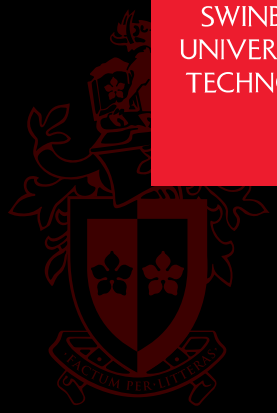
SWINBURNE UNIVERSITY OF TECHNOLOGY



ERGM Algorithms

Peng Wang

Simulation: Technical details

Using the *Metropolis algorithm*:

- random starting graph on fixed number of nodes
- many iterations (e.g. 1,000,000)
- sample every 1000th graph (for example)

The algorithm sets up a Markov Chain on the space of all possible nondirected networks that has Pr(**X**=**x**) as its stationary distribution

- at each step, we consider changing the value of the (*i,j*) tie (from 1 to 0 or 0 to 1) for a randomly selected pair (and relation): $\mathbf{x} \rightarrow \mathbf{x}'_{ij}$
- the change is made with probability

$$\min[1,\exp(\{\sum_Q \lambda_Q (z_Q(\mathbf{x}'_{ij})-z_Q(\mathbf{x})\})]$$

SWIN BUR NE

Simulation: Intuitively

Fixed number of nodes; fixed parameter values

1. Start from a random graph
2. For each step, propose to change one edge at a time
    - If the probability of the graph increases, make the change
    - If the probability decreases, don't make the change (EXCEPT SOMETIMES – this makes it a proper statistical distribution)
3. Throw away the early iterations so the starting graph has no effect on the distribution – "burn-in"
4. Sample as many graphs as needed
5. Stop after a suitable number of iterations

MCMC: important points

- To draw a distribution of graphs we have to rely on iteratively generating graphs using a random walk
- You have to make sure
  - Process has forgotten about the past (burn-in)
  - Chain is “moving freely”
- The “multiplication factor” controls this
  - Large -> more iterations for exploring
  - Large -> more iterations means longer time

Exercise: Illustrate burn-in

Use PNet to simulate a Bernoulli distribution of non-directed 50 node graphs

- Edge parameter = -2
- Take 1,000 samples from a simulation of run of 100,000.
- Set the *burn-in* to be zero

Do the simulation twice

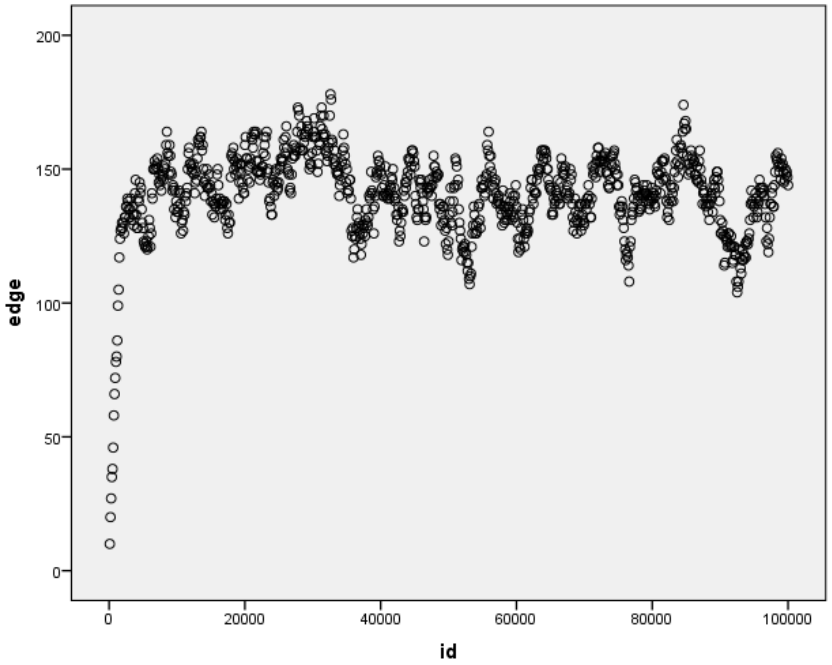
1. Starting density = 0 (empty graph)
2. Starting density = 1 (complete graph)

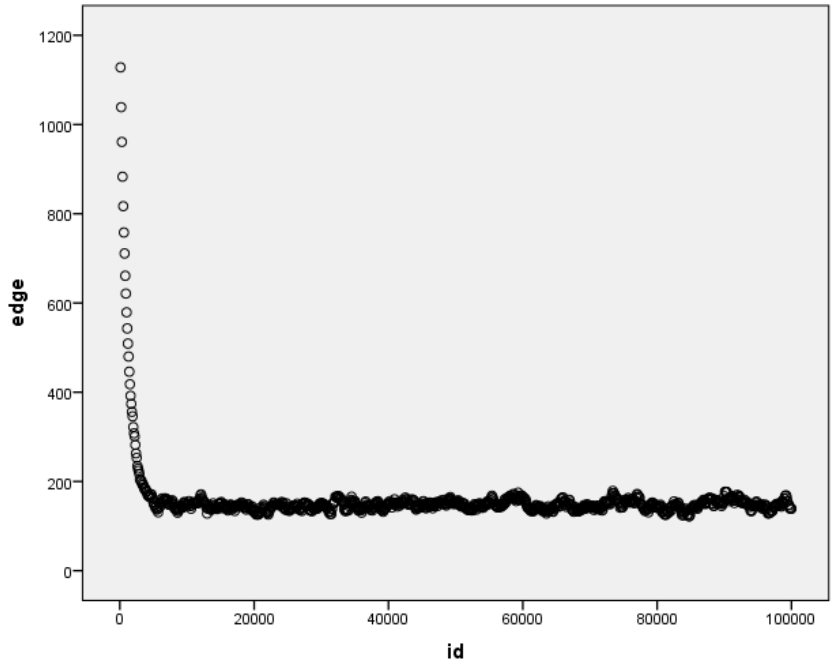
The *simulation_(session name).txt* file contains the number of edges of each sampled graph against the simulation number (ID). Draw a scatterplot for each simulation.

- Notice how each distribution ends up in the same way (same mean number of edges), although starting from a different graph.

SWIN BUR NE

Burn-in





Start simulation from empty or complete graph, end up in same place.
Remove the first few simulations that depend on the starting point – *burn-in*

SWIN BUR NE

Exercise: Markov vs Social circuit
(Chapter 13.1.1, Lusher et al, 2013)

- 30 nodes;
- Undirected graph
- fix density 0.10;
- burnin 100,000;
- Number of iterations 1,000,000
- Number of samples 1,000

Exercise: Markov vs Social circuit
(Chapter 13.1.1, Lusher et al, 2013)

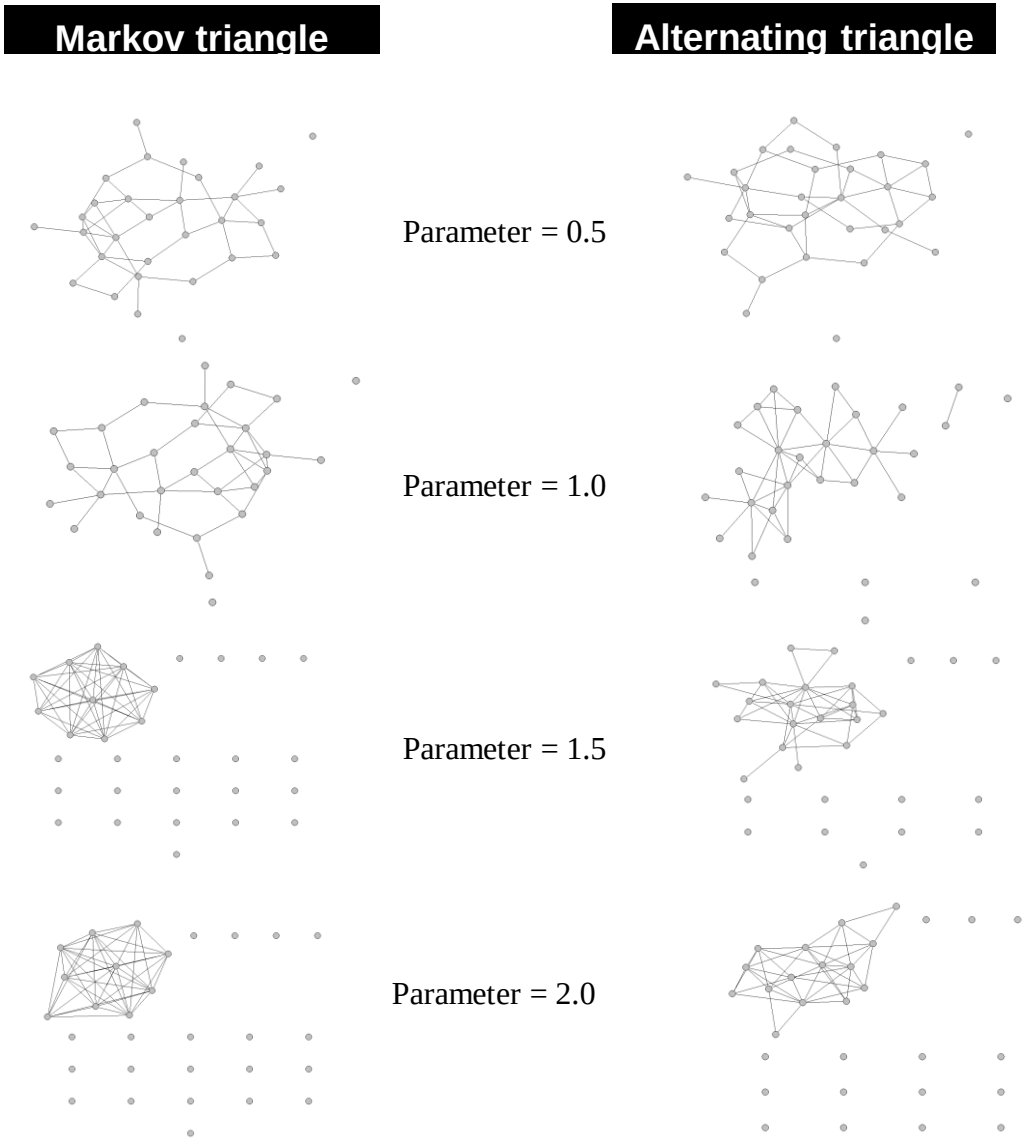
1. Run three simulations with Markov triangle parameter
   - ❑ Triangle = 0.5
   - ❑ Triangle = 1.0
   - ❑ Triangle = 1.5

In each case, draw the final graph. Note a sudden jump to a full clique as the triangle parameter increases

2. To compare, run three simulations with alt. triangle parameter (λ = 2)
   - ❑ AT = 0.5
   - ❑ AT = 1.0
   - ❑ AT = 1.5

SWIN BUR NE

Figure 13.2, Lusher et al

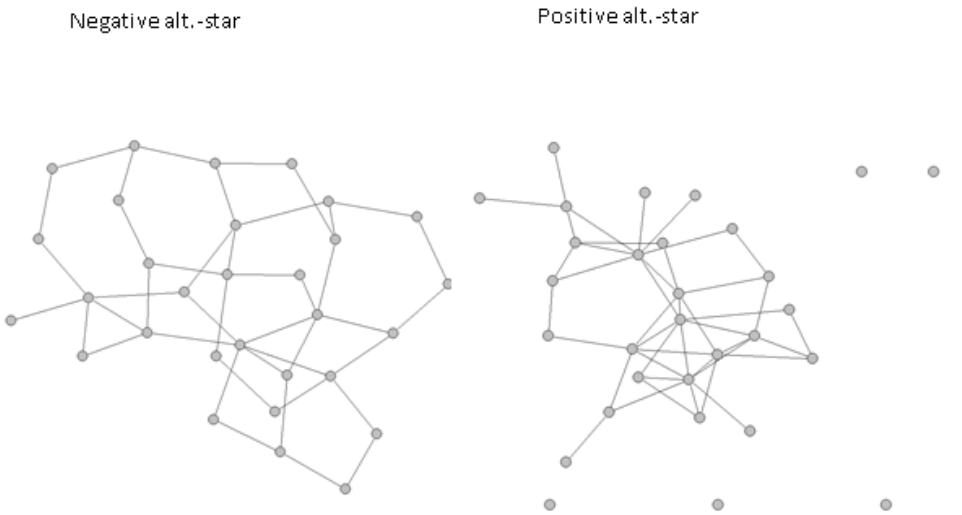


Exercise: Positive and negative stars parameters
(Chapter 13.1.2, Lusher et al, 2013)

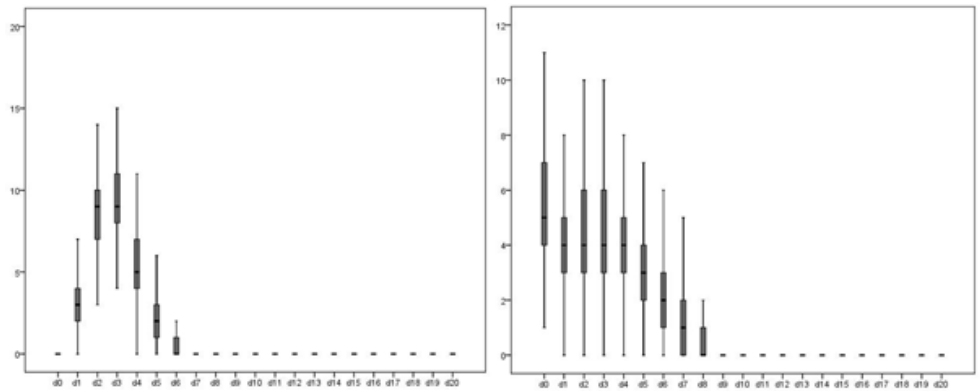
1. Run two simulations with only alternating star parameters (λ = 2)
   - ❑ AS = - 1.0
   - ❑ AS = + 1.0

In each case, draw the final graph. Note that the positive star parameter graph is more centralized with some higher degree nodes. The negative star graph has degrees more evenly spread among nodes.

Figure 13.5, Lusher et al (2013)



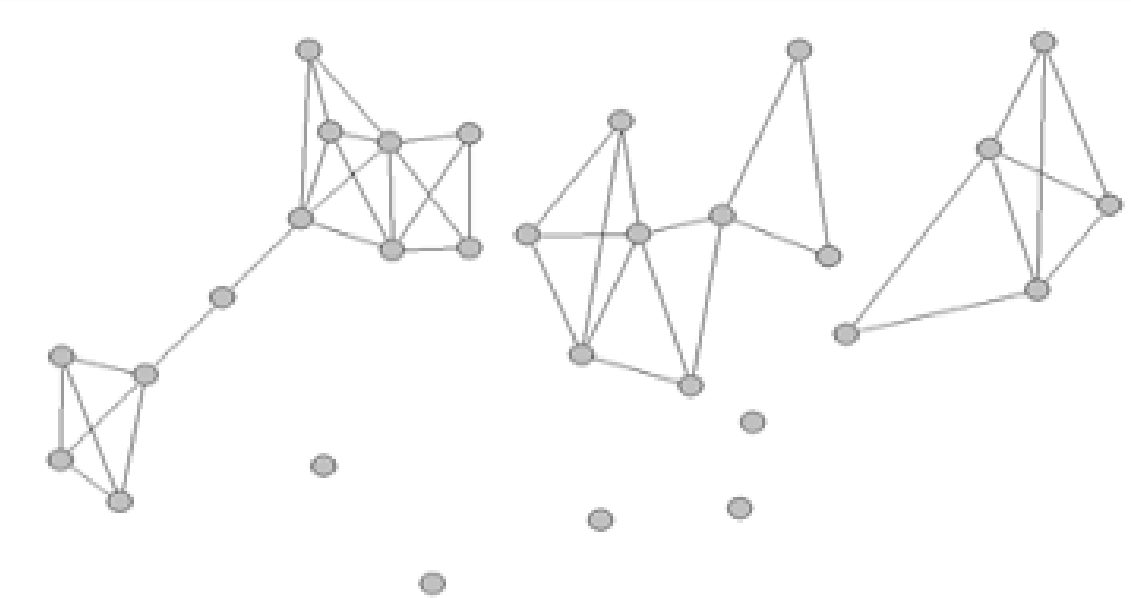
Degree distributions across 1000 graphs



Exercise: Star and triangle parameters together
(Chapter 13.1.3, Lusher et al, 2013)

1. Run a simulation with alternating star and triangle parameters together (λ = 2)
   - ❑ AS = - 1.0
   - ❑ AT = + 2.0

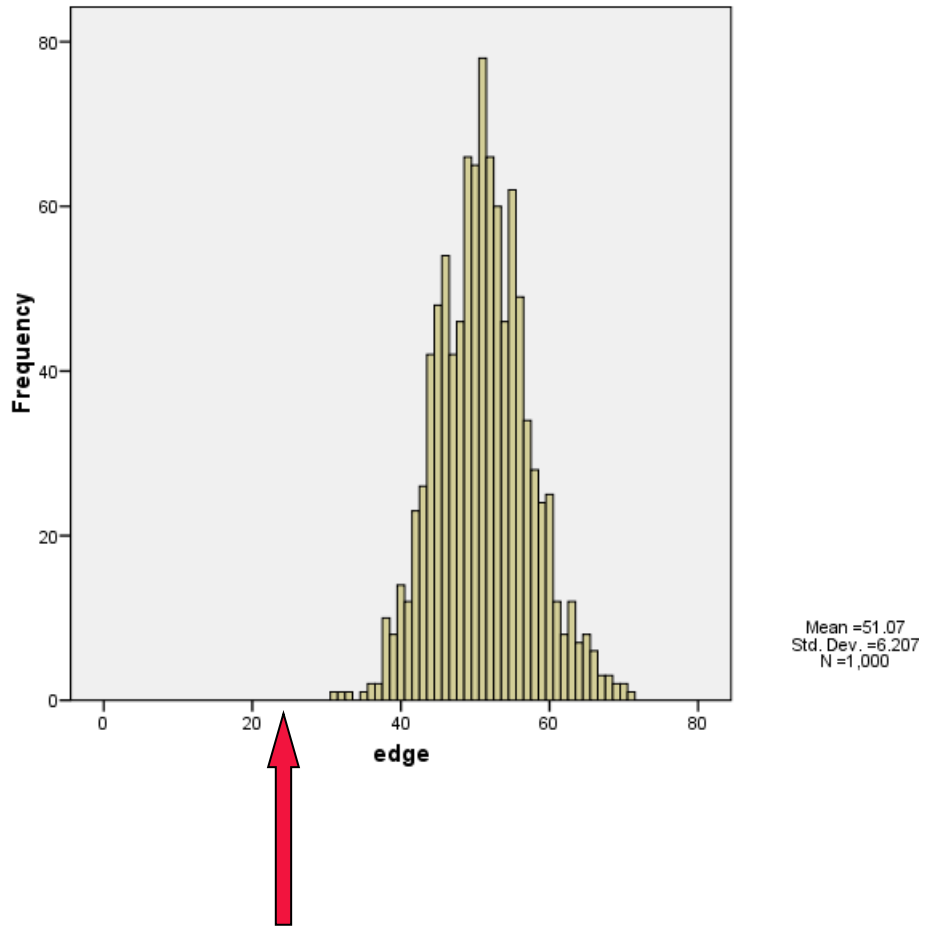
Figure 13.7, Lusher et al



Markov Chain Monte Carlo Maximum Likelihood Estimation (MCMCMLE)

Simulation of a distribution of random graphs from a starting set of parameter values, and subsequent refinement of the parameter values by comparing the distribution of graphs against the observed graph, with this process repeated until the parameter estimates stabilize. (convergence)

Fit a Bernoulli model to a network
First guess at parameter : θ = – 1



Suppose the observed graph on 20 nodes has 23 edges

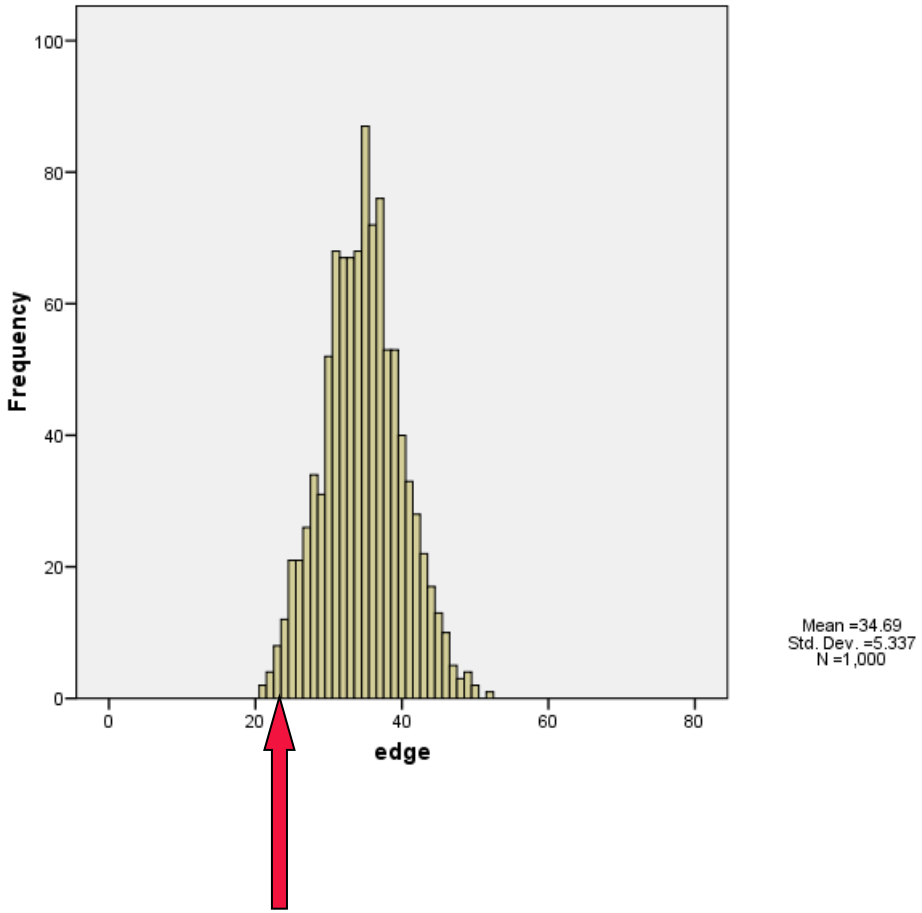
Simulate distribution using θ = -1

t-value for observed graph = – 4.52

ADJUST PARAMETER GUESS

Observed graph is OFF THE SCALE!! **Θ = -1 is not a good estimate**

Fit a Bernoulli model to a network
Second guess at parameter : θ = – 1.5



Observed graph

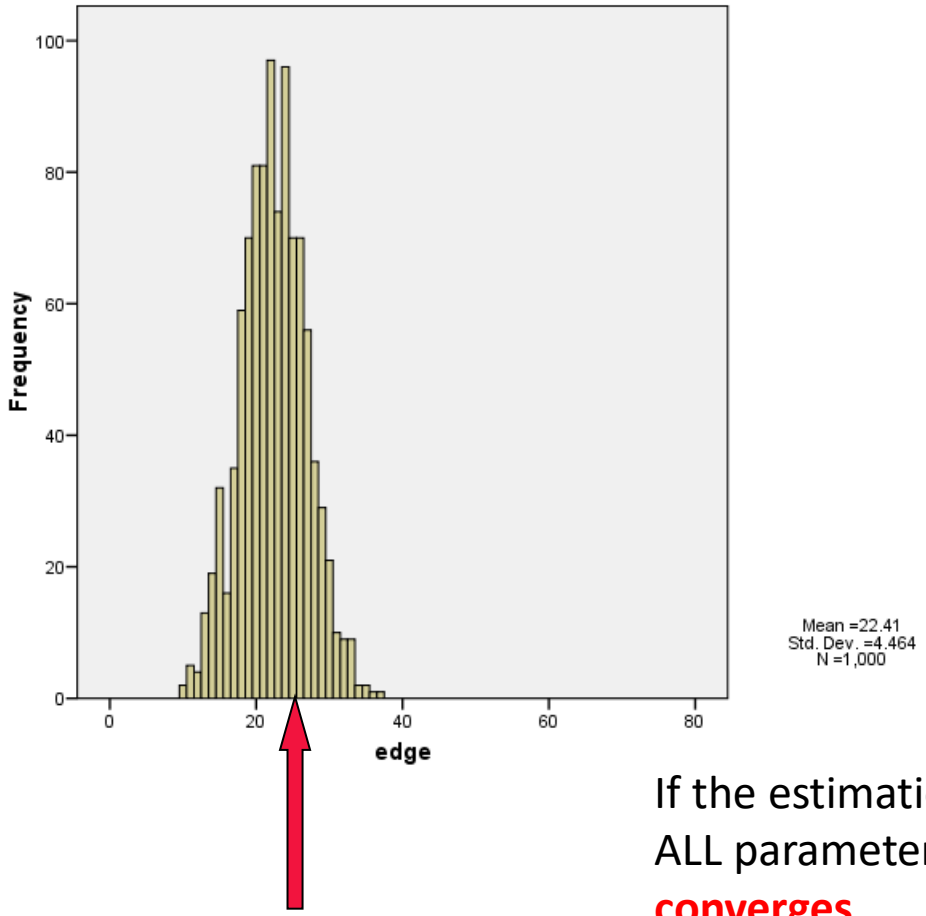
t-value for observed graph
= – 2.19

Still not good!

ADJUST PARAMETER GUESS

SWIN BUR NE

Fit a Bernoulli model to a network
Third guess at parameter : θ = – 2



Observed graph

t-value for observed graph
= + 0.13

A pretty good result!
θ = - 2 is a plausible estimate for this observed graph

If the estimation can find such a good result for ALL parameters simultaneously, the model **converges**

MCMCMLE - "Method of moments"

Snijders (2002): **Polyak-Ruppert variant of the Robbins-Monro procedure to solve the moment equation $E_\lambda\{z(X)\} = z(x)$ for λ:**

- ***Phase 1***: estimate a diagonal matrix D_0 to be used in Phase 2:

  $D_0 = \text{diag}(d_{kk})$ where $d_{kk} = \partial E_\lambda z_k(X)/\partial\lambda_k$ evaluated at initial value of λ

- ***Phase 2***: Over several subphases:
  - generate network statistics according to current estimate $\lambda^{*(n)}$ of λ at step n
  - use update step $\lambda^{*(n+1)} = \lambda^{*(n)} - \alpha_n D_0^{-1} z(X_n)$, where X_n are generated from $P_\lambda(X)$ for $\lambda = \lambda^{*(n)}$, gain sequence $\alpha_n = n^{-c}$, for $.5 < c < 1.0$
  - at end of subphase, set new $\lambda^{*(n+1)}$ as average value over subphase

- ***Phase 3***: estimate the covariance matrix of the estimator as the inverse of the covariance matrix of statistics (and check convergence)

SWIN BUR NE

More intuitively - "Method of moments" as implemented in PNET

- ***Phase 1***: Essentially gets the process started to reach some very rough and ready parameter "guesses"

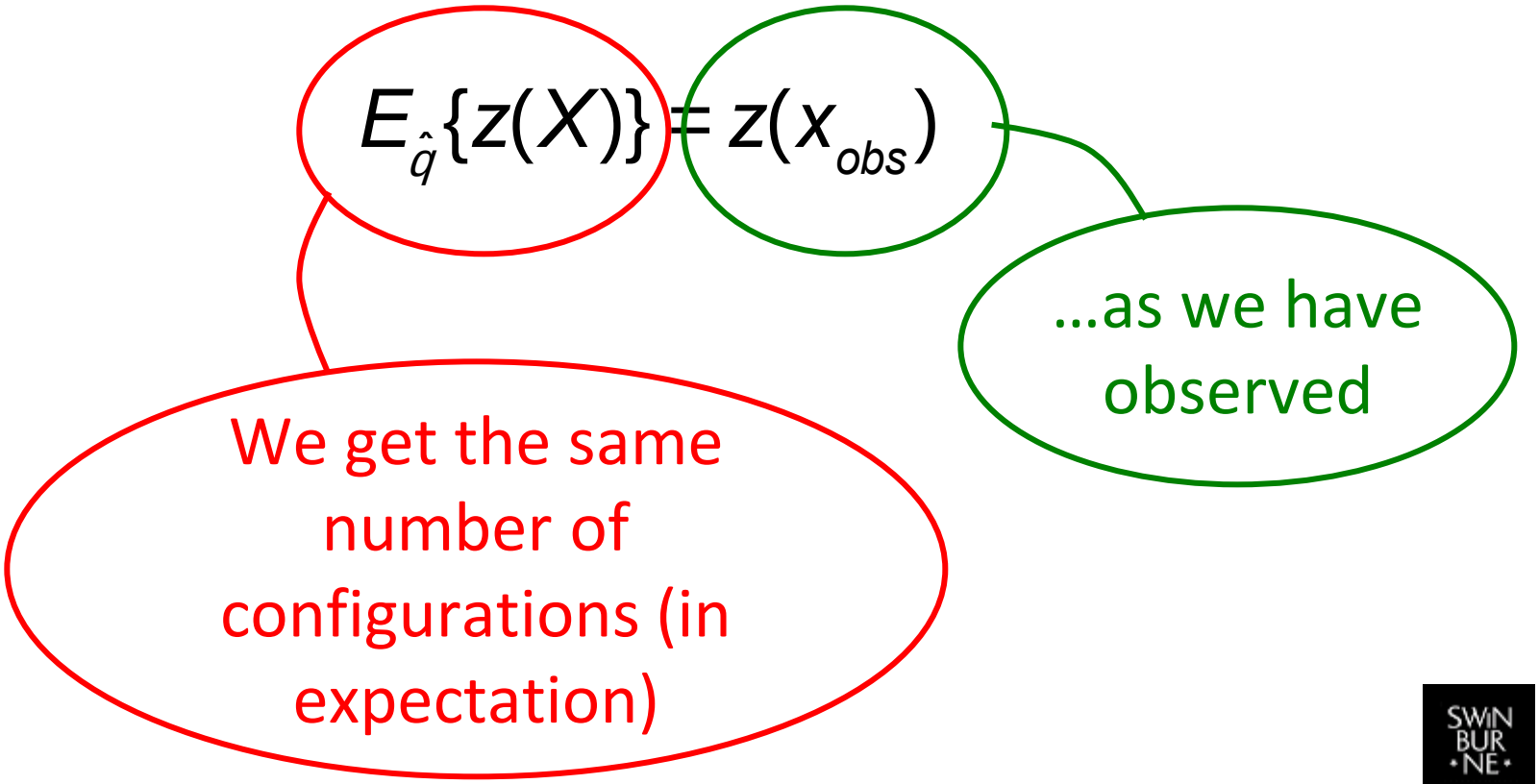
- ***Phase 2***: Several subphases:
  - In each subphase we simulate from the current parameter estimates and take a sample of graphs. The size of the simulation is determined by the "Multiplication factor".
  - Check the observed graph against the simulation
  - Change the parameter estimate multiple times within each subphase – for every subphase, the size of the changes permitted get smaller and smaller – hopefully more and more precise (the "Gaining factor")
  - Stop at the end of the specified number of subphases

- ***Phase 3***: Simulates from the final parameter estimates, checks convergence and estimates standard error.

SWIN BUR NE

Main aim of ML- “Method of moments”

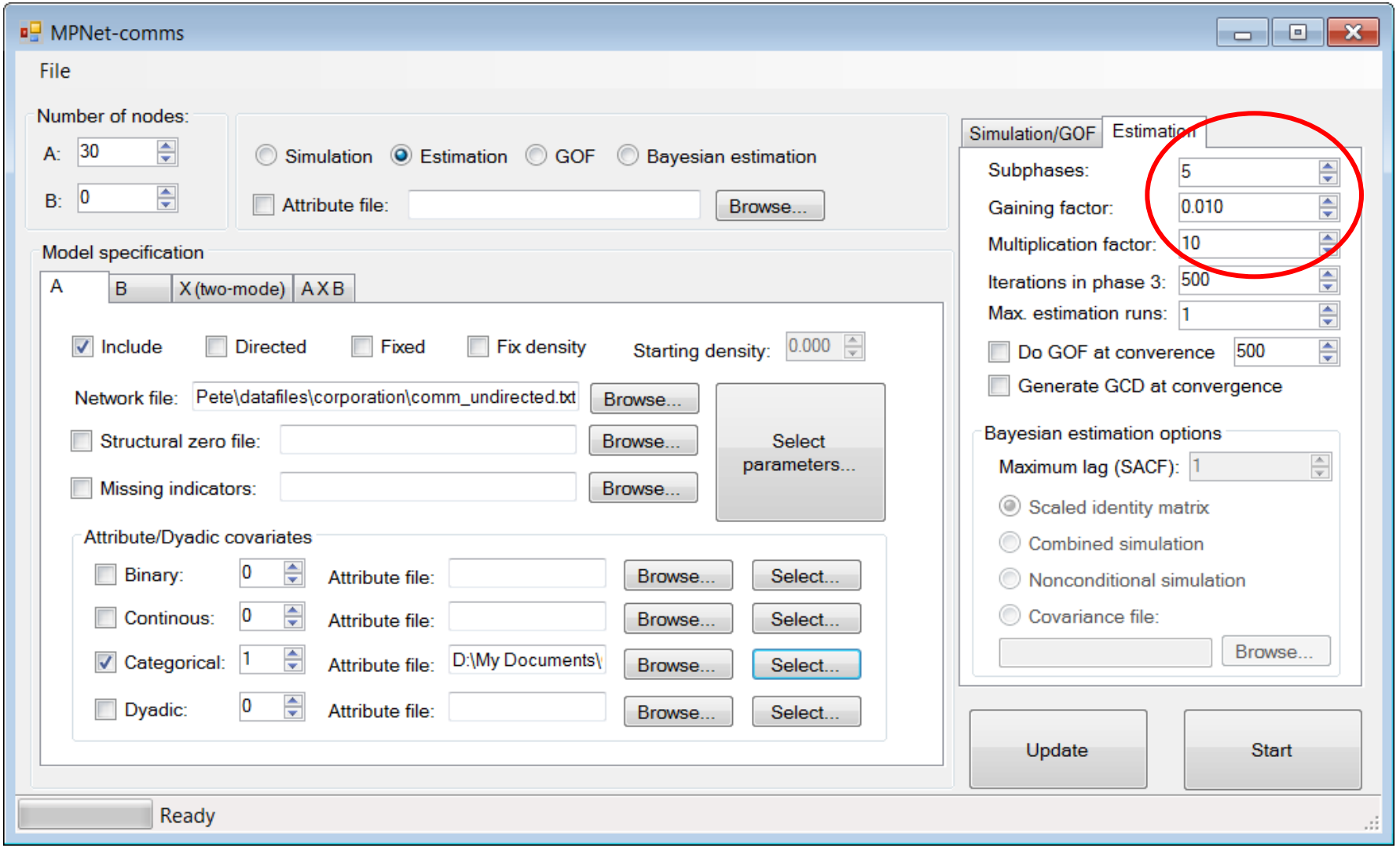
- Find those parameter values such that



SWIN BUR NE

NOTE: **CONVERGENCE IS NOT GUARANTEED!!**

- So:
  - If hard to get convergence, try with bigger multiplication factor (longer simulations mean more of the possible graph space covered, but of course it takes longer)
  - If close to convergence, can use a smaller gaining factor (already quite precise)
  - Results differ slightly from run to run – this is a stochastic algorithm


MPNet-comms
File
Number of nodes:
A: 30
B: 0
Simulation
Estimation
GOF
Bayesian estimation
Attribute file:
Browse...
Model specification
A
B
X (two-mode)
A X B
Include
Directed
Fixed
Fix density
Starting density: 0.000
Network file: Pete\datafiles\corporation\comm_undirected.txt
Browse...
Structural zero file:
Missing indicators:
Select parameters...
Attribute/Dyadic covariates
Binary: 0
Continous: 0
Categorical: 1
Dyadic: 0
Attribute file:
D:\My Documents\
Select...
Simulation/GOF
Estimation
Subphases: 5
Gaining factor: 0.010
Multiplication factor: 10
Iterations in phase 3: 500
Max. estimation runs: 1
Do GOF at converence 500
Generate GCD at convergence
Bayesian estimation options
Maximum lag (SACF): 1
Scaled identity matrix
Combined simulation
Nonconditional simulation
Covariance file:
Update
Start
Ready

The important part of the PNet output

effects	estimates	stderr	t-ratio	
edge	-3.219886	0.69602	-0.07846	*
2-star	0.196034	0.26175	-0.06178	
3-star	-0.200038	0.12604	-0.04406	
Triangle	1.886621	0.27807	-0.05492	*

Asterisk indicates absolute value of estimate more than twice standard error

Parameter estimate

(strength of effect: positive usually means more of the configurations are observed; negative means fewer)

Standard error

(If the value of the parameter estimate is more than twice the standard error, the parameter is significant)

Convergence statistic

(If this value is less than 0.1, then there is good evidence that the estimation has converged.

You can increase this for a very large network.)

What happens if I can't get convergence statistics less than 0.1?

Click the "update" button on pnet

This updates the starting parameter values to those you last estimated.

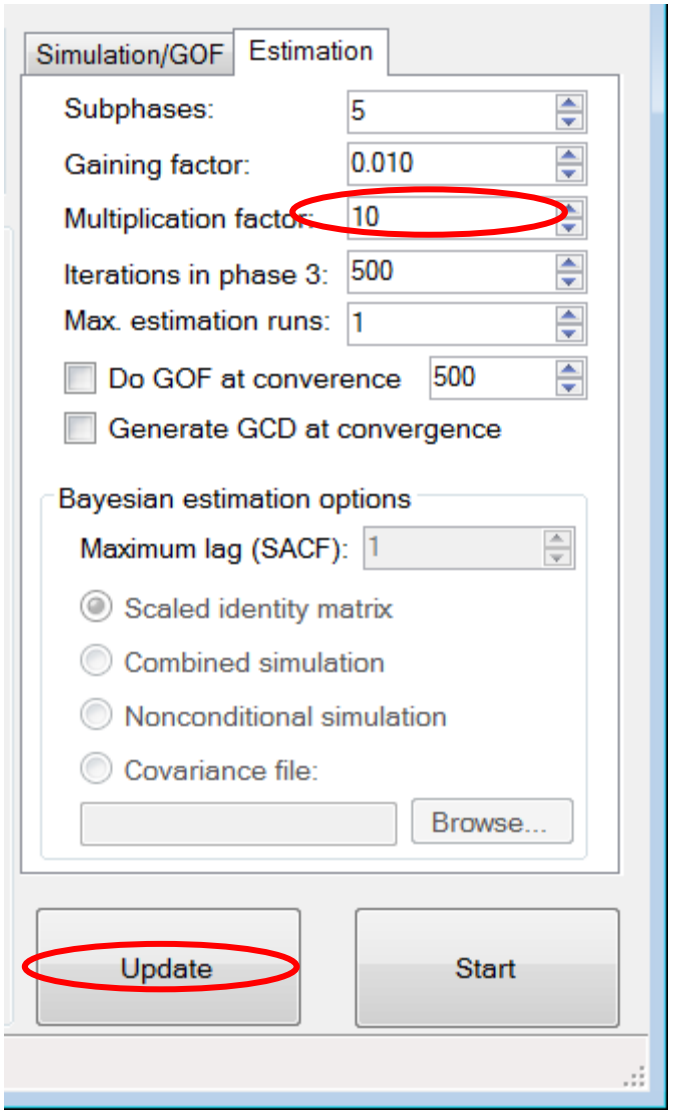
Run it again.

If the model is well-behaved, you should eventually get convergence.

If you still have troubles, try increasing the "Multiplication factor".

Badly behaved models will not converge!!

You then need to specify the model in a different way.



What happens if I can't get convergence statistics less than 0.1?

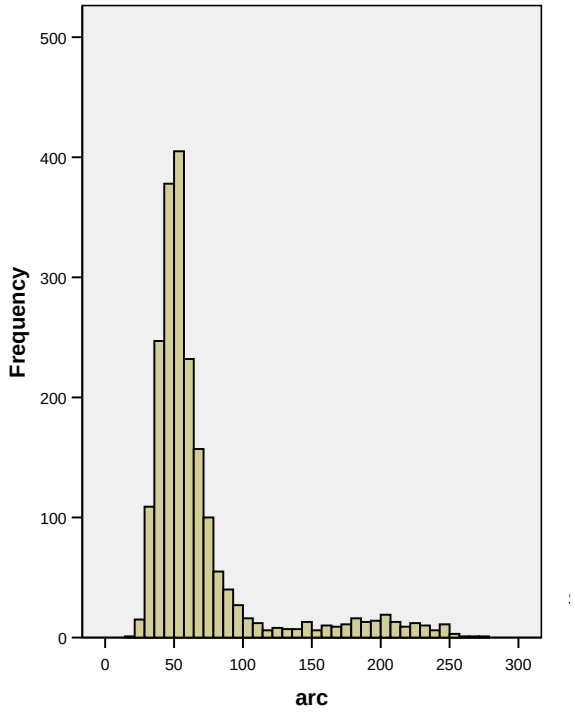
If it looks like you have converged parameter estimates – congratulations!

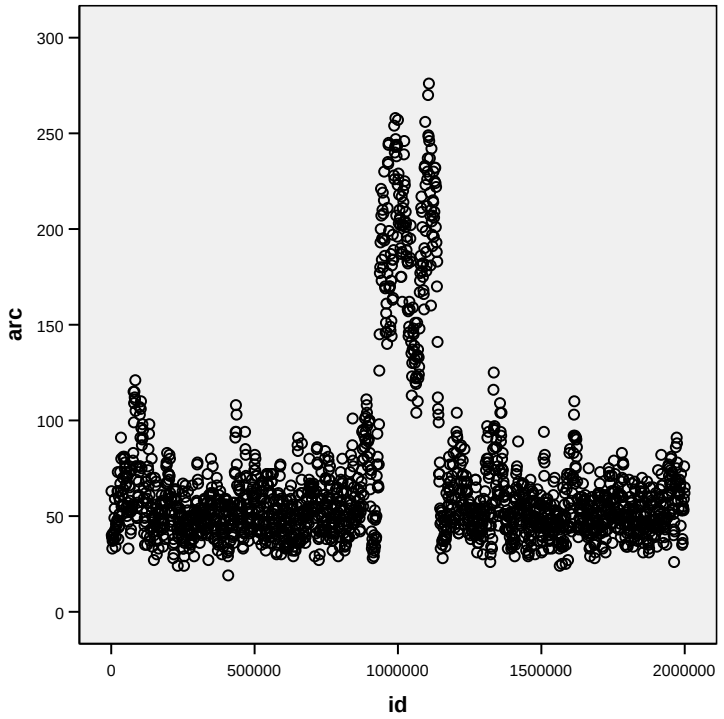
BUT you can't be entirely sure – good to do a long simulation using your good looking estimates.

Sometimes a long simulation run reveals two (or more) REGIONS of graphs

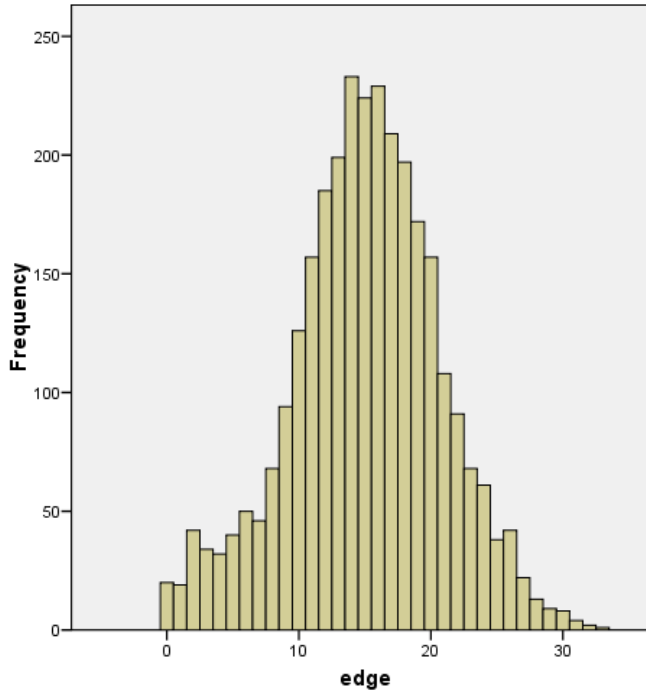
This is called a "near degenerate" model, and is NOT GOOD

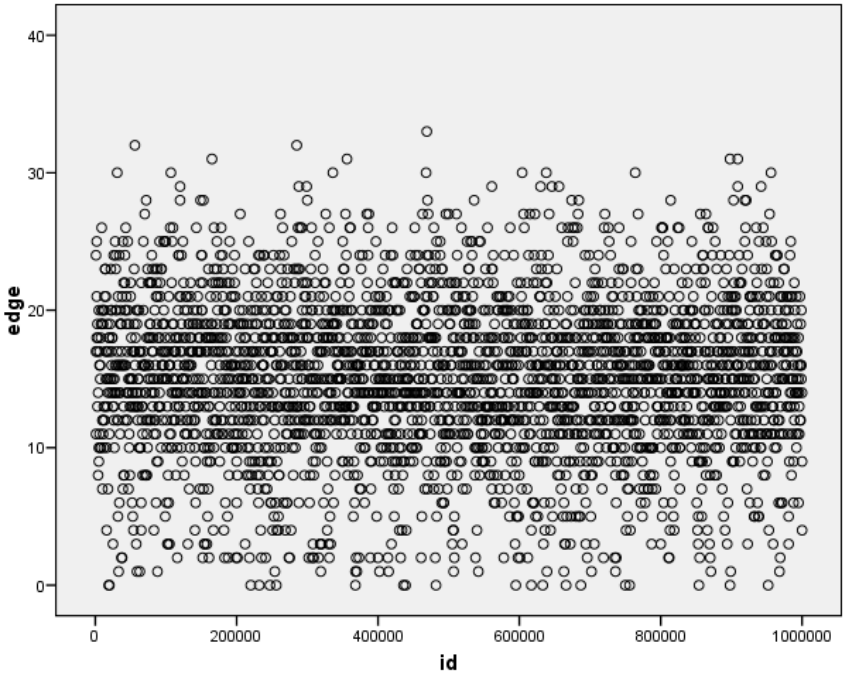
SWIN BUR NE





What we DON'T want to see:
the model exhibits two regions!
This would indicate a bad model.





What we DO want to see:

the model exhibits only one region for all parameters

This suggests a well-behaved model.

SWIN BUR NE