CS 111: Program Design I

Lecture 24: Networks & social networks concluded; predictive policing, plotting

Robert H. Sloan & Richard Warner

University of Illinois at Chicago

November 21, 2019

DEGREE DISTRIBUTIONS

How many close (real world) neighbors

- Estimate number of other people living within 100 feet of where you sleep at night:

A. 0–7

B. **8–15** *We got a Bell curve with peak here*

C. 16–32

D. 32–64

E. 65+

How many close neighbors

- Estimate highest number of other people living within 100 feet of bed of anyone in Chicago area not in dorm, prison, military, or hospital

A. 0–32

B. 32–64

C. 66–125 (Most people thought this or less)

D. 128–250

E. 250+

How many FB friends?

- Estimate your number of FB friends (or followers if larger)

A. 0–200

B. 200–400

C. 400–800

D. 800–2000

E. 2000+

Maximum number FB followers?

- Estimate maximum number of followers of most followed *person* on FB?

A. 5000

B. 50,000

C. 500,000

D. 5,000,000

E. 50,000,000

Ronaldo, Shakira

- Vin Diesel has about 97 million followers
- Shakira has about 101 million followers
- Cristiano Ronaldo has about 122 million followers

Power law degree distributions

- # of followers does *not* approximate classic bell curve distribution of
  - Heights of Homo sapiens
  - Times of runners
  - Number of real-world neighbors
  - Perhaps: Number of FB friends of people in this class?

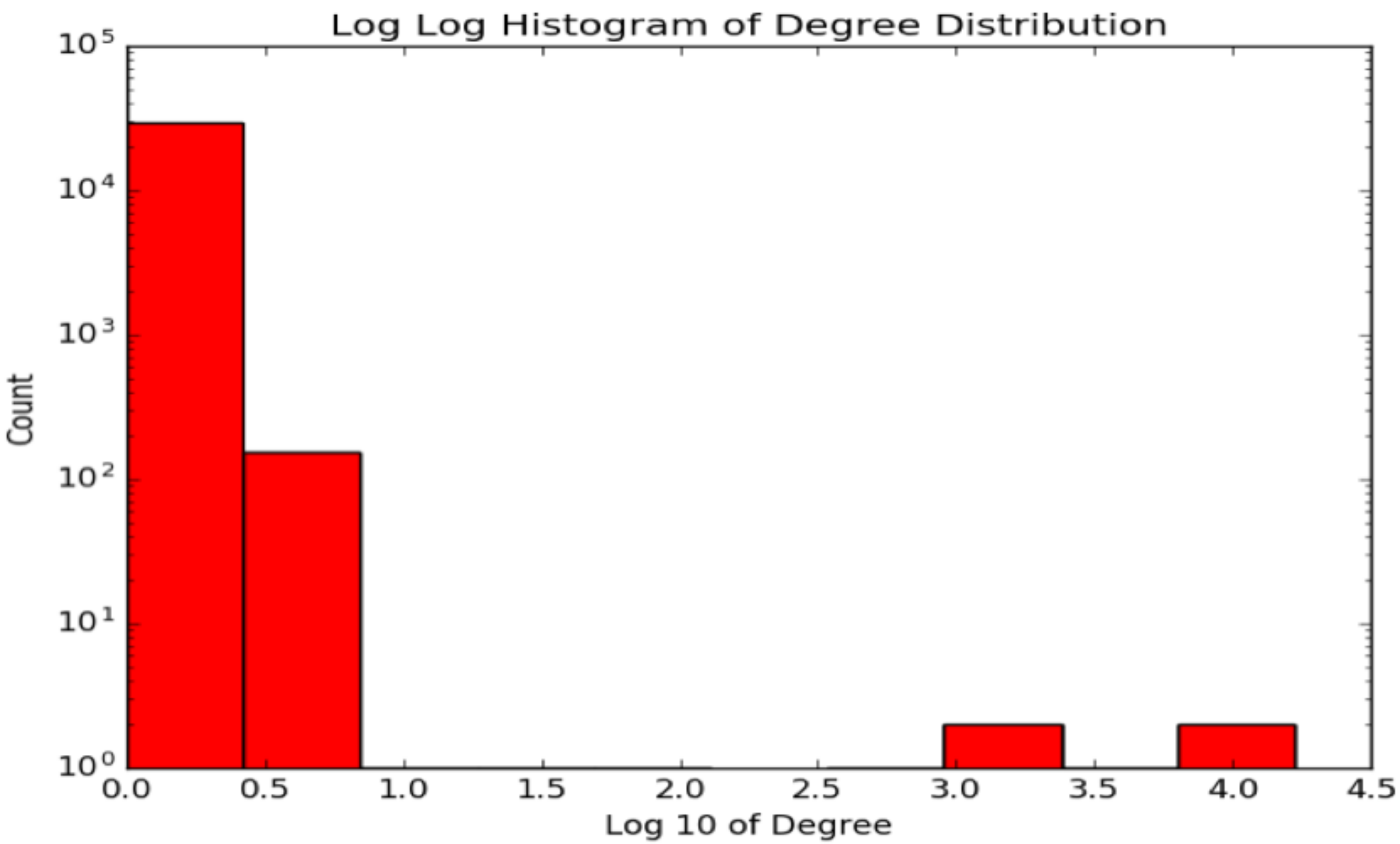
- Compare Ronaldo's FB followers to human height: No 50-foot tall (much less 50 mile tall) outliers!

DEGREE DISTRIBUTION PLOTS (FOR PROJECT)

Power law degree graphs

- Such as social network graphs
- Have *very* large number of low-degree nodes, and very small (but nonzero!) number of *extremely high* degree nodes
- Taking logs can help us view things that are very big
- Will see in lab: count (y) vs. log of degree (x) still hard to see
- Really see something with log vs. log (referred to as "log log")

Log log plot of our twitter data



The code

```
degree_data = pandas.Series(dict(networkx.degree(g)))
log_degree = numpy.log10(degree_data)

log_degree.plot(kind='hist', log=True)
plt.title("Log Log Histogram of Degree Distribution")
plt.xlabel("Log 10 of Degree")
plt.ylabel("Count")
```

Questions on social networks assignment?

PANDAS FILE READING ISSUE RELEVANT TO SOME GRAPH CSV FILES FROM REPOSITORY

Headers and comments up top

- Some graph files start with comments starting with # up top
- And also can have row of column headers.
- Pandas default assumption: 1 row headers, no comments
- Say first 4 lines start with #. Can tell pandas either
  1. Start reading at Python Line 3 (0, 1, 2, 3) *as header* with:
     - header=3
  2. or comment='#', header=None
- file name string type argument okay with either; fileref only with 1!
- *Or* okay to edit file by hand to remove row

pandas read_csv

- Suggestion: use file name version of pd.read_csv
- header: Gives line number to treat as line containing headers, counting lines Pythonically as 0, 1, 2, 3, …
  - Reads headers from that line; *skips earlier lines*; reads data from next line
- comment: character for comment to end of line; all are ignored
  - Next line after comment always taken as header
    - If it's data *must specify* header=None

NESTED LISTS

```
B = [[1,2,3], [5,10,20]]
print(B[1])
```

This will print

Clicker	
A	2
B	[1,2,3]
C	[5,10,20]
D	This will cause an error
E	I don’t know

How confident are you of your answer?

A. Very Highly confident: I've got this

B. Very confident

C. Somewhat confident

D. Not so confident: educated guess

E. Not confident at all: random guess and/or bullied into by the rest of my small group

Matrix

- Famous 1999 Fantasy/Action movie about Neo and the elusive Morpheus
- Way some students believe that they can learn Computer Science: By plugging themselves into it

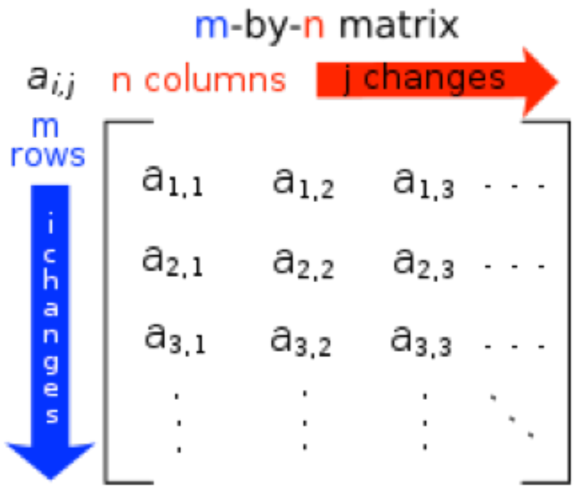
Matrix

- ~~Famous 1999 Fantasy/Action movie about Neo and the elusive Morpheus~~
- ~~Way some students believe that they can learn Computer Science: By plugging themselves into it~~
- **Rectangular array of (usually) numbers, e.g.,**

$$\begin{bmatrix} 1 & 9 & -13 \\ 20 & 5 & -6 \end{bmatrix}.$$

Matrices in Python

- Two common ways to represent:
  - For us: For m-by-n matrix, list of m lists, where each inner nested list is of same length (n) and represents one row
  - (Can also use numpy module)



$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Creating nested list

- Literal notation:

```
matrix = [
    [5, 10, 15, 20, 25],
    [30, 35, 40, 45, 50],
    [55, 60, 65, 70, 75],
    [80, 85, 90, 95, 100],
    [105, 110, 115, 120, 125]
]
```

Building up nested list

- Create *distinct* list of desired row or row of 0s to change later *for each row* append in:

```
matrix = [ ]
for row in range(number_rows):
     new_row = [ ]
     for col in range(number_cols):
          new_row.append(0) #if starting all-0
    matrix.append(new_row)
```

Useful function

```
def make_0array(nrows, ncols):
    '''returns new nrows x ncols 2-d list/array of all 0s'''

    array = [ ] # Build up array of numbers here

    for j in range(nrows):
        new_blank_row = [ ]       # Make a NEW row
        for i in range(ncols):
            new_blank_row.append(0)
        array.append(new_blank_row)
    return array
```

print function: staying on one line (review)

- print() function by default always ends with newline.
  - Not nice to print 2-D m x n array 1 number/line using m*n lines; want whole row per line
- print() has optional argument end= that can give alternate character to put at end instead of newline; e.g., a space:
  - `print (something, end=' ')`

```
def nice_print(A):
    for i in range(len(A)):
        for j in range(len(A[i])):
            print(A[i][j], end=" ")
        print()

A = [[2,5,10],[1,17,0]]
nice_print(A)
```

This will print

Clicker	
A	2 5 10 1 17 0

B	2 1
	17 5
	0 10

C	2 5
	10 1
	17 0

D	This will cause an error
E	I don’t know

How confident are you of your answer?

A. Very Highly confident: I've got this
B. Very confident
C. Somewhat confident
D. Not so confident: educated guess
E. Not confident at all: random guess and/or bullied into by the rest of my small group

```
def col_print(A):
    for i in range(len(A)):
        for j in range(len(A[i])):
            print(A[j][i], end=" ")
        print()

A = [[2,5,10],[1,17,0]]
col_print(A)
```

This will print

Clicker	
A	2 5 10 1 17 0

B	2 1
	17 5
	0 10

C	2 5
	10 1
	17 0

D	This will cause an error
E	I don’t know

Predictive Policing

Contagion Networks

- Viruses spread by contagion over a network of connections.
- Other things spread that way too.
- Chicago’s predictive policing program sees crime as spreading by contagion over networks.
- To predict how things spread, you need to know:
  - Transmission principles—e. g., the virus spreads by contact.
  - The structure of the network.

Very Simple Crime Contagion Example

- Transmission principles:
  - There is one node with a criminal past—”infected” with crime.
    - In the following example, the initial infected node is Bieber, in yellow.
  - Neighbors of infected nodes become infected unless they are immune.
    - In the example, Alice and Ernest are the initial immune node, in blue.
  - Some nodes are immune—cannot be infected by an infected neighbor.
    - A node with only immune neighbors becomes immune.

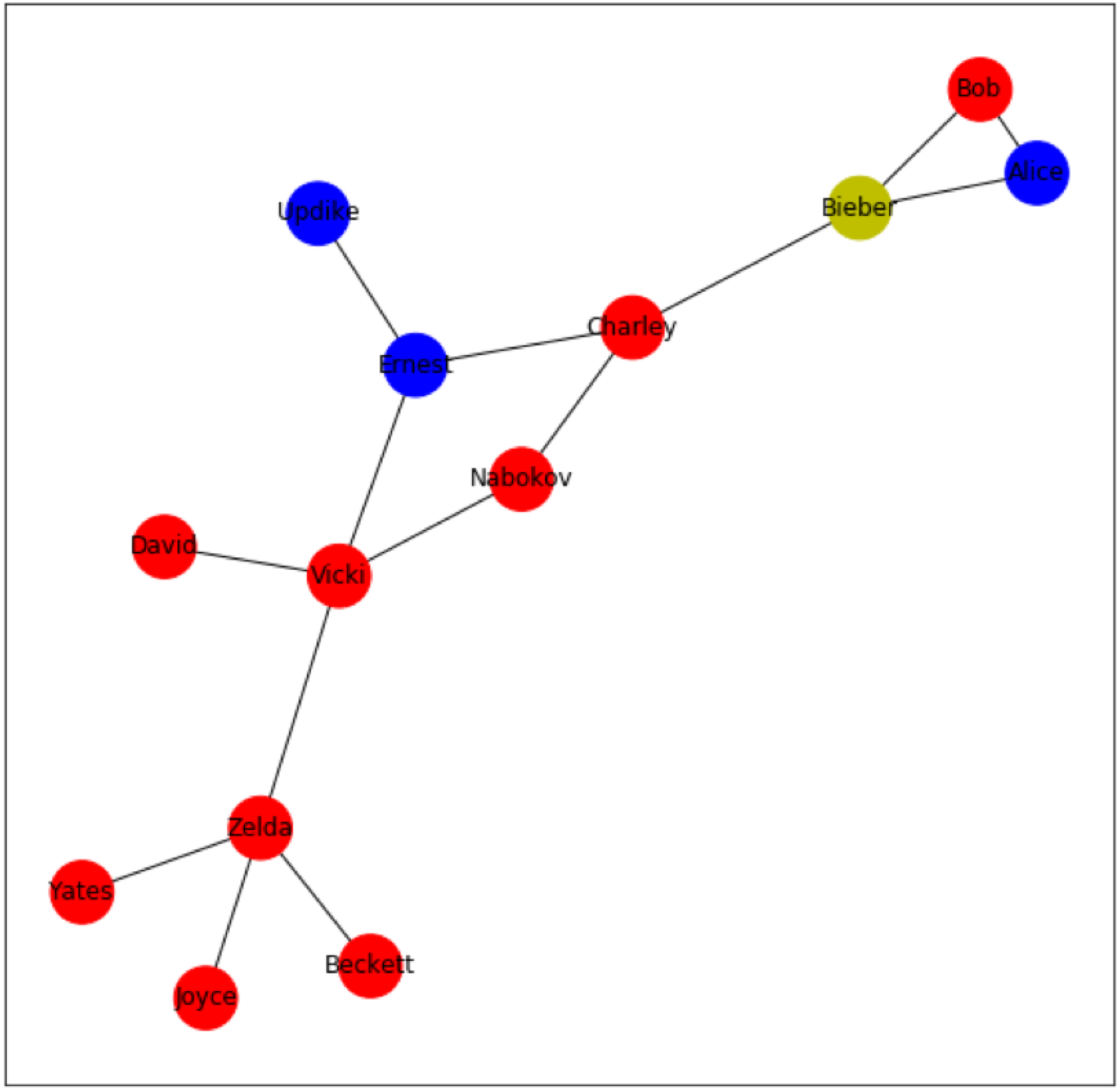
First Example

Yellow (Bieber) = initial infected node

Red = infected

Blue = immune

Alice and Ernest are the initial immune nodes.



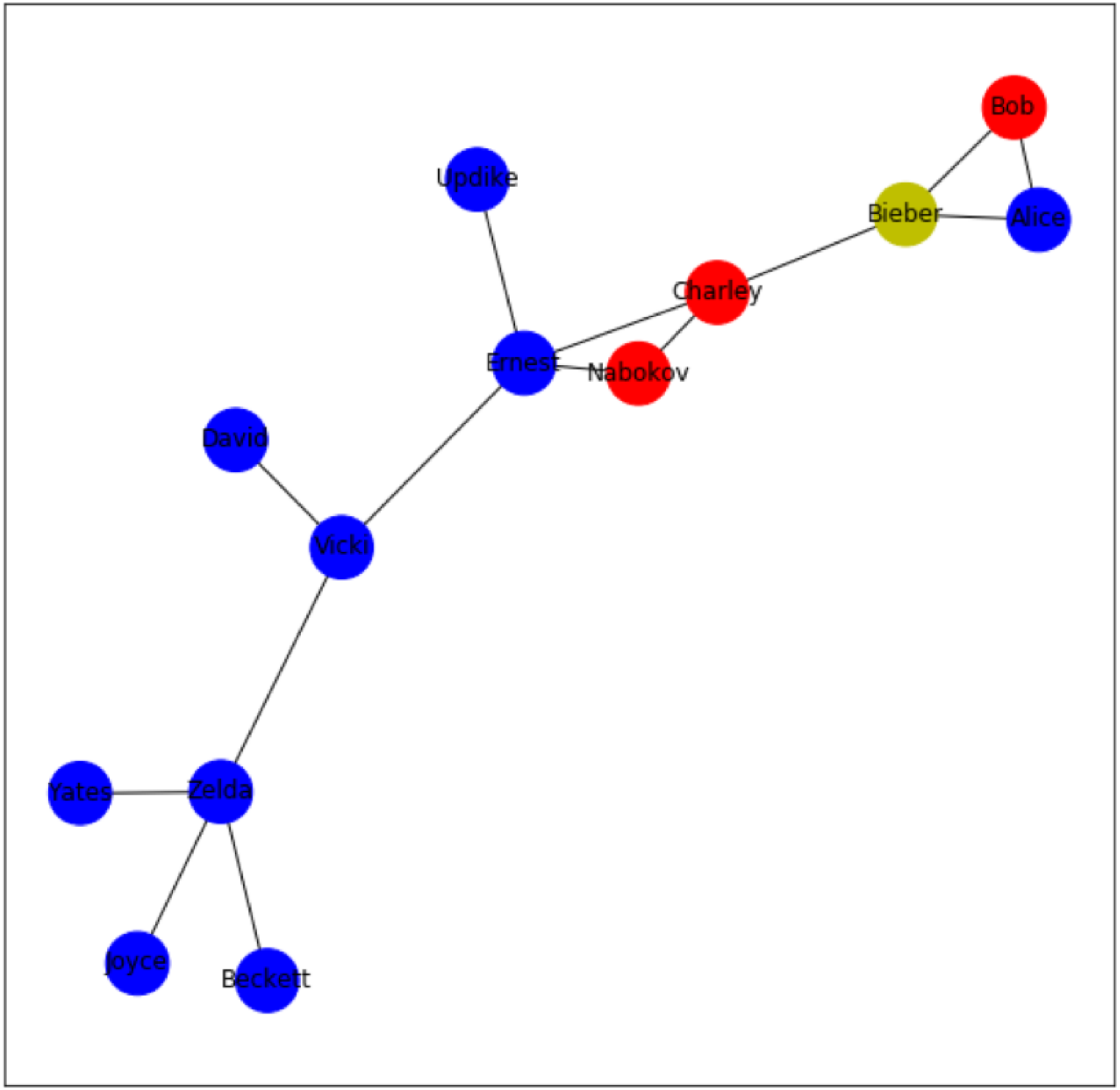
Second Example

Changing the structure changes how the contagion spreads

To draw a network like this:

```
import networkx as nx

pos = nx.circular_layout(g)
pos = nx.spring_layout (g, pos=pos)
nx.draw_networkx(g, pos=pos)
```



Transmission Matters Greatly in Predictive Policing

- We have just seen that structure matters.

- **But** our theory of transmission is *far too simple* to be a model of how crime really spreads.

Chicago’s System

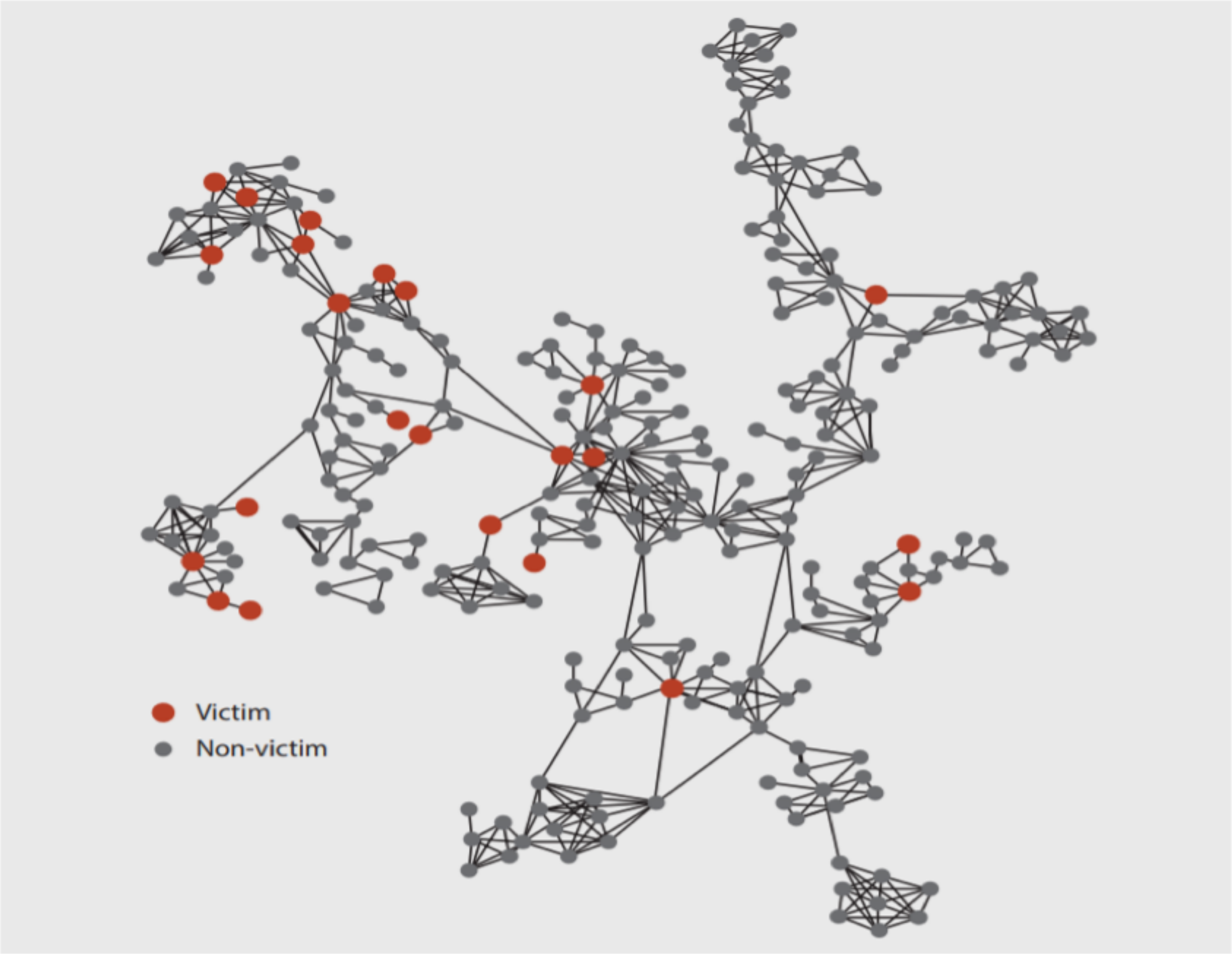
- The Chicago Police’s Strategic Subject List algorithm
  - creates “a risk assessment score known as the Strategic Subject List or ‘SSL.’ Scores “an individual’s probability of being involved in a shooting incident either as a victim or an offender.”
- How do they do this?
  - Not disclosed. But they use a lot of data of various sorts (e. g. social media posts).
  - A reasonable guess:
    - create profiles for “infected,” “susceptible,” and “immune” (all probabilistic).
    - Use network structure and the profiles to generate a score.

The Network Structure

- Uses information about arrests “contained within the CPD data warehouse.”
- From that, the algorithm constructs “social networks . . . to previous homicide victims to predict the likelihood of someone becoming a victim of a homicide.”
- The network is a “co-arrest” network.

A Co-Arrest Network

Figure 3: 200 co-arrest network and gunshot victims in East Palo Alto, California



Co-Arrests

- Generally: co-arrested = arrested together
- Chicago—Two types:
  - X and Y arrested together, Y is murdered later.
    - "A first degree link refers to a relationship between a subject and an individual with whom the subject was previously co-arrested who later became a homicide victim."
  - "X and Y arrested together, and Y later arrested with the murder victim Z.
    - A second degree link refers to a relationship in which a subject was co-arrested with another person who, in turn, was co-arrested with a later homicide victim."

The Underlying Theory

- The more connections you have to co-arrested individuals the more likely you will commit a violent crime or be a victim of one.
- “A series of research studies found that gun violence—just like an infectious disease—can be transmitted from person to person in social networks:
  - i.e., exposure to gun violence not only can lead to a host of negative psychological and cognitive outcomes but also increases the risk of individuals becoming gunshot victims themselves.
  - Furthermore, individuals who associate with a greater number of gunshot victims are at an extremely elevated risk of being victims themselves.”
    - Papachristos and Michael Sierra-Arévalo, *Policing the Connected World*