

# **Network Analysis with UCINET for ABM of Industrial Districts and Economic Geography**

Richard Taylor

Centre for Policy Modelling,

The Business School,

Manchester Metropolitan University,

Manchester, UK.

E-mail: [r.i.taylor@mmu.ac.uk](mailto:r.i.taylor@mmu.ac.uk)

# Outline of Tutorial

## Day 1

- **Introduction to the software**
- The ID network model
- Running the model via GUI
- **Customising the model**
- Java basics
- **The Java-RePast source code**
- **Collecting networks dataset**

## Day 2

- **Introducing UCINET**
- Using DRAW to view your network
- **Concepts of Network Analysis**
- **Characterising your network**
- Static and dynamic networks
- **The Second Industrial Model**
- Presenting the results

# Social Network Analysis (Revision)

## Graph / network theory

A graph  $G(V,E)$  consists of a set of vertices  $V$  representing individuals or objects and a set of edges  $E$  representing relationships between the individuals or objects

- Directed and undirected graphs, valued graphs
- Degree, indegree, outdegree
- Density, connectedness, subgraphs
- Average path length, cliquishness, clustering
- Centrality, periphery
- Small world, scale-free

# Briefly on the small-world notion

A small world is a large, sparse network characterised by:

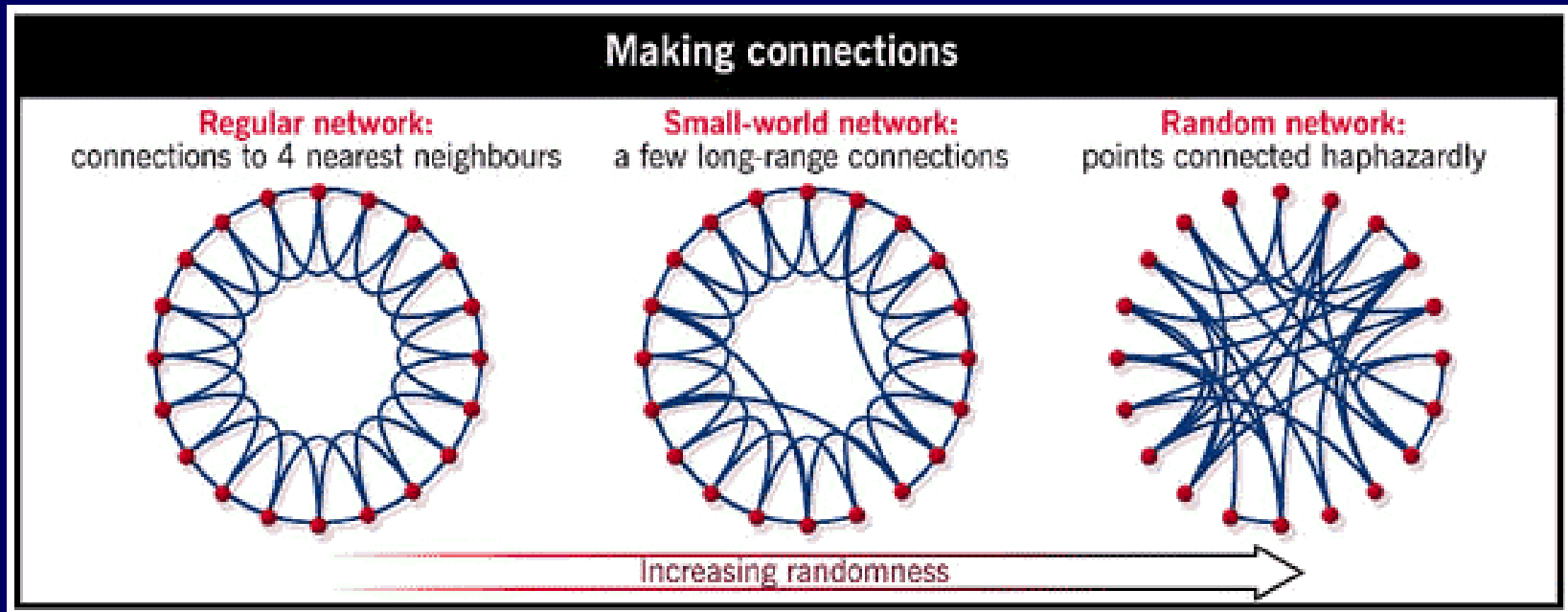
- ❑ Short average path length between any two agents
- ❑ High degree of clustering of network (agents circles of acquaintances tend to overlap – two of my friends are likely to be friends with one another)

## Why Small World networks are interesting to study?

- Systems display enhanced signal propagation speed, computational power, and synchronizability (Watts & Strogatz, 1998)
- Empirical studies show that most social networks are small worlds

# The Small World Model

Watts & Strogatzs' (1998) Model



$p$  is the parameter which governs the randomness

# Network properties of the model

The criteria for identifying the network as small world:

$$\mathcal{L}(t) \cong \mathcal{L}_{\text{random}}(t) \quad C(t) \gg C_{\text{random}}(t)$$

# Scale-free networks

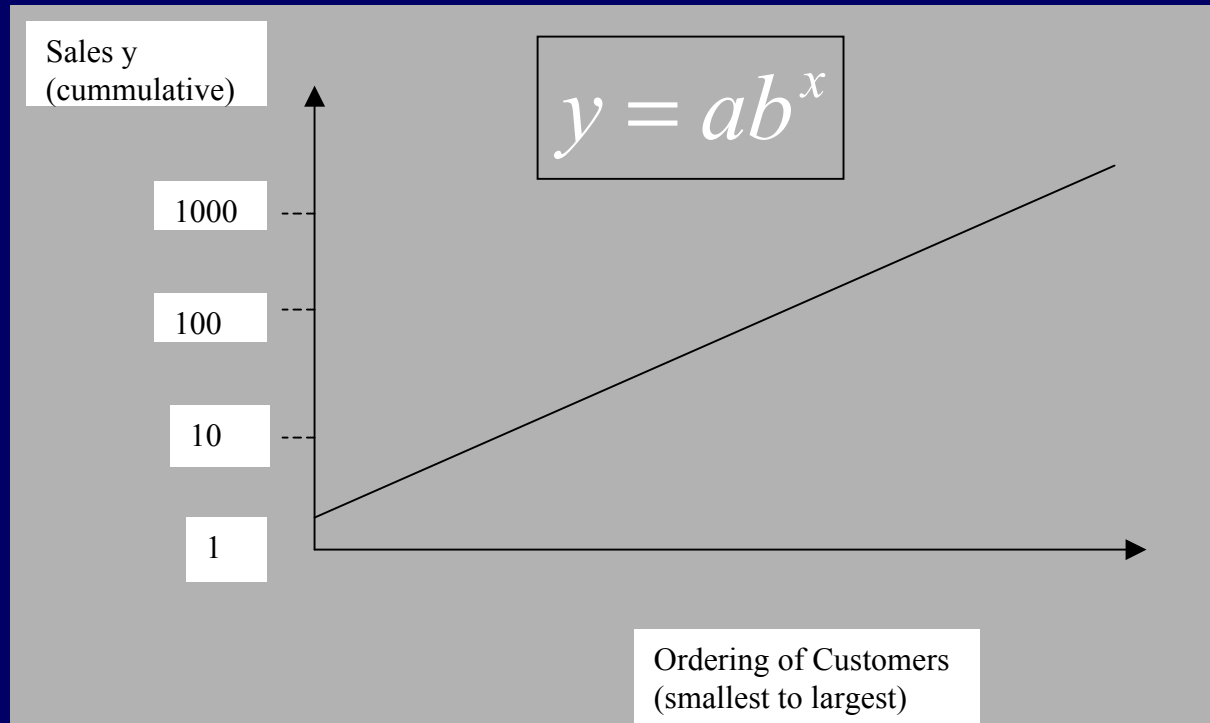
A scale-free network is characterised by an extremely uneven degree distribution, where

- ❑ Some nodes act as "very connected" hubs
- ❑ The majority of nodes are relatively poorly connected

The degree distribution of nodes follows a **power-law**

- i.e. there is a logarithmic relationship between the cumulative frequency of events and the smallest to largest ordering of objects to which those events are ascribed

# Scale-free networks



- As well as being small world distributed, many natural and artificial systems also have the scale-free property
- The term “scale-free” was first used by Barabasi et al. (1998)

# Introducing UCINET

Objective: to become familiar with UCINET and to display and arrange your network using visualisation tools

Guidelines for the practical (assuming you have the output file: IDNetwork.dl)

1. Open UCINET and import the dataset as a data language file (File-Import-Import as DL File) and see the matrix display for the network
2. Launch the Draw program and open your network (IDNetwork.#h)
3. Rearrange the network by hand and/or arrange it by one of the layout algorithms provided by Draw (Layout-Choose Layout- *your choice*)
4. Launch the Mage program and view the network 3-dimensionally

# Analysis with UCINET

**Objective:** learn how to use the analysis tools of UCINET to characterise the ID network model

Guidelines for the practical:

1. Calculate the density (Network-Properties-Density) of your network – (the file IDNetwork.#h)
2. Symmetrise (Transform-Symmetrise) your network to make an undirected network (creating a file SymmetriseIDNetwork.#h)
3. Use the symmetrised network to calculate the geodesic distance matrix and the average path length (Network-Cohesion-Distance)
4. Identify the components of your network (Network-Regions-Components) and identify the cliques (Network-Subgroups-Cliques)
5. In groups of 2s / 3s, compare your results and explain the differences

# Principles of dynamic models

Agent-based models can be used to describe dynamic processes

Dynamic simulations have various uses:

- to find a solution to a computational problem (iteratively) which may be difficult to solve analytically
- to answer the question ‘what will happen next?’ given an initial state and a set of inference rules to describe state transitions

Often the problem we try to model is quite complex and has a large ‘possibility space’ of different states that could be reached

# Principles of dynamic models

Even with the fastest computer (and the cleverest search processes) we cannot explore every possible solution / transition

This is not necessarily a problem, but we must find ways to handle the incomplete nature of simulation results. For example,

- classifying each result obtained into a smaller number of categories
- ‘sampling’ the state space by running repeated experiments with a range of different parameter settings
- doing analysis upon the sample and inferring from these findings the behaviour of the model

# Principles of dynamic models

Dynamic social simulation models allow us to investigate different kinds of problems. For example:

- The effect of the introduction of a new technology or other environmental variable in the model
- The effect of a change in rules set in place by institutional actors (e.g. legal rules, tax rates etc.) i.e. policy modelling
- The emergence of macro-level behaviours of the system which result from lower-level interactions and feedback mechanisms building up (and recognisable only) with the passage of time

# Principles of dynamic models

Dynamic agent-based models describe state-changes at different points of simulated time

Time is normally divided into discrete steps, **time-steps** but AKA cycles or, in RePast, **ticks**

The simulation may consist of several ‘nested’ time levels

RePast models would implement BuildSchedule method for scheduling ‘actions’ that change the simulations state i.e., that describe a dynamic simulation

For our purpose, we are interested in studying IDs not only by measuring networks, but by understanding the social processes by which they are formed, i.e. the generative mechanisms

# Using BuildSchedule to describe the dynamics

Objective: to produce dynamic simulations via the BuildSchedule component of the model

BuildSchedule uses BasicAction objects to set up calls to several different methods provided in the demonstration code

These methods all manipulate the network in some way, by adding, removing, or resetting its nodes and edges

Example use of the scheduling for BasicAction objects:

```
schedule.scheduleActionAt(100, this, "stop" );
```

This schedules the model object 'this' to "stop" at the 100<sup>th</sup> tick

# Using BuildSchedule to describe the dynamics

Objective: to produce dynamic simulations via the BuildSchedule component of the model

Guidelines for the practical:

1. Read the comments in the demonstration sourcecode for the ID network model and decide which dynamic process you would like to simulate
2. Uncomment (remove the // characters) the corresponding lines of the program and modify them or write your own lines of code to build the schedule
3. Compile the program, run it, and start it through the toolbar as before
4. Observe the behaviour of the network through the RePast display (note that you may have to tick Options-Update display to see these changes)
5. Describe how the end state differs from the initial state of the network

# Characterising the network evolution

Objective: to examine the effects of the dynamics of the simulation upon the ID network topology

The ID model should now be evolving over the course of the simulation, the network looking quite different at beginning, middle, and end phases.

To analyse the network characteristics as they evolve, it is necessary to schedule a call to write the network data to output file

To avoid confusion, we shall append additional network data to the same file used for the initial data (“IDNetwork.dl”)

# Characterising the network evolution

Objective: to examine the effects of the dynamics of the simulation upon the ID network topology

Guidelines for the practical:

1. In BuildSchedule, uncomment the relevant lines to schedule calls to write the data at the end of the simulation (or at any other appropriate point)
2. Compile the program, run it, and start it through the toolbar as before
3. When the data are successfully appended to the same file, you must then cut and paste them into separate files, which should be named after their dynamic parameters and tick count (contained in the block header)
4. Import individually into UCINET each file created in this way and carry out the same procedure for analysis for each one

5. Now you should be able to make a more measured and precise description of the evolution of your ID network. Good luck!

**Thank you very much!**

**Grazie mille!**