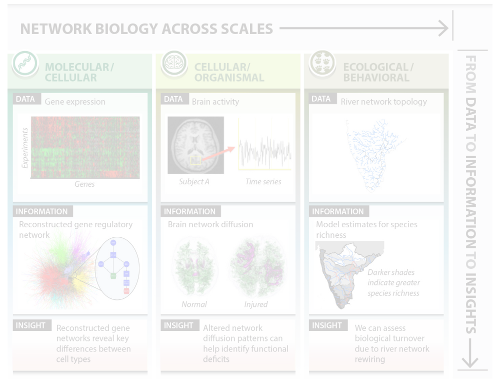

NETWORK BIOLOGY ACROSS SCALES
MOLECULAR/ CELLULAR
CELLULAR/ ORGANISMAL
ECOLOGICAL/ BEHAVIORAL
DATA
Gene expression
Brain activity
River network topology
INFORMATION
Reconstructed gene regulatory network
Brain network diffusion
Model estimates for species richness
INSIGHT
Reconstructed gene networks reveal key differences between cell types
Altered network diffusion patterns can help identify functional deficits
We can assess biological turnover due to river network rewiring
FROM DATA TO INFORMATION TO INSIGHTS

Network Preliminaries

Héctor Corrada Bravo

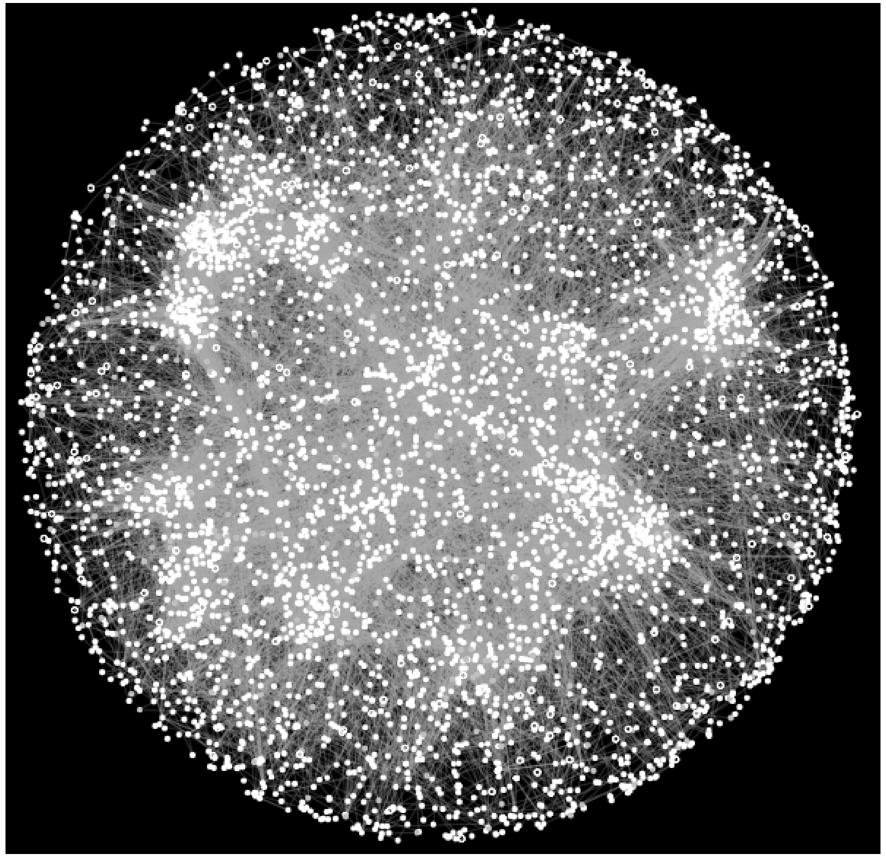
University of Maryland, College Park, USA

CMSC828O 2019-09-04

Genetic Interaction Network

- Yeast high-throuput double-knockdown assay
- ~5000 genes
- ~800k interactions

http://www.geneticinteractions.org/



Costanzo et al. (2016) Science. DOI: 10.1126/science.aaf1420

Genetic Interaction Network

- Yeast high-throuput double-knockdown assay
- ~5000 genes
- ~800k interactions

http://www.geneticinteractions.org/

Costanzo et al. (2016) Science. DOI: 10.1126/science.aaf1420

Genetic Interaction Network

- Number of vertices: 2803
- Number of edges: 67,268

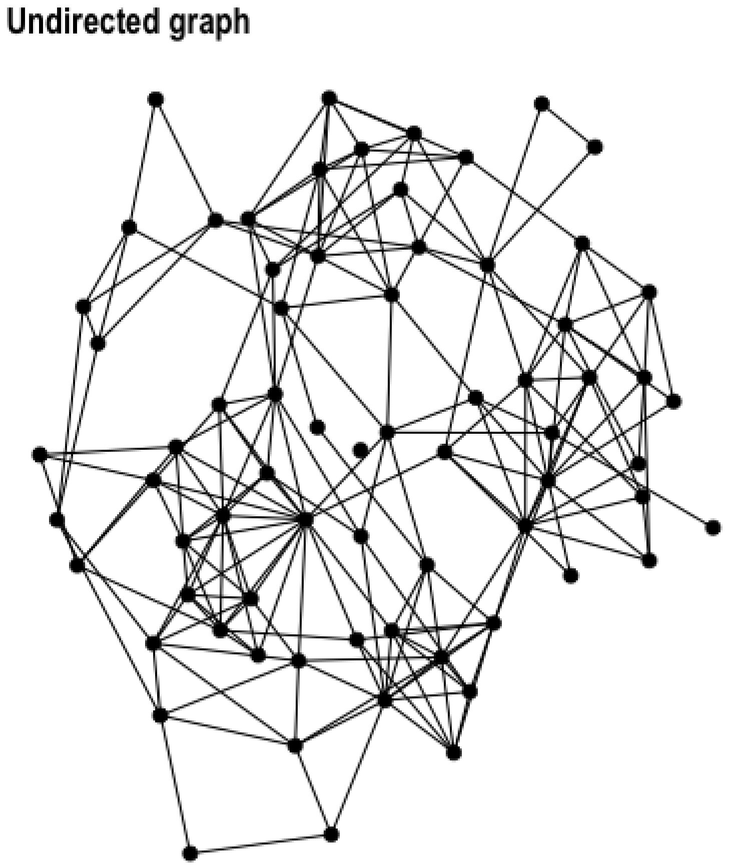
Preliminaries

Network: abstraction of *entities* and their interactions

Graph: mathematical representation

vertices: nodes

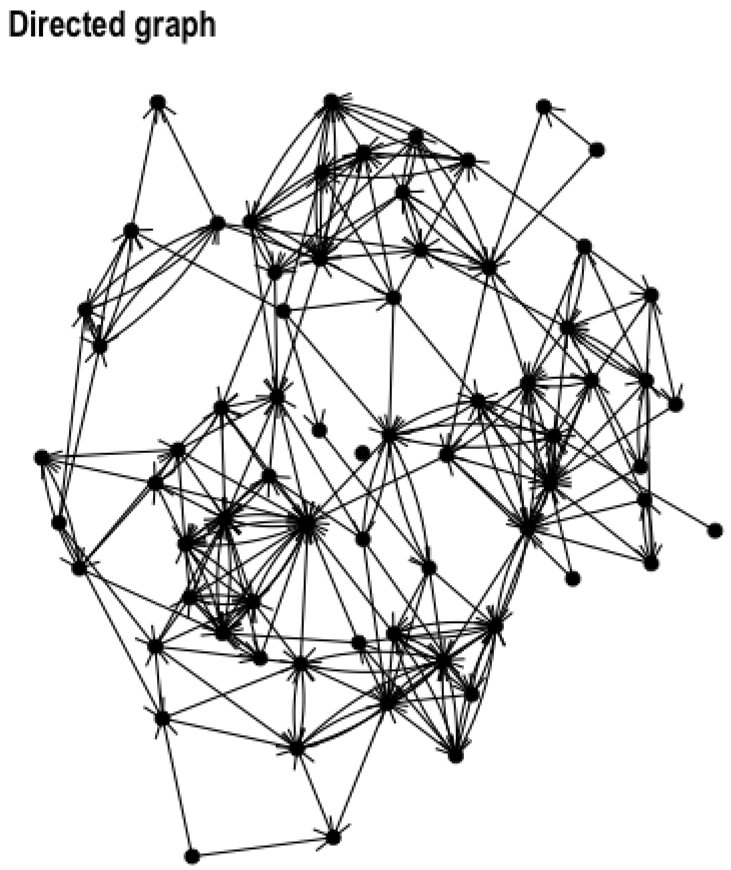
edges: links



Preliminaries

Network: abstraction of *entities* and their interactions
Graph: mathematical representation

vertices: nodes
edges: links



Network statistics: notation

Number of vertices: n

In our example: *number of genes*

Network statistics: notation

Number of vertices: n

In our example: *number of genes*

Number of edges: m

In our example: *number of genetic interactions*

Network statistics: notation

Number of vertices: n

In our example: *number of genes*

Number of edges: m

In our example: *number of genetic interactions*

Degree of vertex i: k_i

Number of genetic interactions for gene i

Network statistics: notation

On the board:

- Calculate number of edges m using degrees k_i (for both directed and undirected networks)
- Calculate *average degree* c
- Calculate *density* ρ

Network statistics: notation

On the board:

- Calculate number of edges m using degrees k_i (for both directed and undirected networks)
- Calculate *average degree* c
- Calculate *density* ρ

In our example:

Average degree: 47.9971459

Density: 0.0171296

(On the board)

Number of edges using degrees (undirected)

$$m = \frac{1}{2} \sum_{i=1}^{n} k_i$$

Number of edges using degrees (directed)

$$m = \sum_{i=1}^{n} k_i^{\text{in}} = \sum_{i=1}^{n} k_i^{\text{out}}$$

(On the board)

Average degree

$$c = \frac{1}{n}\sum_{i=1}^{n} k_i$$

Density

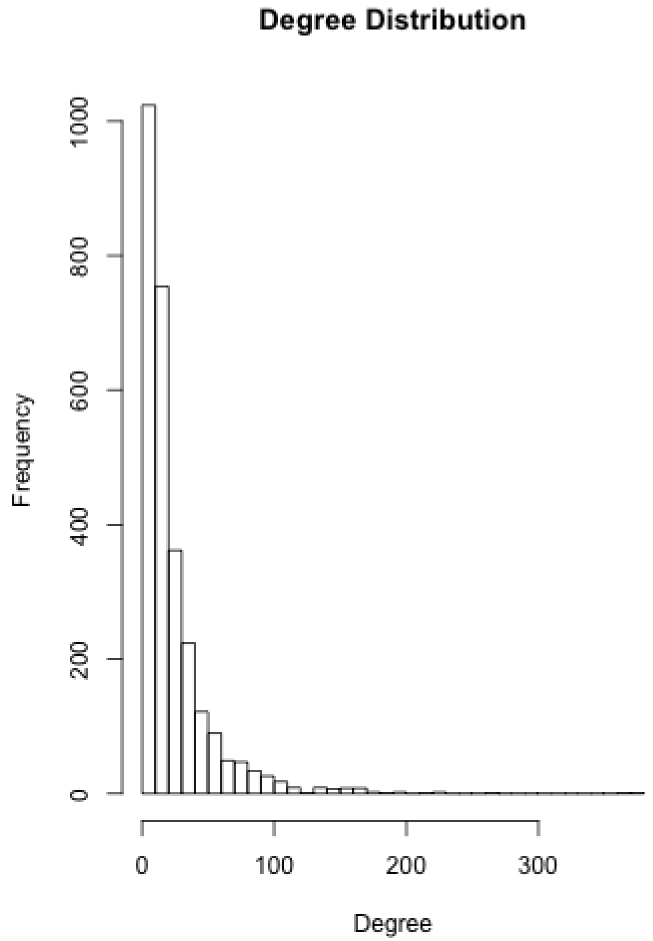
$$\rho = \frac{m}{\binom{n}{2}} = \frac{2m}{n(n-1)} = \frac{c}{n-1} \approx \frac{c}{n}$$

Degree distribution

Fundamental analytical tool to characterize networks

p_k: probability randomly chosen vertex has degree k

On the board: how to calculate p_k and how to calculate average degree c using degree distribution.



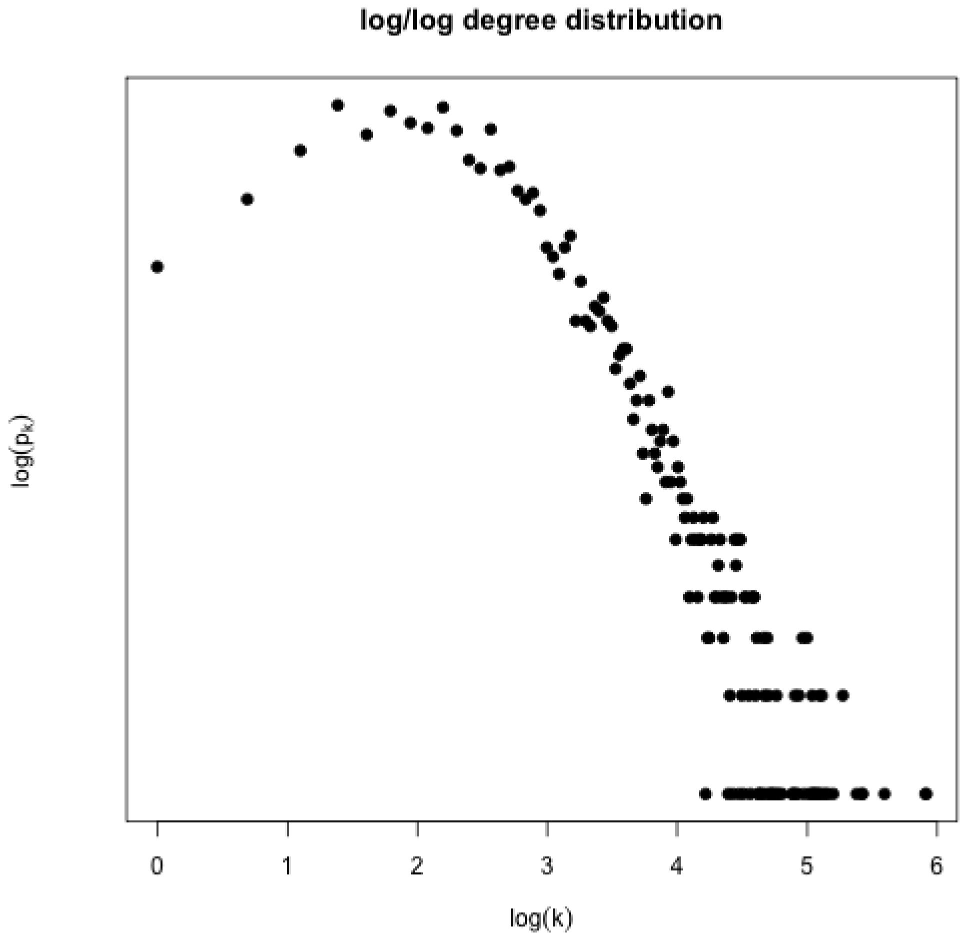
(On the board)

Degree distribution

$$p_k = \frac{n_k}{n}$$

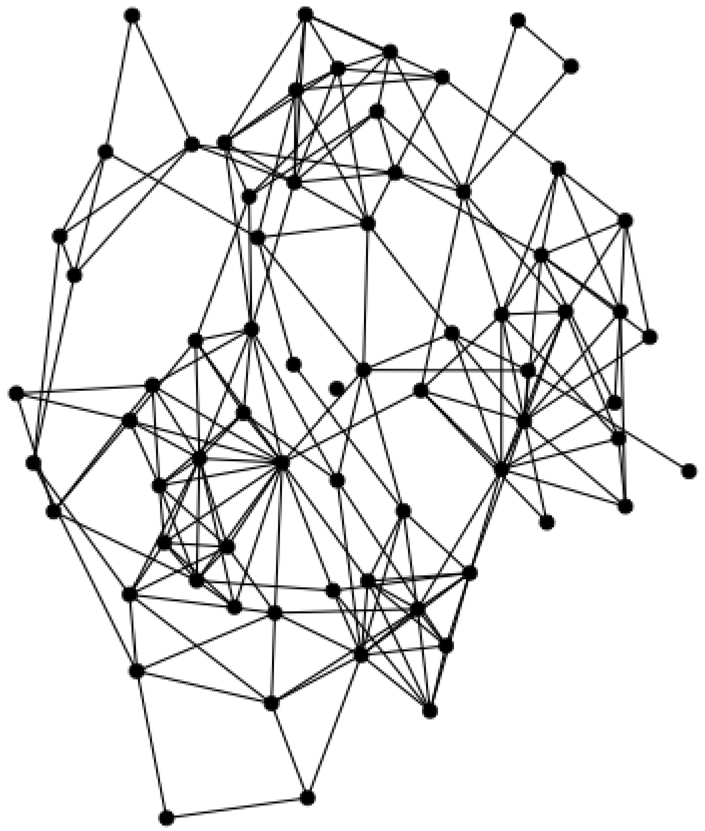
n_k: number of nodes in graph with degree k

Degree Distribution



Paths and Distances

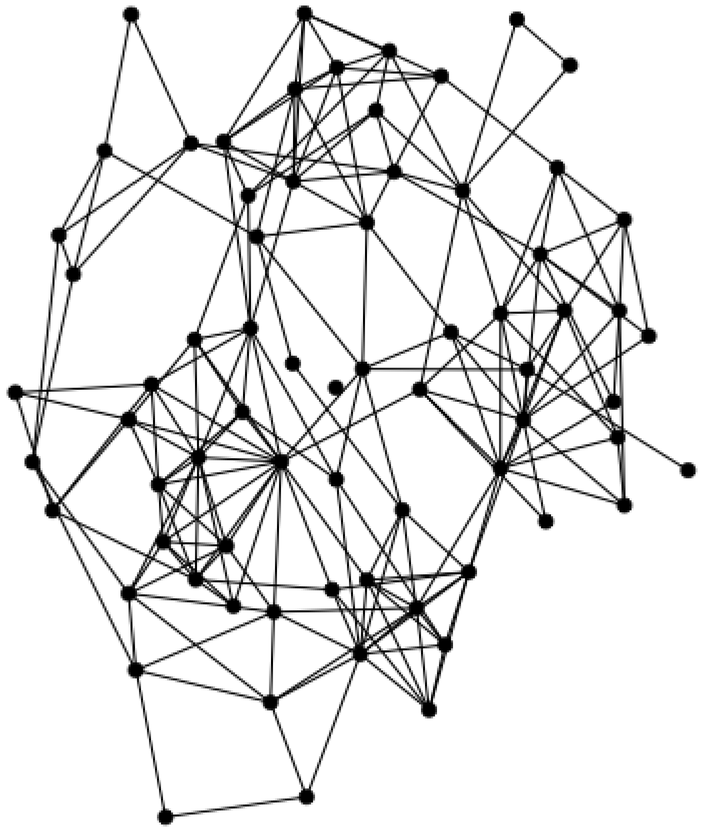
Distance d_{ij}: length of **shortest** path betwen vertices i and j.



Paths and Distances

Distance d_{ij}: length of **shortest** path betwen vertices i and j.

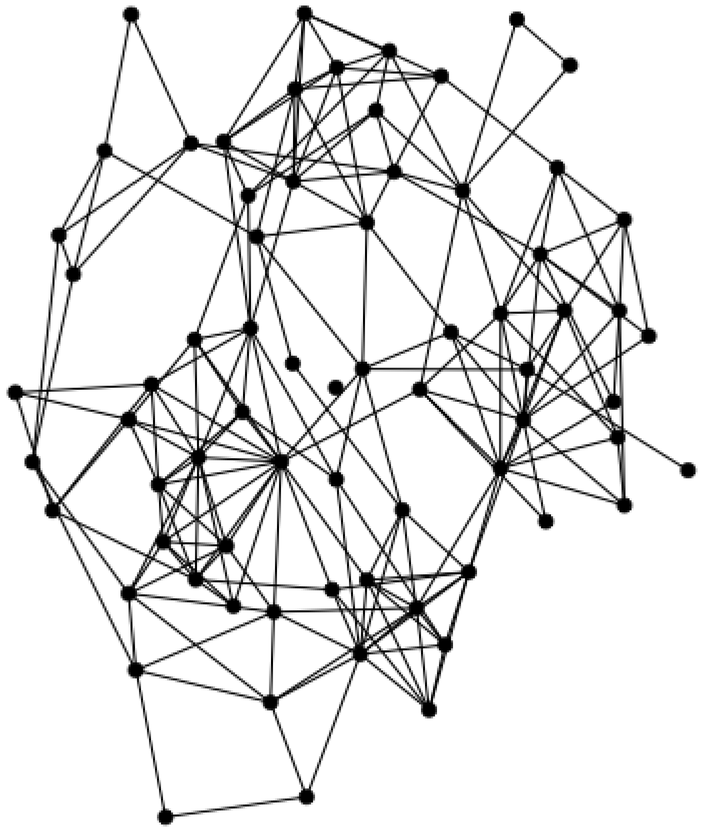
Diameter: longest shortest path $\max_{ij} d_{ij}$



Paths and Distances

Distance d_{ij}: length of **shortest** path betwen vertices i and j.

On the board: average path length

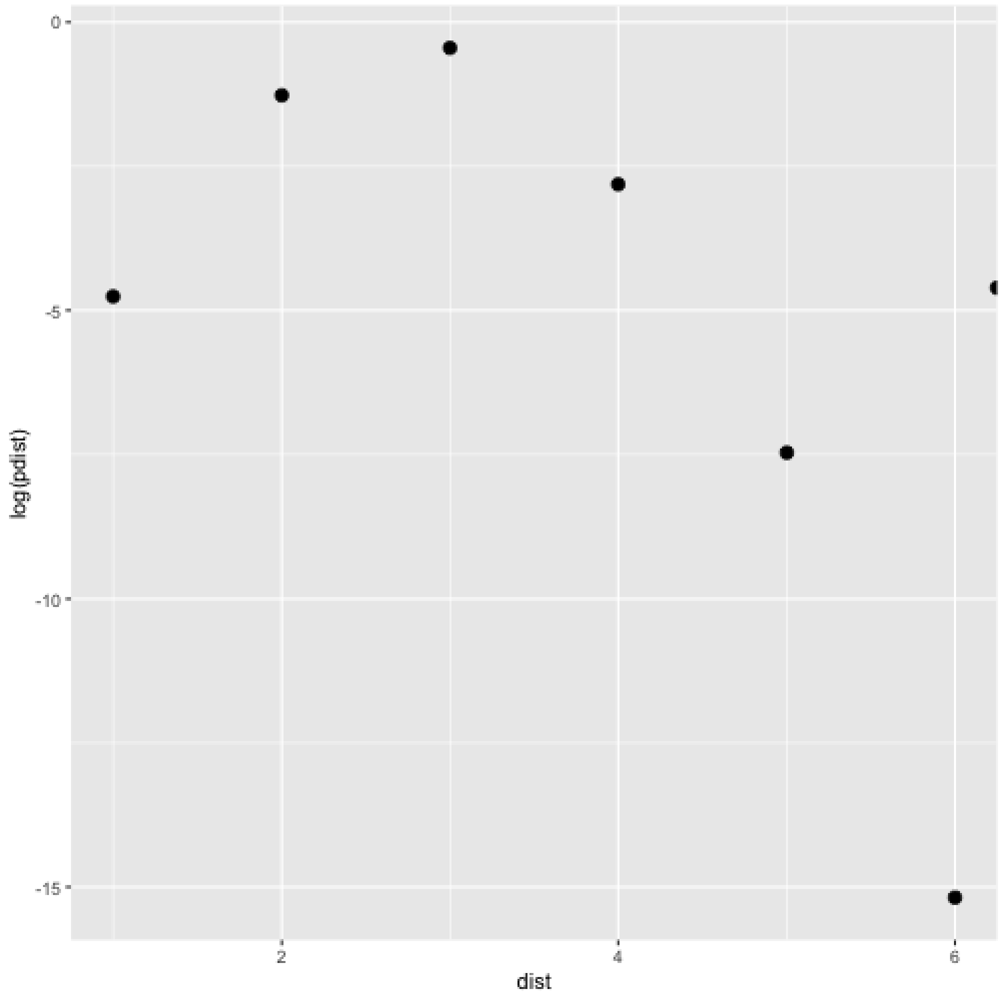


(On the board)

Average path length

$$\bar{d} = \frac{1}{n(n-1)} \sum_{i,j;i \neq j} d_{ij}$$

Distance Distribution



Distances and paths

By convention: if there is no path between vertices i and j then $d_{ij} = \infty$

Distances and paths

By convention: if there is no path between vertices i and j then $d_{ij} = \infty$

Vertices i and j are *connected* if $d_{ij} < \infty$

Distances and paths

By convention: if there is no path between vertices i and j then $d_{ij} = \infty$

Vertices i and j are *connected* if $d_{ij} < \infty$

Graph is connected if $d_{ij} < \infty$ for all i, j

Distances and paths

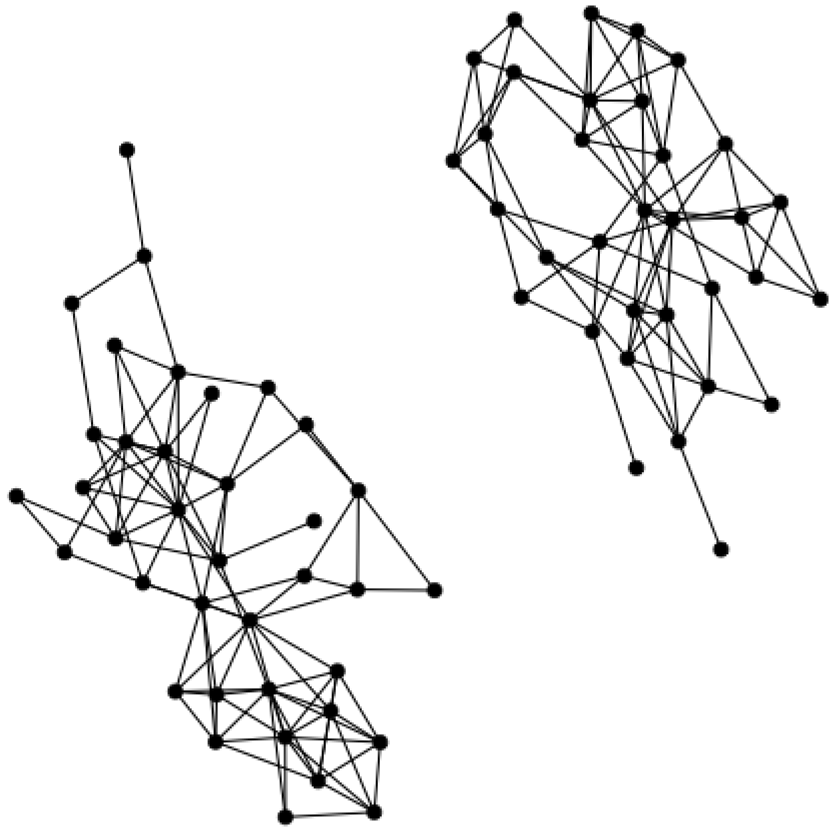
By convention: if there is no path between vertices i and j then $d_{ij} = \infty$

Vertices i and j are *connected* if $d_{ij} < \infty$

Graph is connected if $d_{ij} < \infty$ for all i, j

Components maximal subset of connected components

Components



Clustering Coefficient

One more quantity of interest: how dense is the neighborhood around vertex i?

Do the genes that interact with me also interact with each other?

Definition on the board

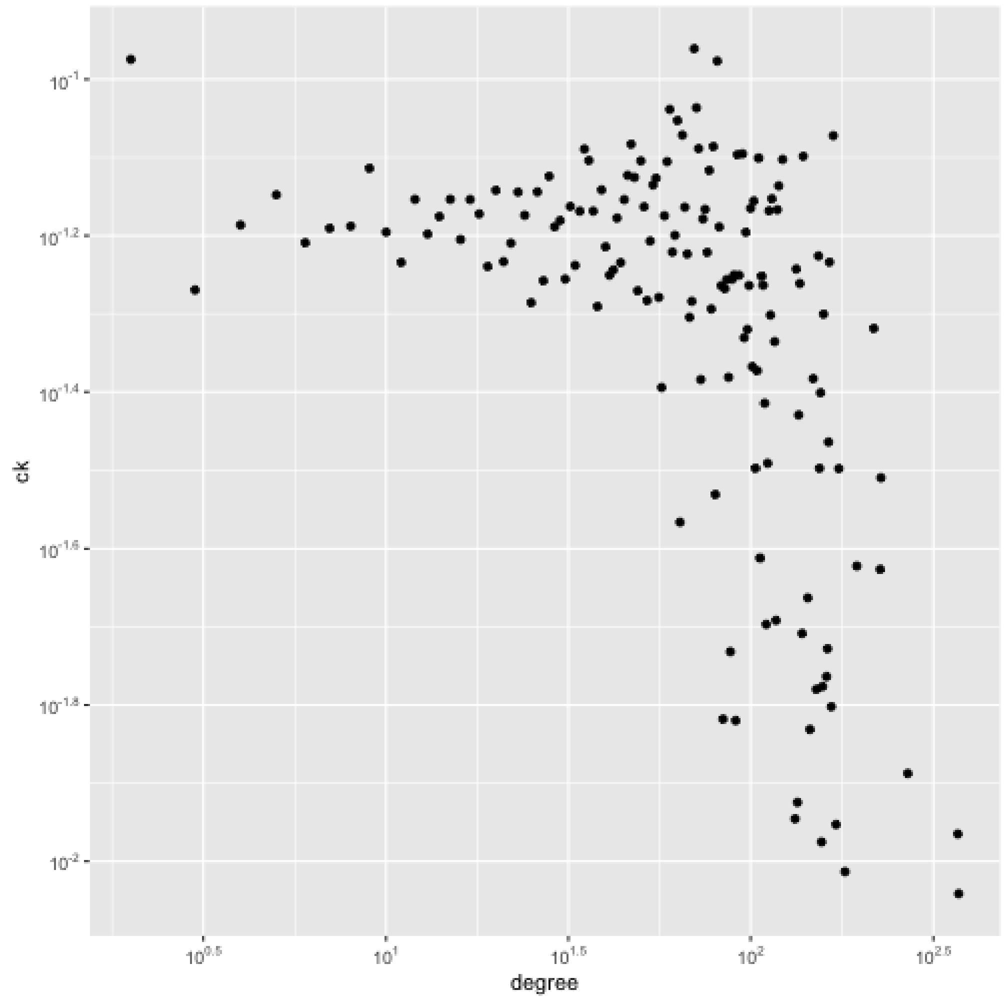
(On the board)

Clustering coefficient

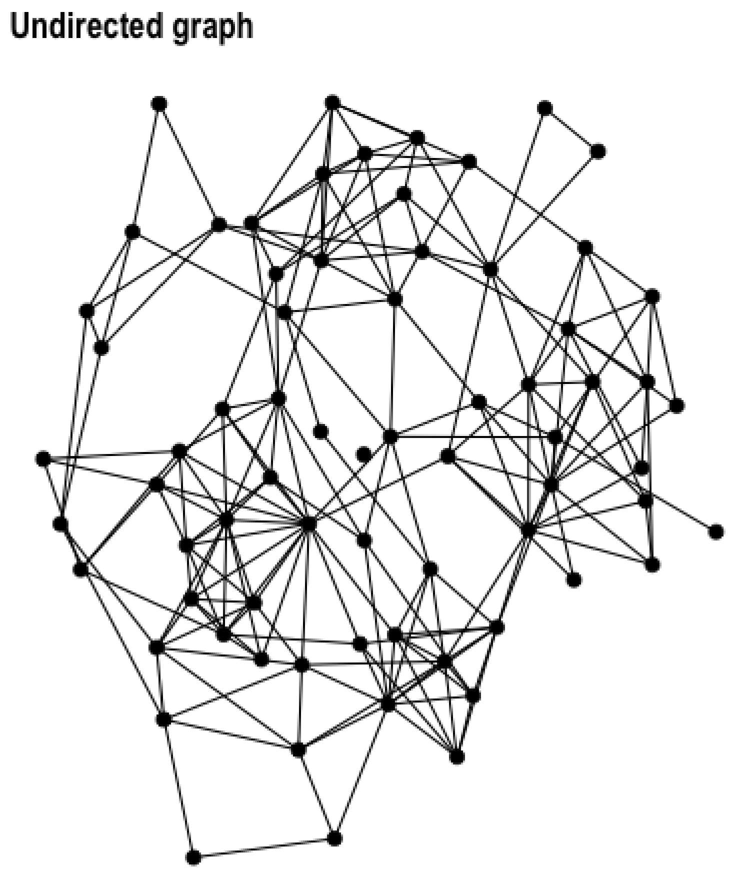
$$c_i = \frac{2m_i}{k_i(k_i - 1)}$$

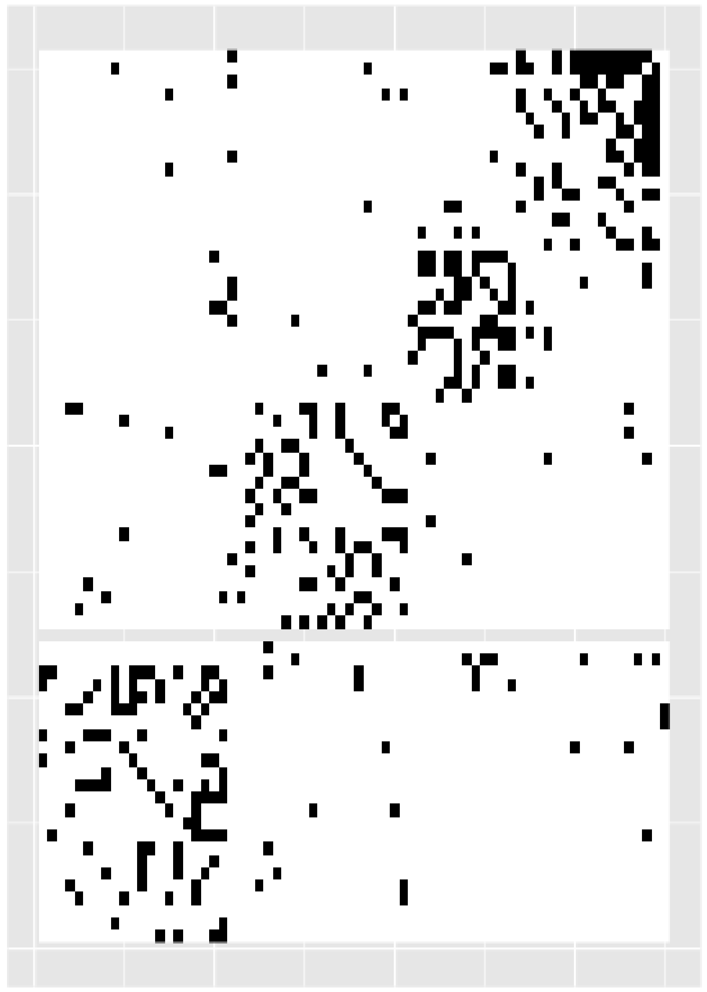
m_i: number of edges between neighbors of vertex i

Clustering coefficient



Adjacency Matrix

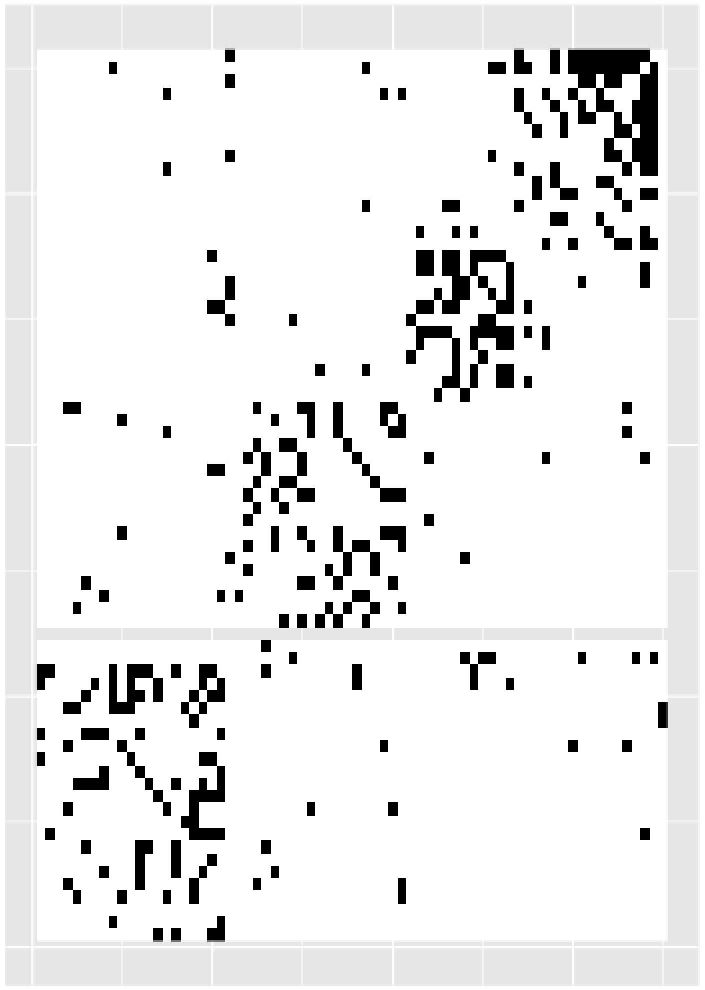




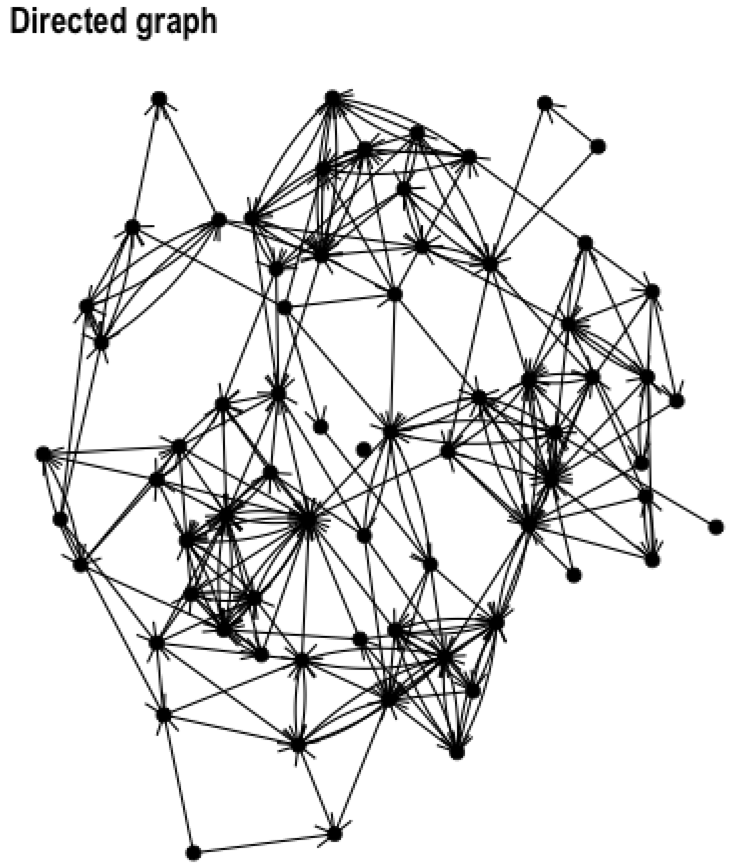
Adjacency Matrix

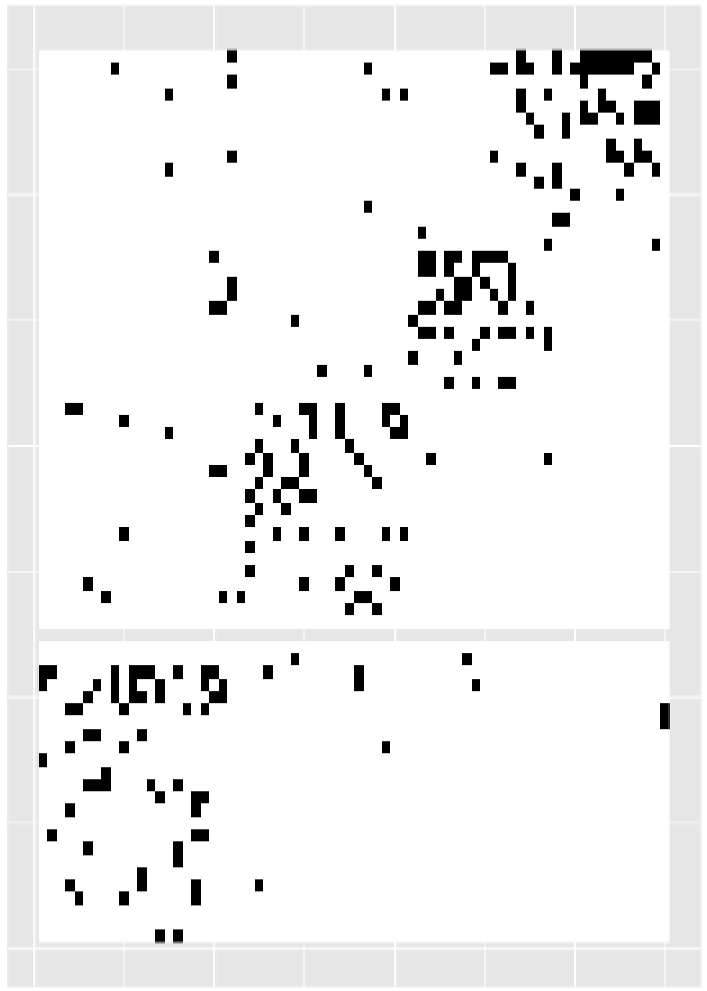
On the board:

- Definition
- Computing degree with adj. matrix
- Computing num. edges m with adj. matrix
- Computing paths with adj. matrix



Adjacency Matrix





Weighted networks

Edges are assigned a weight indicating quantitative property of interaction

Weighted networks

Edges are assigned a weight indicating quantitative property of interaction

- Strength of genetic interaction (evidence from experiment)
- Rates in a metabolic network
- Spatial distance in an ecological network

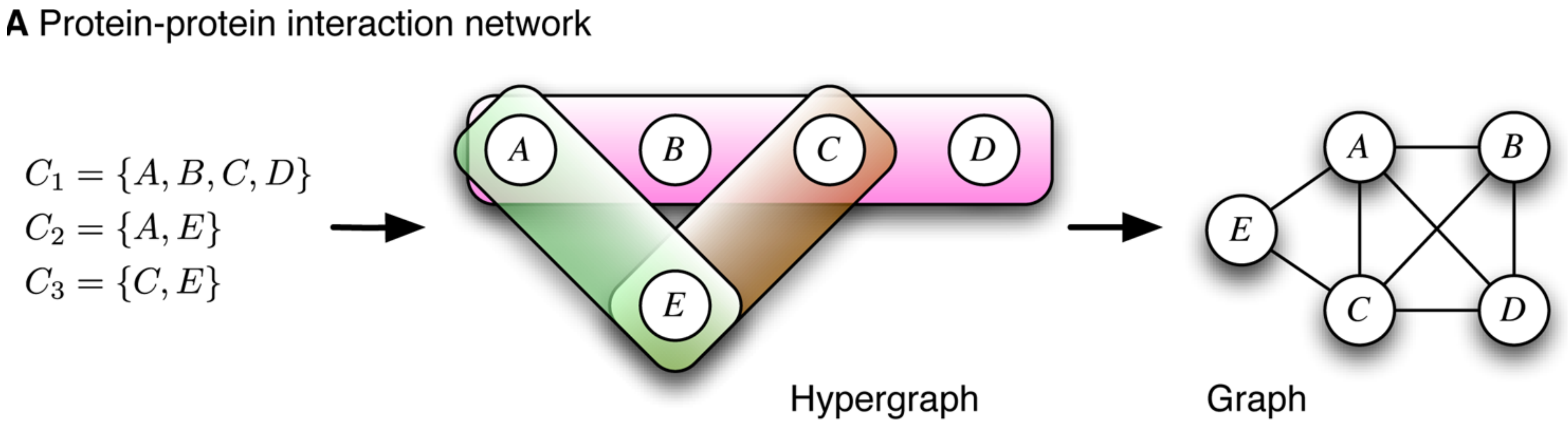
Adjacency matrix contains weights instead of 0/1 entries

Adjacency matrix contains weights instead of 0/1 entries

Path lengths are the sum of edge weights in a path

Hypergraphs

Edges connect more than two vertices

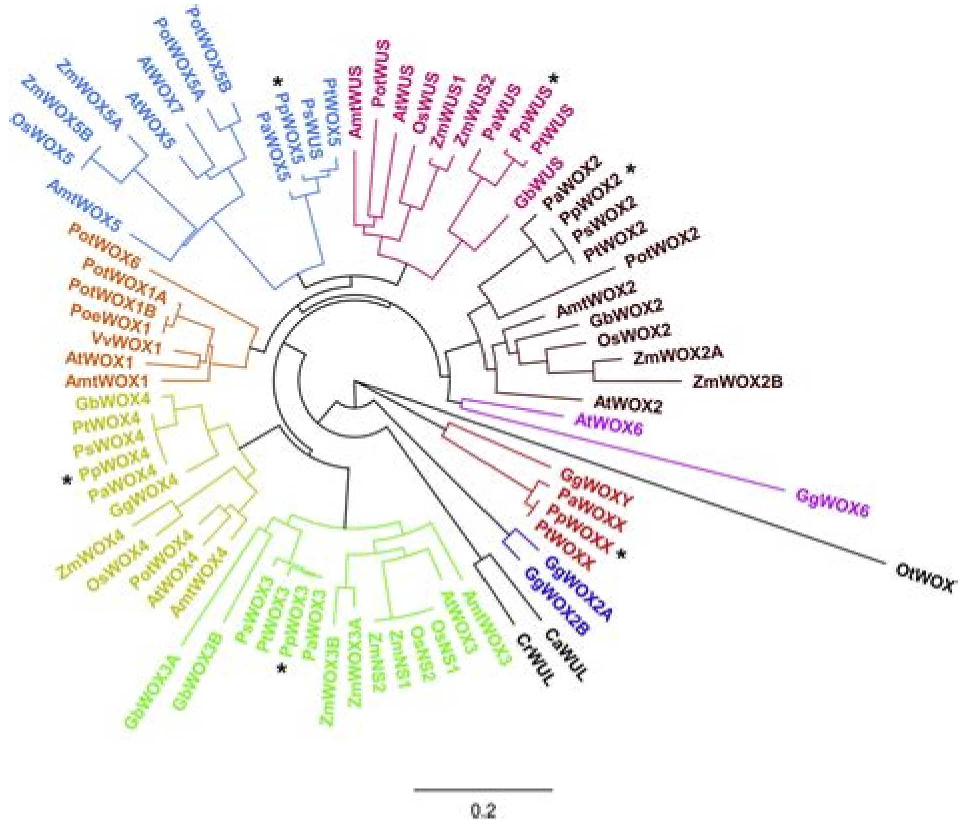


https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000385

Trees

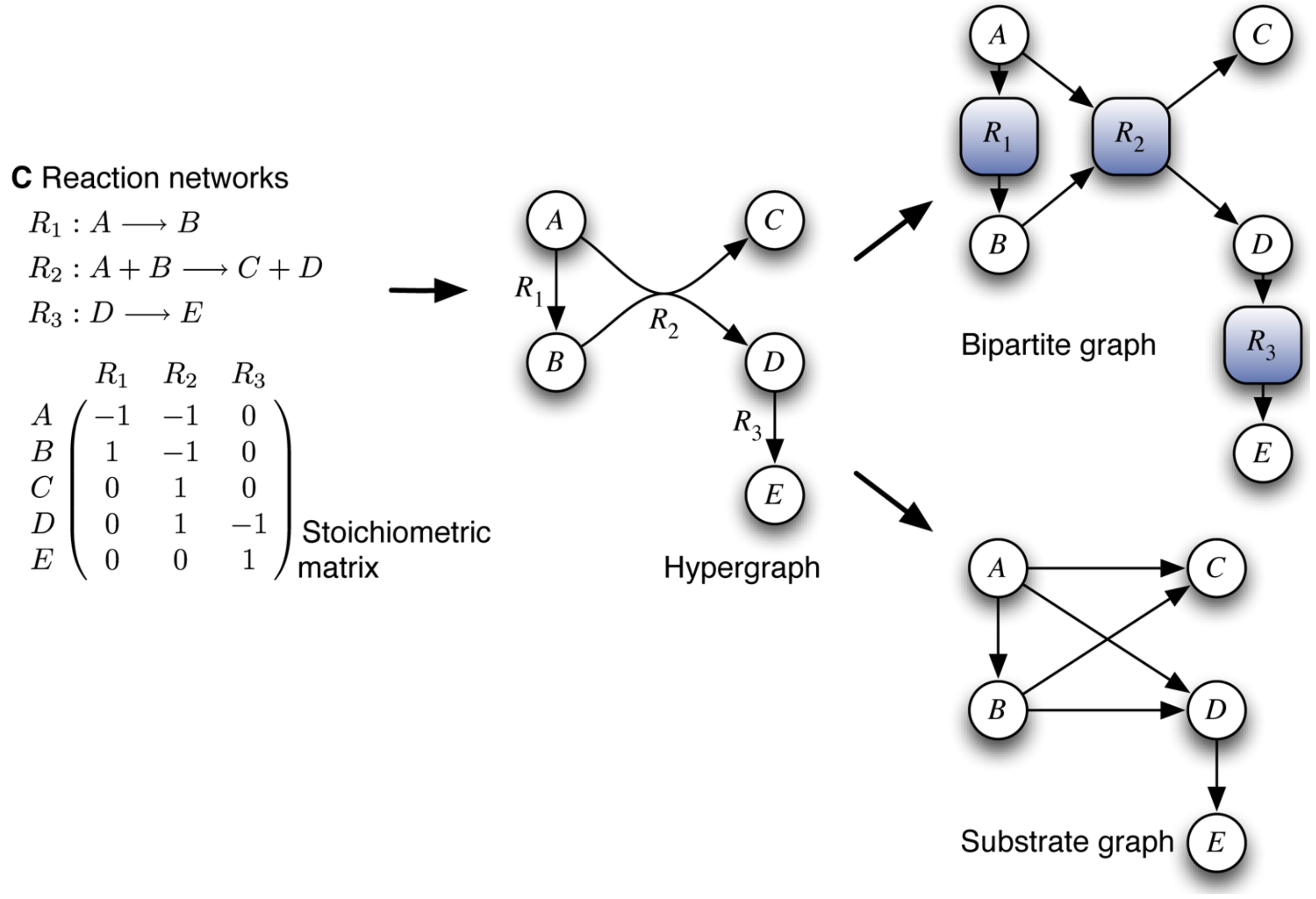
Acyclic graphs

Single path between any pair of vertices



https://www.sciencedirect.com/science/article/pii/S0981942817304321

Bipartite Networks



$R_1 : A \longrightarrow B$

$R_2 : A + B \longrightarrow C + D$

$R_3 : D \longrightarrow E$

$$\begin{array}{c|ccc} & R_1 & R_2 & R_3 \\ \hline A & -1 & -1 & 0 \\ B & 1 & -1 & 0 \\ C & 0 & 1 & 0 \\ D & 0 & 1 & -1 \\ E & 0 & 0 & 1 \end{array}$$

Bipartite Networks

We use an *Incidence Matrix* B instead of *Adjacency Matrix*

(On the board): definition

Bipartite Networks

Projections

vertex projection: P_{ij}, num. of groups in which vertices i and j co-occur

group projection: P'_{ij}, num. of members groups i and j share

Bipartite Networks

Projections

vertex projection: P_{ij}, num. of groups in which vertices i and j co-occur

group projection: P'_{ij}, num. of members groups i and j share

(On the board)

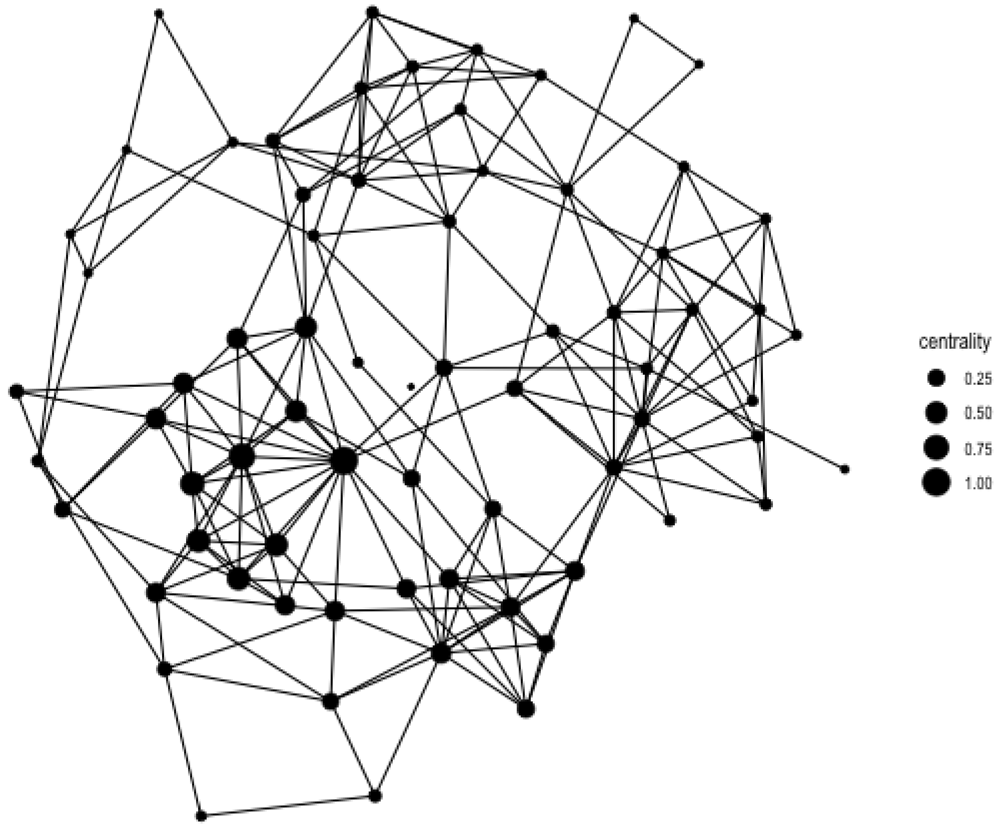
$$P = B^T B$$

$$P' = BB^T$$

Centrality

What are the *important* nodes in the network?

What are *central* nodes in the network?



Centrality

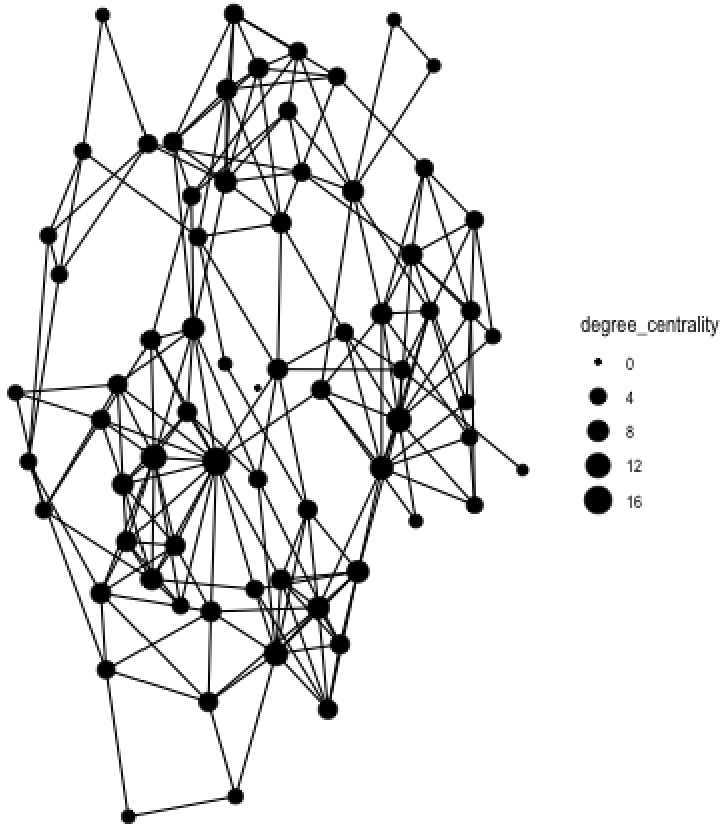
Undirected Graphs

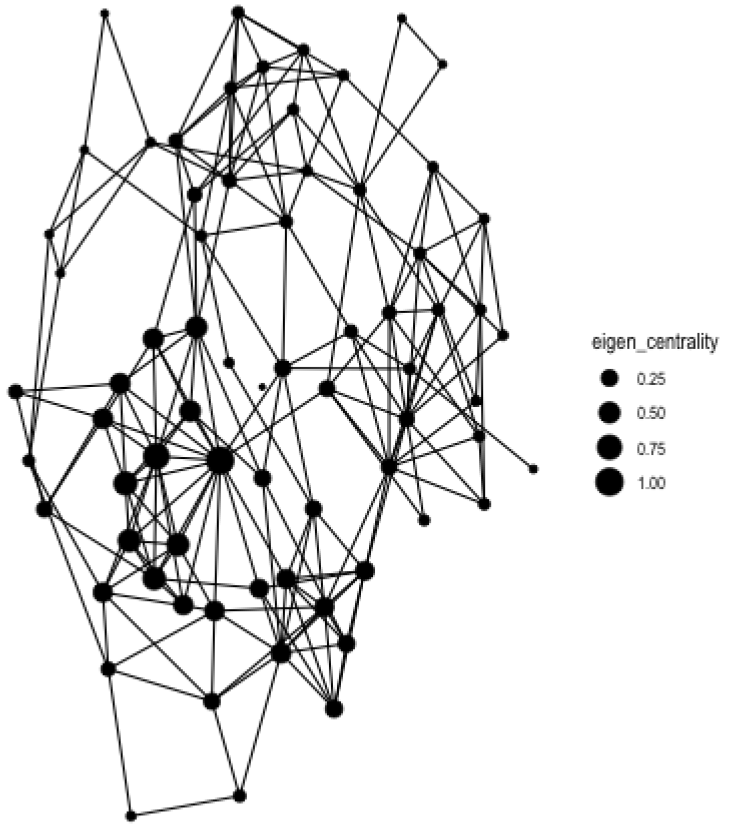
- Eigenvalue Centrality

Directed Graphs

- Katz Centrality
- Pagerank

Centrality

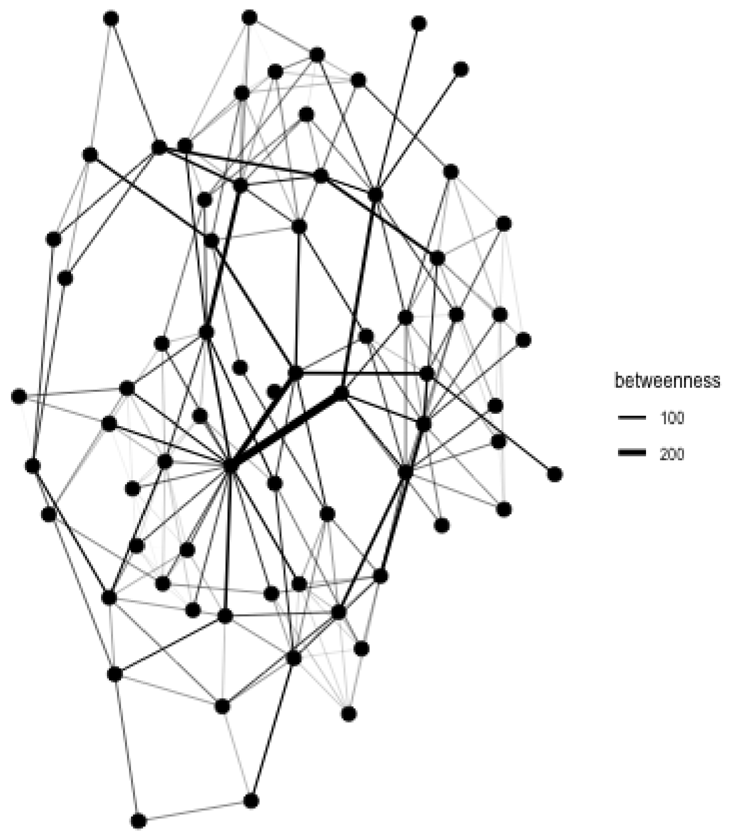

degree_centrality
0
4
8
12
16


eigen_centrality
0.25
0.50
0.75
1.00

Betweenness

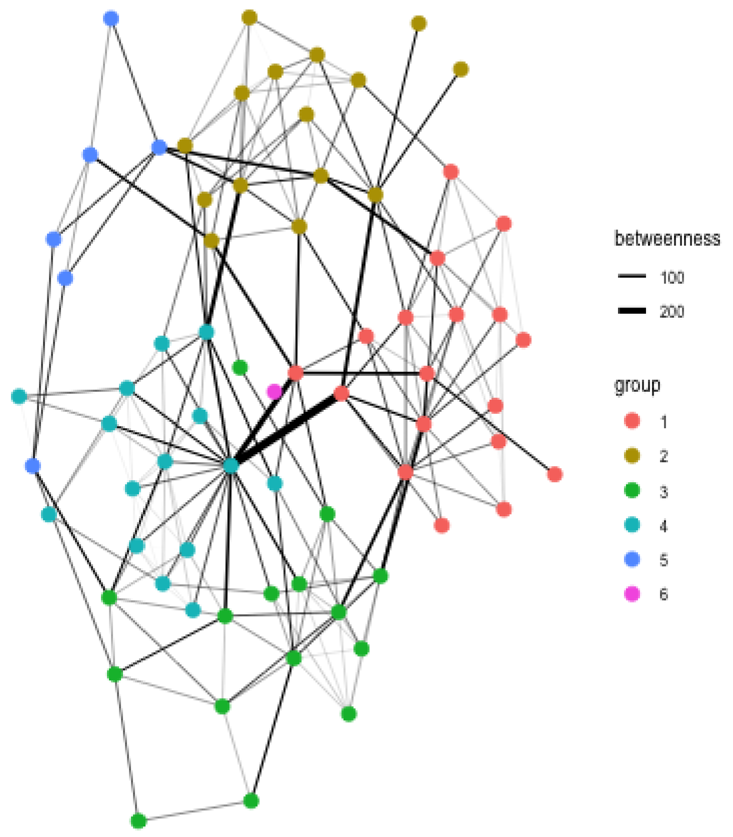
What are the *important* edges in the network?

What are edges that may connect clusters of nodes in the network?



Betweenness

Girvan-Newman Algorithm - hierarchical method to partition nodes into communities using edge betweenness



Girvan-Newman Algorithm

Two phases:

Phase One: Compute betweenness for every edge

Phase Two: Discover communities by removing *high* betweenness edges (similar to hierarchical clustering)

Girvan-Newman Algorithm

Calculating Betweenness

Formally, $\mathrm{betweenness}(e)$: fraction of vertex pairs (x, y) where shortest path crossess edge e

Path Counting: For each vertex x, use breadth-first-search to count number of shortest paths through each edge e in graph between x and every other vertex y.

Sum result across vertices for each edge e, and divide by two

Presentation from *Mining Massive Datasets* Leskovec, et al. http://mmds.org/ (Ch. 10)

Girvan-Newman Algorithm

Counting Paths

Algorithm (starting from node x)

1. Construct breadth-first search tree
2. (Root->Leaf) Label each vertex v with the number of shortest paths between x and v: sum of labels of parents
3. (Leaf->Root) Count the (weighted) number of shortest paths that go through each edge: next slide

Girvan-Newman Algorithm

Counting Paths

Step 3, counting number of shortest paths through each edge

a. Leafs v in search tree get a *credit* of $C_v = 1$

b. Incoming edge $e_i = (y_i, v)$ to vertex v in search tree gets *credit* $C_{e_i} = C_v * \frac{p_i}{\sum_j p_j}$

- p_i: number of shortest paths between x and y_i (computed in Step 2)
- sum $\sum_j$ is over parents of v

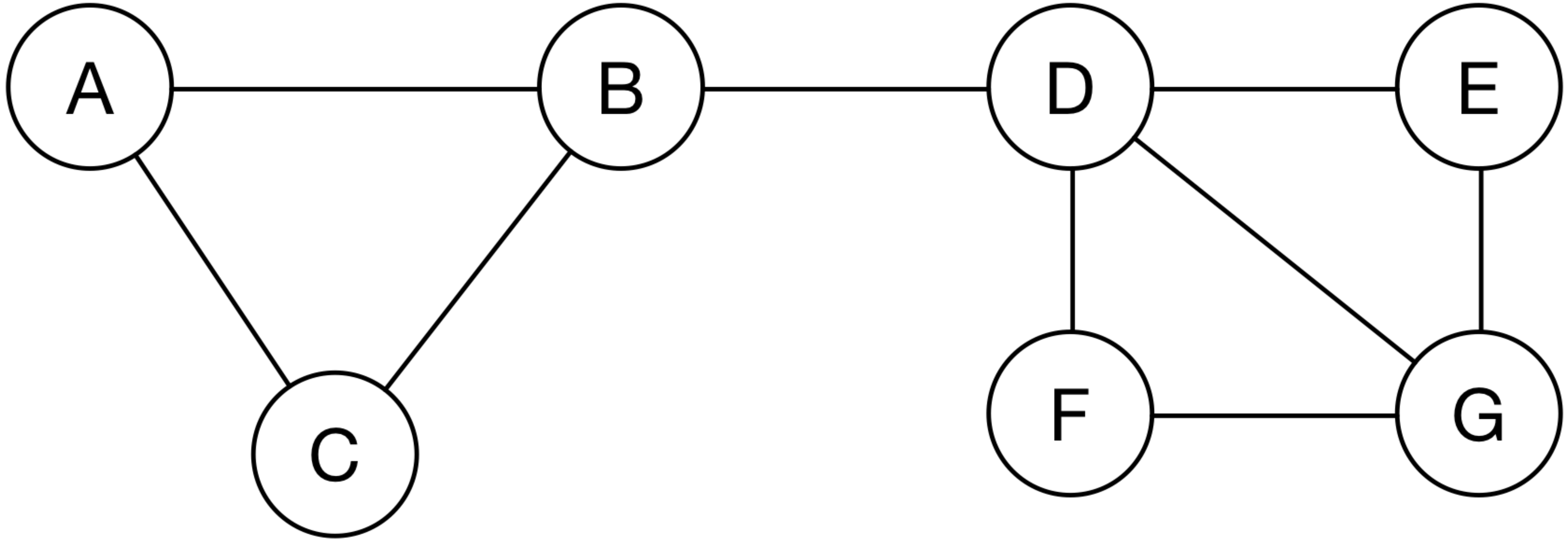
Girvan-Newman Algorithm

Counting Paths

c. Non-leaf vertex v gets credit $C_v = 1 + \sum_j e_j$ where sum j is over outgoing edges e_j in search tree

Girvan-Newman Algorithm

Example



Resources

Cross-language

igraph: http://igraph.org/

Boost Graph Library:

https://www.boost.org/doc/libs/1_71_0/libs/graph/doc/

Resources

Python

- `igraph`
- `networkx`

Resources

R

Workhorses:

- `igraph`
- `Rgraphviz`

Tidyverse (https://tidyverse.org):

- `tidygraph`
- `ggraph`

Resources

For data analysis it is helpful to think in terms of rectangular datasets

For networks, we need to have two distinct tables to represent this data.

- One table represents entities and their attributes:

```
## # A tibble: 70 x 2
##    name  .tidygraph_node_index
##    <chr>                 <int>
##  1 1                         1
##  2 2                         2
##  3 3                         3
##  4 4                         4
```

Resources

- Second table to represent edges and their attributes:

```
## # A tibble: 202 x 4
##      from    to .tidygraph_edge_index .orig_data
##     <int> <int> <list>                <list>
##   1     1    14 <int [2]>             <tibble [2 × 3]>
##   2     1    16 <int [1]>             <tibble [1 × 3]>
##   3     1    20 <int [1]>             <tibble [1 × 3]>
##   4     1    21 <int [1]>             <tibble [1 × 3]>
##   5     2     9 <int [1]>             <tibble [1 × 3]>
##   6     2    21 <int [1]>             <tibble [1 × 3]>
##   7     4     5 <int [2]>             <tibble [2 × 3]>
```

Network-derived attributes

Besides attributes measured for each node, we have seen we can derive node and edge attributes based on the structure of the network.

For instance, we can compute the *degree* of a node, that is, the number of edges incident to the node.

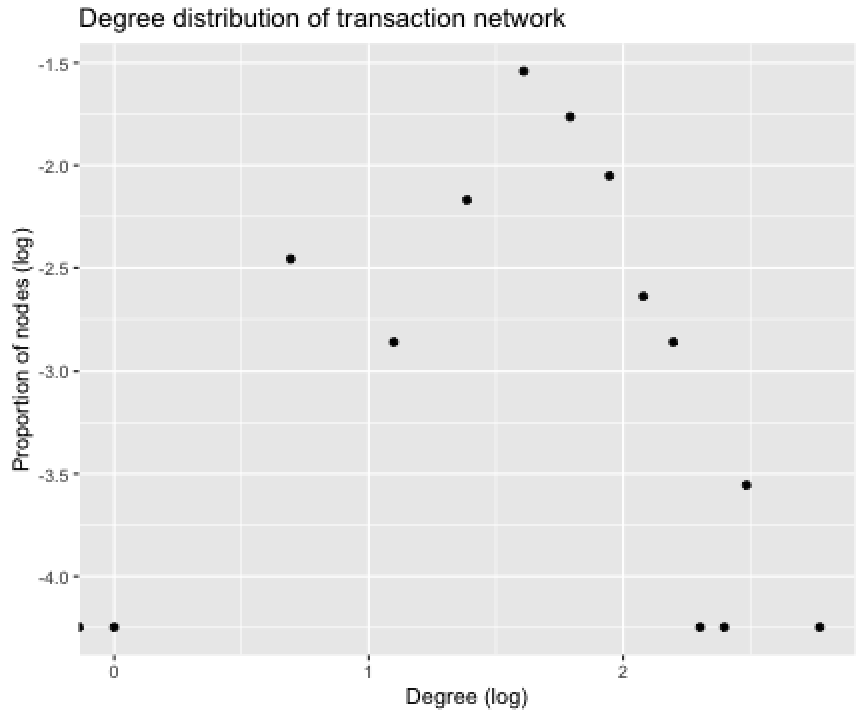
Network-derived attributes

```
## # A tibble: 70 x 3
##    name  .tidygraph_node_index degree
##    <chr>                 <int>  <dbl>
##  1 1                         1      4
##  2 2                         2      2
##  3 3                         3      0
##  4 4                         4      5
##  5 5                         5      5
##  6 6                         6      5
##  7 7                         7      3
##  8 8                         8      5
##  9 9                         9      6
```

Network-derived attributes

The distribution of newly created attributes are fundamental analytical tools to characterize networks.

```
undirected_graph_1958 %>%
  activate(nodes) %>%
  as_tibble() %>%
  group_by(degree) %>%
  summarize(n=n()) %>%
  ungroup() %>%
  mutate(num_nodes = sum(n)) %>%
  mutate(deg_prop = n / num_nodes) %>%
```


Degree distribution of transaction network
Proportion of nodes (log)
Degree (log)