INTRODUCTION TO DATA SCIENCE

JOHN P DICKERSON

Midterm Review – 10/27/2020

CMSC320
Tuesdays & Thursdays
5:00pm – 6:15pm
(… or anytime on the Internet)

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

ANNOUNCEMENTS

Mini-Project #2 was due late last week!

- Deliverable was an .ipynb file submitted to ELMS, **but moving forward this will be .pdf / .html files, for TA grading ease**
- Some folks had trouble getting the .pdf export to render figures – that's okay, if we run into an issue grading, we'll ping you
- In the future: can export to .html and then convert to .pdf

Mini-Project #3 will be released after the midterm!

- Due before Thanksgiving (TBD)

PROJECT 1 GRADES ARE UP!

General comments:

People did really well!

We used a fairly strict rubric, but if you have a real bone to pick with your grade, please triage through TAs/office hours!

Comments for our sanity, moving forward:

- `df.head(n)` -- defaults to n = 5, use ~10, 20, 50 as needed
- Please label your `pdf` file something like `<lastname>_<firstname>_project3.pdf`
- E.g., `dickerson_john_project3.pdf`

MIDTERM: STRUCTURE

50 points = 25% of the total grade
Rough breakdown (may change a little):

10 points:

- 10 True/False questions, 1 point each

10 points:

- 5 multiple choice questions, 2 points each

30 points:

- 10 short answer questions, 3 points each

Compared to the CMSC320 midterm I posted from an earlier semester, this midterm is more qualitative.

QUICK MIDTERM REVIEW

As discussed in previous lectures and on Piazza, the midterm can cover:

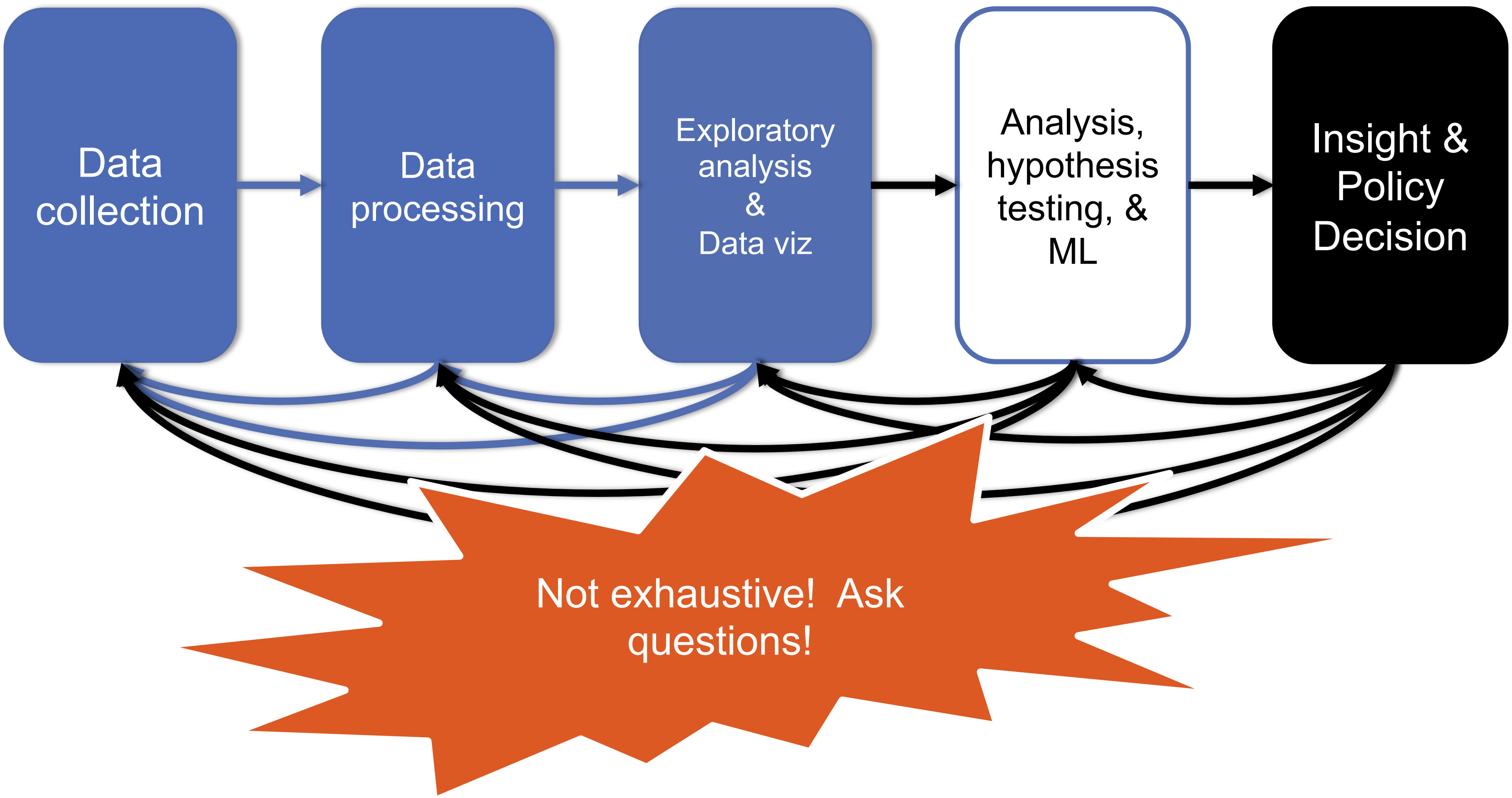
- Up to and including last Thursday's lecture (10/22)
- Quizzes that were due on or before today
- Stuff that you should know from doing P1 and P2

Everything is online: https://cmsc320.github.io/

I know this is a lot of material.

- Rule of thumb: open up a slide deck
- Do you feel "comfortable" with the material?
- Test will be more qualitative than prior 1xx, 2xx, 3xx tests

QUICK MIDTERM REVIEW



DATA COLLECTION (DC) & DATA PROCESSING (DP)

We talked about:

- Scraping data
- RESTful APIs
- Structured data formats (JSON, XML, etc)
- Regexes

Data manipulation via Numpy Stack (Numpy, Pandas, etc)

- Indexing, slicing, groups, joins, aggregate queries, etc

Tidy data + melting

Version control (just know how this works qualitatively)

RDMS, a little bit of SQL

Entity resolution & other data integration issues

Storing stuff as a graph, and manipulating it

DC: HTTP REQUESTS

https://www.google.com/**?q=cmsc320&tbs=qdr:m**

Google ??????????

HTTP GET Request:

GET /?q=cmsc320&tbs=qdr:m HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:10.0.1) Gecko/20100101 Firefox/10.0.1

```
params = { “q”: “cmsc320”, “tbs”: “qdr:m” }
r = requests.get(   “https://www.google.com”,
                    params = params )
```

*be careful with https:// calls; `requests` will not verify SSL by default

DC: RESTFUL APIS

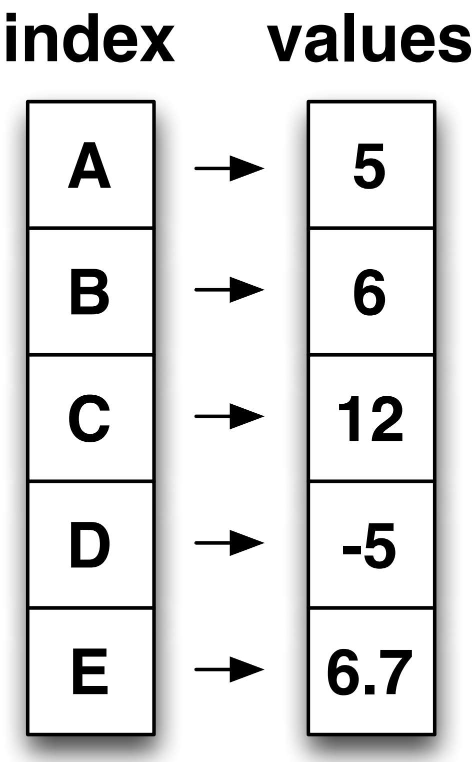
This class will just query web APIs, but full web APIs typically allow more.

Representational State Transfer (RESTful) APIs:

- GET: perform query, return data
- POST: create a new entry or object
- PUT: update an existing entry or object
- DELETE: delete an existing entry or object

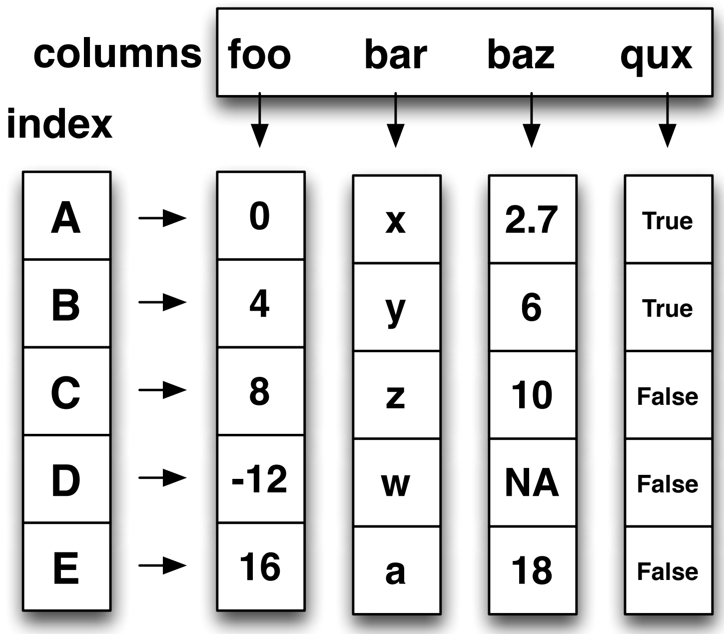
Can be more intricate, but verbs ("put") align with actions

DC: PANDAS: SERIES



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

DC: 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

DC: STORING A GRAPH

Three main ways to represent a graph in memory:

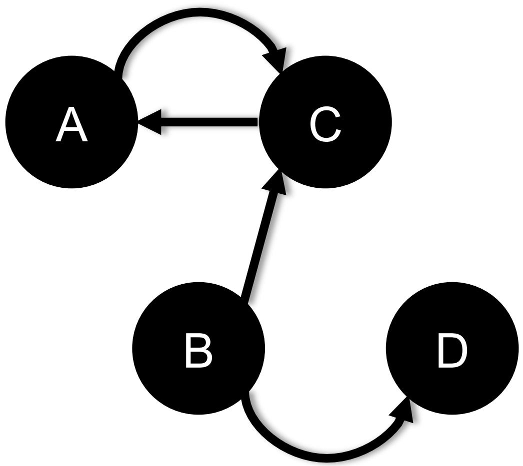
- Adjacency lists
- Adjacency dictionaries
- Adjacency matrix

The storage decision should be made based on the expected use case of your graph:

- Static analysis only?
- Frequent updates to the structure?
- Frequent updates to semantic information?

DC: ADJACENCY LISTS

For each vertex, store an array of the vertices it connects to



Vertex	Neighbors
A	[C]
B	[C, D]
C	[A]
D	[]

Pros: ????????

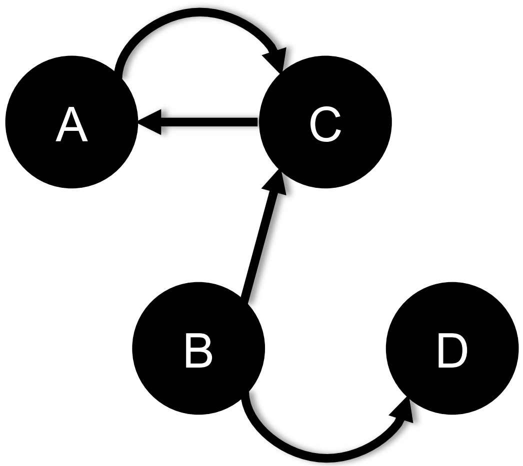
- Iterate over all outgoing edges; easy to add an edge

Cons: ????????

- Checking for the existence of an edge is O(|V|), deleting is hard

DC: ADJACENCY DICTIONARIES

For each vertex, store a dictionary of vertices it connects to



Vertex	Neighbors
A	{C: 1.0}
B	{C: 1.0, D: 1.0}
C	{A: 1.0}
D	{}

Pros: ?????????

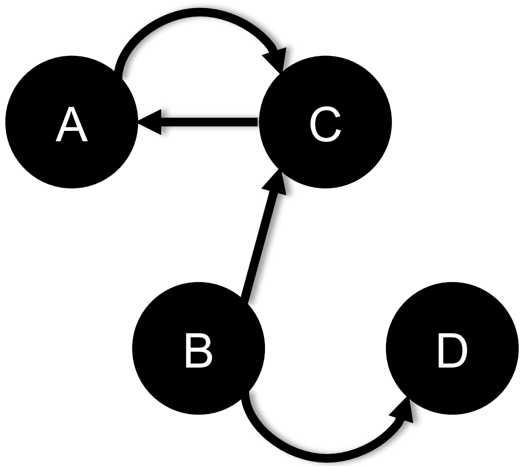
- O(1) to add, remove, query edges

Cons: ?????????

- Overhead (memory, caching, etc)

DC: ADJACENCY MATRIX

Store the connectivity of the graph in a matrix



From

To	A	B	C	D
A	0	0	1	0
B	0	0	0	0
C	1	1	0	0
D	0	1	0	0

Cons: ?????????

- $O(|V|^2)$ space regardless of the number of edges

Almost always stored as a sparse matrix

DP: 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

DP: 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

DP: 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")

DP: 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

By ‘A’ →

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

DP: 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

By 'B' →

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

DP: 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

By 'A', 'B' →

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

DP: 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

DP: 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

DP: 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

∪

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

DP: 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

$\bowtie$

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**

DP: 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

DP: GOOGLE IMAGE SEARCH ONE SLIDE SQL JOIN VISUAL

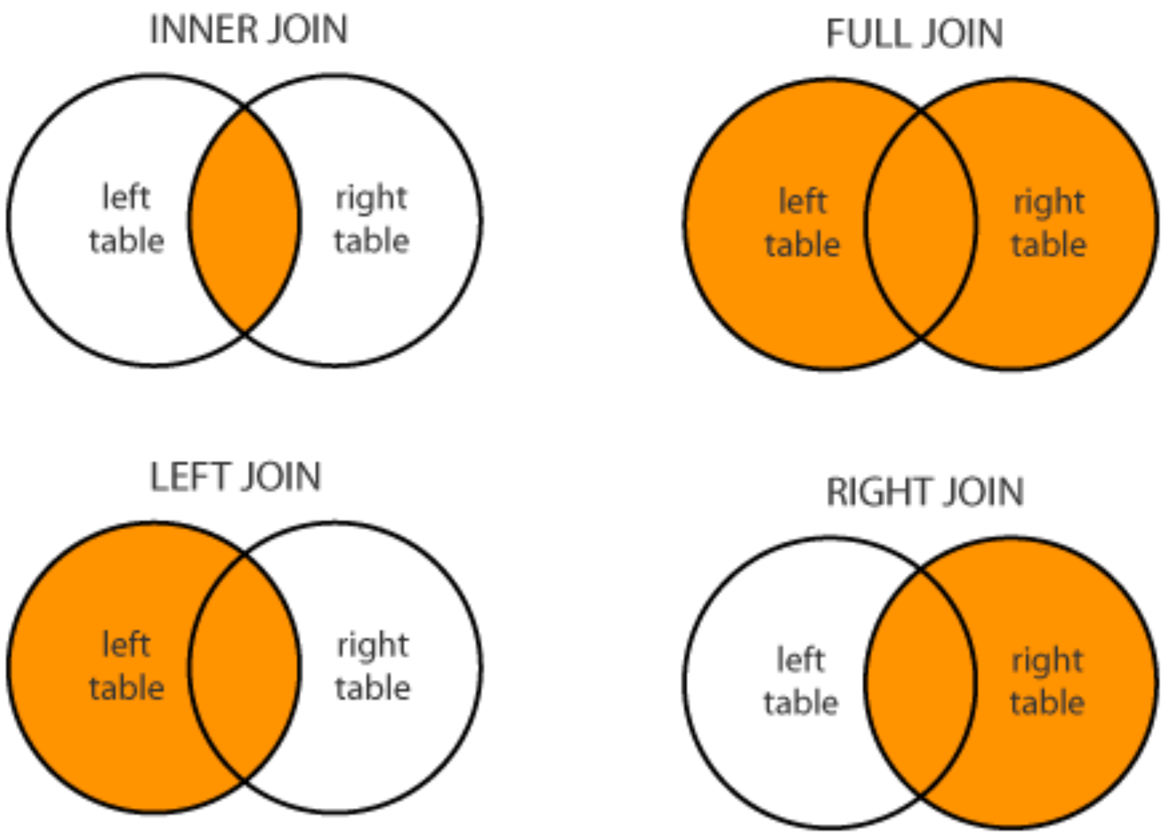
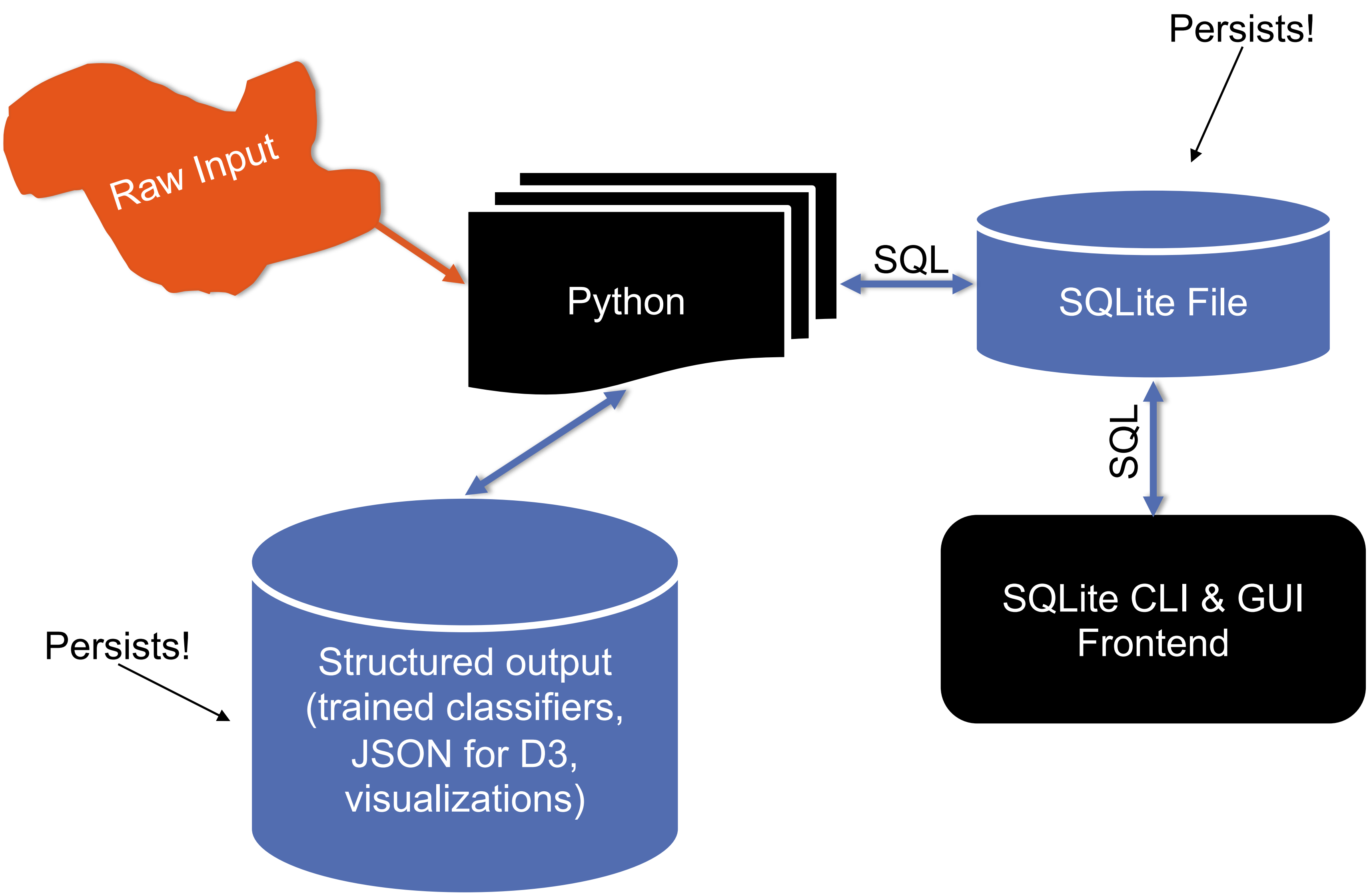


Image credit: http://www.dofactory.com/sql/join

DC/DP: HOW A RELATIONAL DB FITS INTO YOUR WORKFLOW



DP: ADDITIONAL STUFF

Data integration

- Extraction, schema alignment & mapping, querying over multiple schema / global schema

Data quality issues

- Single- vs multi-source quality issues

Data cleaning

- Outlier detection, constraint-based cleaning

Entity resolution (~part of data cleaning)

- Deduplication, record linkage, reference matching
- Fuzzy matching, etc.

EDA & VIZ

Missing data

- MCAR
- MAR
- MNAR
- Single & multiple imputation

Analysis

- Basic linear regression
- Summary statistics / robust statistics
- Variance, stdev, covariance, Pearson's correlation coefficient
- Hypothesis testing
- Bayes' rule

EDA: 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]

EDA: 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]

EDA: YOUR SAMPLE

Hair Color	> 6ft	Grade
Red	Y	A
Brown	N	A
Black	N	B
Black	Y	A
Brown	Y	
Brown	Y	
Brown	N	
Black	Y	B
Black	Y	B
Brown	N	A
Black	N	
Brown	N	C
Red	Y	
Red	N	A
Brown	Y	A
Black	Y	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]

EDA: 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	>6ft	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]

EDA: 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]

EDA: 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 MAR, 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

EDA: 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”:

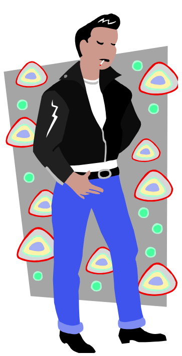
- Missing data mechanism must be considered as you deal with the missing data
- Must include model for why the data are missing, and best guesses as to what the data might be

EDA: BACK TO IRIBE …

Is the the missing data:

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

???????????



Hair Color	> 6ft	Grade
Red	Y	A
Brown	N	A
Black	N	B
Black	Y	A
Brown	Y	
Brown	Y	
Brown	N	
Black	Y	B
Black	Y	B
Brown	N	A
Black	N	
Brown	N	C
Red	Y	
Red	N	A
Brown	Y	A
Black	Y	A

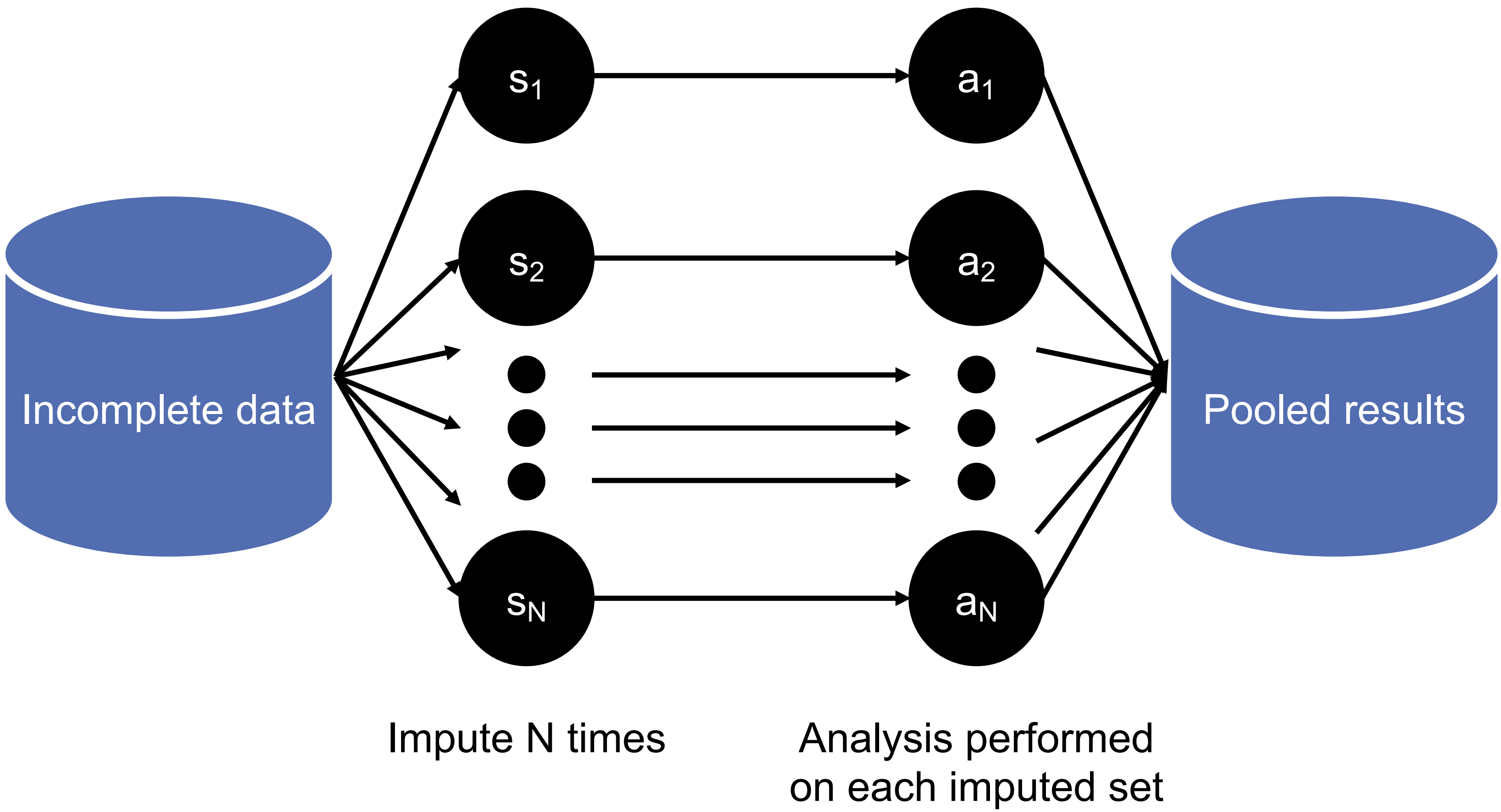
EDA: ADD A VARIABLE

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

EDA: MULTIPLE IMPUTATION PROCESS



ANALYSIS: IMPORTANCE OF VERTICES

Not all vertices are equally important

Centrality Analysis:

- Find out the most important node(s) in one network
- Used as a feature in classification, for visualization, etc …

Commonly-used Measures

- Degree Centrality
- Closeness Centrality
- Betweenness Centrality
- Eigenvector Centrality

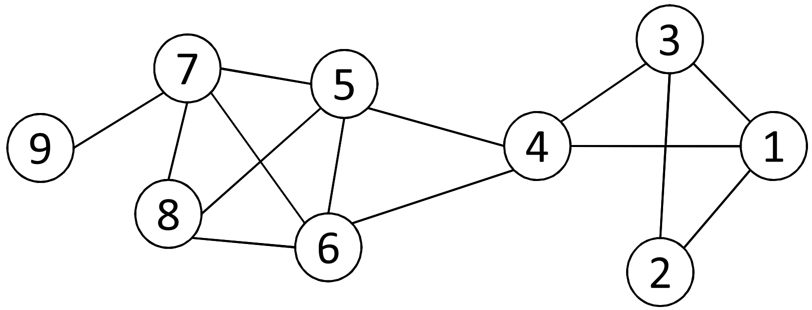
ANALYSIS: DEGREE CENTRALITY

The importance of a vertex is determined by the number of vertices adjacent to it

- The larger the degree, the more important the vertex is
- Only a small number of vertex have high degrees in many real-life networks

Degree Centrality: $$C_D(v_i) = d_i = \sum_j A_{ij}$$

Normalized Degree Centrality: $$C'_D(v_i) = d_i/(n-1)$$



For vertex 1, degree centrality is 3; Normalized degree centrality is 3/(9-1)=3/8.

ANALYSIS: BETWEENNESS CENTRALITY

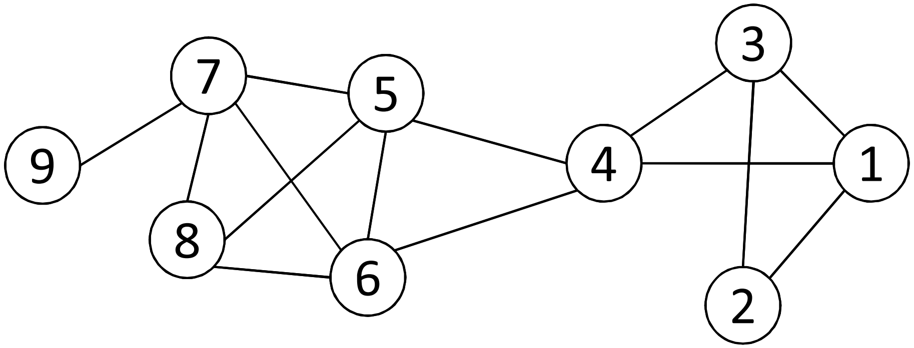


Table 2.2: $\sigma_{st}(4)/\sigma_{st}$

	$s=1$	$s=2$	$s=3$
$t=5$	1/1	2/2	1/1
$t=6$	1/1	2/2	1/1
$t=7$	2/2	4/4	2/2
$t=8$	2/2	4/4	2/2
$t=9$	2/2	4/4	2/2

σ_{st} : The number of shortest paths between s and t

$\sigma_{st}(v_i)$: The number of shortest paths between s and t that pass v_i

$$C_B(v_i) = \sum_{v_s \neq v_i \neq v_t \in V, s<t} \frac{\sigma_{st}(v_i)}{\sigma_{st}}$$

What is the betweenness centrality for node 4 ?????????

ANALYSIS: TERM FREQUENCY

Term frequency: the number of times a term appears in a specific document

- tf_{ij}: frequency of word *j* in document *i*

This can be the raw count (like in the BOW in the last slide):

- $tf_{ij} \in \{0,1\}$ if word *j* appears or doesn't appear in doc *i*
- $\log(1 + tf_{ij})$ – reduce the effect of outliers
- $tf_{ij} / \max_j tf_{ij}$ – normalize by document i's most frequent word

What can we do with this?

- Use as features to learn a classifier $w \rightarrow y$ …!

ANALYSIS: INVERSE DOCUMENT FREQUENCY

Recall:

- tf_{ij}: frequency of word *j* in document *i*

Any issues with this ??????????

- Term frequency gets overloaded by common words

Inverse Document Frequency (IDF): weight individual words negatively by how frequently they appear in the corpus:

$$\text{idf}_j = \log\left(\frac{\#\text{documents}}{\#\text{documents with word } j}\right)$$

IDF is just defined for a word j, not word/document pair j, i

ANALYSIS: TF-IDF

How do we use the IDF weights?

Term frequency inverse document frequency (TF-IDF):

- TF-IDF score: tf_{ij} x idf_j

	the	CMSC320	you	he	I	quick	dog	me	CMSCs	…	than
Document 1	0.8	0	0	0	0	1.1	1.1	0	0		0
Document 2	0	0	2.2	0.8	1.1	0	0	1.1	0	…	0
Document 3	0.8	1.1	0	0.4	0	0	0	0	1.1		1.1

This ends up working better than raw scores for classification and for computing similarity between documents.

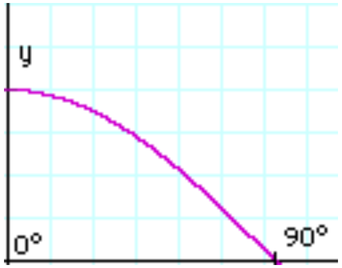
ANALYSIS: SIMILARITY BETWEEN DOCUMENTS

Given two documents x and y, represented by their TF-IDF vectors (or any vectors), the cosine similarity is:

$$\mathrm{similarity}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^{\mathsf{T}}\mathbf{y}}{|\mathbf{x}| \times |\mathbf{y}|}$$

Formally, it measures the cosine of the angle between two vectors x and y:

- $\cos(0^o) = 1$, $\cos(90^o) = 0$??????????



Similar documents have high cosine similarity; dissimilar documents have low cosine similarity.