WRAP-UP FROM LAST LECTURE ...

- Data collection
- Data processing
- Exploratory analysis & Data viz
- Analysis, hypothesis testing, & ML
- Insight & Policy Decision
DISCRETE TO CONTINUOUS VARIABLES

Some models only work on continuous numeric data
Convert a binary variable to a number ??????????????

• health_insurance = {“yes”, “no”} → {1, 0}

Why not {-1, +1} or {-10, +14}?  
• 0/1 encoding lets us say things like “if a person has healthcare then their income increases by $X.”
• Might need {-1,+1} for certain ML algorithms (e.g., SVM)
DISCRETE TO CONTINUOUS VARIABLES

What about non-binary variables?
My main transportation is a \{BMW, Bicycle, Hovercraft\}
One option: \{ BMW \to 1, Bicycle \to 2, Hovercraft \to 3 \}
• Problems ???????????

One-hot encoding: convert a categorical variable with $N$ values into a $N$-bit vector:
• BMW \to [1, 0, 0]; Bicycle \to [0, 1, 0]; Hovercraft \to [0, 0, 1]

```python
# Converts dtype=category to one-hot-encoded cols
cols = [‘BMW’,’Bicycle’,’Hovercraft’]
df = pd.Series(cols)
df = pd.get_dummies( df )
```
CONTINUOUS TO DISCRETE VARIABLES

Do doctors prescribe a certain medication to older kids more often? Is there a difference in wage based on age?

Pick a discrete set of bins, then put values into the bins

Equal-length bins:
- Bins have an equal-length range and skewed membership
- Good/Bad ?????????

Equal-sized bins:
- Bins have variable-length ranges but equal membership
- Good/Bad ?????????
BIN SIZE

DIFFERENT NUMBER OF BINS

https://web.ma.utexas.edu/users/mks/statmistakes/dividingcontinuousintocategories.html
SKewed data often arises in multiplicative processes:

- Some points float around 1, but one unlucky draw $\rightarrow 0$

Logarithmic transforms reduce skew:

- If values are all positive, apply log$_2$ transform
- If some values are negative:
  - Shift all values so they are positive, apply log$_2$
  - Signed log: sign(x) * log$_2$( |x| + 1)
SKEWED DATA

$\log_2$ transform on airline takeoff delays
NEXT UP:

VISUALIZATION, GRAPHS, & NETWORKS
AND NOW!

Graph Processing
• Representing graphs
• Centrality measures
• Community detection

Natural Language Processing
• Bag of Words, TF-IDF, N-grams
• (If we get to this today …)

Thank you to: Sukumar Ghosh (Iowa), Lei Tang (Yahoo!), Huan Liu (ASU), Zico Kolter (CMU)
**NETWORKS? GRAPHS?**

**Networks** are systems of interrelated objects

**Graphs** are the mathematical models used to represent networks

In data science, we will use algorithms on graphs to answer questions about real-world networks.
A graph $G = (V,E)$ is a set of vertices $V$ and edges $E$

Edges can be undirected or directed.

Examples of directed vs undirected graphs

Undirected:
- $V = \{A, B, C, D\}$
- $E = \{(A,B), (B,C), (C,D), (A,C)\}$

Directed:
- $V = \{A, B, C, D\}$
- $E = \{(A,C), (C,A), (B,C), (B,D)\}$
GRAPHs

Edges can be unweighted or weighted
- Unweighted \(\rightarrow\) all edges have unit weight

Examples of unweighted and weighted graphs ?????????????
Facebook posts (in black), and users liking or commenting on those posts
**NETWORKX**

**NetworkX** is a Python library for storing, manipulating, and analyzing (small- and medium-sized) graphs

- Uses Matplotlib for rendering
- [https://networkx.github.io/](https://networkx.github.io/)
- `conda install -c anaconda networkx`

```python
import networkx as nx

G = nx.Graph()
G.add_node("spam")
G.add_edge(1, 2)

print(list(G.nodes()))
print(list(G.edges()))
```

```
[1, 2, 'spam']
[(1, 2)]
```
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?
ADJACENCY LISTS

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

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
ADJACENCY DICTIONARIES

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

Pros: ??????????
• O(1) to add, remove, query edges

Cons: ??????????
• Overhead (memory, caching, etc)

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{C: 1.0}</td>
</tr>
<tr>
<td>B</td>
<td>{C: 1.0, D: 1.0}</td>
</tr>
<tr>
<td>C</td>
<td>{A: 1.0}</td>
</tr>
<tr>
<td>D</td>
<td>{}</td>
</tr>
</tbody>
</table>
**ADJACENCY MATRIX**

Store the connectivity of the graph in a matrix

Cons: ?????????
- \(O(|V|^2)\) space regardless of the number of edges

Almost always stored as a **sparse matrix**
NetworkX uses an adjacency dictionary representation

- Built-ins for reading from/to SciPy/NumPy matrices

```python
# Make a directed 3-cycle
G=nx.DiGraph()
G.add_edges_from([('A','B'), ('B', 'C'), ('C', 'A')])

# Get all out-edges of vertex 'B'
print(G['B'])

# Loop over vertices
for v in G.nodes(): print(v)

# Loop over edges
for u,v in G.edges(): print(u, v)
```
ASIDE: GRAPH DATABASES

Traditional relational databases store relations between entities directly in the data (e.g., foreign keys)
• Queries search data, JOIN over relations

Graph databases directly relate data in the storage system using edges (relations) with attached semantic properties

Image thanks to Wikipedia
Two *people*, John and Sally, are *friends*. Both John and Sally have *read the book*, Graph Databases.

Nodes ???????????
- John
- Sally
- Graph Databases

Thanks to: http://neo4j.com
Two people, John and Sally, are friends. Both John and Sally have read the book, Graph Databases.

A named construct that groups nodes into sets

Labels ???????????
- Person
- Book

Next: assign labels to the nodes
EXAMPLE GRAPH DATABASE

Two people, John and Sally, are friends. Both John and Sally have read the book, Graph Databases.

Relationships ????????

- John is a friend of Sally; Sally is a friend of John
- John has read Graph Databases; Sally has read Graph Databases
EXAMPLE GRAPH DATABASE

Can associate **attributes** with entities in a key-value way

- Attributes on nodes, relations, labels
Querying graph databases needs a language other than SQL.
Recall: graph databases explicitly represent relationships:
- Adhere more to an object-oriented paradigm
- May be more suitable for managing ad-hoc data
- May scale better, depending on the query types (no JOINs)

```
# When did Sally and John become friends?
MATCH (sally:Person { name: 'Sally' })
MATCH (john:Person { name: 'John' })
MATCH (sally)-[r:FRIEND_OF]-(john)
RETURN r.since AS friends_since
```

Cypher query
BULBFLOW

Many graph databases out there:
• List found here: https://en.wikipedia.org/wiki/Graph_database

neo4j and Titan are popular, easy-to-use solutions
• https://neo4j.com/
• http://titan.thinkaurelius.com/

Bulbflow is a Python framework that connects to several backing graph-database servers like neo4j
• http://bulbflow.com/
• https://github.com/espeed/bulbs