Big Data BUS 41201

Week 6: Networks

Veronika Ročková
University of Chicago Booth School of Business

http://faculty.chicagobooth.edu/veronika.rockova/
Networks

✓ Network data examples
✓ Graph representations
✓ Summarization: nodes and edges, direction
✓ Measuring connectivity and betweenness
✓ Community detection
✓ Market basket analysis
✓ Page Rank for relevance ordering
The network has **nodes** (vertices), such as a website or worker, and **edges** are the (directed or undirected) links between nodes.
Internet — 50 billion Webpages

www.opte.org  www.worldwidewebsize.com
Facebook — 1.2 Billion Users
Media networks

Connections between political blogs (Adamic, Glance, 2005)
Organizational networks

9/11 terrorist network (Krebs, 2002)

Figure 2 - All nodes within 1 step [direct link] of original suspects
Many more examples

who follows whom?

who calls whom?

who buys what?
Network Analysis

A rich set of tools to help us understand complex relationships, learn about behaviors, preferences and trends.

We’ll begin by summarizing important properties.

In particular, we’ll focus on measures of network connectivity.

Each node has connectivity statistics

**Degree**: How many other nodes are you connected to?

**Betweenness**: How many node-to-node paths go through you?

You can also make a lot of cool illustrations for graphs. However, how to effectively visualize large networks is an open problem.
Basics: How to represent a network?

We will use graphs, which consist of vertices and edges.

Visually

Adjacency matrix

\[ A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \]

Adjacency list

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

Edge list

\{(1, 2), (1, 5), (2, 3), (2, 4), (3, 5), (3, 4), (3, 6)\}
More exotic networks

Visually

Adjacency matrix

<table>
<thead>
<tr>
<th>Source node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Adjacency list

<table>
<thead>
<tr>
<th>Target node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Edge list

1, 2, 1
1, 3, 3
2, 4, 2
3, 2, 1
Network Graphs in R

`igraph` is a toolbox for visualizing and summarizing graphs. It has front-ends for R and Python. Others: Gephi, Pajek, etc.

Unlike most R packages, `igraph` is well documented. Type `help(igraph)` to get started.

For most applications, you’ll read graphs from an edgelist:

```
1 2
1 3
2 1
3 2
```

```
edgemat <- as.matrix(read.table("edgelist.txt"))
graph <- graph.edgelist(edgemat)
```
Descriptive statistics of networks

Degree of a node

Number of edges connected to a node $d_i = \sum_j A_{ij}$

$$A = \begin{pmatrix}
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{pmatrix}$$

Degree distribution

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\Pr(k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/6</td>
</tr>
<tr>
<td>2</td>
<td>3/6</td>
</tr>
<tr>
<td>3</td>
<td>1/6</td>
</tr>
<tr>
<td>4</td>
<td>1/6</td>
</tr>
</tbody>
</table>
Marriage and Power

Early Renaissance Florence was ruled by an oligarchy of powerful families.

Padget and Ansell: “Robust Action and the Rise of Medici”

By the 15th century, the Medicis emerged supreme, & Medici Bank became the largest in Europe.

Political ties were established via marriage. How did Medici win?
Network links can be used to measure “social capital”.

A node’s degree is its number of edges.

> sort(degree(marriage))
Ginori ... Strozzi Medici
1 4 6

Medicis are connected!
The Medici network is very centralized (a star shaped network)

\[ \Rightarrow \text{Medici relatives were connected almost solely through Medici’s themselves.} \]

The rival Strozzi’s network is far denser (competing claims for network leadership).
Deeper network structure with **betweenness**

A node is important if it lies on many shortest-paths so it is essential in passing information through the network.

An alternative to degree, **betweenness** measure the proportion of **shortest paths** containing a given node.

**Shortest path:** fewest steps from $i$ to $j$ (direction matters).

\[
\text{betweenness}(k) = \sum_{i,j: i \neq j, k \notin \{i,j\}} \frac{s_{ij}(k)}{s_{ij}}
\]

\[s_{ij}(k)\] number of shortest paths from $i$ to $j$ that go through $k$

\[s_{ij}\] number of shortest paths from $i$ to $j$

Measures how much influence a node has over connections between others, i.e. how often a node serves as the “bridge”.
Betweenness vs Degree

Medicis have the highest degree, but only by a factor of 3/2 over the Strozzi.

But their betweenness is 5 times higher!

Betweenness measures deep graph connectivity, rather than just counting neighbors.

> sort(betweenness(marriage))
        Ginori ... Strozzi Medici
                0.0    9.3   47.5
Structural Holes

A structural hole is a low-degree node in an organization chart with high betweenness.

Social Capital exists where people have an advantage because of their location in a social structure (brokerage opportunities).

Holes can act like bottlenecks in companies, and lead to unexpected employees having excess power and influence. But if you’re the employee, it’s a fast track to promotion.

“People who stand near the holes in a social structure are at higher risk of having good ideas.”


igraph has constraint for finding structural holes.
Community Detection

Networks are often organized into **communities**: groups of nodes that are *densely connected inside* but only *loosely connected across*.

The goal is to identify meaningful communities in an automated way.
Karate Club Example

Zachary’s karate club network (H: Instructor, A: Club president)

http://networkkarate.tumblr.com/
Karate club: Adjacency matrix

Adjacency matrix: Karate Club

![Adjacency matrix: Karate Club](image-url)
### Karate club: Reordered adjacency matrix

#### Permuted Adjacency matrix: Karate Club

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Permuted Adjacency matrix: Karate Club](image)
Edge Betweenness and Community Detection

The *edge betweenness* is a measure of traffic flow through an edge rather than a node.

It is the proportion of shortest paths between pairs of nodes that run through the edge, summed over all possible node pairs.

If a network contains communities that are only loosely connected, then all shortest paths between different communities must go along only a few edges.

Community detection can be achieved by removing high-betweenness edges in a graph \(\rightsquigarrow\) *Girvan-Newman algorithm*

1. Calculate betweenness of all edges in a graph.
2. Remove the edge with the highest betweenness.
3. Repeat (1) until no edges remain.
Community Detection: Edge Betweenness

Successively remove edges of highest betweenness, breaking up the network into separate components

(a) Step 1

(b) Step 2
Collaborative Filtering

A common question in data mining: what do one person’s choices say about anothers? As amazon says: “people who buy this book also bought...”

These types of tasks are referred to as ‘collaborative filtering’: using shared choices to predict preferences.

It’s a big field, with many tools

- logistic regression of each product on to all other choices.
- principal components analysis: underlying taste factors.

Many of the tools from this class apply (projects?).

But as an easy start, there are good fast algorithms for discovering low dimensional association rules.
Association Rules

_Frequent itemset mining_ is a popular tool for discovering purchasing patterns from large commercial databases.

Consider \( X = \{X_1, \ldots, X_p\} \) a set of _binary attributes_ called _items_.

Let \( T = \{T_1, \ldots, T_n\} \) be a _database of transactions_, where each transaction \( T_i \) is associated with an item set \( S_i \) (subset of variables in \( X \) for which \( X_j = 1 \)).

For two item sets \( S_i \) and \( S_j \), we define an implication

\[
S_i \Rightarrow S_j
\]

as an _association rule_ when \( S_j \) occurs more frequently when \( S_i \) occurs.

The LHS item \( S_i \) is the _antecedent_ and the RHS item \( S_j \) is the _consequent_.

Association Rules: Toy Example

The goal is to find interesting item co-occurrences and build association rules.

Assume $X = \{\text{milk, bread, butter, beer}\}$.

Database of transactions

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>milk, bread</td>
</tr>
<tr>
<td>2</td>
<td>bread, butter</td>
</tr>
<tr>
<td>3</td>
<td>beer</td>
</tr>
<tr>
<td>4</td>
<td>milk, bread, butter</td>
</tr>
<tr>
<td>5</td>
<td>bread, butter</td>
</tr>
</tbody>
</table>

An example of rule can be

$$\{\text{milk, bread}\} \Rightarrow \text{butter}.$$
Association Rules

There are various measures of informativeness among item sets and association rules.

*Support of item set* $S$: $\text{supp}(S)$ is the proportion of transactions that contain $S$

\[
\text{supp}({\{\text{milk, bread}\}}) = \frac{2}{5}
\]

It is an estimate of the probability of purchasing {milk, bread}

*Confidence of a rule* $S_i \Rightarrow S_j$ is the proportion of $S_i$ transactions that include also $S_j$:

\[
\text{conf}(S_i \Rightarrow S_j) = \frac{\text{supp}(S_i \text{ and } S_j)}{\text{supp}(S_i)}
\]

\[
\text{conf}({\{\text{milk, bread}\}} \Rightarrow {\{\text{butter}\}}) = \frac{\text{supp}({\{\text{milk, bread, butter}\}})}{\text{supp}({\{\text{milk, bread}\}})} = \frac{1}{2}
\]

It is an estimate of the *conditional* probability of purchasing butter *given* {milk, bread}
Association Rules

Association rules (AR’s) should have enough of support and confidence. To narrow down focus to even fewer and even more interesting AR’s, one can use *lift*.

*Lift of a rule* \( S_i \Rightarrow S_j \) measures the increase in support of \( S_j \) when \( S_i \) occurs

\[
\text{lift}(S_i \Rightarrow S_j) = \frac{\text{supp}(S_i \text{ and } S_j)}{\text{supp}(S_i)\text{supp}(S_j)} = \frac{\text{conf}(S_i \Rightarrow S_j)}{\text{supp}(S_j)}
\]

It can be regarded as a ratio of a *conditional* probability of purchasing butter, *given* \{milk, bread\}, and an *unconditional* probability of purchasing butter.

\[
\text{lift}(\{\text{milk, bread}\} \Rightarrow \{\text{butter}\}) = \frac{1}{2} \times \frac{5}{3}
\]

Greater lift values indicate stronger associations.
Support, Association, and Lift

Ex: when you buy chips, you need beer to wash them down.

Suppose that beer is purchased 10% of the time in general, but 50% of the time when the consumer grabs chips.

- The support for ‘beer’ is 10%
- The confidence of this rule is 50%.
- It’s lift is 5: 50% is 5 times higher than 10%.

Given this information, you could put some chips by the beer.

Generally, association rules with high lift are most useful.

Low support does not preclude high confidence or high lift.

Chips ⇒ Beer is high support, but low lift if everybody always buys beer.

Vodka ⇒ Caviar is low support, but high lift if people caviar for their parties.
Finding Association Rules with R

There’s no deep theory around ARules. We just scan the high-lift or high-confidence rules to find interesting rules.

To find confidence and lift, just count the number of times RHS and LHS happen, and how often they happen together.

\[
supp(event) = \frac{\text{number of times event occurs}}{\text{number of observations}}
\]

However, counting all possible combinations can take forever. **Apriori**: algorithm for finding rules over a support threshold.

The `apriori` function is available in the `arules` package. You need to get the data in a certain format, but after this it is straightforward to use.
## Binary Incidence Matrix

Convenient representation of transactions with binary incidence matrix.

<table>
<thead>
<tr>
<th>Transaction</th>
<th>$X_1$ milk</th>
<th>$X_2$ bread</th>
<th>$X_3$ butter</th>
<th>$X_4$ beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$T_2$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$T_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$T_4$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$T_5$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The number of purchase opportunities as well as products can be huge. Coding as a sparse matrix is extremely helpful.
**Last.fm Artist Plays**

Online radio keeps track of everything you play, for recommending music & focused marketing.

This ‘network’ shows artists sized by play count, with lines (edges) for shared users.

metal, rock, pop, jazz, electronica, hip-hop reggae/ska, classical, folk/country/world.
## Association rules for Music Taste

<table>
<thead>
<tr>
<th>lhs</th>
<th>rhs</th>
<th>support</th>
<th>confidence</th>
<th>lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>t.i. =&gt; kanye west</td>
<td>0.0104</td>
<td>0.5672</td>
<td>8.8544</td>
<td></td>
</tr>
<tr>
<td>pink floyd, the doors =&gt; led zeppelin</td>
<td>0.0106</td>
<td>0.5387</td>
<td>6.8020</td>
<td></td>
</tr>
<tr>
<td>beyonce =&gt; rihanna</td>
<td>0.0139</td>
<td>0.4686</td>
<td>10.8810</td>
<td></td>
</tr>
<tr>
<td>morrissey =&gt; the smiths</td>
<td>0.0112</td>
<td>0.4655</td>
<td>8.8961</td>
<td></td>
</tr>
<tr>
<td>megadeth =&gt; iron maiden</td>
<td>0.0132</td>
<td>0.4307</td>
<td>7.2677</td>
<td></td>
</tr>
<tr>
<td>jimi hendrix =&gt; the doors</td>
<td>0.0120</td>
<td>0.3062</td>
<td>5.3170</td>
<td></td>
</tr>
<tr>
<td>nelly furtado =&gt; madonna</td>
<td>0.0100</td>
<td>0.2750</td>
<td>5.0374</td>
<td></td>
</tr>
<tr>
<td>bright eyes =&gt; the shins</td>
<td>0.0102</td>
<td>0.2698</td>
<td>5.4623</td>
<td></td>
</tr>
<tr>
<td>elliott smith =&gt; modest mouse</td>
<td>0.0109</td>
<td>0.2679</td>
<td>5.1732</td>
<td></td>
</tr>
<tr>
<td>britney spears =&gt; lady gaga</td>
<td>0.0120</td>
<td>0.2612</td>
<td>7.7292</td>
<td></td>
</tr>
<tr>
<td>ramones =&gt; the clash</td>
<td>0.0104</td>
<td>0.2586</td>
<td>5.9052</td>
<td></td>
</tr>
<tr>
<td>franz ferdinand =&gt; kaiser chiefs</td>
<td>0.0132</td>
<td>0.2224</td>
<td>7.1153</td>
<td></td>
</tr>
</tbody>
</table>

**Example:** Given a new user that listens to a lot of Morrissey, we’re 46% positive that they’ll also like the Smiths; This is 9 times higher than if we didn’t know about Morrissey.
From association to networks

Graphs can be a useful way to summarize all sorts of data. We can define networks using any measure of connectivity.

For example, an association network:

Say there's an edge between \( \text{lhs} \) and \( \text{rhs} \) if \( \text{support} \) and \( \text{confidence} \) are greater than some thresholds.

If we just look at any shared membership in a playlist, we get our monster graph from the beginning.

For example, in the \texttt{lastfm.R} code we use rules from

\begin{verbatim}
apriori(playtrans,  
    parameter=list(support=.001, confidence=.1, maxlen=2))
\end{verbatim}

...to define a network with 1k nodes and 36k edges.
0.1% support and 10% confidence lastfm network

The network has four very distinct cliques.

These look something like metal, hip-hop, alt, pop.

See code for plotting.
There’s lots you can do.
Search for “California”

*Search is one great example of network analysis;*

Consider ranking sites for the query “California”.

⇾ Take 200 pages with heavy traffic and high term-frequency for “California”.

⇾ Follow links to build a neighborhood.

We’re left with about 10,000 sites, with links, to rank.

caedges <- read.csv("CaliforniaEdges.csv")
casites <- scan("CaliforniaNodes.txt", "character")
edgemat <- cbind(casites[caedges$from], casites[caedges$to])
The query provides a very large directed network
Look at neighborhoods to get a workable plot.

```
latimes <- graph.neighborhood(calink, order=1, ...)
```

**LA Times network**

Neighborhood order is the number of included steps away from the node.

At order=1, we just have sites pointing to latimes.com.
Just going one extra step creates a much bigger network. Yellow points are the first-order connections from before.
Google’s PageRank Algorithm

Google has been pretty successful. Page Rank is a key ingredient.

Suppose we have $N$ webpages and we want to rank them in terms of their likely relevance to the websurfer.

PageRank algorithm, famously invented by Larry Page and Sergei Brin (founders of Google), assigns a score to each webpage.

Paper: The Anatomy of a Large-Scale Hypertextual Web Search Engine
PageRank Algorithm

Page Rank labels a site more important *if many sites link to it*.

However, we don’t want to treat all linking webpages equally.

Instead, we *weight* the links from different webpages

1. Webpages that link to \(i\), and have *high PageRank* scores themselves, should be given *more weight*

2. Webpages that link to \(i\), but *link to a lot of other webpages* in general, should be given *less weight*

The rank \(r_i\) of \(i^{th}\) page depends on all the other ranks \(r_j\) of pages that point to it \(\Rightarrow\) circular argument.
PageRank Algorithm

The circularity can be exploited in recursive calculations.

\[ r_i = \sum_{j=1}^{N} \frac{e_{ij}}{c_j} r_j \]

\( r_i \) is the page rank,
\( e_{ij} \) is a binary edge indicator, and
\( c_j = \sum_{i=1}^{N} e_{ij} \) is the number of nodes pointed at by node j.

Start with an initial vector \( r^{(0)} \) and update \( r^{(k)} = A r^{(k-1)} \) until convergence, where \( A \) is a sparse weighted adjacency matrix.

Very efficient and simple calculation.
Page rank of “california” search response

We can run PageRank to organize our list of sites.

```r
> search <- page.rank(calink)$vector
> casites[order(search, decreasing=TRUE)]
```

I don’t think this alone would have made google famous!
Nodes need to be weighted by traffic and by successful clicks.