Network basics
  ▶ Unit of analysis
  ▶ Data representation
  ▶ Analysis & Visualization

Thoughts on research design
  ▶ Edge contexts
  ▶ A bit on social graph data and ethics

Digging into the data
  ▶ Where to get it
  ▶ Our first scrape and build!

Real-time social graph analysis
  ▶ Build the network of Twitter users in the room
  ▶ Hold our breath...
Last week Bruno asked how many of you use Twitter

- 15 (54%) members said yes, which may be enough to do some interesting stuff

Please tweet the following hash-tag:

#analyticsnyc

Some example tweets:

- Many thanks to the @GiltGroupe for hosting tonight’s #analyticsnyc meetup
- Hanging out with @DKALab and @brunocm at #analyticsnyc
- I cannot wait to watch @drewconway crash and burn with his #analyticsnyc live demo
Networks and the study of relationships

Network theory uses the language of graph theory:

\[ G = \{V, E\} \]

In the abstract, this is very powerful, we have a general purpose way to represent any number of relations:

- \[ G = \{\text{Routers, Packet Traffic}\} \]
  *Edges have non-binary values*

- \[ G = \{\text{U.S. Airports, Commercial Routes}\} \]
  *Edges can represent distance, cost, frequency, etc.*

- \[ G = \{\text{New York City nerds, Co-membership in Meetups}\} \]
  *Edges are implied!*

While both nodes and edges are needed to have a network of any substance, it is the edge (relationship) that will always be the primary focus of our analyses.
Consider a very simple example of two people meeting for the first time...

If we focus on the nodes, we might consider this meeting creates a dyad:

We know, however, that people do not exist as isolates and this assumption is ignoring all of exogenous social structure these individuals brings with them.

In reality, the meeting may reveal a large degree of shared structure:

By expressing this meeting in terms of the nodes as a function of their edges we may gain a much richer understanding of the structural dynamics.
Perhaps the most natural way to represent the relationships between $N$ actors is with an $N \times N$ matrix, often referred to as a “sociomatrix”

$$
\begin{array}{c|ccccc}
 & X_1 & X_2 & \ldots & X_N \\
\hline
X_1 & 0 & 1 & \ldots & 0 \\
X_2 & 1 & 0 & \ldots & 1 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
X_N & 0 & 1 & \ldots & 0 \\
\end{array}
$$

- Very intuitive representation
- Can support directional and weighted edges
- Application of matrix algebraic operation for analysis

As an introduction to representing relationship, a matrix provides insight to a network by itself. In practice, however, this representation has many limitations

- Unwieldy as network sizes scale up
- Most networks are sparse; too many zeroes
- Difficult to publish and share

Fortunately, there are many other options for representing network data!
Representing network data: the practical

Remember, it is all about the edges; therefore, more efficient representations will be limited to edge data.

**Edge list**

A text file with two columns (usually delimited by a space or tab), where first column is source, and second is target:

1 2
1 3
2 1
6 7
6 8
10 12
10 13

**Adjacency list**

Also a text file, however, here the first column is the source, and all subsequent entries are “adjacent” nodes:

1 2 3
2 1
6 7 8
10 12 13

While these are the most universal data formats, network data representation is a bit of a cottage industry:

- Pajek (.net)
- GraphML (.gml)
- GraphViz (.dot)
- ..and domain specific formats (eek!)
The number of software suites and packages available for conducting social network analysis has exploded over the past ten years.

In general, this software can be categorized in two ways:

- **Type** - many SNA tools are developed to be standalone applications, while others are language specific packages.
- **Intent** - consumers and producer of SNA come from a wide range of technical expertise and/or need, therefore, there exist simple tools for data collection and basic analysis, as well as complex suites for advanced research.

### Standalone Apps

<table>
<thead>
<tr>
<th>Basic</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>- ORA (Windows)</td>
<td>- UCINet (Windows)</td>
</tr>
<tr>
<td>- Analyst Notebook (Windows)</td>
<td>- Pajek (Multi)</td>
</tr>
<tr>
<td>- KrakPlot (Windows)</td>
<td>- Network Workbench (Multi)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modules &amp; Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>- libSNA (Python)</td>
</tr>
<tr>
<td>- UrlNet (Python)</td>
</tr>
<tr>
<td>- NodeXL (MS Excel)</td>
</tr>
<tr>
<td>- NetworkX (Python)</td>
</tr>
<tr>
<td>- JUNG (Java)</td>
</tr>
<tr>
<td>- igraph (Python, R &amp; Ruby)</td>
</tr>
</tbody>
</table>

Many of the above tools have visualization components, but several tools are designed specifically for visualization: Graphviz, NetDraw, Tom Sawyer, Gephi, etc.

**What I use**

Drew Conway
Often network analysis is used to identify key actors within a social group. To identify these actors, various centrality metrics can be computed based on a network’s structure:

- **Degree** (number of connections)
- **Betweenness** (number of shortest paths an actor is on)
- **Closeness** (relative distance to all other actors)
- **Eigenvector centrality** (leading eigenvector of sociomatrix)

One method for using these metrics to identify key actors is to plot actors’ scores for Eigenvector centrality versus Betweenness. Theoretically, these metrics should be approximately linear; therefore, any non-linear outliers will be of note.

- An actor with very high betweenness but low EC may be a critical gatekeeper to a central actor
- Likewise, an actor with low betweenness but high EC may have unique access to central actors
First, visualize the data

Visualization can be the best first step in the analytical process

- This will give you a good feel for what is going on with your relationships.
- For large networks this is often not possible.
- For this example, we will use the main component of the social network collected on drug users in Hartford, CT. The network has 194 nodes and 273 edges.
- Here I am using the GUESS visualizer in NWB with a Kamada-Kawai layout.
The first steps are to load the data into memory, and perform some basic centrality analysis. 

### Load the data into igraph

```r
library(igraph)
G<-read.graph("drug_main.txt",format="edgelist")
G<-as.undirected(G)
# By default, igraph inputs edgelist data as a directed graph.
# In this step, we undo this and assume that all relationships are reciprocal.
```

### Store metrics in new data frame

```r
cent<-data.frame(bet=betweenness(G),eig=evcent(G)$vector)
# evcent returns lots of data associated with the EC, but we only need the
# leading eigenvector
res<-lm(eig~bet,data=cent)$residuals
cent<-transform(cent,res=res)
# We will use the residuals in the next step
```

---

Finding Key Actors with R

Plot the data

```r
library(ggplot2)
# We use ggplot2 to make things a
# bit prettier
p <- ggplot(cent, aes(x=bet, y=eig,
    label=rownames(cent), colour=res,
    size=abs(res)))+
xlab("Betweenness Centrality")+
ylab("Eigenvector Centrality")
# We use the residuals to color and
# shape the points of our plot,
# making it easier to spot outliers.
p+geom_text()+opts(title="Key Actor Analysis for Hartford Drug Users")
# We use the geom_text function to plot
# the actors' ID's rather than points
# so we know who is who
```

Key Actor Analysis for Hartford Drug Users

Betweenness Centrality
Eigenvector Centrality

res
-0.2
0
0.2
0.4
0.6
abs(res)
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Network basics

Key Actor Plot

Drew Conway

Mining and Analyzing Online Social Graph Data

# Create positions for all of the nodes w/ force directed
l<-layout.fruchterman.reingold(G, niter=500)

# Set the nodes’ size relative to their residual value
V(G)$size<-abs(res)*10

# Only display the labels of key players
nodes<-as.vector(V(G)+1)

# Key players defined as have a residual value > .25
nodes[which(abs(res)<.25)]<-NA

# Save plot as PDF
pdf(‘actor_plot.pdf’,pointsize=7)
plot(G,layout=l,vertex.label=nodes, vertex.label.dist=0.25, vertex.label.color='red',edge.width=1)
dev.off()
I have spent a lot of time tonight describing network analysis as a way to understand relationships.

- Depending on the source and context of the data, these relationships can be interpreted as many different things.
- Must consider the **data generation** process by which the edge was created.
- With respect to online social graph data, ask: how do people use this service?

**Twitter**

Recent study shows that Twitter is used as a news aggregation service.

- Ties here are driven by personal contact: me → offline → you.
- Geography and history less important.
- Networks may cluster around communities of interest.

**Facebook**

- Ties here are driven by personal contact: me → offline → you.
- Offline relationships are driving “friendeing”:
  - Geographical and historical information less important.
  - Networks may cluster around communities of interest.
  - Considerable amount of meta-data already in FB, what does this add?

**Google SocialGraph**

Connecting all of Google’s social sites together:

- Ties here are driven by personal contact: me → anything → you.
- The combining of multiple platforms into a single “network” makes analysis and interpretation difficult.
Ethics and network data

Why should we be discuss ethics at a data analytics seminar?

“People have really gotten comfortable not only sharing more information and different kinds, but more openly and with more people. That social norm is just something that has evolved over time.”

“Money is a terrible master but an excellent servant”

“There’s a sucker born every minute”

Mark Zuckerberg, CEO, Facebook

P.T. Barnum

In isolation, the data we provide online about our relationships and preferences are fairly innocuous. Their summation, however, can illuminate aspects of our lives that can be exploited for any number of ways.

▶ Social networking services are moving previously private data to the public
▶ Simply because data is available does not mean that individuals want or realize that it can be used to make inferences about who they are or their lifestyles
▶ Analysts are caught in the middle. There is no IRB on the Internet!
Examples of ethically questionable network analyses

MIT study predicts sexual orientation from Facebook friends
- Two undergraduate students start a project ‘Gaydar’
- Claim they can predict which men are homosexual simply based on their friend structure
- **Problem**: Extremely private information, questionable methods

Pleaserobme.com
- Combined information from foursquare.com and Twitter to publicize when people were clearly not at home
- Used as a proof of concept to show the danger of publicizing localization information
- **Problem**: Useful for raising awareness, hurtful to anyone who was actually robbed

Project Grey Goose
- Large group of former and current DoD/IC analysts collaborated to study the identity of hackers
- I was personally involved in this project
- Using public web forum data, built up user profiles and social networks to attempt to identify hackers affiliated with Russian government
- **Problem**: While no names were published, bordering on vigilantism
Where to get social graph data

Recently, there has been an explosion of resources for scraping social graph data.

<table>
<thead>
<tr>
<th>Service</th>
<th>Data</th>
<th>API Docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>twitter</td>
<td>Following(ers), @-replies, date/time/geo</td>
<td><a href="http://apiwiki.twitter.com/">http://apiwiki.twitter.com/</a></td>
</tr>
<tr>
<td>facebook</td>
<td>Friends, Wall Posts, date/time</td>
<td><a href="http://developers.facebook.com/docs/api">http://developers.facebook.com/docs/api</a></td>
</tr>
<tr>
<td>Google</td>
<td>All SocialGraph relationships</td>
<td><a href="http://code.google.com/apis/socialgraph/">http://code.google.com/apis/socialgraph/</a></td>
</tr>
</tbody>
</table>

There is clearly no shortage of data:

- Each service provides different relational context
- Data formats are generally JSON, Atom, XML, or some combination
- For a more extensive list of API resources, see HackNY wiki of local startups
Using a “seed” user, we will build out a network

- In Python, use NetworkX, cjson and other standard scientific libraries, parse the SocialGraph data
- Through a process called “k-snowball searching”
  - seed → friend → ··· → friend_k
    - Seed: imichaeldotorg.livejournal.com
    - k = 3
- Note the low value of k
The code, part 1

Loading the libraries and setting things up

```python
from cjson import *
from urllib import *
from networkx import *
from time import *
from scipy import array, unique
...
if __name__ == "__main__":
    seed_url='http://imichaeldotorg.livejournal.com'
    sg=get_sg(seed_url)
    net,newnodes=create_egonet(sg)
    info(net)
```

Get the JSON from SocialGraph

```python
def get_sg(seed_url):
    sgapi_url="http://socialgraph.apis.google.com/lookup?q="+seed_url+"&edo=1&edi=1&fme=1&pretty=0"
    try:
        furl=urlopen(sgapi_url)
        fr=furl.read()
        furl.close()
        return fr
    except IOError:
        print "Could not connect to website"
        print sgapi_url
        return
```

Name: ['http://imichaeldotorg.livejournal.com/']
Type: DiGraph
Number of nodes: 5
Number of edges: 5
Average in degree: 1.0
Average out degree: 1.0
def create_egonet(s):
    try:
        raw=decode(s)
        G=DiGraph()
        pendants=[]
        n=raw['nodes']
        nk=n.keys()
        G.name=str(nk)
        pendants=[]
        for a in range(0,len(nk)):
            for b in range(0,len(nk)):
                if a!=b:
                    G.add_edge(nk[a],nk[b])
        for k in nk:
            ego=n[k]
            ego_out=ego['nodes_referenced']
            for o in ego_out:
                G.add_edge(k,o)
                pendants.append(o)
        ego_in=ego['nodes_referenced_by']
        for i in ego_in:
            G.add_edge(i,k)
            pendants.append(i)
        pendants=array(pendants,dtype=str)
        pendants.flatten()
        pendants=unique(pendants)
        return G,pendants
    except DecodeError:
        ...
    except KeyError:
        ...

def snowball_round(G,seeds,myspace=False):
    t0=time()
    if myspace:
        seeds=get_myspace_url(seeds)
        sb_data=[]
    for s in range(0,len(seeds)):
        s_sg=get_sg(seeds[s])
        new_ego,pen=create_egonet(s_sg)
        for p in pen:
            sb_data.append(p)
        if s<1:
            sb_net=compose(G,new_ego)
        else:
            sb_net=compose(new_ego,sb_net)
        del new_ego
        if s==round(len(seeds)*0.2):
            sb_net.name='20% complete'
            sb_net.info()
            print 'AT: '+strftime('%m/%d/%Y, %H:%M:%S', gmtime())
            print ''
    # More time keeping, probably a MUCH better way to do this
    sb_data=array(sb_data)
    sb_data.flatten()
    sb_data=unique(sb_data)
    sb_net.info()
    return sb_net,sb_data
### Build the whole network

<table>
<thead>
<tr>
<th>Step</th>
<th>Nodes</th>
<th>Edges</th>
<th>Mean Degree</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seed</td>
<td>5</td>
<td>5</td>
<td>2.0</td>
<td>0.25</td>
</tr>
<tr>
<td>$k = 2$</td>
<td>75</td>
<td>115</td>
<td>3.0</td>
<td>0.02</td>
</tr>
<tr>
<td>$k = 3$</td>
<td>4,938</td>
<td>8,659</td>
<td>3.5</td>
<td>$3.6(10^{-4})$</td>
</tr>
</tbody>
</table>

- Our seed is abnormally isolated, with only four neighbors
- Large jump after first snowball
- Massive structural leap at $k = 3$
The full network

To get a feeling for the size of the full network...
Live demonstration time!