# Outline

- Community detection
  - Generative model for modular networks
  - Variational Bayesian inference
  - Validation
- Applications

# Community detection

- **Model *global* structure** (e.g. summarize data)
- **Visualize** structure (e.g. graph layout)
- **Analyze** interactions (e.g. affinities within/between groups)
- **Predict** (e.g. function, attributes, links)

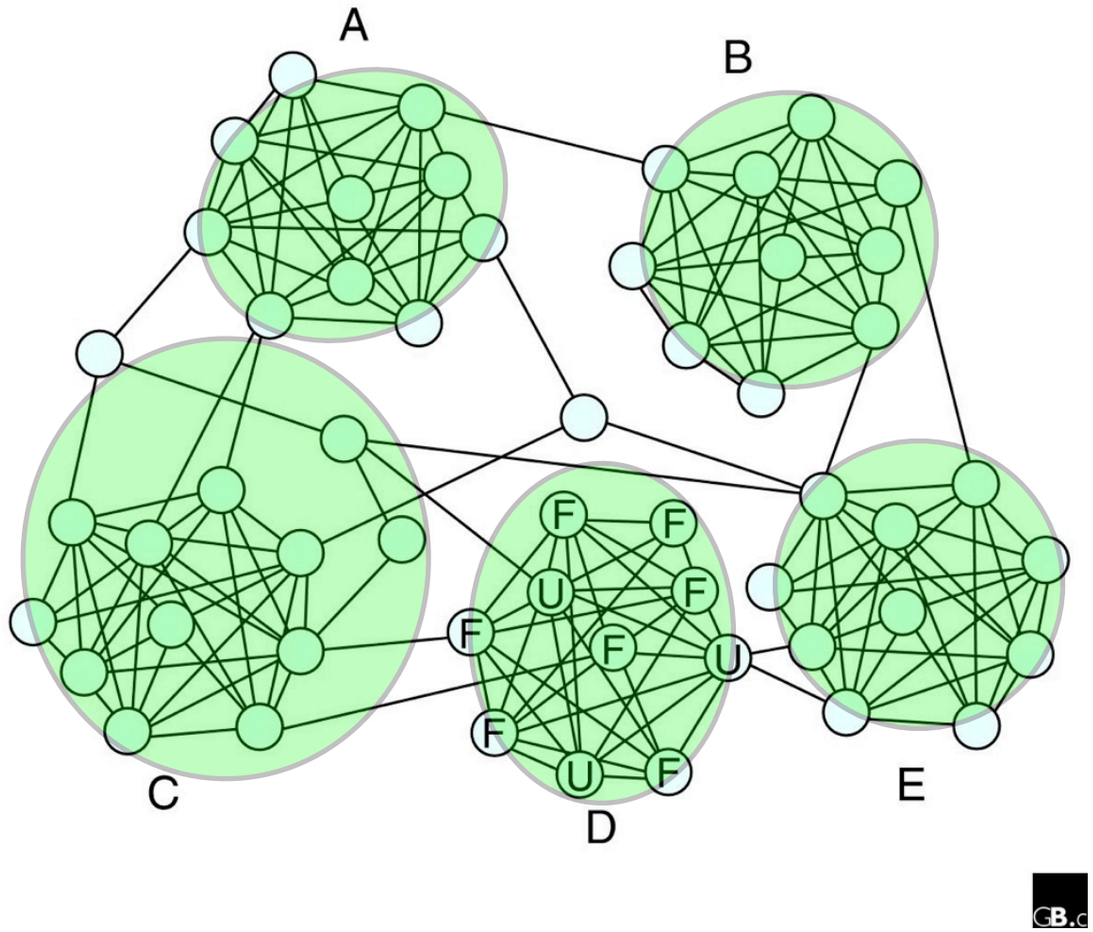


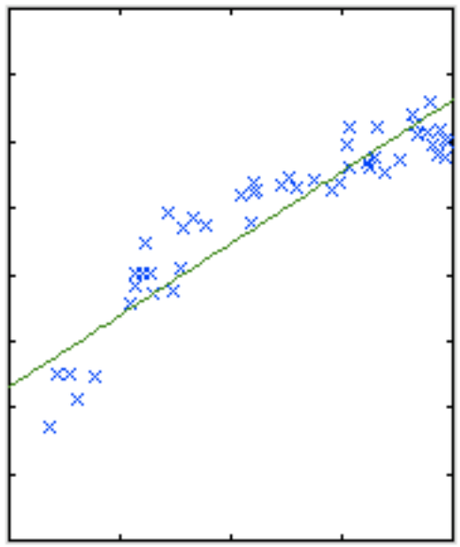

Image from Giorgini, et. al., Genome Biol, 2005

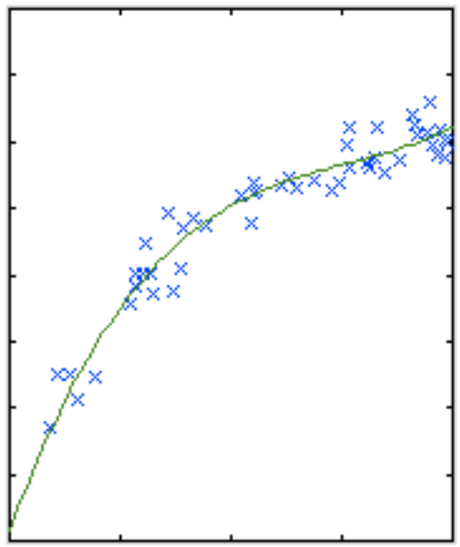
# Community detection: Background

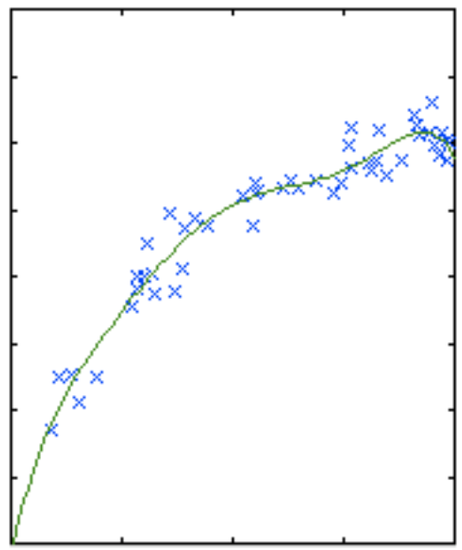
- Physics literature
  - Newman et. al. (2002,...)
  - Bornholdt & Reichardt (2006)
  - Hastings (2006)
  - ...
- Parametrized cost function (energy), mostly focus on ***how* to optimize**

- Machine learning literature
  - Nowicki & Snijders (2001)
  - Kemp et. al. (2004)
  - Leicht & Newman (2007)
  - Airoldi et. al. (2007)
  - Xu et. al. (2007)
  - Sinkkonen et. al. (2007)
- **Complex models**, approximate inference (often **expensive**)

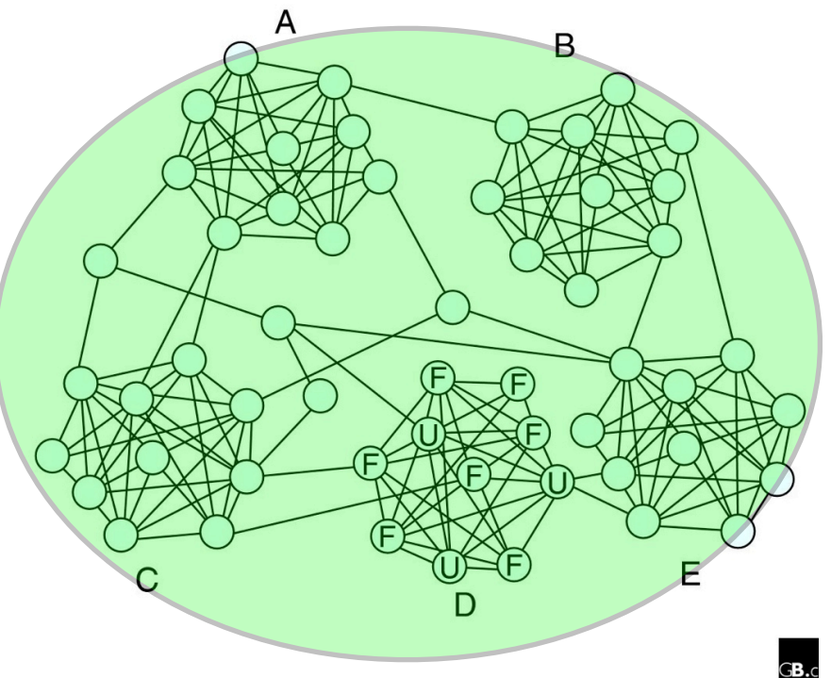
# Complexity control

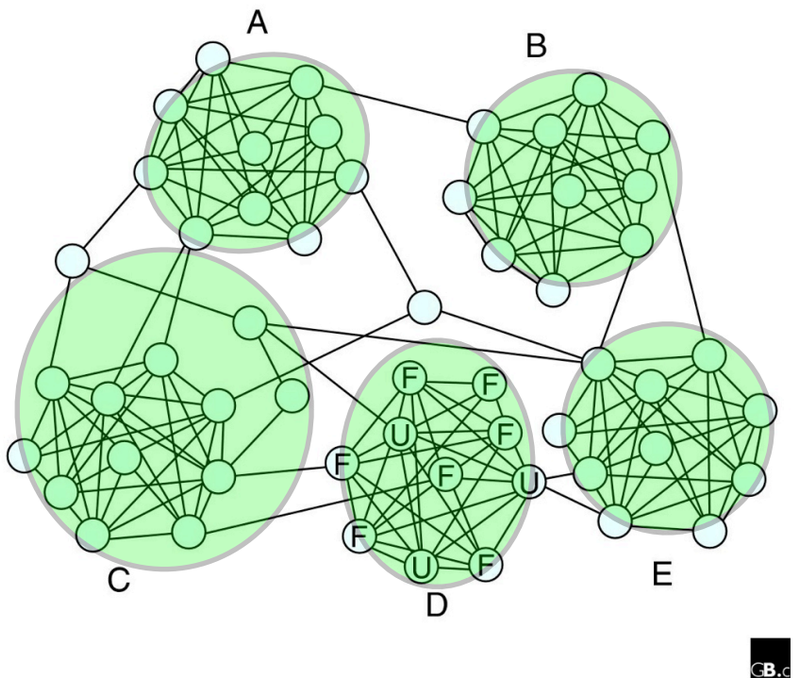


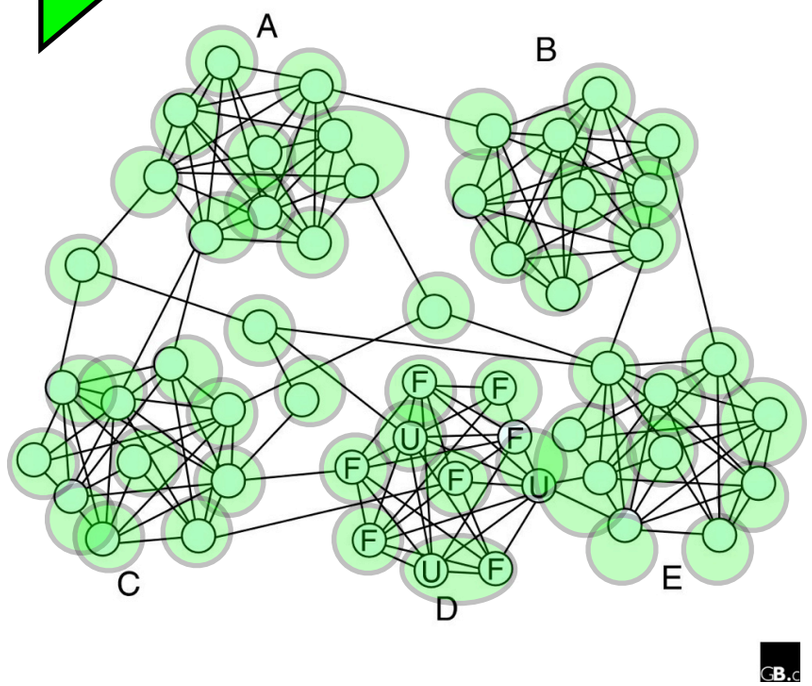




Increasing complexity



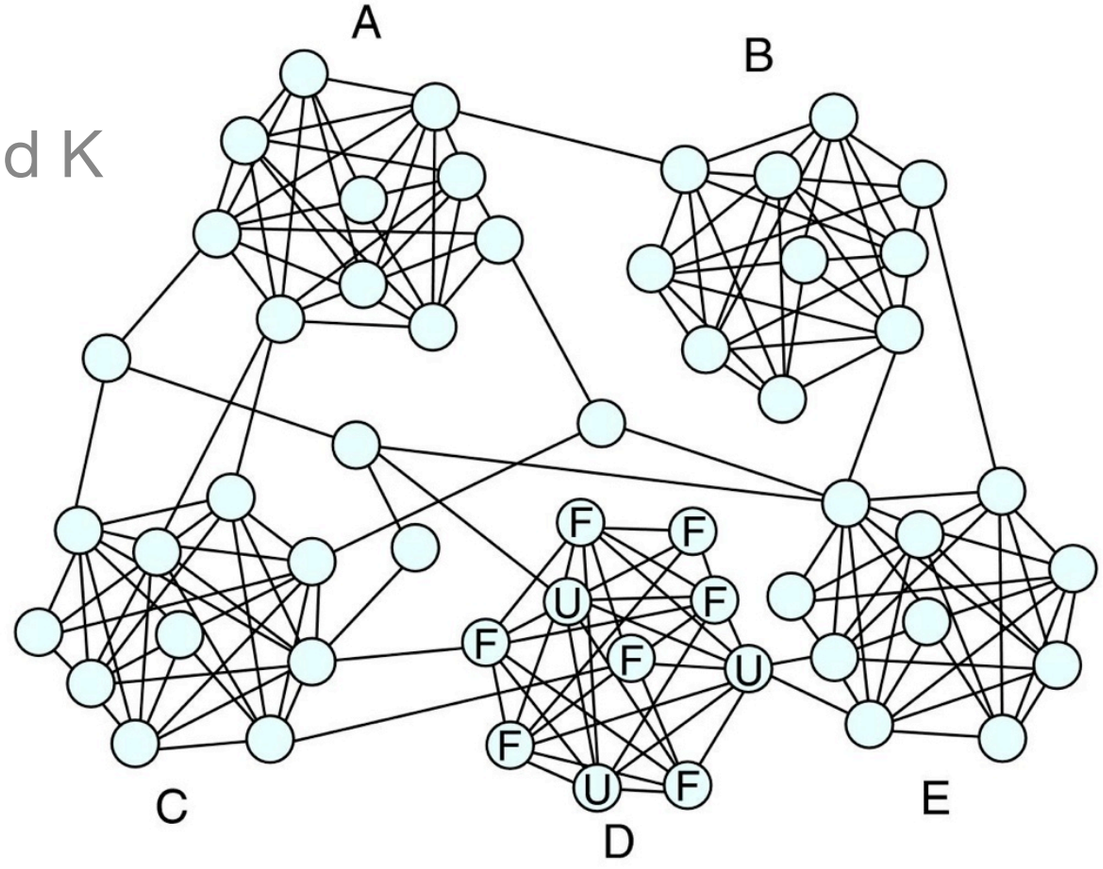




Images from http://research.microsoft.com/~minka/statlearn/demo/ and Giorgini, et. al., Genome Biol, 2005

# Community detection as inference

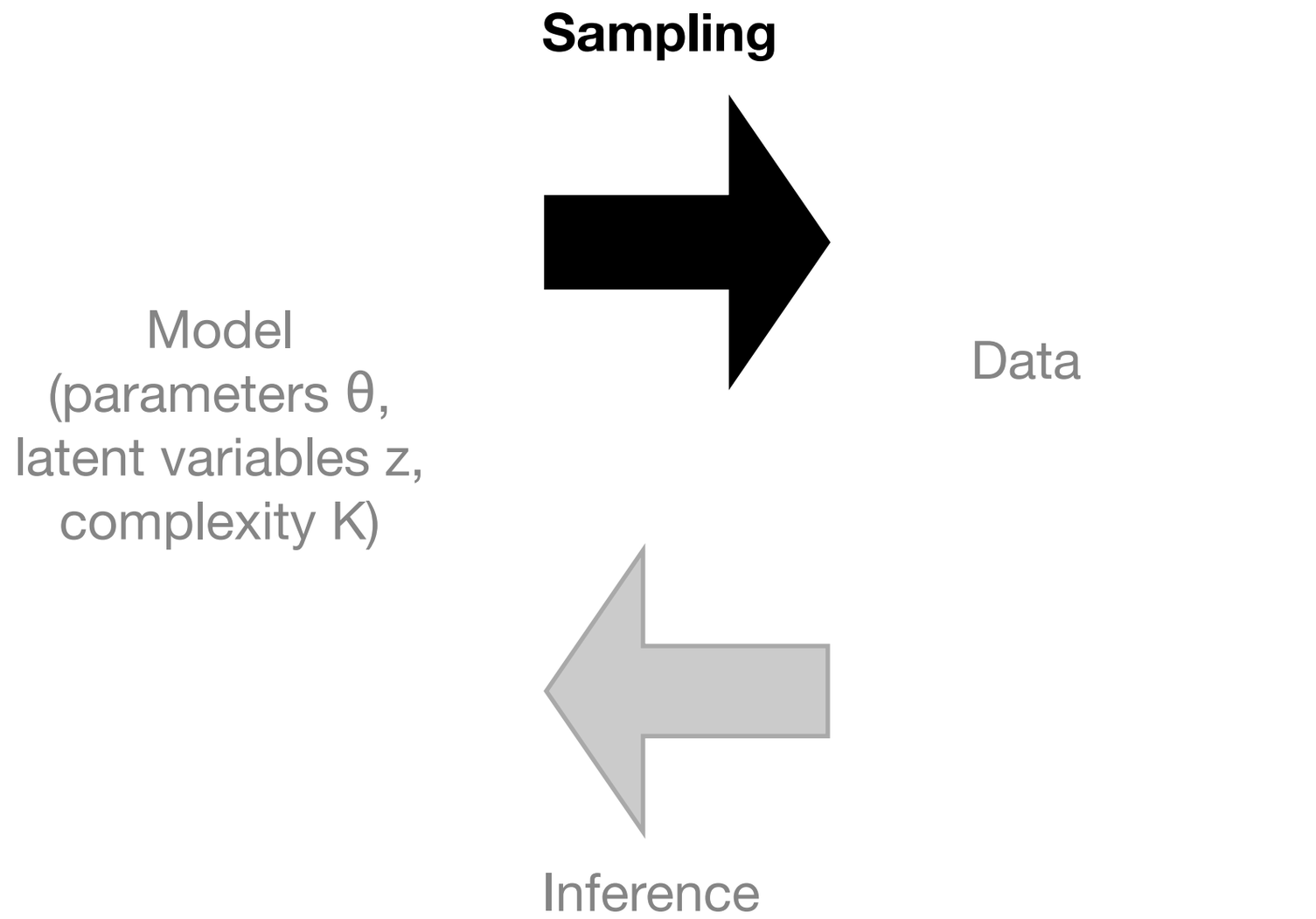
- Data $D=\{A_{ij}\}$ $i,j=1,...,N$; $A_{ij}=1$ if nodes i and j connected
- Parameters bias of die $\pi$, bias of coins $\theta$
- Latent variables $\{z_i\}$, assignments of nodes to communities
- Given D, infer z, Θ and K



# Outline

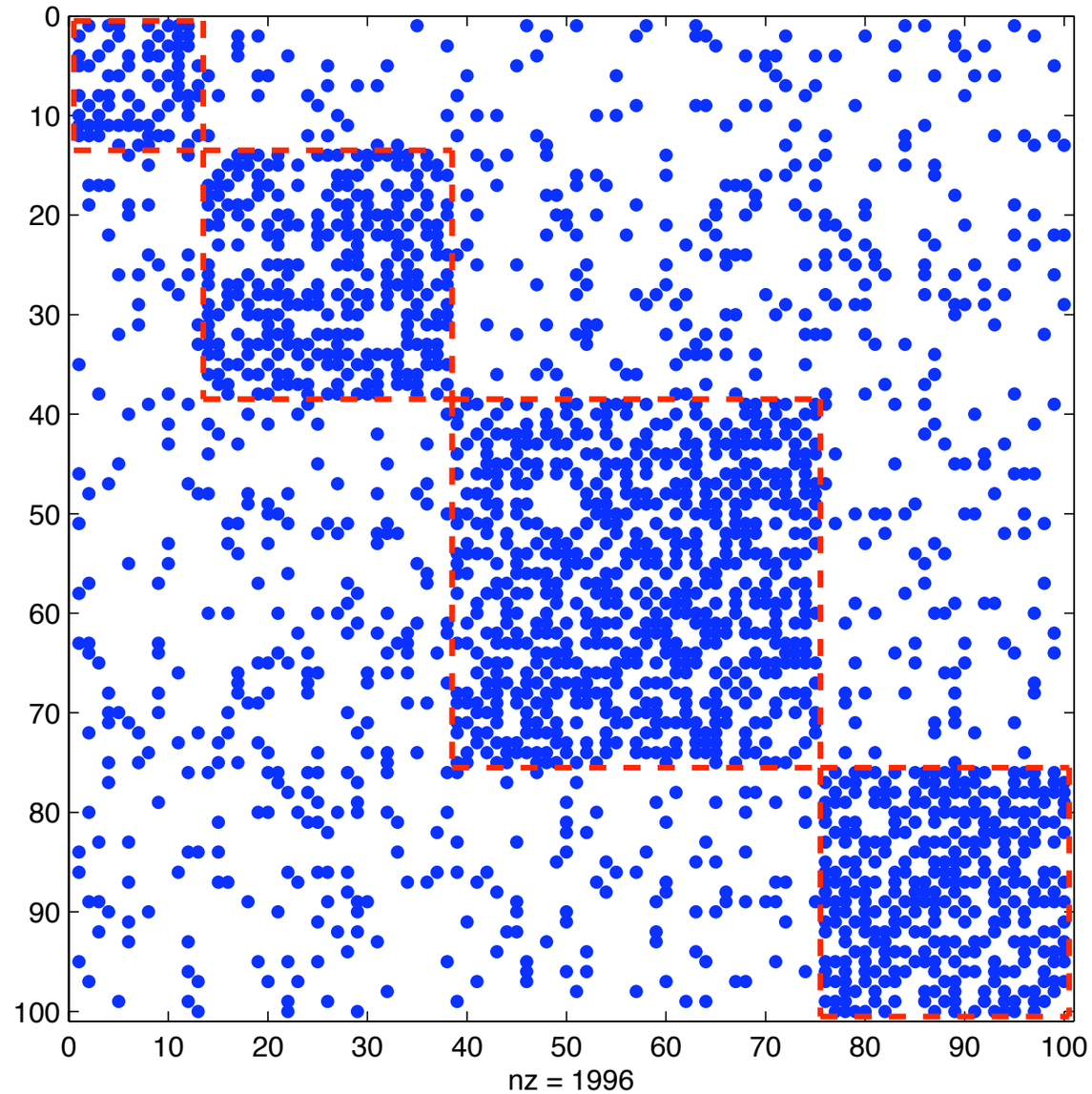
- Community detection
  - **Generative model for modular networks**
  - Variational Bayesian inference
  - Validation
  - Applications

# Community detection as inference



# Constrained stochastic block model

- **Nodes belong to "blocks"** of varying size
  - Roll die for assignment of nodes to blocks
- Probability of **edge** between two nodes **depends only on block membership**
  - Flip (one of two) coins for edges
- Result: **mixture of Erdos-Renyi** graphs



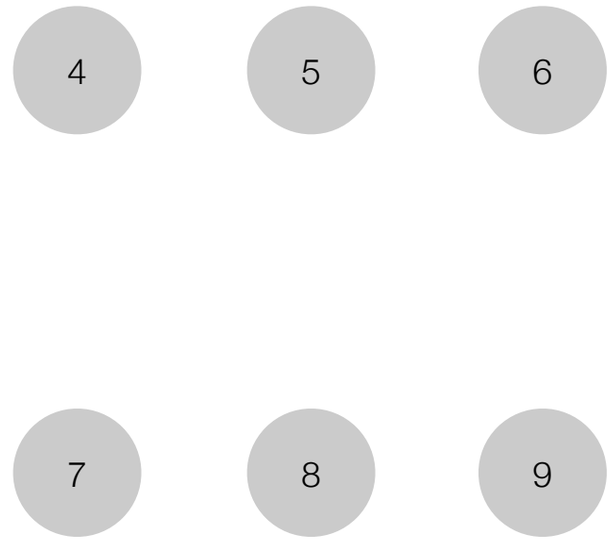

Holland, Laskey, Leinhardt 1983; Wang and Wong, 1987

# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias $\pi$ to determine $z_i=1,...,K$, the (unobserved) module assignment for $i^{th}$ node
- For each pair of nodes (i,j):
  - If $z_i=z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i \neq z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$

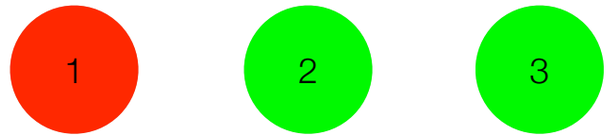
# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node
- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$

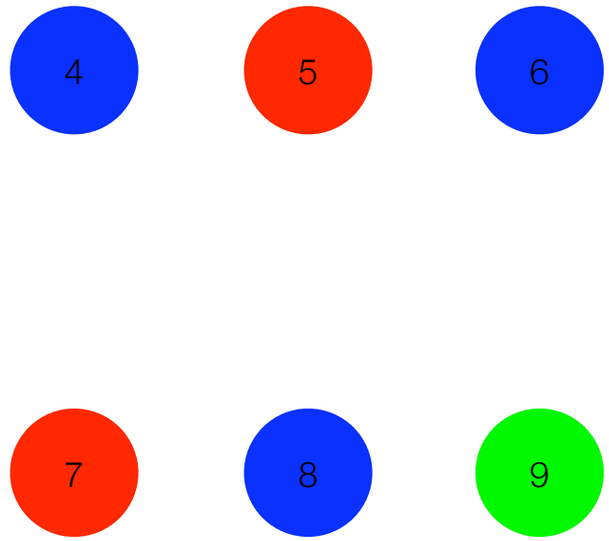



# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node

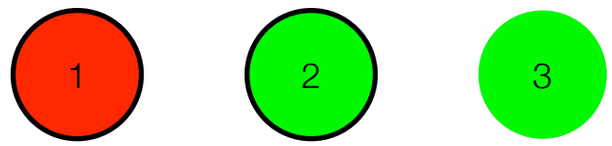



- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$

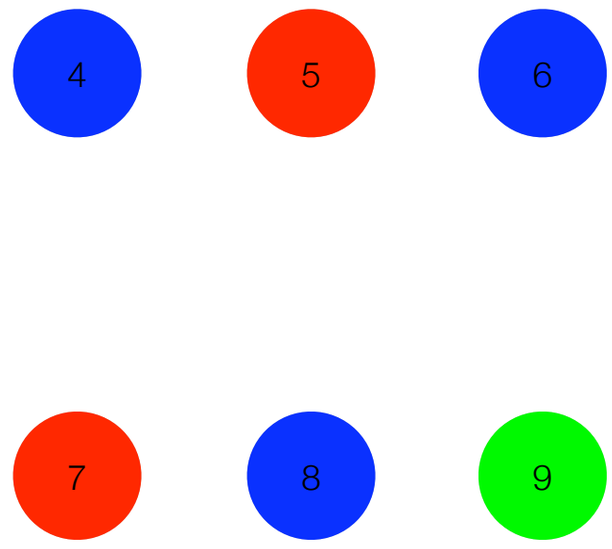


# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node

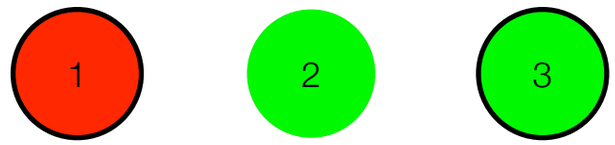



- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$

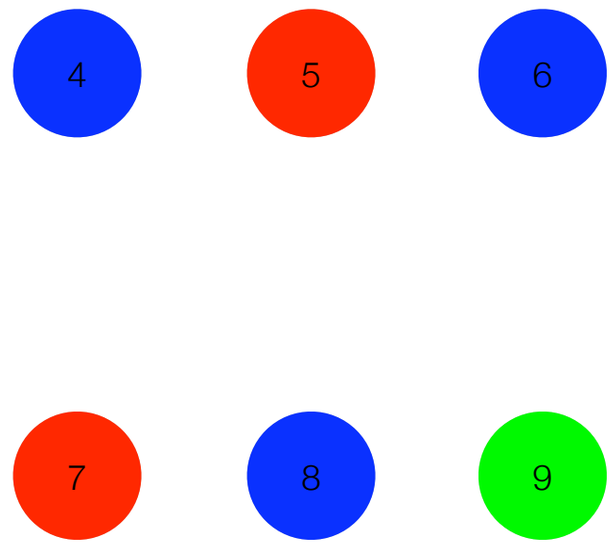


# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node

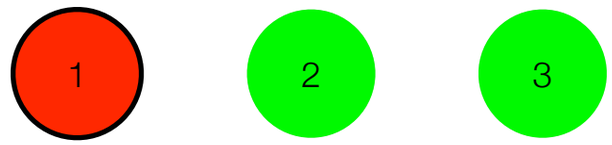



- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$

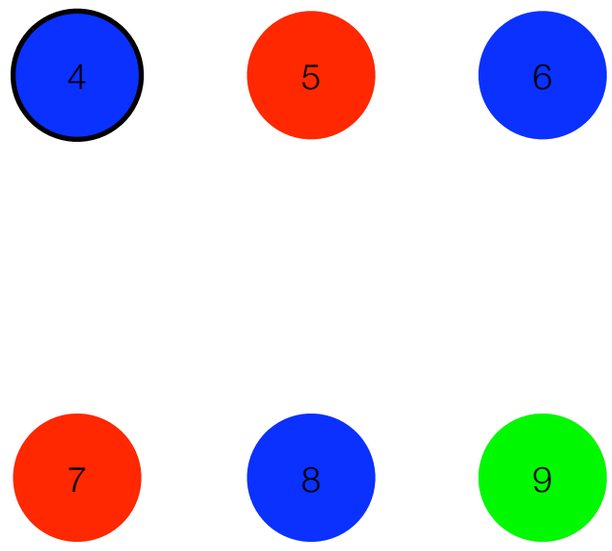


# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node

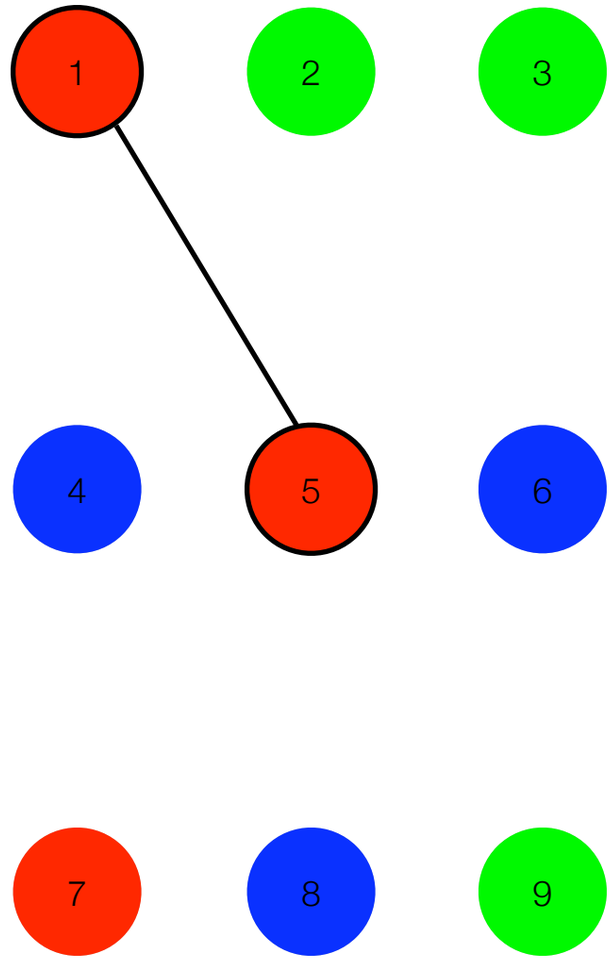



- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$



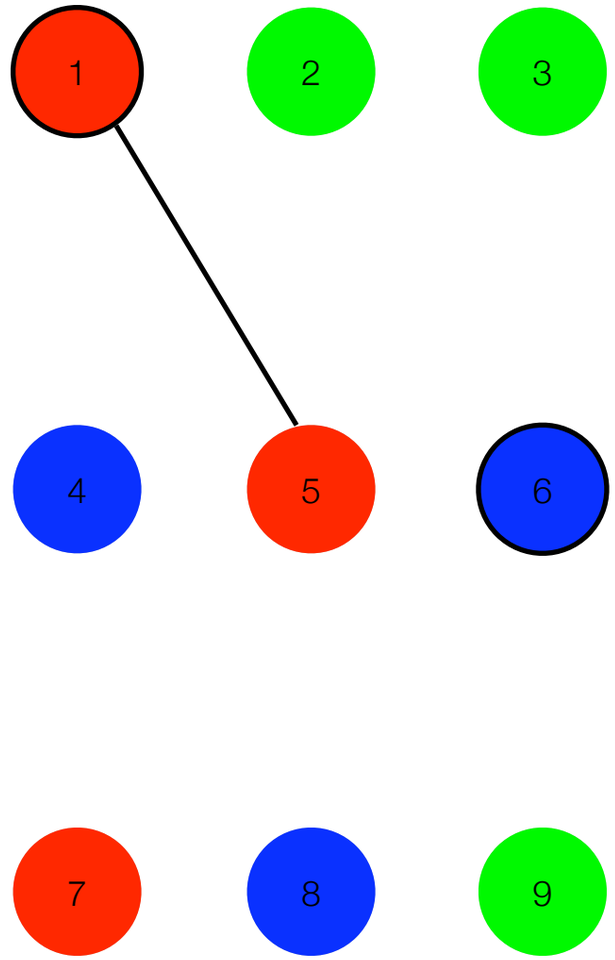
# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node
- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$



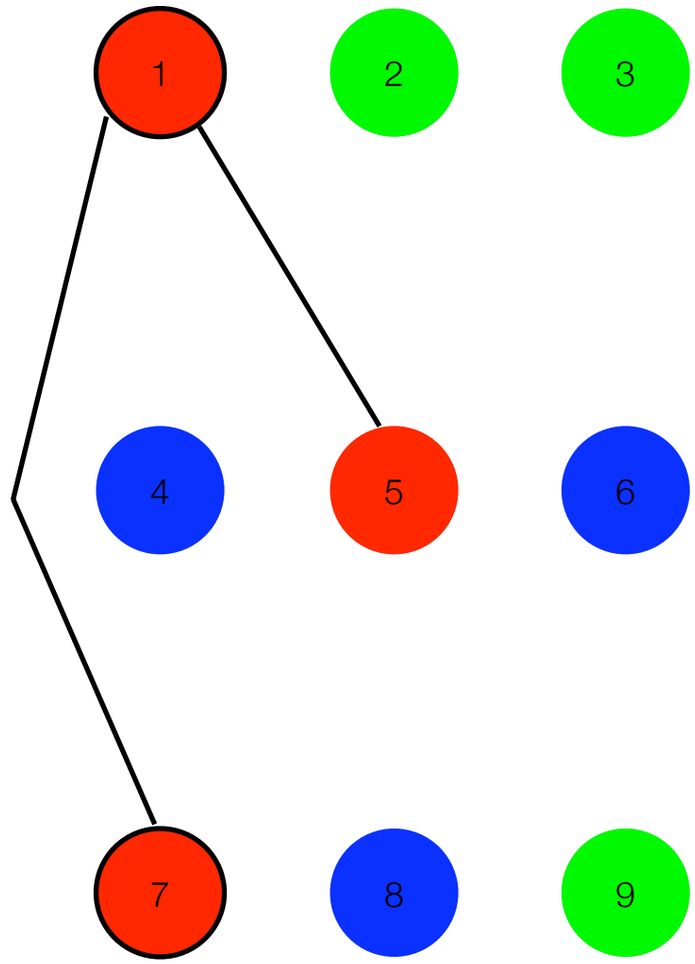
# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node
- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$



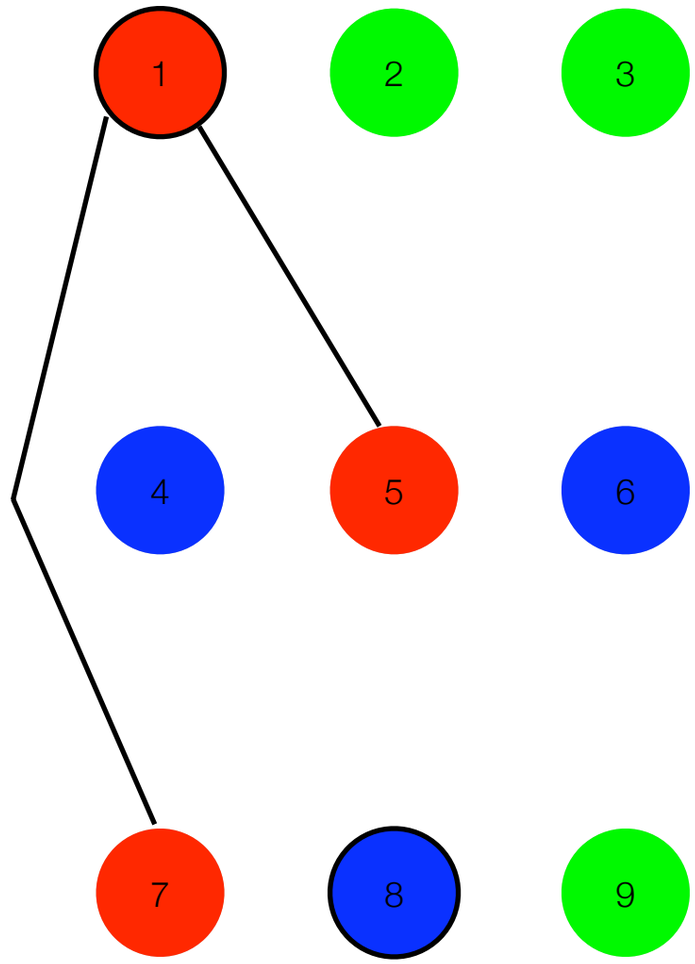
# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node
- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$



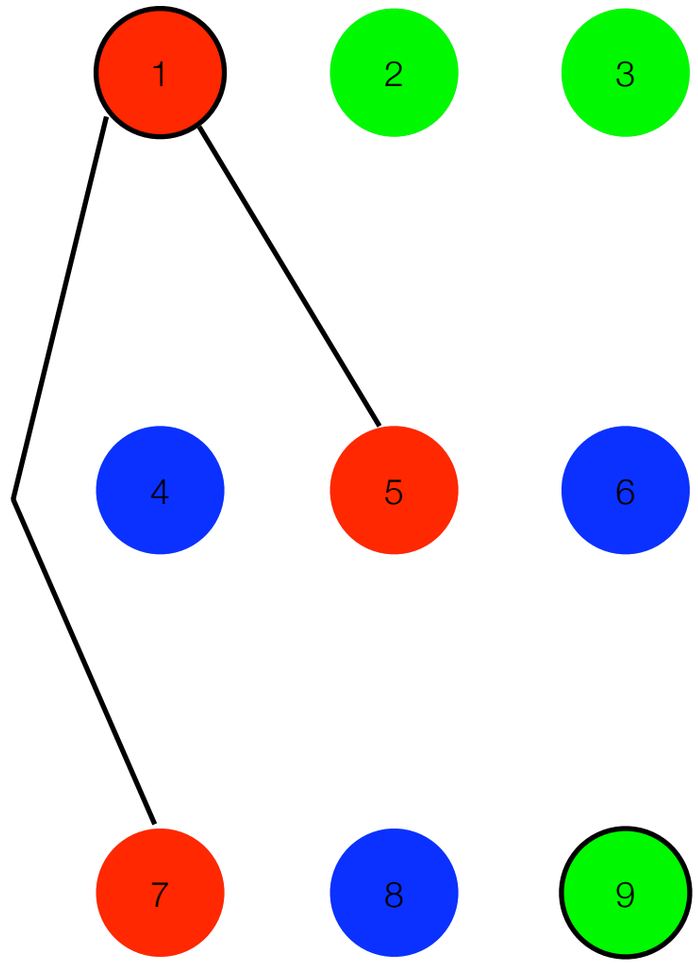
# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node
- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$



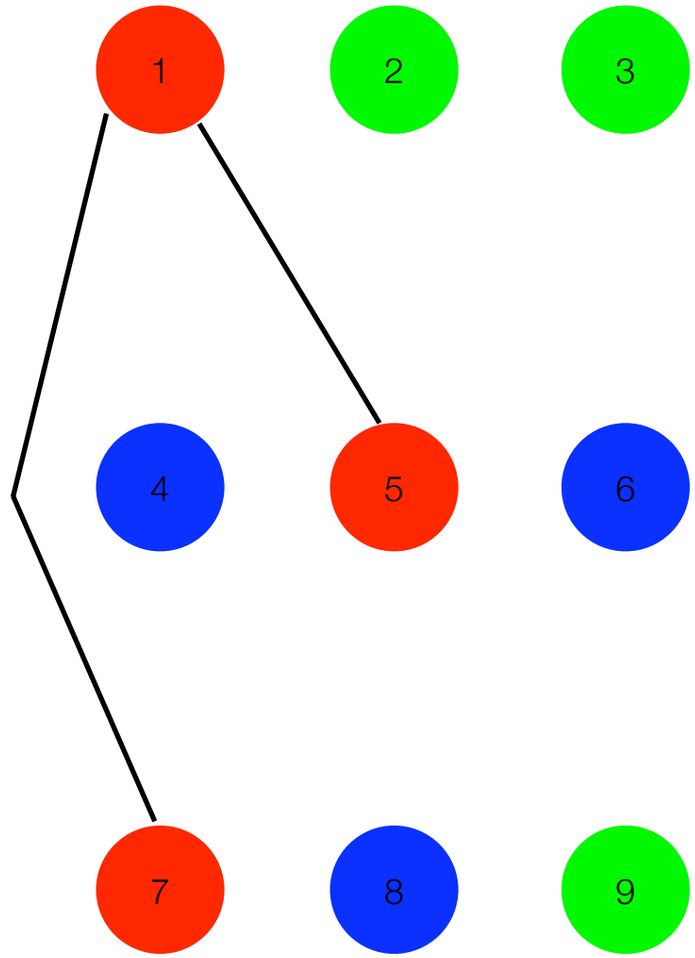
# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node
- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$



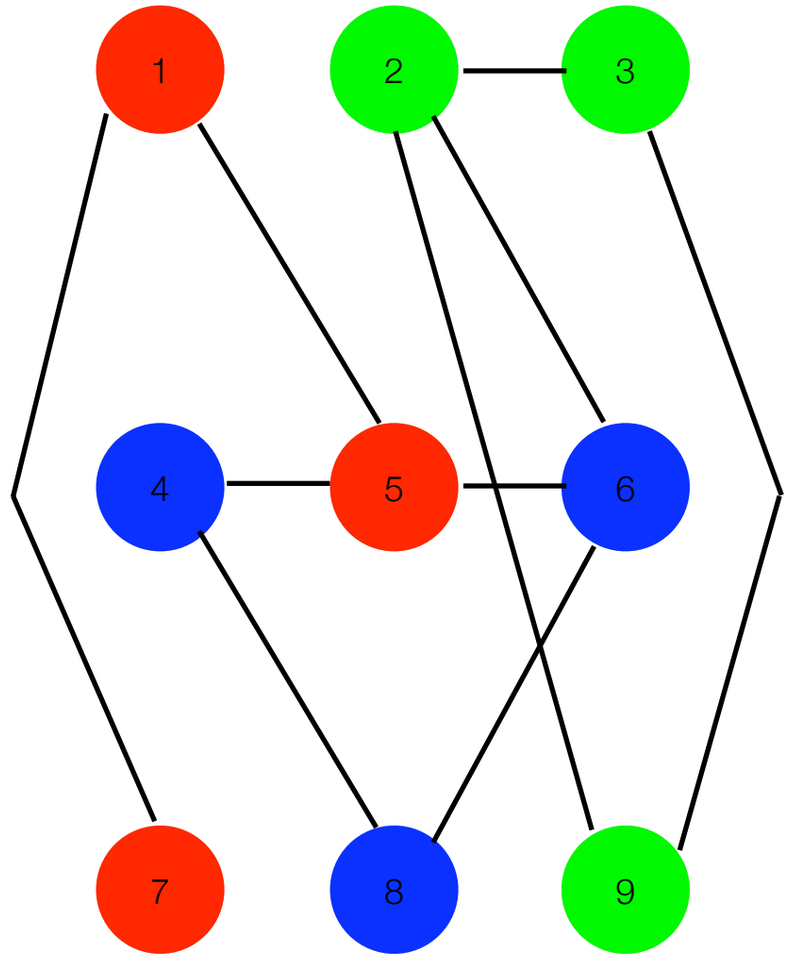
# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node
- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$



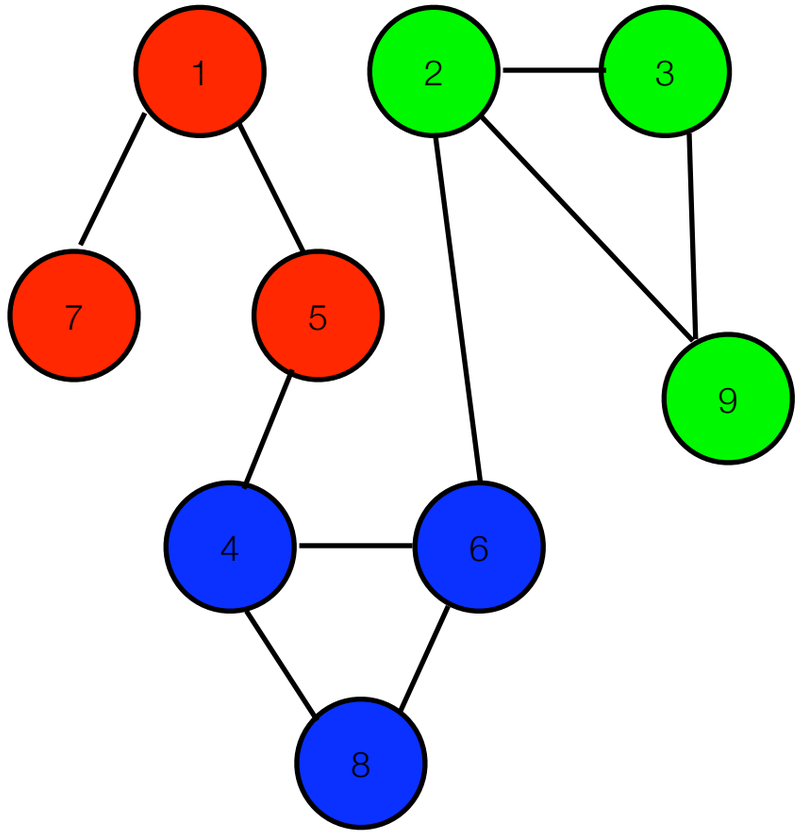
# Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node
- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$



# Generating modular networks

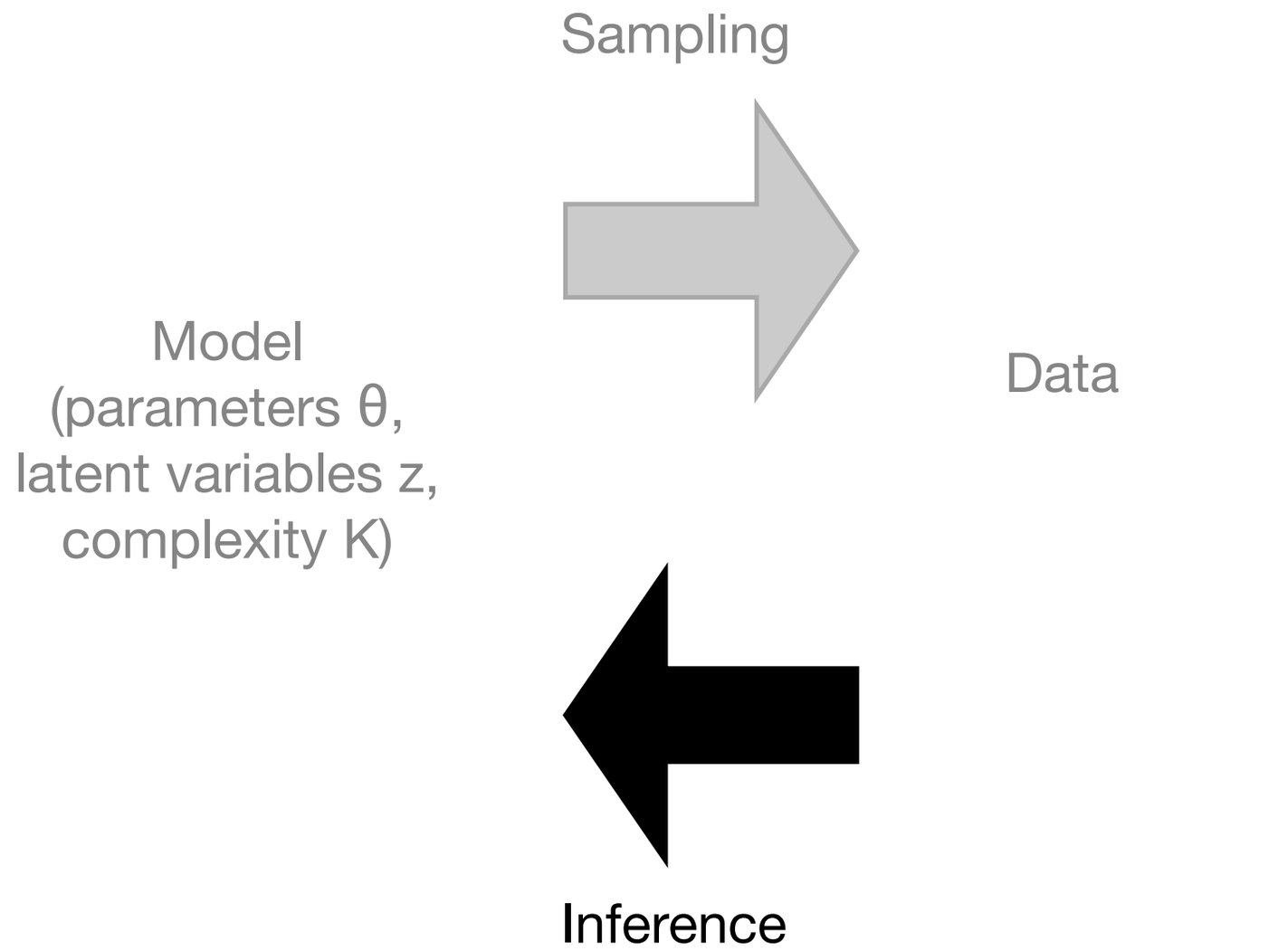
- For each node:
  - **Roll K-sided die** with bias π to determine $z_i$=1,...,K, the (unobserved) module assignment for $i^{th}$ node
- For each pair of nodes (i,j):
  - If $z_i$=$z_j$, **flip "in community" coin** with bias $\theta_c$ to determine edge $A_{ij}$
  - If $z_i$≠$z_j$, **flip "between communities" coin** with bias $\theta_d$ to determine edge $A_{ij}$



# Outline

- Community detection
  - Generative model for modular networks
  - **Variational Bayesian inference**
  - Validation
  - Applications

# Community detection as inference

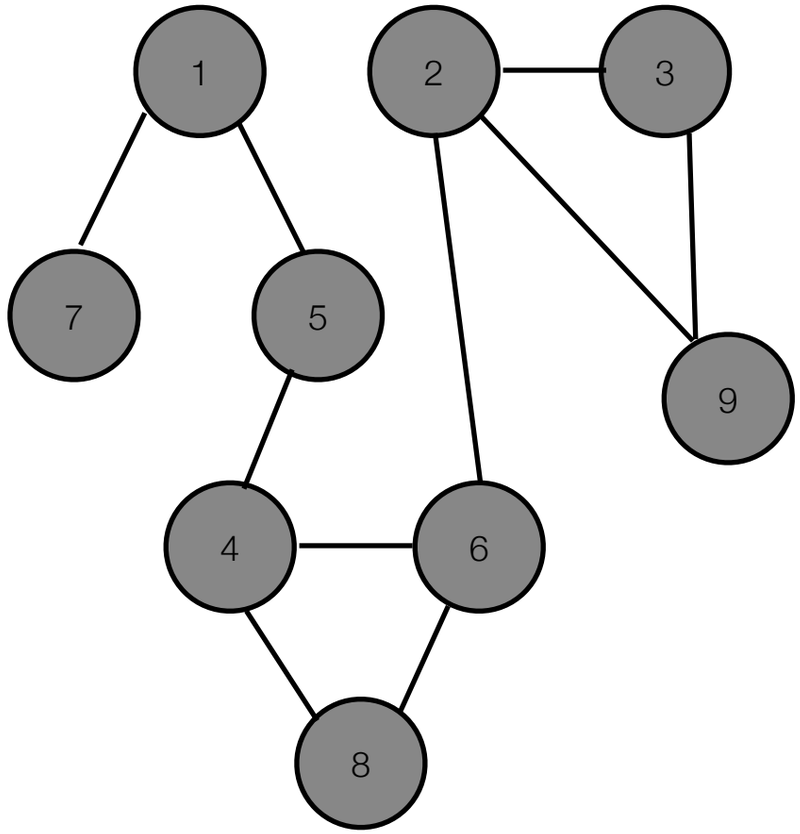


# Community detection as inference

- From observed graph structure, infer distributions over module assignments, model parameters, and model complexity

$$p(\vec{\pi}, \vec{\theta}|\mathbf{A}, K) = \frac{p(\mathbf{A}|\vec{\pi}, \vec{\theta}, K)p(\vec{\pi}, \vec{\theta}|K)}{p(\mathbf{A}|K)}$$

$$p(\vec{z}|\mathbf{A}, K) = \frac{p(\mathbf{A}|\vec{z}, K)p(\vec{z}|K)}{p(\mathbf{A}|K)}$$

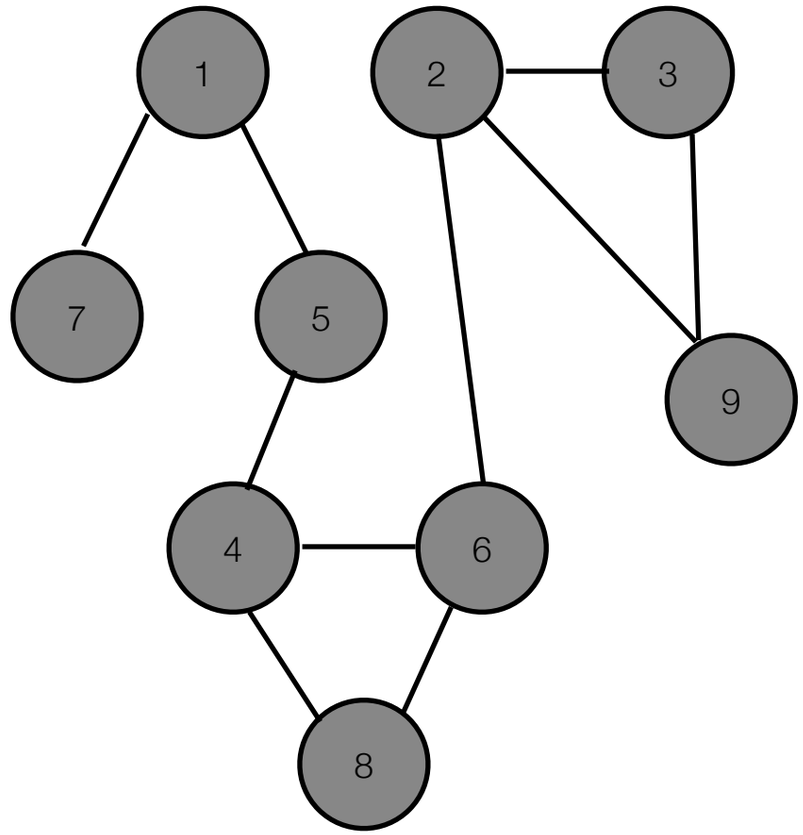


# Community detection as inference

- From observed graph structure, infer distributions over module assignments, model parameters, and model complexity

$$p(\vec{\pi}, \vec{\theta}|\mathbf{A}, K) = \frac{p(\mathbf{A}|\vec{\pi}, \vec{\theta}, K)p(\vec{\pi}, \vec{\theta}|K)}{p(\mathbf{A}|K)}$$

$$p(\vec{z}|\mathbf{A}, K) = \frac{p(\mathbf{A}|\vec{z}, K)p(\vec{z}|K)}{p(\mathbf{A}|K)}$$




Multiplication is easy, but normalization is intractable $O(K^N)$; use mean-field variational approach

# Variational Bayes

- Jensen's inequality (log of expected value bounds expected value of log) for any distribution q

$$
\begin{aligned}
-\ln p(\mathbf{A}|K) &= -\ln \sum_{\vec{z}} \int d\vec{\theta} \int d\vec{\pi}\ p(\mathbf{A}, \vec{z}, \vec{\pi}, \vec{\theta}|K) \\
&= -\ln \sum_{\vec{z}} \int d\vec{\theta} \int d\vec{\pi}\ q(\vec{z}, \vec{\pi}, \vec{\theta}) \frac{p(\mathbf{A}, \vec{z}, \vec{\pi}, \vec{\theta}|K)}{q(\vec{z}, \vec{\pi}, \vec{\theta})} \\
&\leq \underbrace{-\sum_{\vec{z}} \int d\vec{\theta} \int d\vec{\pi}\ q(\vec{z}, \vec{\pi}, \vec{\theta}) \ln \frac{p(\mathbf{A}, \vec{z}, \vec{\pi}, \vec{\theta}|K)}{q(\vec{z}, \vec{\pi}, \vec{\theta})}}_{F\{q;A\}}
\end{aligned}
$$

Variational Bayes (Feynman 1972; MacKay, Jordan, Ghahramani, Jaakola, Saul 1999)

# Variational Bayes for modular networks

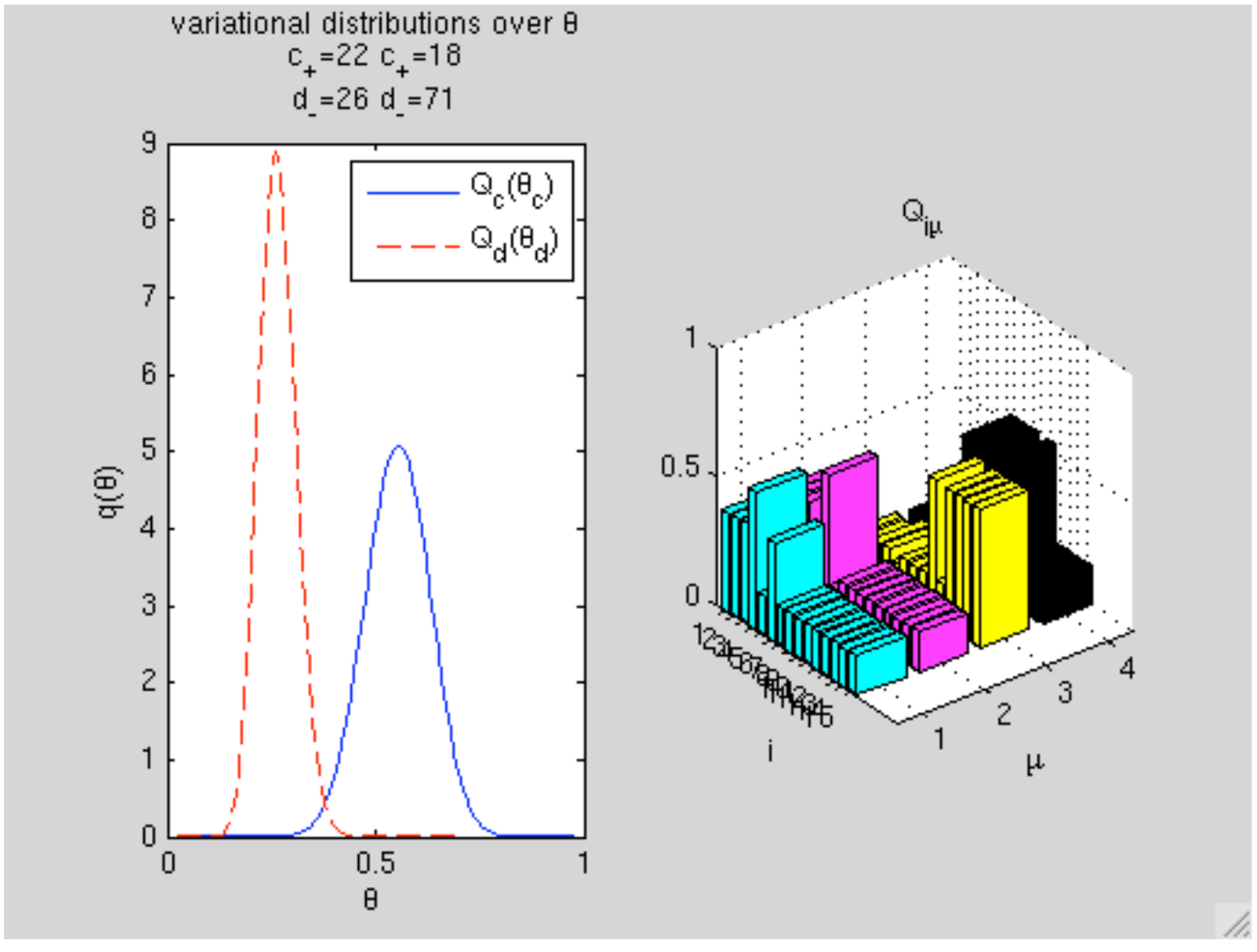
- Iteratively optimize F{q;A} by updating distributions over parameters {π, θ} and latent variables {z}

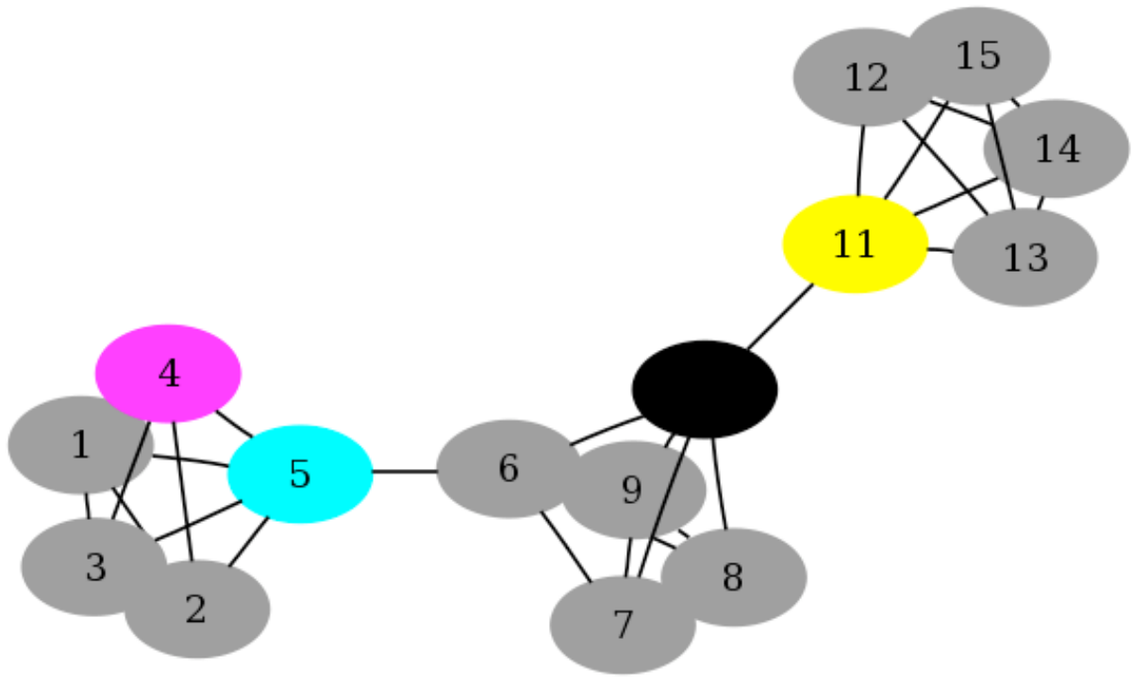
**Algorithm 2** Variational Bayes for maximum evidence inference

1: t=0
2: choose initial distributions $q^{(0)}(Z), q^{(0)}(\Theta)$
3: **repeat**
4: E-step: calculate $\ln q^{(t+1)}(Z) \propto \langle \ln p(\mathcal{D}, Z|\Theta, K) p(\Theta|K) \rangle_{q^{(t)}(\Theta)}$
5: M-step: calculate $\ln q^{(t+1)}(\Theta) \propto \langle \ln p(\mathcal{D}, Z|\Theta, K) p(\Theta|K) \rangle_{q^{(t+1)}(Z)}$
6: $t \leftarrow t + 1$
7: **until** $\mathcal{F}[q^{(t+1)}(Z), q^{(t+1)}(\Theta)] - \mathcal{F}[q^{(t)}(Z), q^{(t)}(\Theta)] \leq \delta$ or $t = T_{max}$

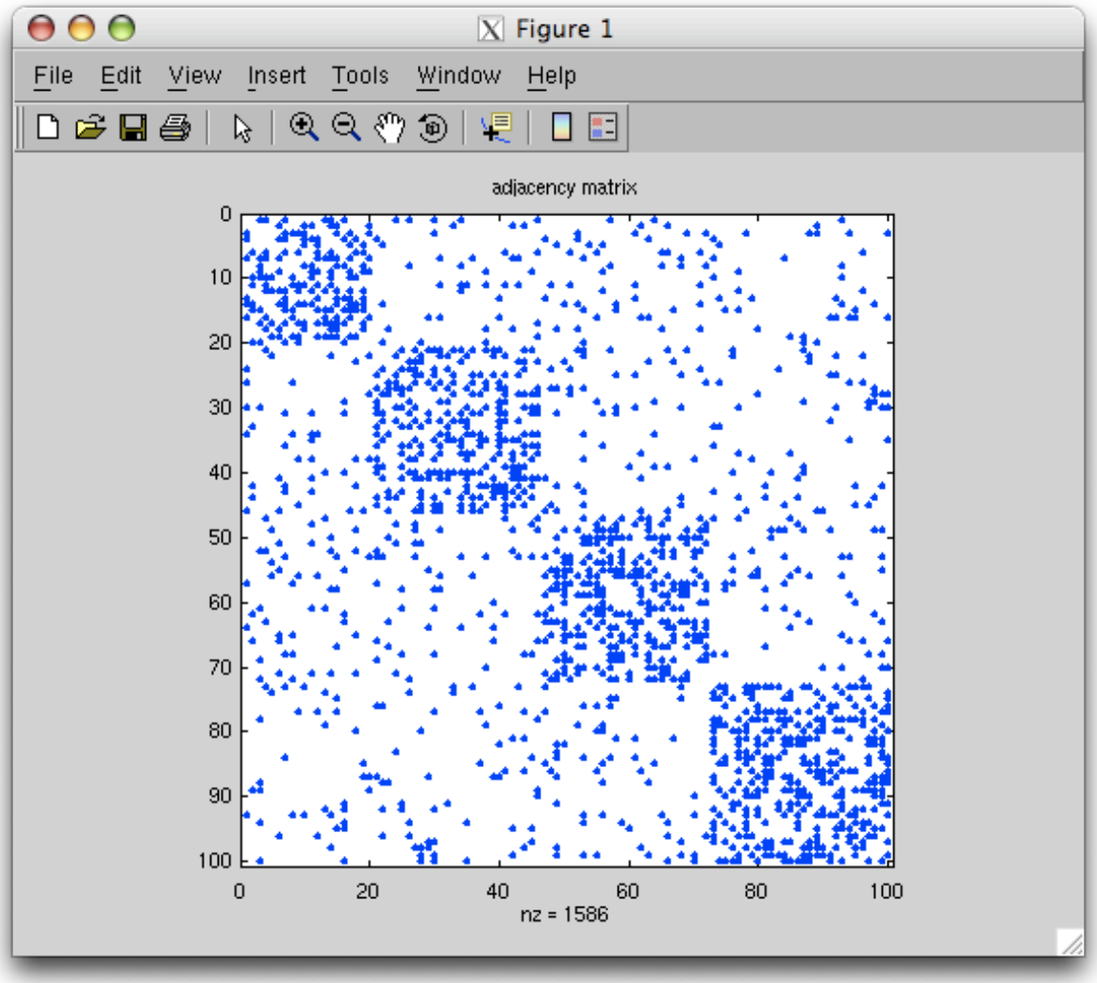
# Validation: complexity control

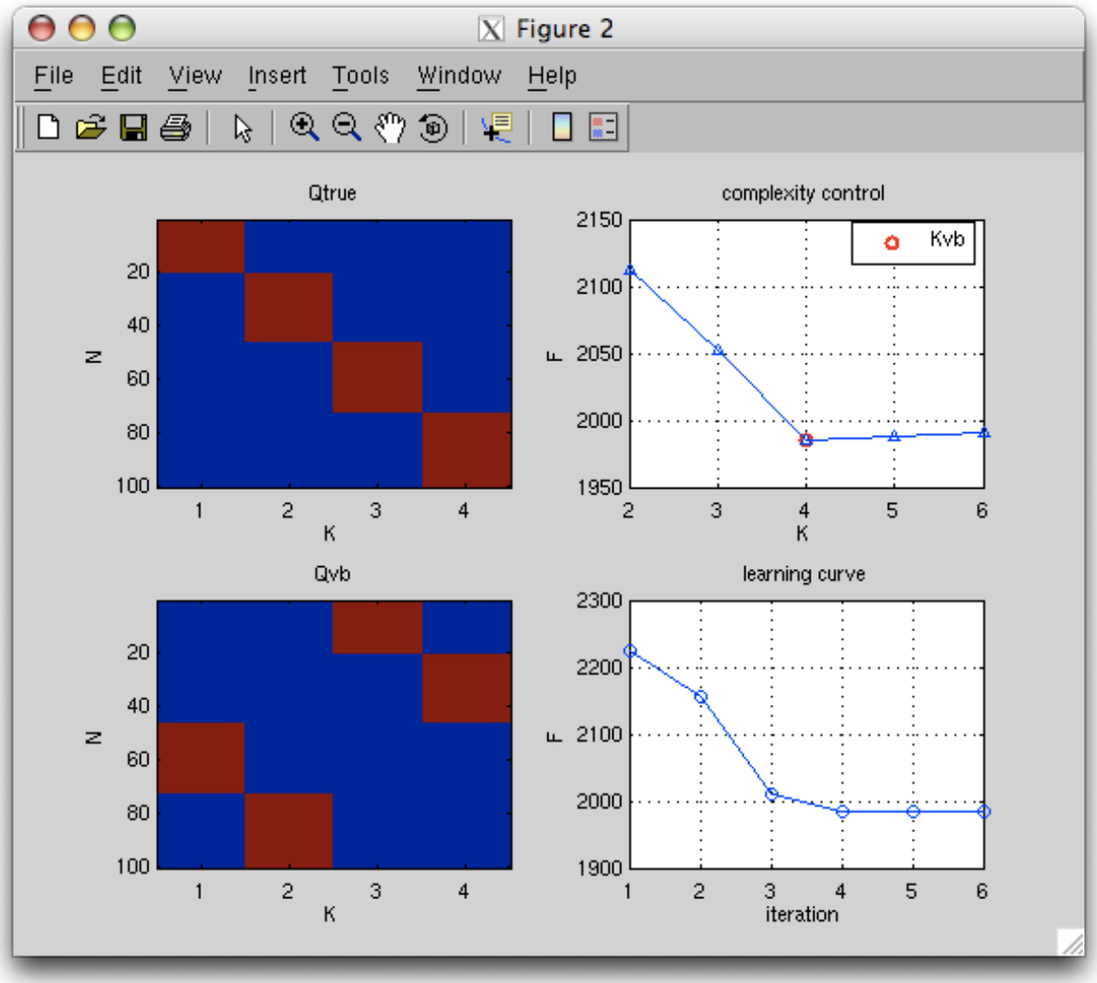
- Automatic complexity control: probability of occupation for extraneous modules goes to zero



# http://vbmod.sourceforge.net



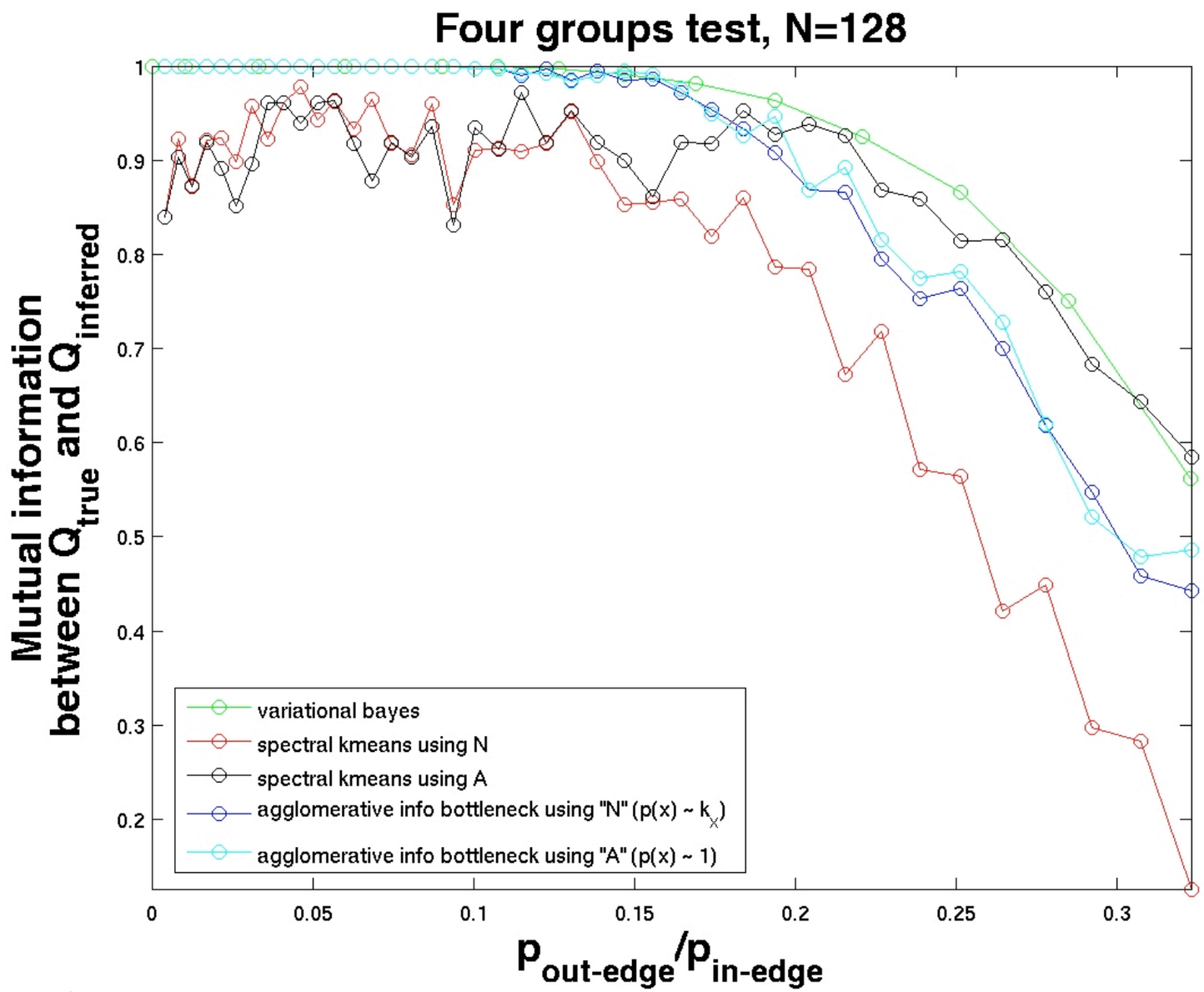

Phys. Rev. Lett. 2008 vol. 100

# Outline

- Community detection
  - Generative model for modular networks
  - Variational Bayesian inference
  - **Validation**
  - Applications

# Validation: “four groups” test

- Mutual information between true and inferred latent variable assignments for N=128 nodes, K=4 modules, average node degree 16




Danon et. al (2005)

# Validation: Complexity control

- Comparison of our method (VB) to alternative method (ICL, based on BIC) for synthetic N=60 node networks and $K_{True}$=3,4,5 modules

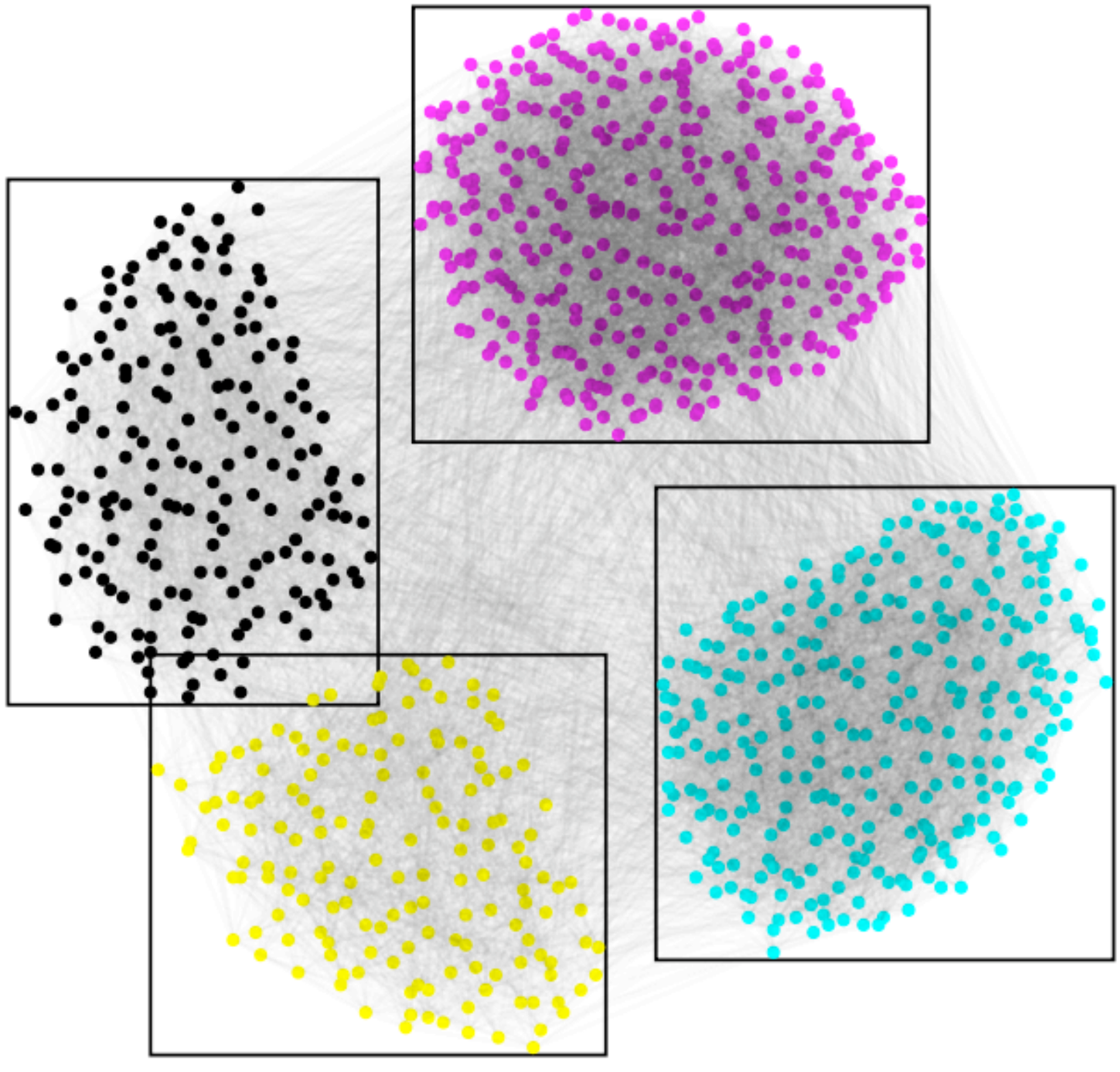
$K_{\mathrm{True}}/K_{\mathrm{VB}}$

| | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 3 | 0 | **99** | 1 | 0 | 0 |
| 4 | 0 | 0 | **90** | 10 | 0 |
| 5 | 0 | 1 | 5 | **91** | 3 |

$K_{\mathrm{True}}/K_{\mathrm{ICL}}$

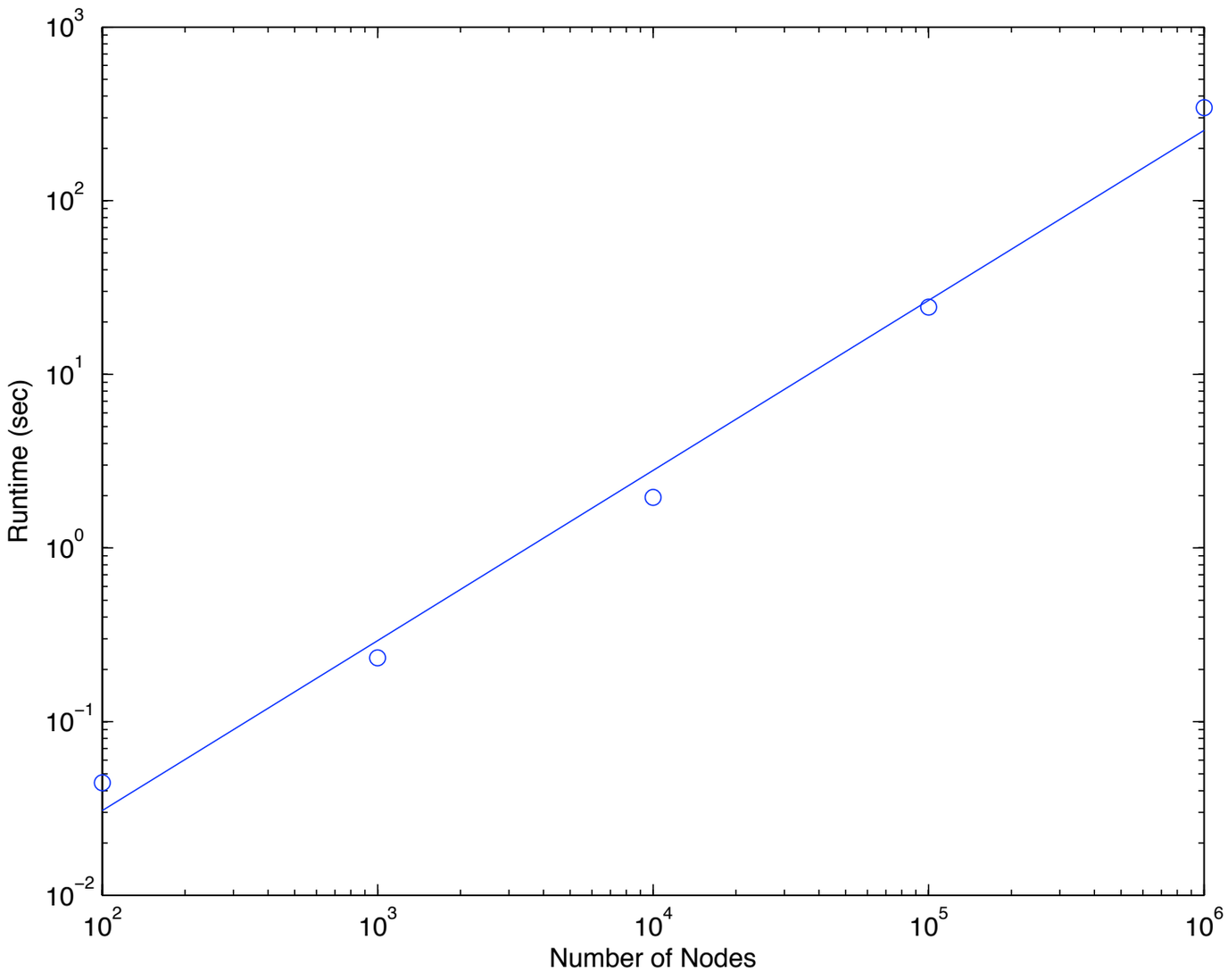
| | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 3 | 0 | **100** | 0 | 0 | 0 |
| 4 | 4 | 25 | **71** | 0 | 0 |
| 5 | 26 | 55 | 17 | **2** | 0 |

# Validation: Large-scale network

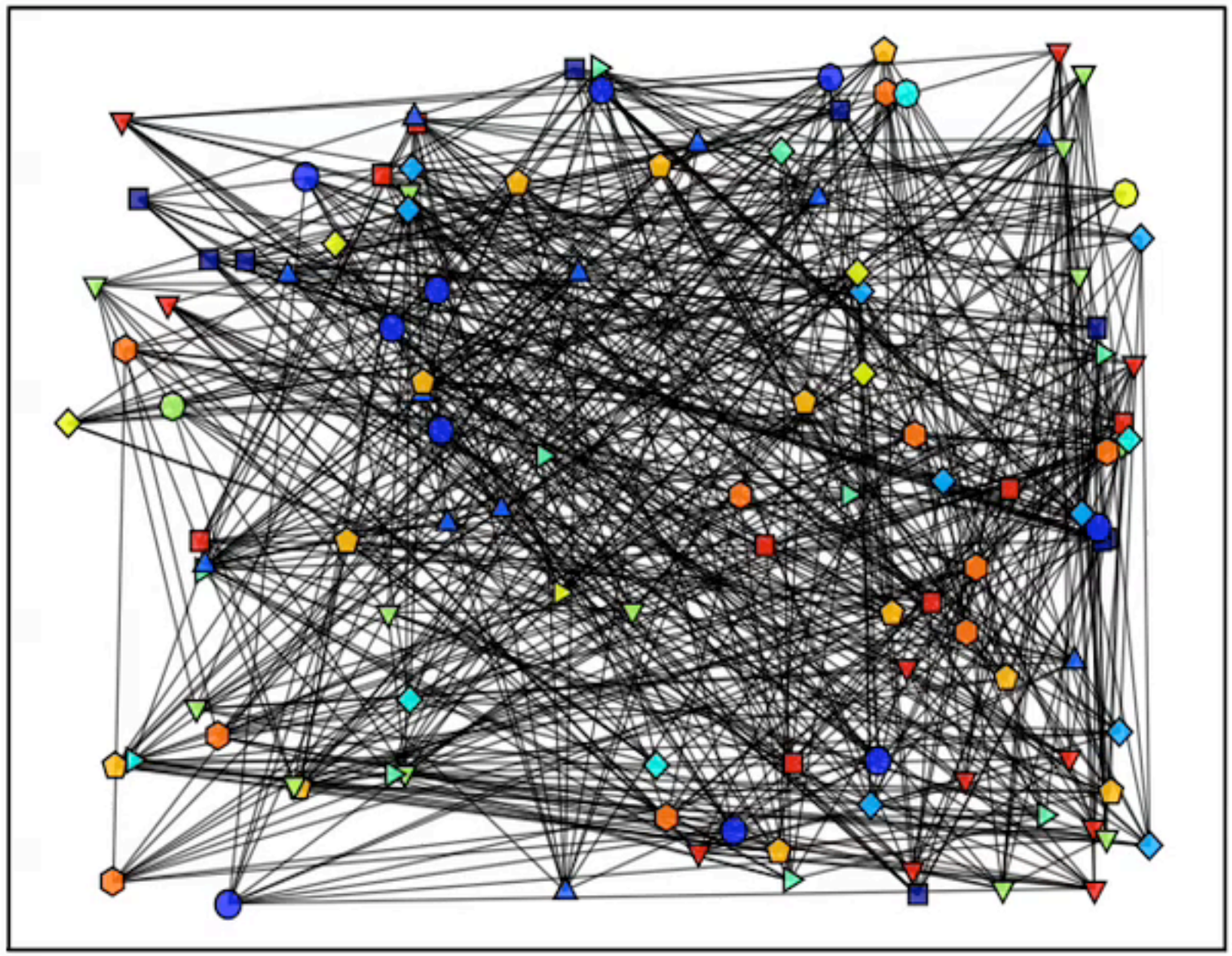


# Validation: Runtime

- O(MK) runtime; ~400 sec for N=$10^6$ nodes, K=4 modules, average node degree 16



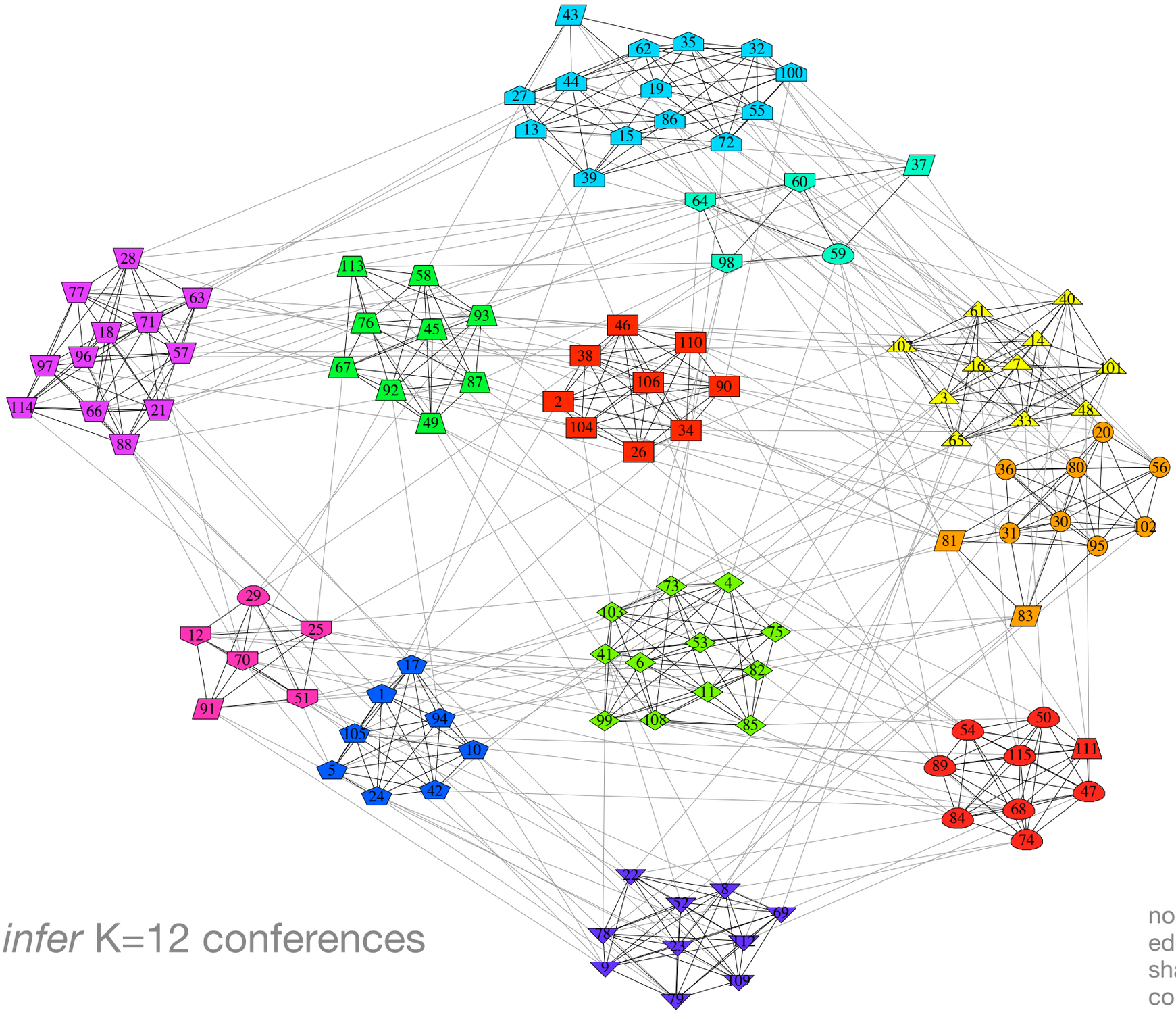
# Validation: NCAA football schedule



- Correctly *infer* K=12 conferences

nodes: teams
edges: games
shape: conference
color: inferred module

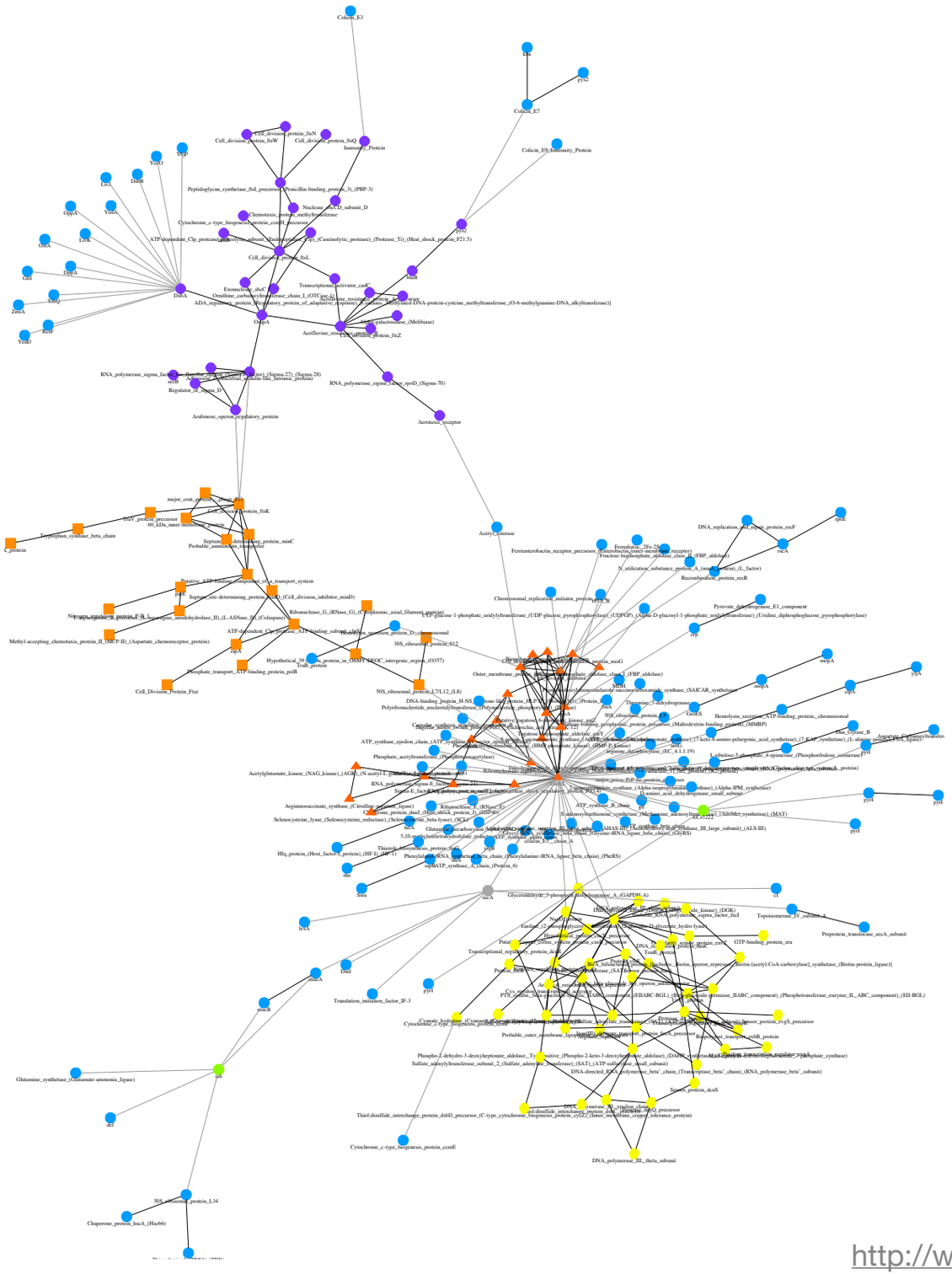
# Validation: NCAA football schedule



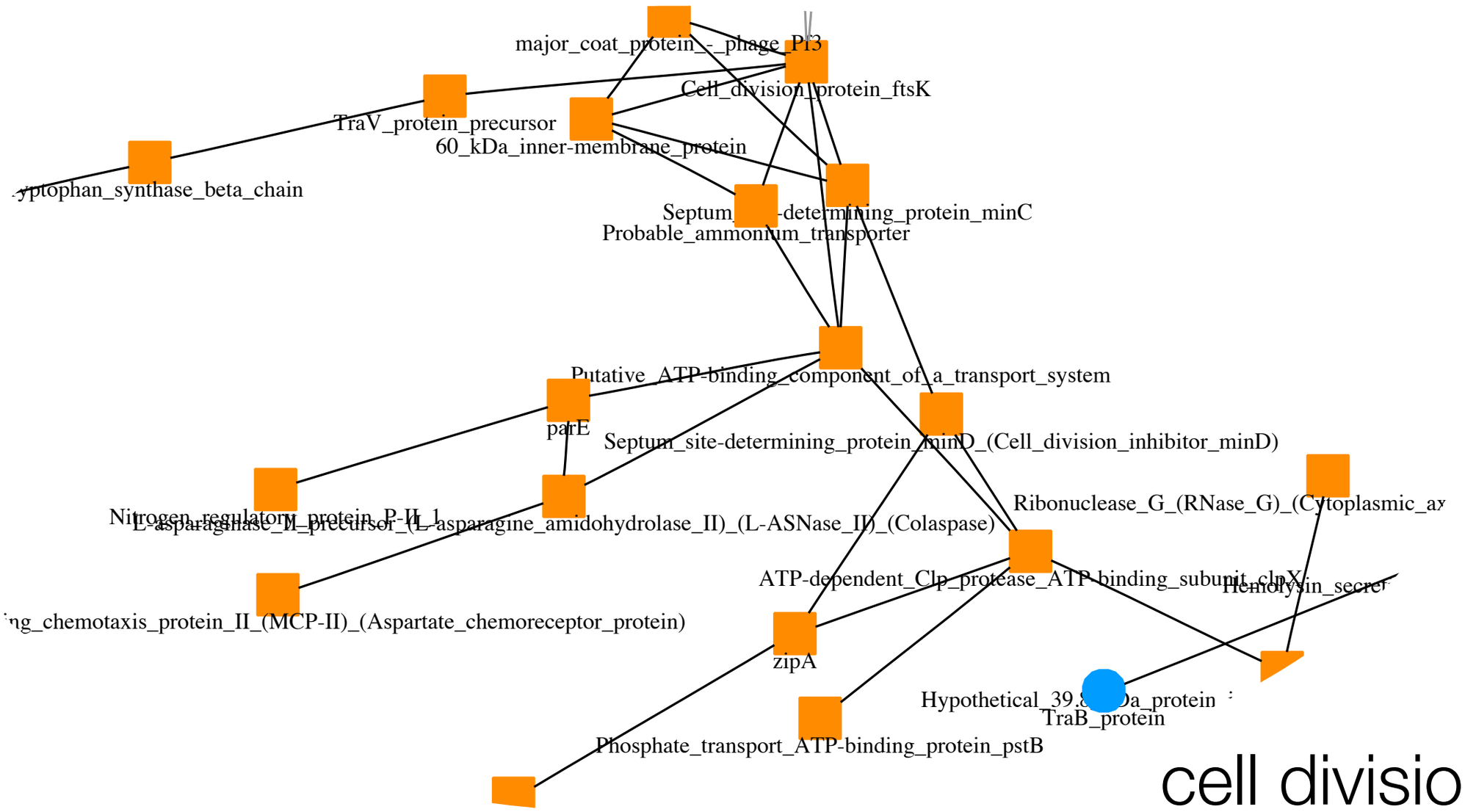

- Correctly *infer* K=12 conferences

nodes: teams
edges: games
shape: conference
color: inferred module

# Application: E. Coli protein-protein network



# Application: E. Coli protein-protein network




http://www.cs.purdue.edu/homes/koyuturk/mawish/

# Outline

- Community detection
  - Generative model for modular networks
  - Variational Bayesian inference
  - Validation
- **Applications**

# Application: APS March Meeting 2008 co-authorship

APS Meetings

Login
Create Account
Meeting Home
APS Home
Meeting Announcement
Invited Speakers
Author Index
Session Index
Epitome
Session Chairs
Word Search
Affiliation Search
Using the Scheduler
BAPS PDFs

**2008 APS March Meeting**

**Monday–Friday, March 10–14, 2008; New Orleans, Louisiana**

## Session P39: Applications of Complex Networks

Show Abstracts

Sponsoring Units: GSNP
Chair: Narayan Menon, University of Massachusetts, Amherst
Morial Convention Center - 231

| | |
|---|---|
| Wednesday, March 12, 2008 8:00AM - 8:12AM | P39.00001: Effects of quenched randomness on predator-prey interactions in a stochastic Lotka-Volterra lattice model<br>Uwe C. Tauber , Ulrich Dobramysl<br>Preview Abstract |
| Wednesday, March 12, 2008 8:12AM - 8:24AM | P39.00002: Dynamical Clustering in Reaction-Dispersal Processes on Complex Networks<br>Vincent David , Marc Timme , Theo Geisel , Dirk Brockmann<br>Preview Abstract |
| Wednesday, March 12, 2008 8:24AM - 8:36AM | P39.00003: Fluctuations and Food-web Structures in Individual-based Models of Biological Coevolution<br>Per Arne Rikvold , Volkan Sevim<br>Preview Abstract |
| Wednesday, March 12, 2008 8:36AM - 8:48AM | P39.00004: Metabolic disease network and its implication for disease comorbidity<br>Deok-Sun Lee , Zoltan Oltvai , Nicholas Christakis , Albert-Laszlo Barabasi<br>Preview Abstract |
| Wednesday, March 12, 2008 8:48AM - 9:00AM | P39.00005: The Human Phenotypic Disease Network<br>Cesar Hidalgo , Nicholas Blumm , Albert-Laszlo Barabasi , Nicholas Christakis<br>Preview Abstract |

http://meetings.aps.org/Meeting/MAR08

# Conclusions

- Phrased community detection as Bayesian inference
- Variational methods give interpretable, fast, and scalable method for (approximate) inference
- Empirical validation of model selection and comparison for constrained and full stochastic block models
- Some correlation between topological communities and node attributes, but unclear without additional information

# Acknowledgements


- **Wiggins Lab**
  - Andrew Mugler
  - Anil Raj
  - Chris Wiggins
  - Jon Bronson
- **Yahoo! Research**
  - Duncan Watts
- **Useful discussions**
  - Edo Airoldi (Princeton)
  - Joel Bader (John Hopkins)
  - David Blei (Princeton)
  - Aaron Clauset (SFI)
  - Jonathan Goodman (NYU)
  - Matt Hastings (LANL)