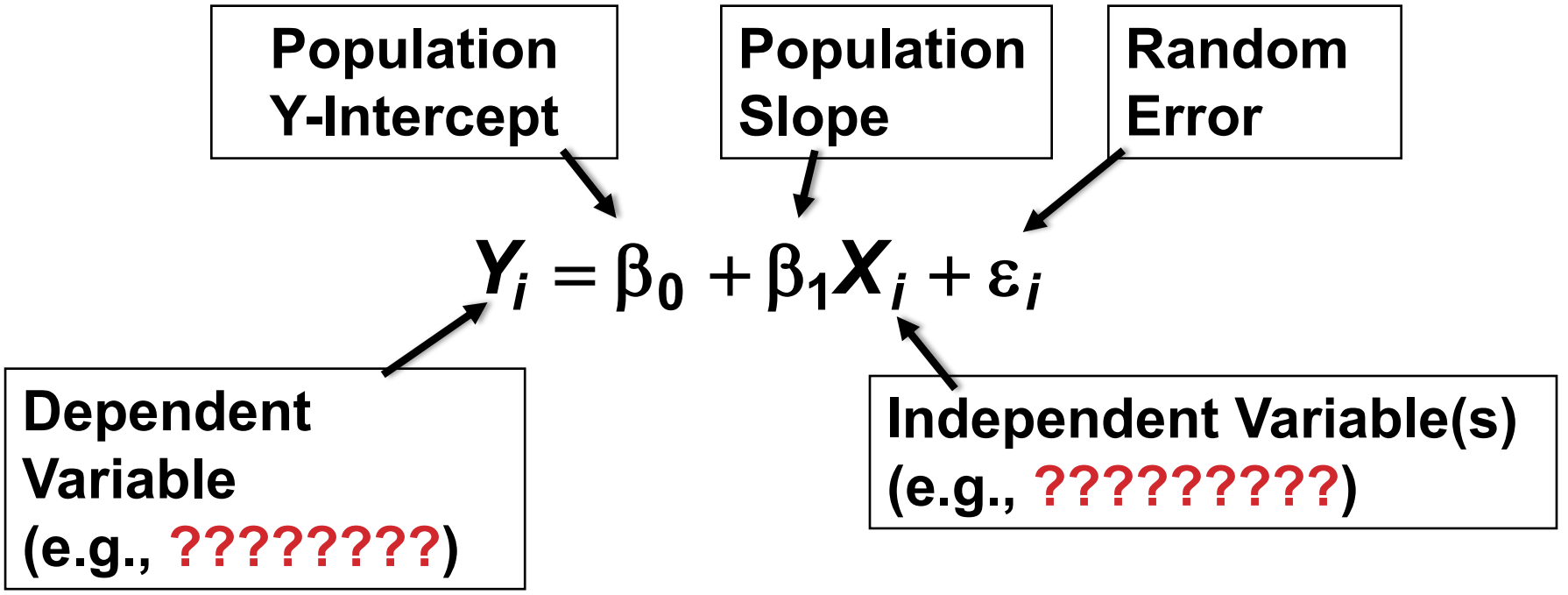
What is the relationship between the predictors and the outcome?

What is the conditional expectation of the dependent variable given fixed values for the dependent variables?

LINEAR REGRESSION

Assumption: relationship between variables is linear:

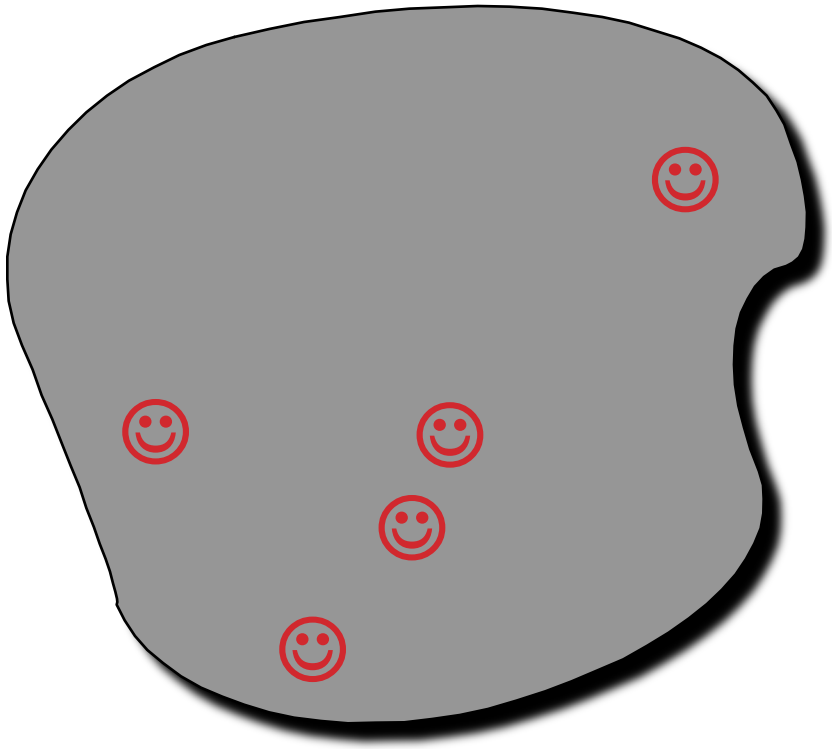
- (We'll relax linearity, study in more depth later.)



[WF]

POPULATION & SAMPLE REGRESSION MODELS

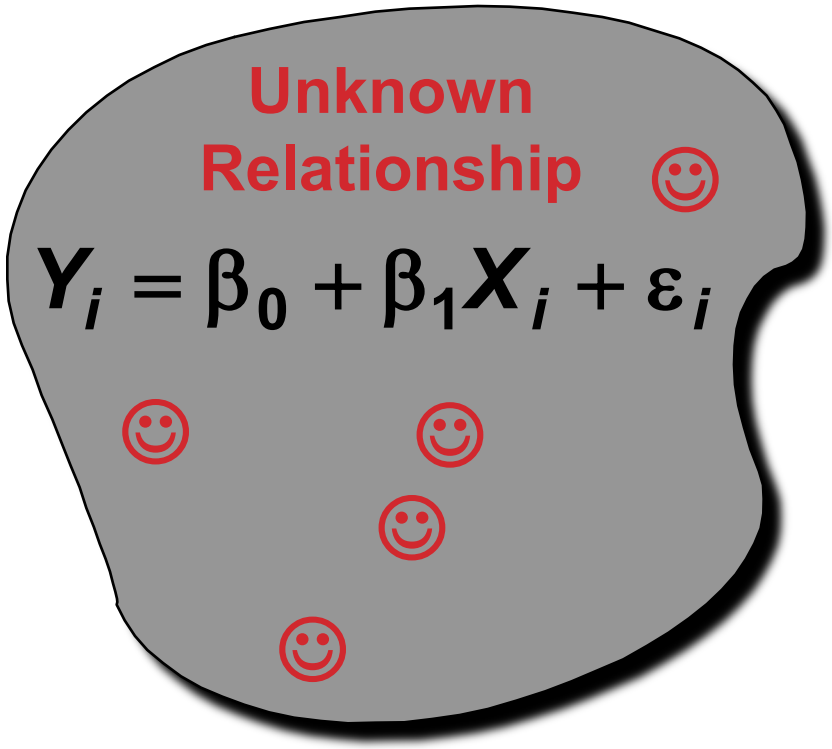
Population



[WF]

POPULATION & SAMPLE REGRESSION MODELS

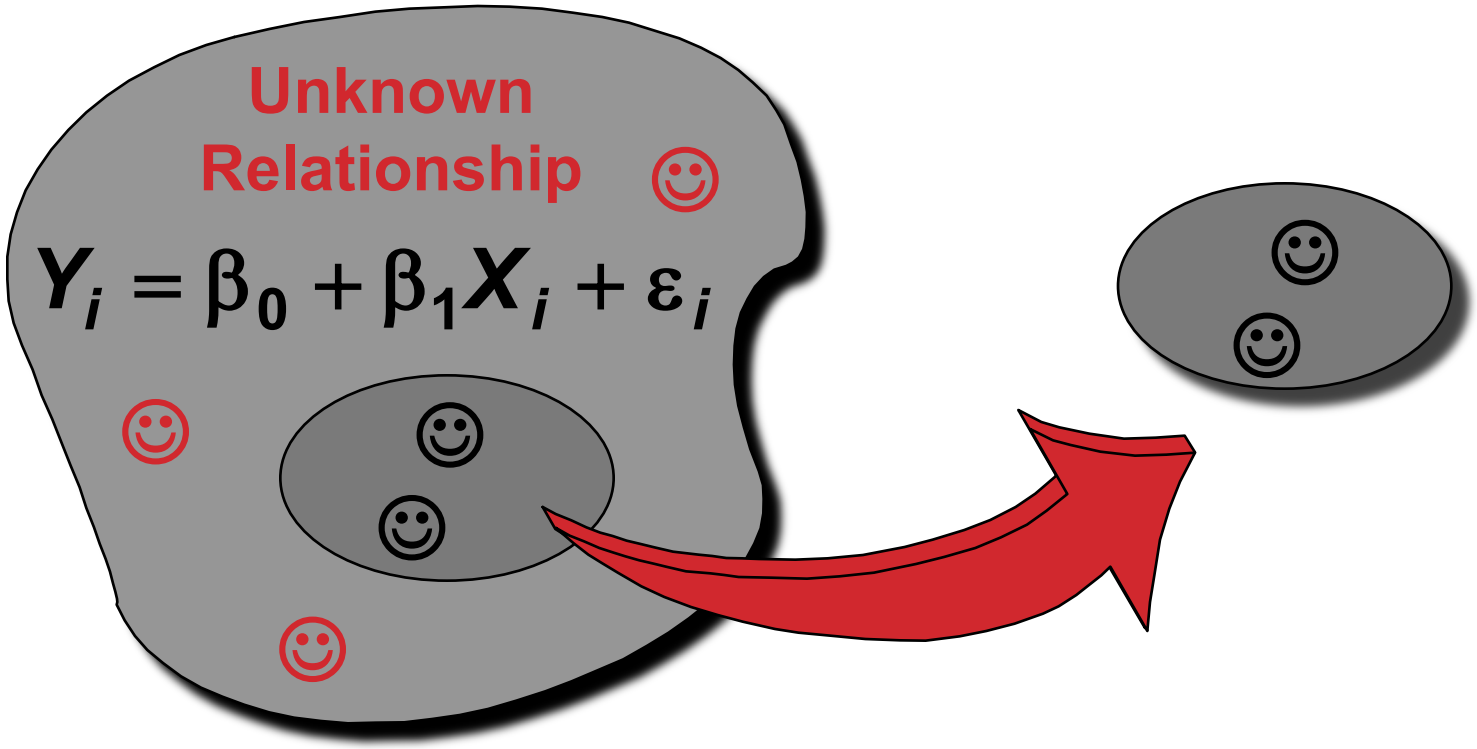
Population



[WF]

POPULATION & SAMPLE REGRESSION MODELS

Population **Random Sample**



[WF]

POPULATION & SAMPLE REGRESSION MODELS

Population

Random Sample

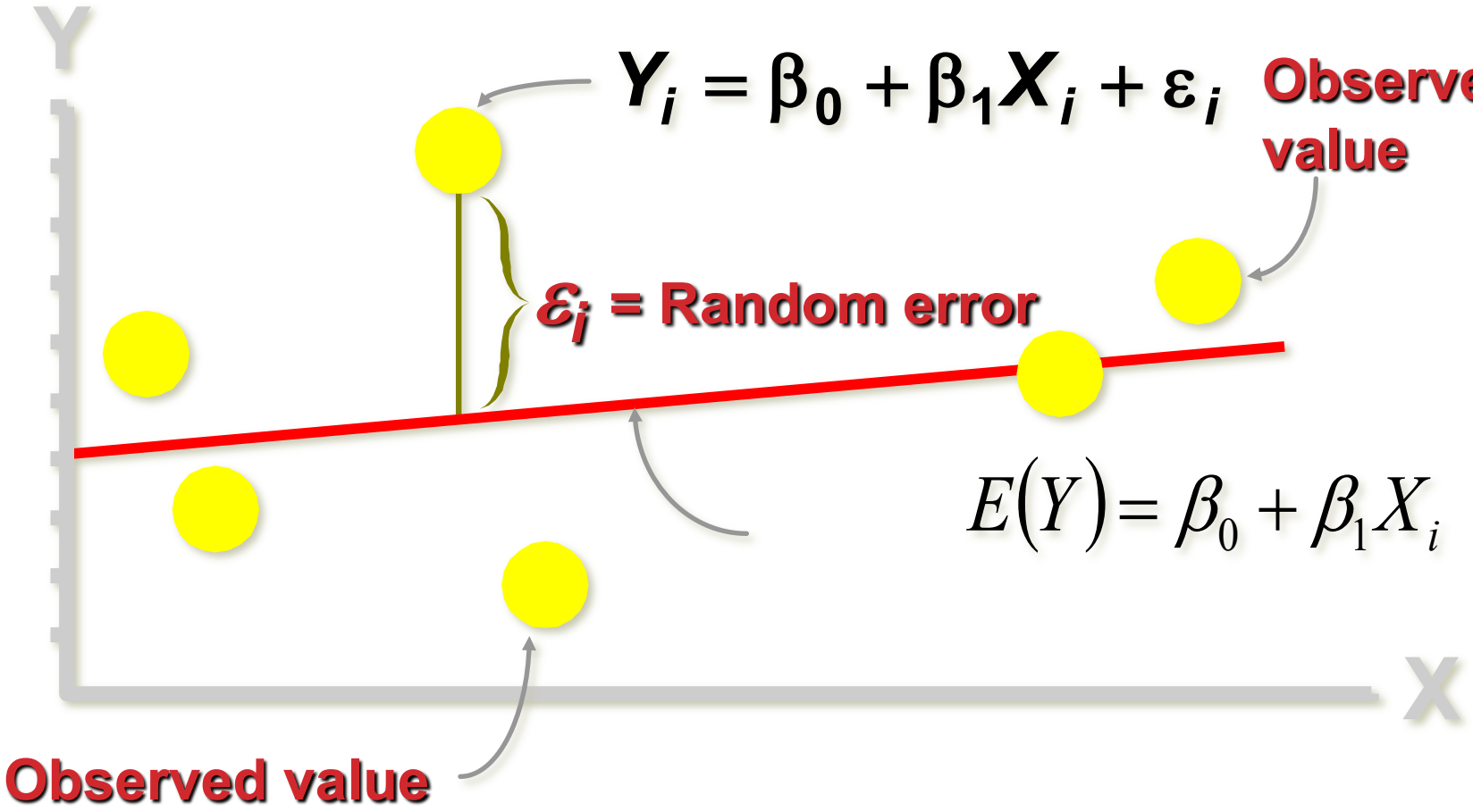
Unknown Relationship

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

$$Y_i = \hat{\beta}_0 + \hat{\beta}_1 X_i + \hat{\varepsilon}_i$$

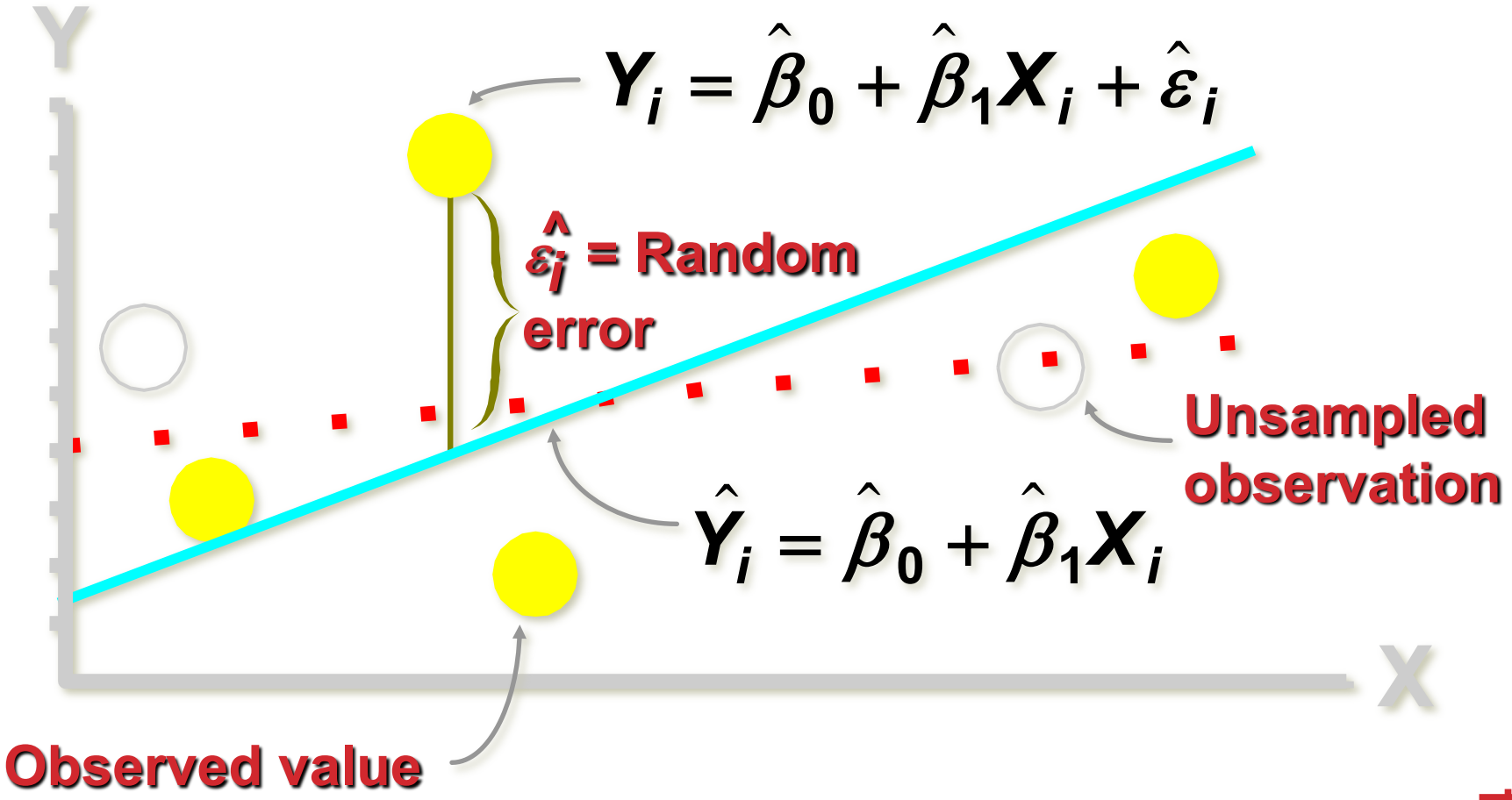
[WF]

LINEAR REGRESSION



[WF]

SAMPLE LINEAR REGRESSION MODEL



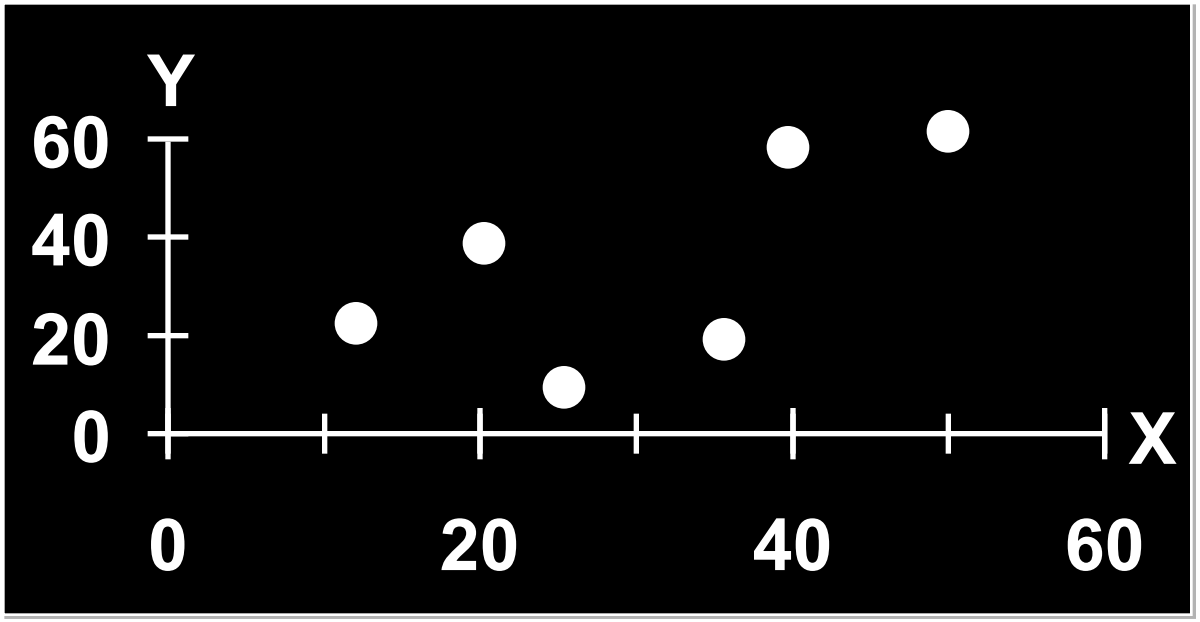
[WF]

ESTIMATING PARAMETERS: LEAST SQUARES METHOD

SCATTER PLOT

Plot all (X_i, Y_i) pairs, and plot your learned model

If you squint, suggests how well the model fits the data



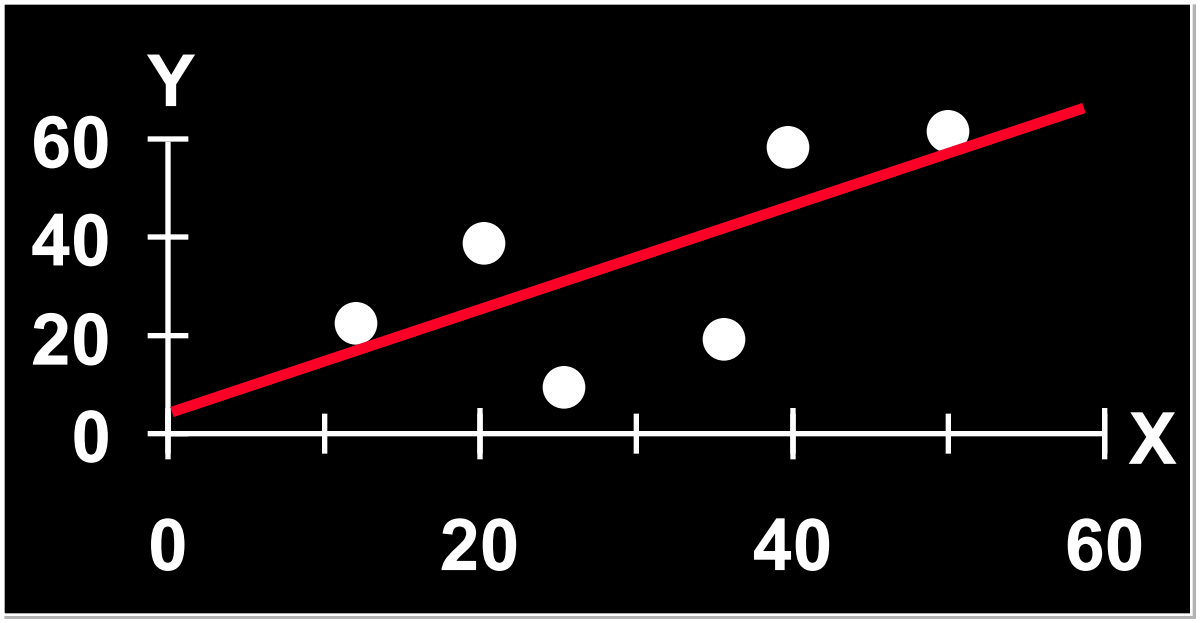
[WF]

QUESTION

How would you draw a line through the points?

How do you determine which line "fits the best" …?

?????????

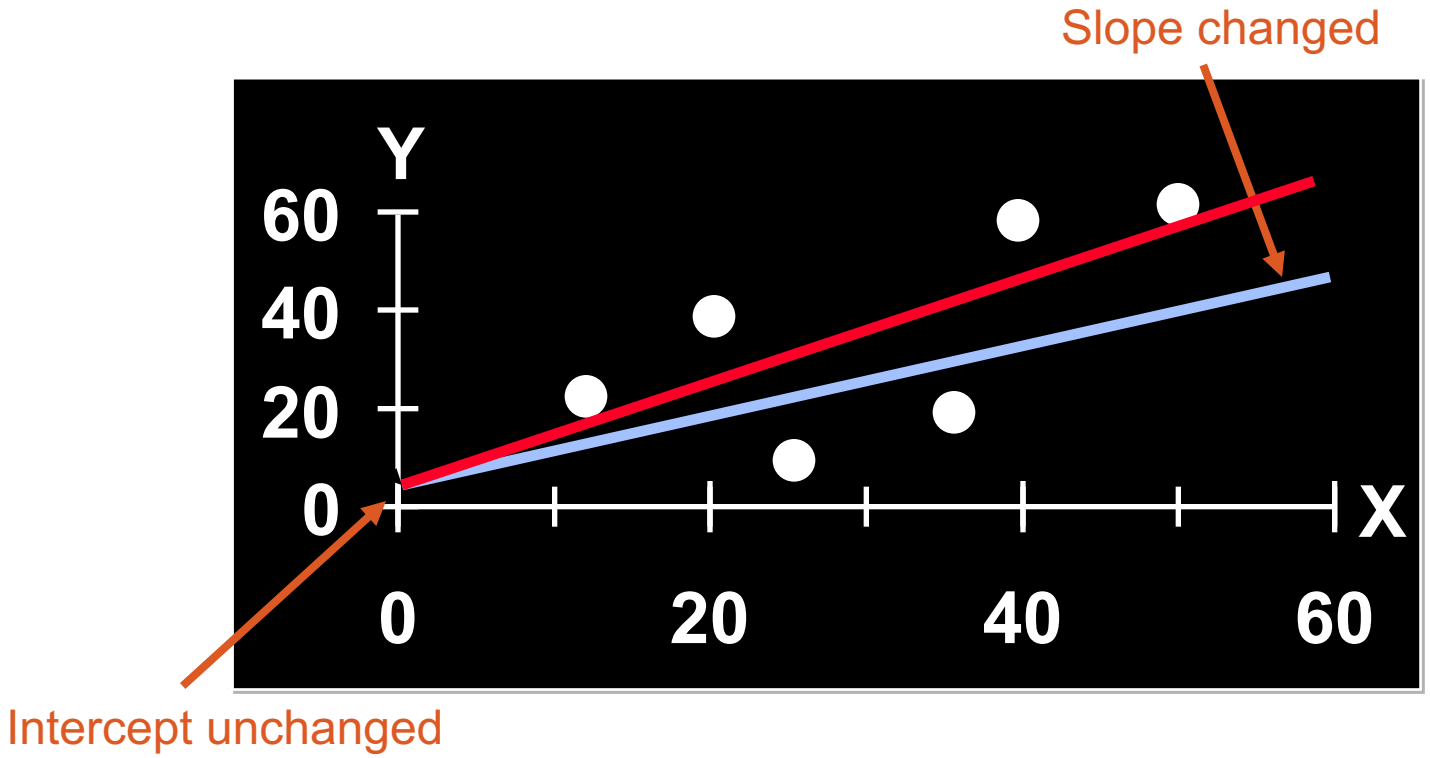


[WF]

QUESTION

How would you draw a line through the points?

How do you determine which line "fits the best" ?????????

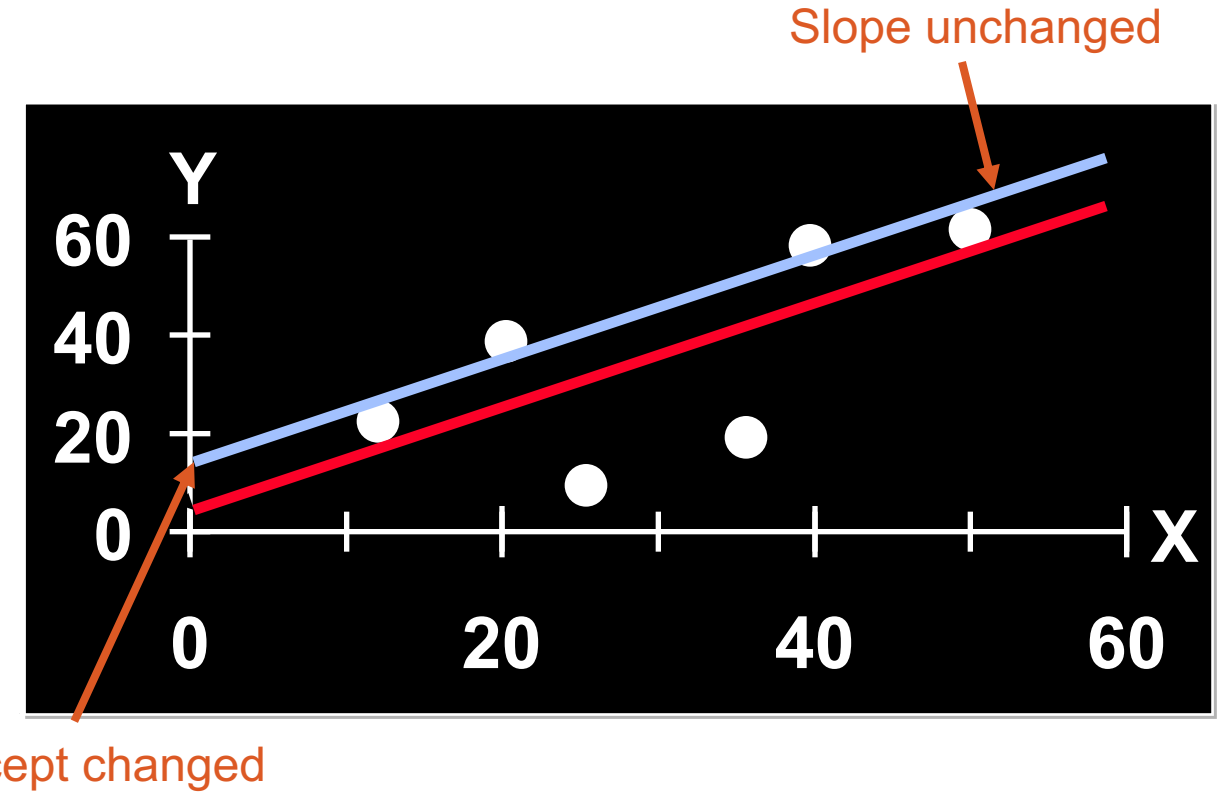


[WF]

QUESTION

How would you draw a line through the points?

How do you determine which line "fits the best" ?????????

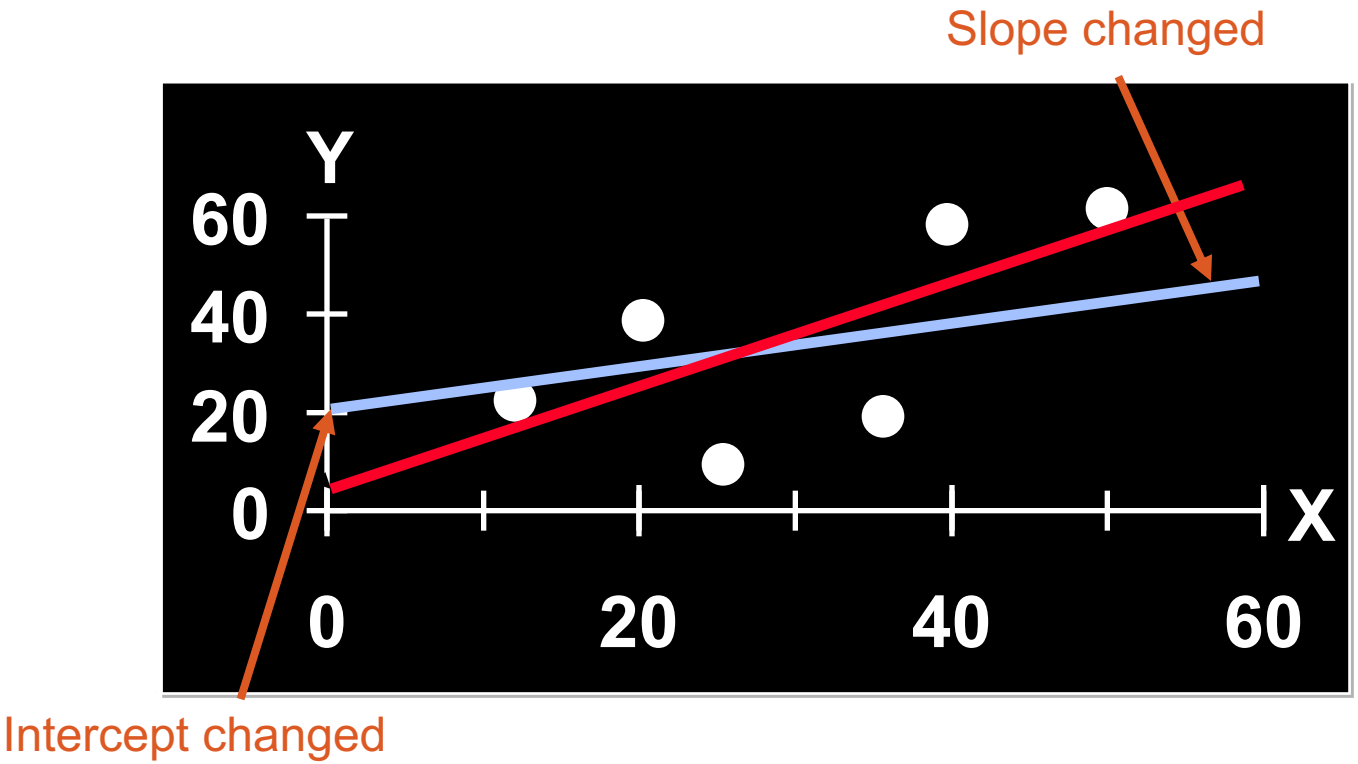


[WF]

QUESTION

How would you draw a line through the points?

How do you determine which line "fits the best" ?????????



[WF]

LEAST SQUARES

Best fit: difference between the true Y-values and the estimated Y-values is minimized:

- Positive errors offset negative errors …
- … square the error!

$$\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 = \sum_{i=1}^{n}\hat{\varepsilon}_i^2$$

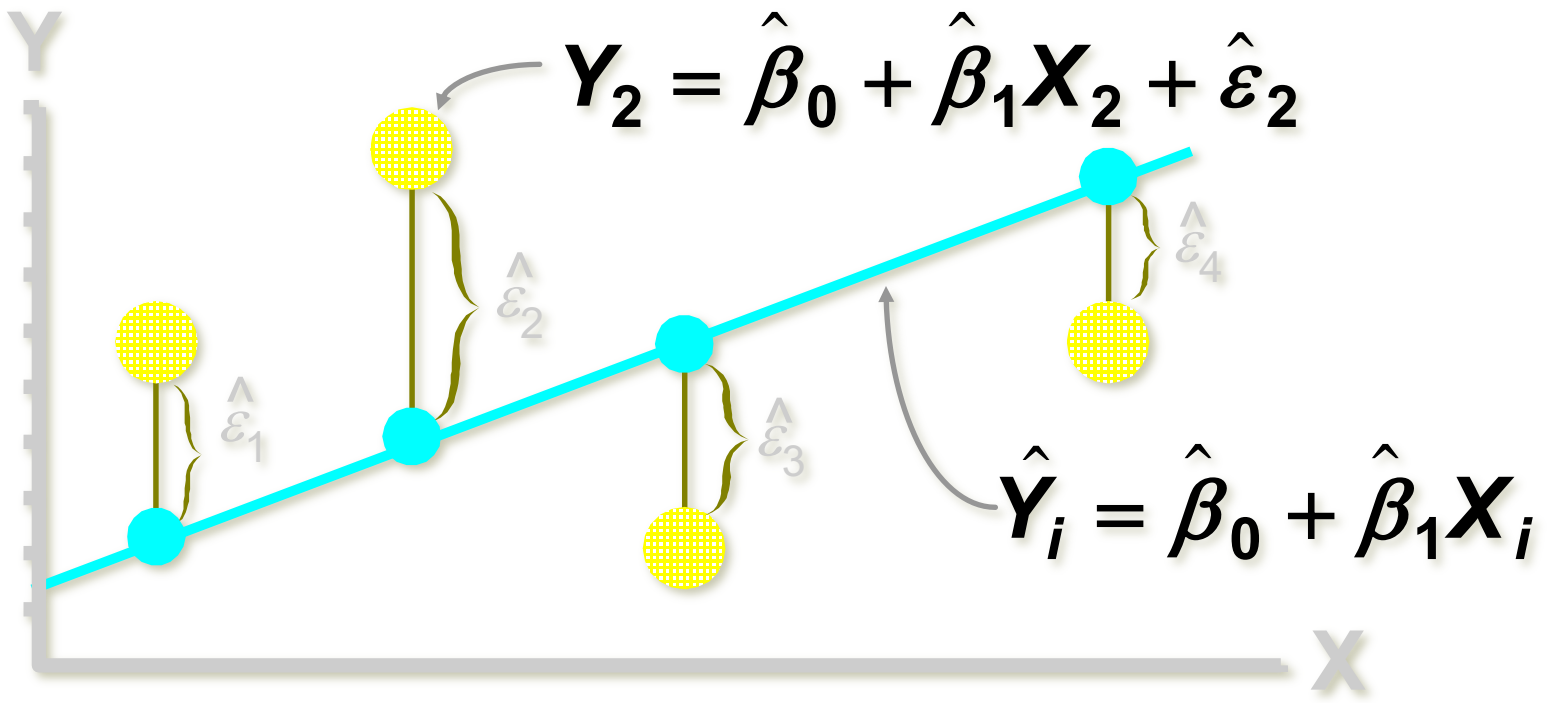
Least squares minimizes the sum of the squared errors

- Why squared? We'll cover this in more depth in March.
- Until then: http://www.benkuhn.net/squared

[WF]

LEAST SQUARES, GRAPHICALLY

LS minimizes $\sum_{i=1}^{n} \hat{\varepsilon}_i^2 = \hat{\varepsilon}_1^2 + \hat{\varepsilon}_2^2 + \hat{\varepsilon}_3^2 + \hat{\varepsilon}_4^2$



[WF]

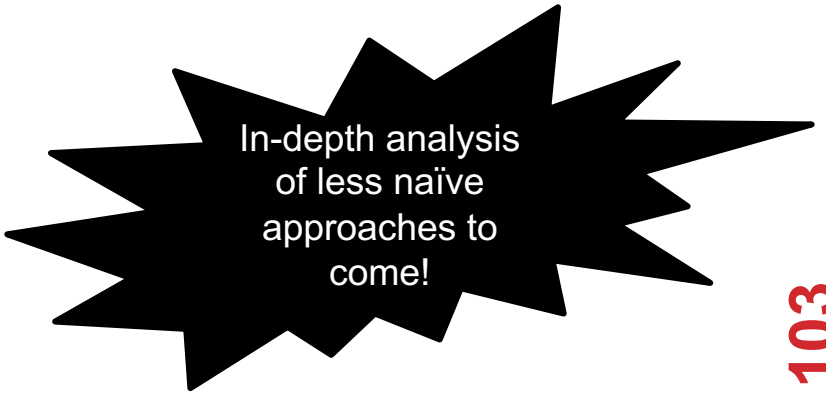
INTERPRETATION OF COEFFICIENTS

Slope ($\hat{\beta}_1$):

- Estimated *Y* changes by $\hat{\beta}_1$ for each unit increase in *X*
- If $\beta_1 = 2$, then *Y* Is expected to increase by 2 for each 1 unit increase in *X*

Y-Intercept ($\hat{\beta}_0$)

- Average value of *Y* when *X* = 0
- If $\hat{\beta}_0 = 4$, then average *Y* is expected to be 4 when *X* Is 0



[WF]

NOW, BACK TO MISSING DATA ...

EXAMPLE

Question: Does the circumference of certain body parts predict BF%?

Assumption: BF% is a linear function of measurements of various body parts and other features …

Analysis: Results from a regression model with BF% …

Predictor	Estimate	S.E.	p-value
Age	0.0626	0.0313	0.0463
Neck	-0.4728	0.2294	0.0403
Forearm	0.45315	0.1979	0.0229
Wrist	-1.6181	0.5323	0.0026

(Interpretation ???????????)

[JA]

WHAT IF DATA WERE MISSING?

In this case, the dataset is complete:

- But what if 5 percent of the participants had missing values? 10 percent? 20 percent?

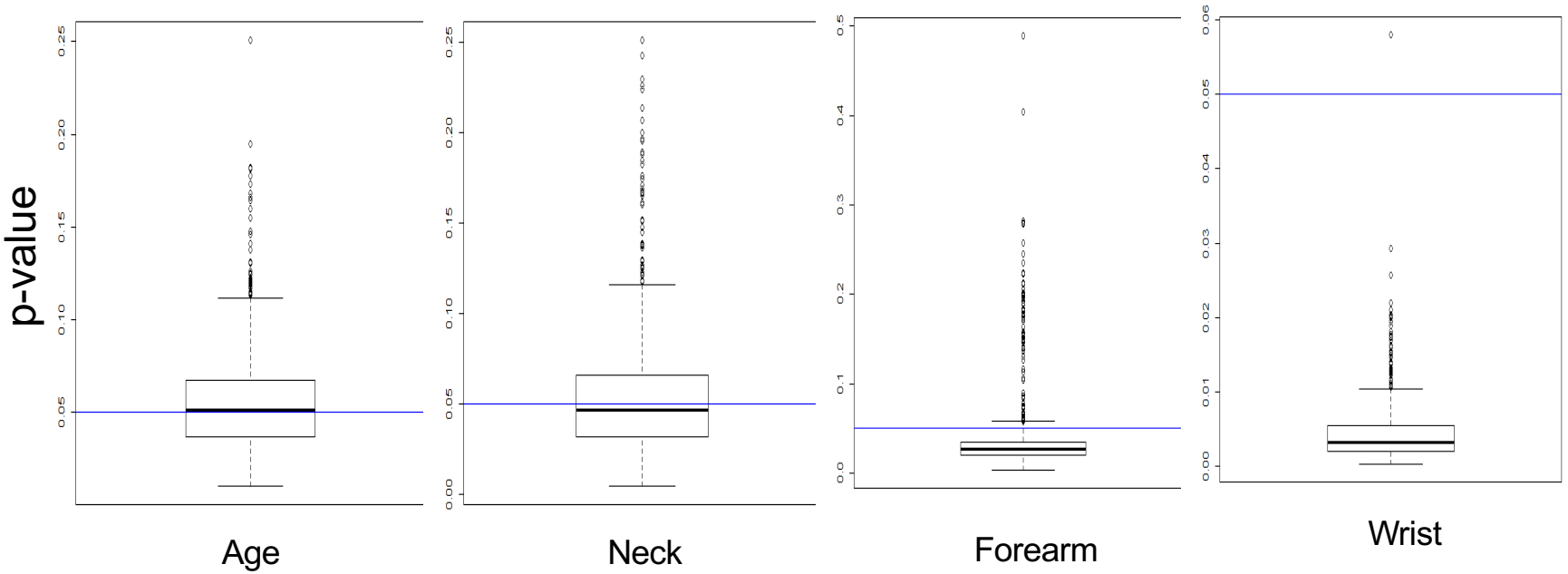
What if we performed complete case analysis and removed those who had missing values?

First let's examine the effect if we do this if when the data is missing completely at random (MCAR)

- Removed cases at random, reran analysis, stored the p-values
- p-value: probability of getting at least as extreme a result as what we observed given that there is no relationship
- Repeat 1000 times, plot p-values …

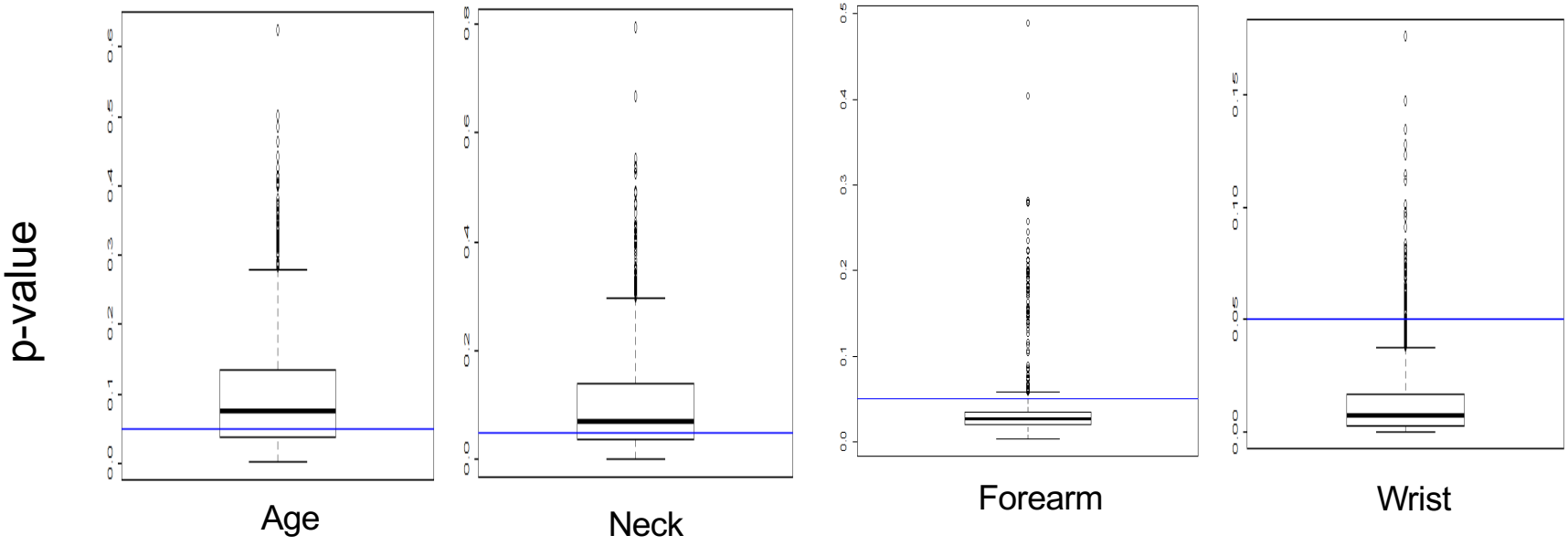
[JA]

~5% DELETED (N=13)



[JA]

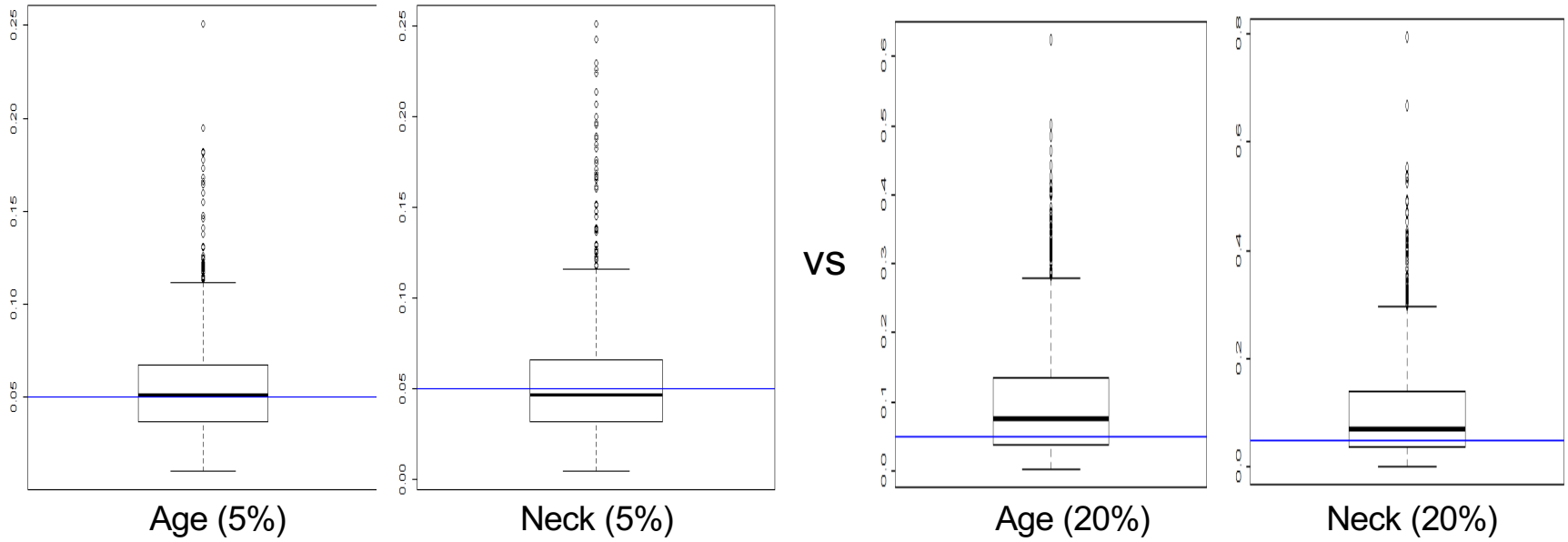
~20% DELETED (N=50)



[JA]

CONCLUSIONS SEEM TO CHANGE …

Age/Neck: fail to reject the null hypothesis usually?



Still reject Forearm/Wrist most of the time

This is assuming the missing subjects' distribtion does not differ from the non-missing. This would cause bias …

[JA]

TYPES OF MISSING-NESS

Missing Completely at Random (MCAR)

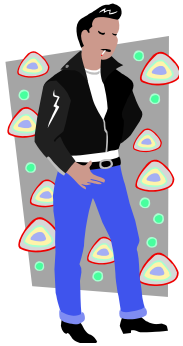
Missing at Random (MAR)

Missing Not at Random (MNAR)

[JA]

WHAT DISTINGUISHES EACH TYPE OF MISSING-NESS?

Suppose you're loitering outside of CSIC one day …



Students just received their mid-semester grades

You start asking passing undergrads their CMSC131 grades

- **You don't force them to tell you or anything**
- **You also write down their gender and hair color**

[JA]

YOUR SAMPLE

Hair Color	Gender	Grade
Red	M	A
Brown	F	A
Black	F	B
Black	M	A
Brown	M	
Brown	M	
Brown	F	
Black	M	B
Black	M	B
Brown	F	A
Black	F	
Brown	F	C
Red	M	
Red	F	A
Brown	M	A
Black	M	A

Summary:

- 7 students received As
- 3 students received Bs
- 1 student received a C

Nobody is failing!

- But 5 students did not reveal their grade …

[JA]

WHAT INFLUENCES A DATA POINT'S PRESENCE?

Same dataset, but the values are replaced with a "0" if the data point is observed and "1" if it is not

Question: for any one of these data points, what is the probability that the point is equal to "1" …?

What type of missing-ness do the grades exhibit?

Hair Color	Gender	Grade
0	0	0
0	0	0
0	0	0
0	0	0
0	0	1
0	0	1
0	0	1
0	0	0
0	0	0
0	0	0
0	0	1
0	0	0
0	0	1
0	0	0
0	0	0
0	0	0

[JA]

MCAR: MISSING COMPLETELY AT RANDOM

If this probability is not dependent on any of the data, observed or unobserved, then the data is Missing Completely at Random (MCAR)

Suppose that X is the observed data and Y is the unobserved data. Call our "missing matrix" R

Then, if the data are MCAR, P(R|X,Y) = ??????????

$$P(R|X,Y) = P(R)$$

Probability of those rows missing is independent of anything.

[JA]

TOTALLY REALISTIC MCAR EXAMPLE

You are running an experiment on plants grown in pots, when suddenly you have a nervous breakdown and smash some of the pots

You will probably not choose the plants to smash in a well-defined pattern, such as height age, etc.

Hence, the missing values generated from your act of madness will likely fall into the MCAR category

[JA]

APPLICABILITY OF MCAR

A completely random mechanism for generating missing-ness in your data set just isn't very realistic

Usually, missing data is missing for a reason:

- **Maybe older people are less likely to answer web-delivered questions on surveys**
- **In longitudinal studies people may die before they have completed the entire study**
- **Companies may be reluctant to reveal financial information**

MAR: MISSING AT RANDOM

Missing at Random (MAR): probability of missing data is dependent on the observed data but not the unobserved data

Suppose that X is the observed data and Y is the unobserved data. Call our "missing matrix" R

Then, if the data are MCAR, P(R|X,Y) = ??????????

$$P(R|X,Y) = P(R|X)$$

Not exactly random (in the vernacular sense).

- There is a probabilistic mechanism that is associated with whether the data is missing
- Mechanism takes the observed data as input

EXAMPLES?

MAR: KEY POINT

We can model that latent mechanism and compensate for it

Imputation: replacing missing data with substituted values

- Models today will assume MAR

Example: if age is known, you can model missing-ness as a function of age

Whether or not missing data is MAR or the next type, Missing Not at Random (MNAR), is not* testable.

- Requires you to “understand” your data

*unless you can get the missing data (e.g., post-study phone calls)

MNAR: MISSING NOT AT RANDOM

MNAR: missing-ness has something to do with the missing data itself

Examples: ??????????

- Do you binge drink? Do you have a trust fund? Do you use illegal drugs? What is your sexuality? Are you depressed?

Said to be "non-ignorable":

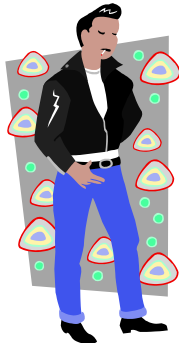
- Missing data mechanism must be considered as you deal with the missing data
- Must include model for why the data are missing, and best guesses as to what the data might be

BACK TO CSIC …

Is the the missing data:

- **MCAR;**
- **MAR; or**
- **MNAR?**

???????????



Hair Color	Gender	Grade
Red	M	A
Brown	F	A
Black	F	B
Black	M	A
Brown	M	
Brown	M	
Brown	F	
Black	M	B
Black	M	B
Brown	F	A
Black	F	
Brown	F	C
Red	M	
Red	F	A
Brown	M	A
Black	M	A

ADD A VARIABLE

Bring in the GPA:

Does this change anything?

Hair Color	GPA	Gender	Grade
Red	3.4	M	A
Brown	3.6	F	A
Black	3.7	F	B
Black	3.9	M	A
Brown	2.5	M	
Brown	3.2	M	
Brown	3.0	F	
Black	2.9	M	B
Black	3.3	M	B
Brown	4.0	F	A
Black	3.65	F	
Brown	3.4	F	C
Red	2.2	M	
Red	3.8	F	A
Brown	3.8	M	A
Black	3.67	M	A

HANDLING MISSING DATA ...

SINGLE IMPUTATION

Mean imputation: imputing the average from observed cases for all missing values of a variable

Hot-deck imputation: imputing a value from another subject, or "donor," that is most like the subject in terms of observed variables

- Last observation carried forward (LOCF): order the dataset somehow and then fill in a missing value with its neighbor

Cold-deck imputation: bring in other datasets

Old and busted:

- All fundamentally impose too much precision.
- Have uncertainty over what unobserved values actually are
- Developed before cheap computation

MULTIPLE IMPUTATION

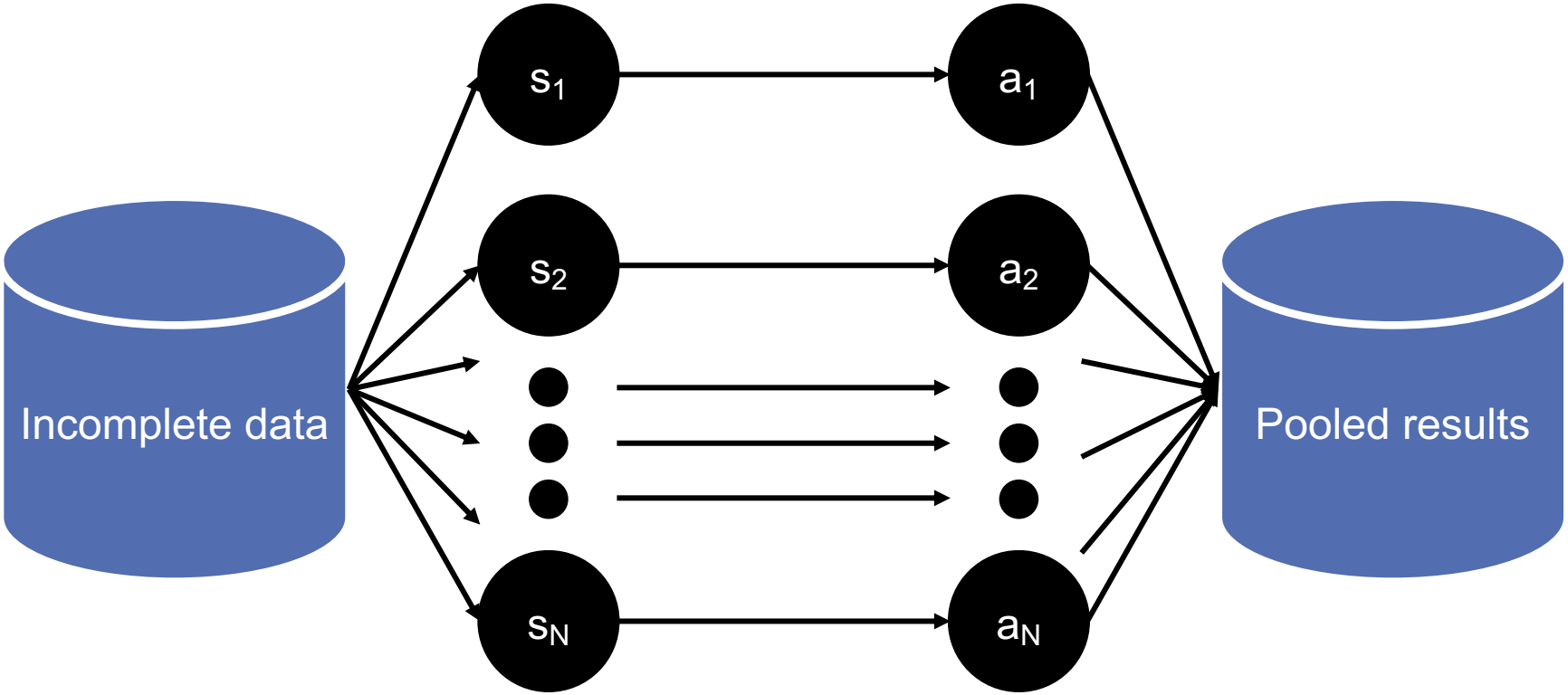
Developed to deal with noise during imputation

- Impute once → treats imputed value as observed

We have uncertainty over what the observed value would have been

Multiple imputation: generate several random values for each missing data point during imputation

IMPUTATION PROCESS



TINY EXAMPLE

X	Y
32	2
43	?
56	6
25	?
84	5

Independent variable: X
Dependent variable: Y
We assume Y has a linear relationship with X

LET'S IMPUTE SOME DATA!

Use a predictive distribution of the missing values:

- Given the observed values, make random draws of the observed values and fill them in.
- Do this N times and make N imputed datasets

X	Y
32	2
43	5.5
56	6
25	8
84	5

X	Y
32	2
43	7.2
56	6
25	1.1
84	5

For very large values of N=2 …

INFERENCE WITH MULTIPLE IMPUTATION

Now that we have our imputed data sets, how do we make use of them? ??????????

- Analyze each of the separately

X	Y
32	2
43	5.5
56	6
25	8
84	5

Slope	-0.8245
Standard error	6.1845

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

X	Y
32	2
43	7.2
56	6
25	1.1
84	5

Slope	4.932
Standard error	4.287

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

POOLING ANALYSES

Pooled slope estimate is the average of the N imputed estimates

Our example, $\beta_{1p} = \frac{\beta 11+\beta 12}{2}$ = (4.932-.8245) x 0.5 = 2.0538

The pooled slope variance is given by

$$s = \frac{\sum Zi}{m} + (1 + \frac{1}{m}) \text{ x } \frac{1}{m-1} * \sum(\beta 1i - \beta_{1p})^2$$

Where Z_i is the standard error of the imputed slopes

Our example: (4.287 + 6.1845)/2 + (3/2)*(16.569) = 30.08925

Standard error: take the square root, and we get 5.485

PREDICTING THE MISSING DATA GIVEN THE OBSERVED DATA

Given events A, B; and P(A) > 0 …

Bayes' Theorem:

$$P(B|A) = \frac{P(A|B) * P(B)}{P(A)}$$

Probability of seeing evidence given the hypothesis

In our case:

$$P(\mathbf{H}|\mathbf{E}) = \frac{P(\mathbf{E}|\mathbf{H}) * P(\mathbf{H})}{P(\mathbf{E})}$$

Prior probability of hypotheses

Posterior probability of the hypothesis given the evidence

Prior over the evidence

BAYESIAN IMPUTATION

Establish a prior distribution:

- **Some distribution of parameters of interest θ before considering the data, $P(\theta)$**
- **We want to estimate θ**

Given θ, can establish a distribution $P(X_{obs}|\theta)$

Use Bayes Theorem to establish $P(\theta|X_{obs})$ ...

- **Make random draws for θ**
- **Use these draws to make predictions of Y_{miss}**

HOW BIG SHOULD N BE?

Number of imputations N depends on:

- Size of dataset
- Amount of missing data in the dataset

Some previous research indicated that a small N is sufficient for efficiency of the estimates, based on:

- $(1 + \frac{\lambda}{N})$-1
- N is the number of imputations and λ is the fraction of missing information for the term being estimated [Schaffer 1999]

More recent research claims that a good N is actually higher in order to achieve higher power [Graham et al. 2007]

MORE ADVANCED METHODS

Interested? Further reading:

- Regression-based MI methods
- Multiple Imputation Chained Equations (MICE) or Fully Conditional Specification (FCS)
  - Readable summary from JHU School of Public Health: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/
- Markov Chain Monte Carlo (MCMC)
  - We'll cover this a bit, but also check out CMSC422!

NEXT CLASS:

SUMMARY STATISTICS &VISUALIZATION