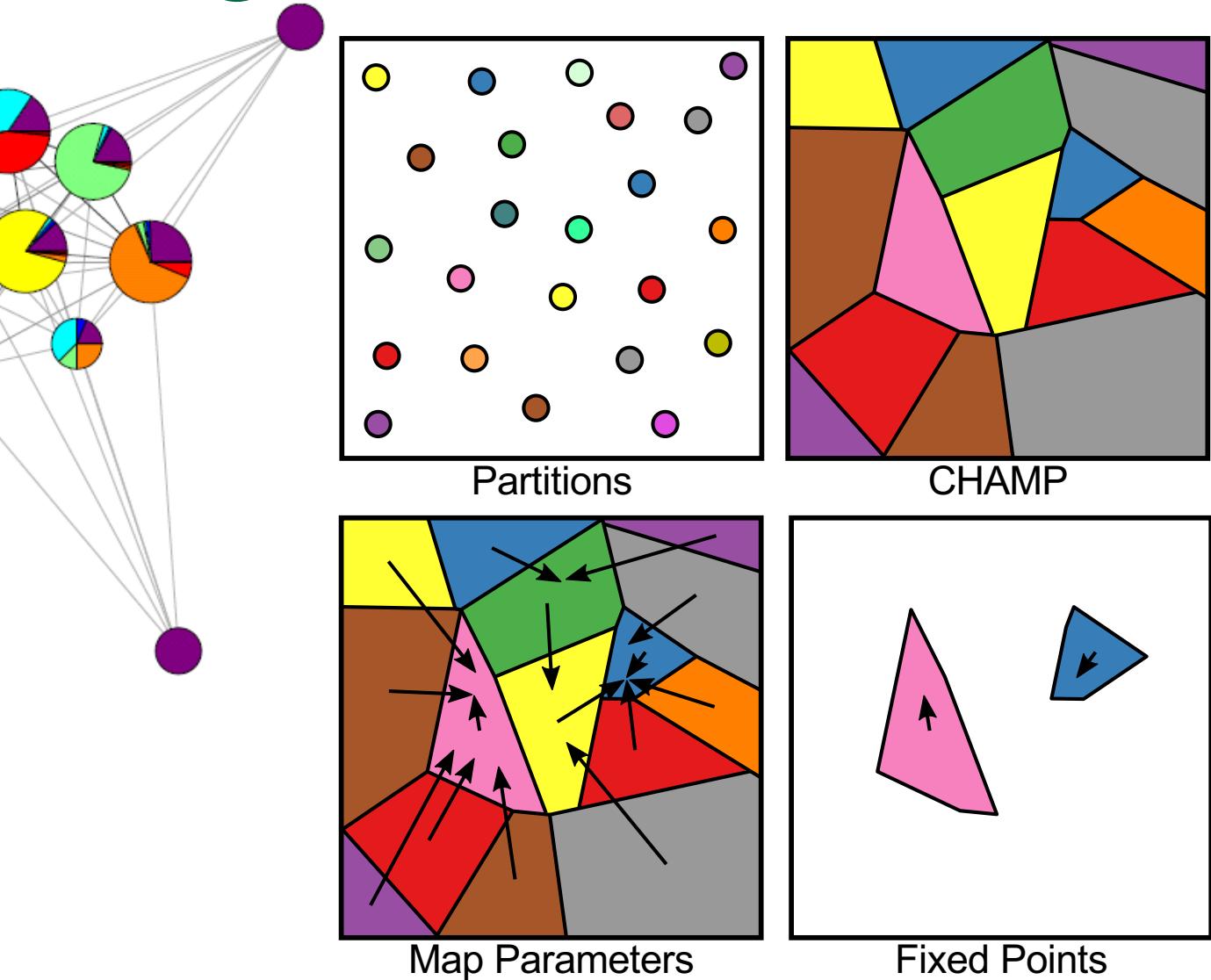
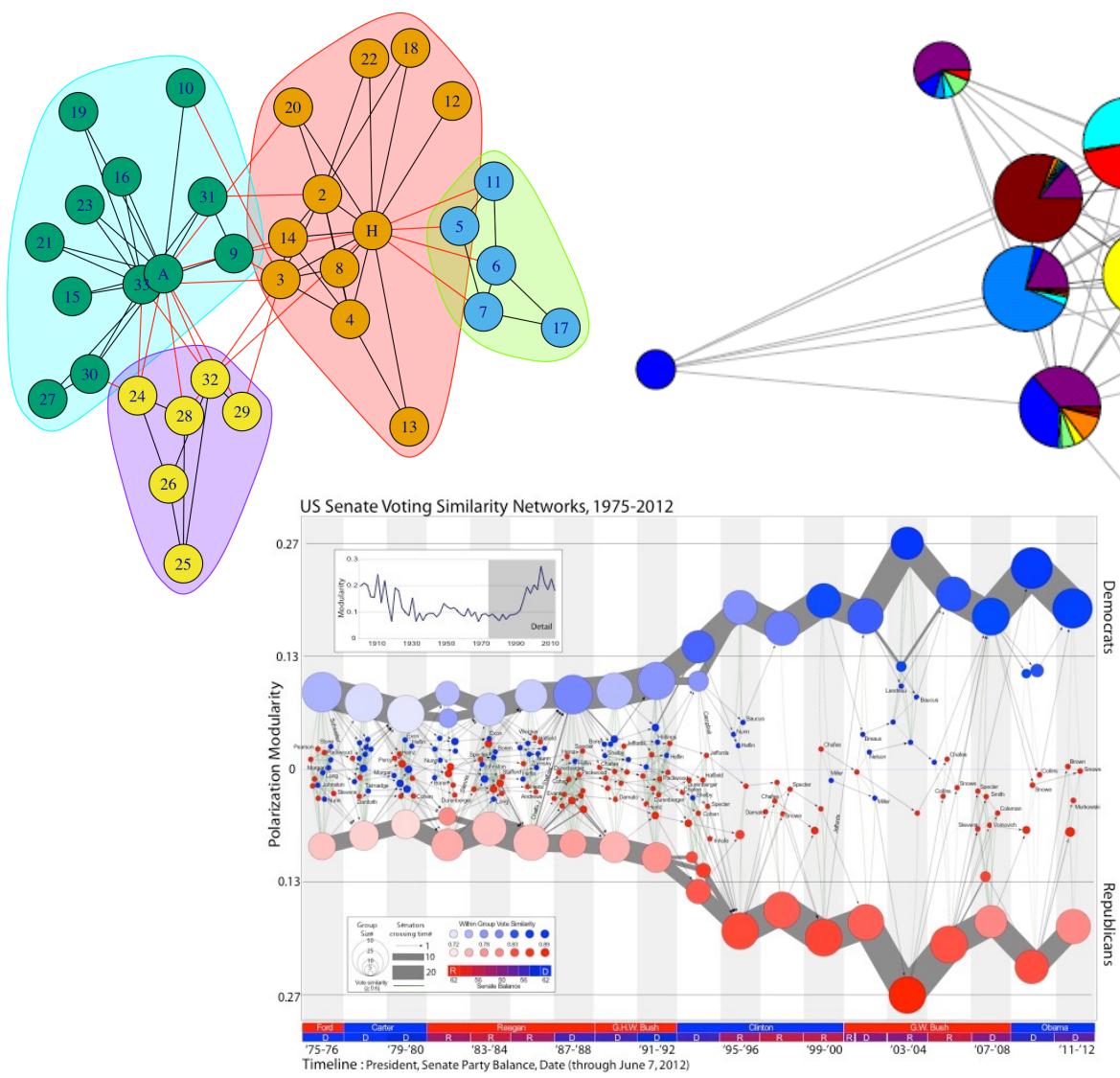


Community Detection in R at SNH 2023

Peter J. Mucha, Dartmouth College

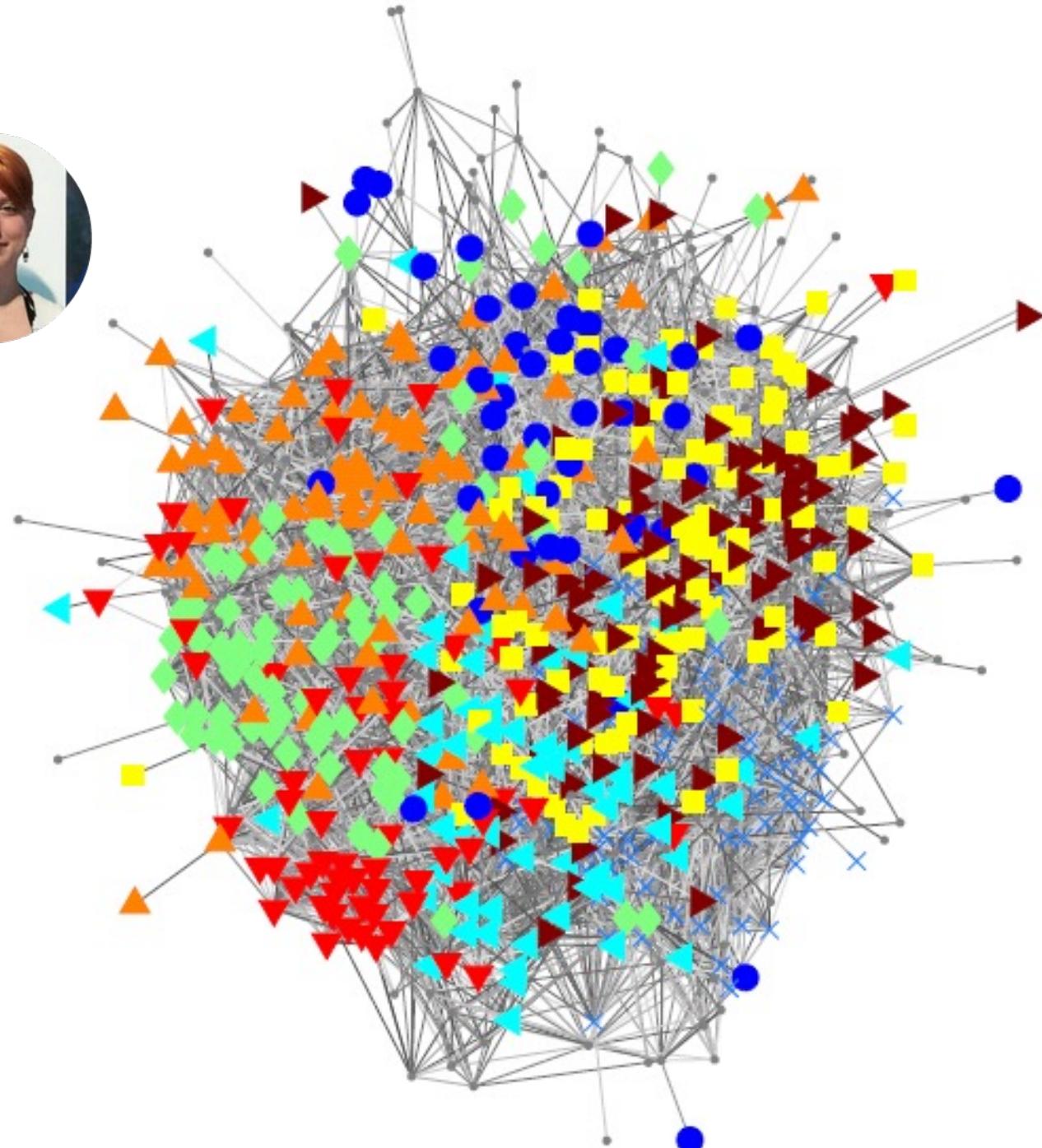


Facebook

Traud *et al.*, “Comparing community structure to characteristics in online collegiate social networks” (2011)

Traud *et al.*, “Social structure of Facebook networks” (2012)

Caltech 2005:
Colors indicate residential
“House” affiliations

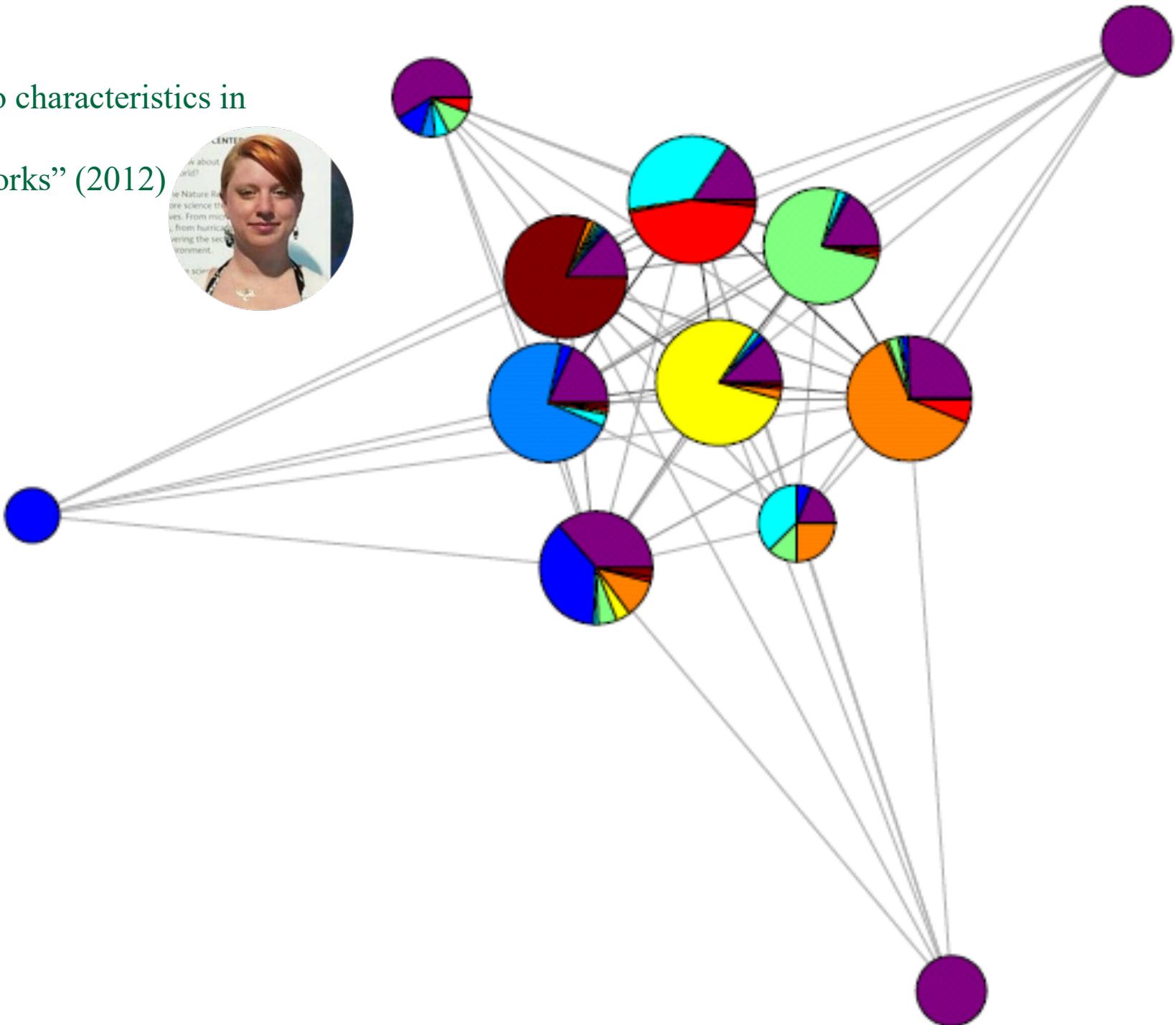


Facebook

Traud *et al.*, “Comparing community structure to characteristics in online collegiate social networks” (2011)

Traud *et al.*, “Social structure of Facebook networks” (2012)

Caltech 2005:
Colors indicate residential
“House” affiliations



Facebook

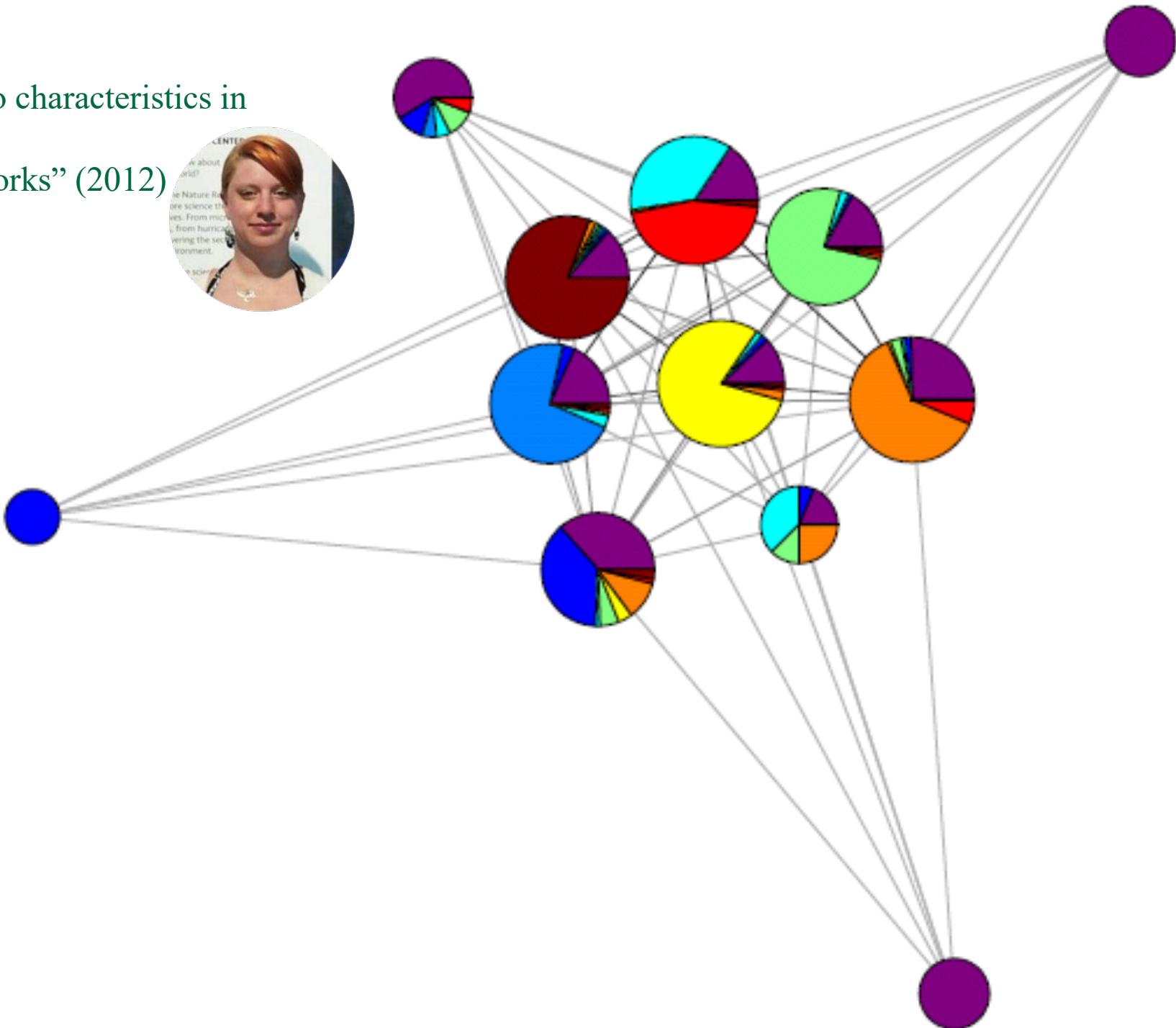
Traud *et al.*, “Comparing community structure to characteristics in online collegiate social networks” (2011)

Traud *et al.*, “Social structure of Facebook networks” (2012)

Caltech 2005:

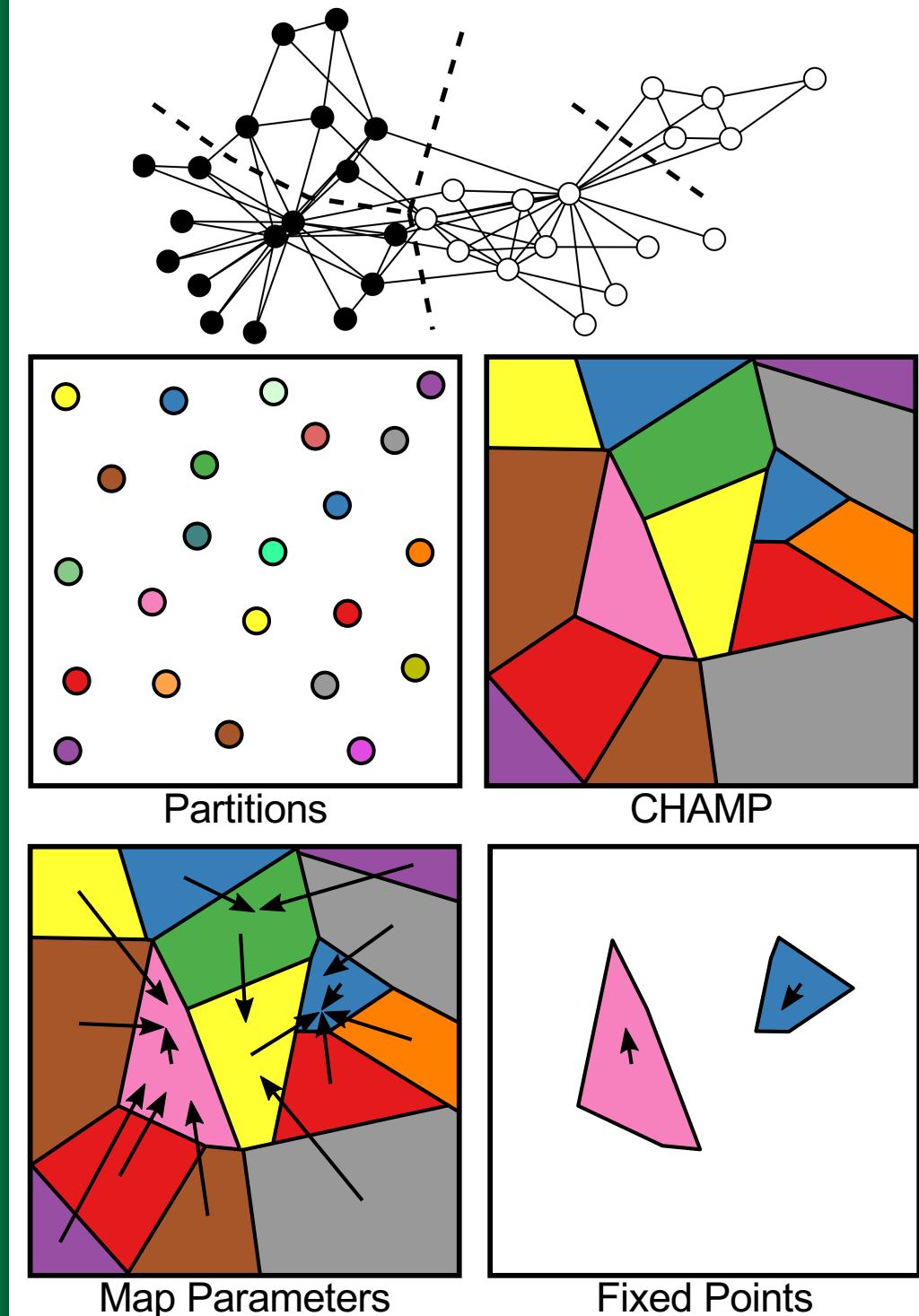
Colors indicate residential
“House” affiliations

Purple = Not provided



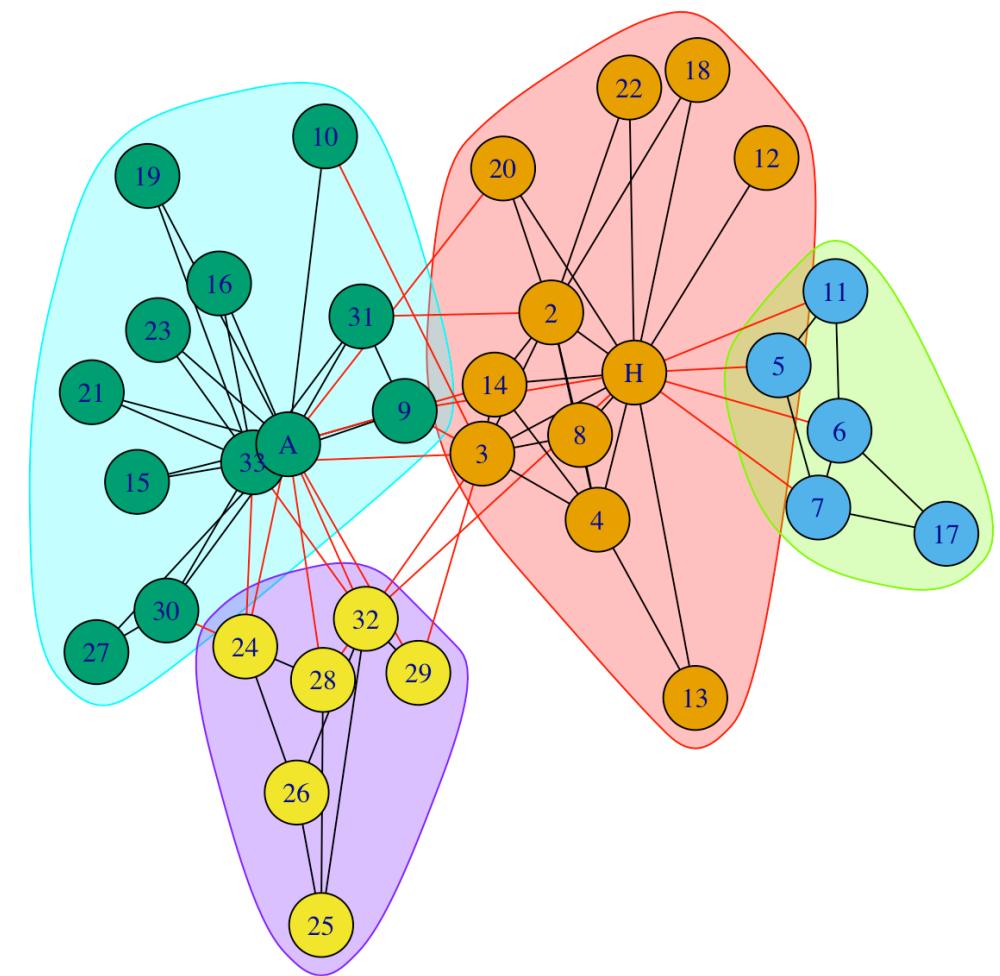
Takeaway Messages

1. CommunitiesSNH2023.Rmd file designed to highlight upcoming IDEANet capabilities
2. Community detection is “just” unsupervised clustering of nodes in the network:
 - NFL: No single “right” way to cluster (different tools, knobs)
 - “Meaning” of identified groups depends on your application
3. CHAMP and Iterative Parameter Maps in R:
 - Multilayer modularity capabilities “coming soon”
github.com/wweir827/CHAMP
github.com/ragibson/ModularityPruning



This is the typical user:

```
> library(igraphdata)
> data(karate)
> partition1 <- cluster_louvain(karate)
> partition1
IGRAPH clustering multi level, groups: 4, mod: 0.44
+ groups:
$`1`
[1] "Mr Hi"      "Actor 2"     "Actor 3"     "Actor 4"     "Acto
[8] "Actor 14"    "Actor 18"    "Actor 20"    "Actor 22"
$`2`
[1] "Actor 5"     "Actor 6"     "Actor 7"     "Actor 11"    "Actor
$`3`
[1] "Actor 9"     "Actor 10"    "Actor 15"    "Actor 16"    "Acto
[8] "Actor 27"    "Actor 30"    "Actor 31"    "Actor 33"    "John
+ ... omitted several groups/vertices
> plot(partition1,karate)
>
```

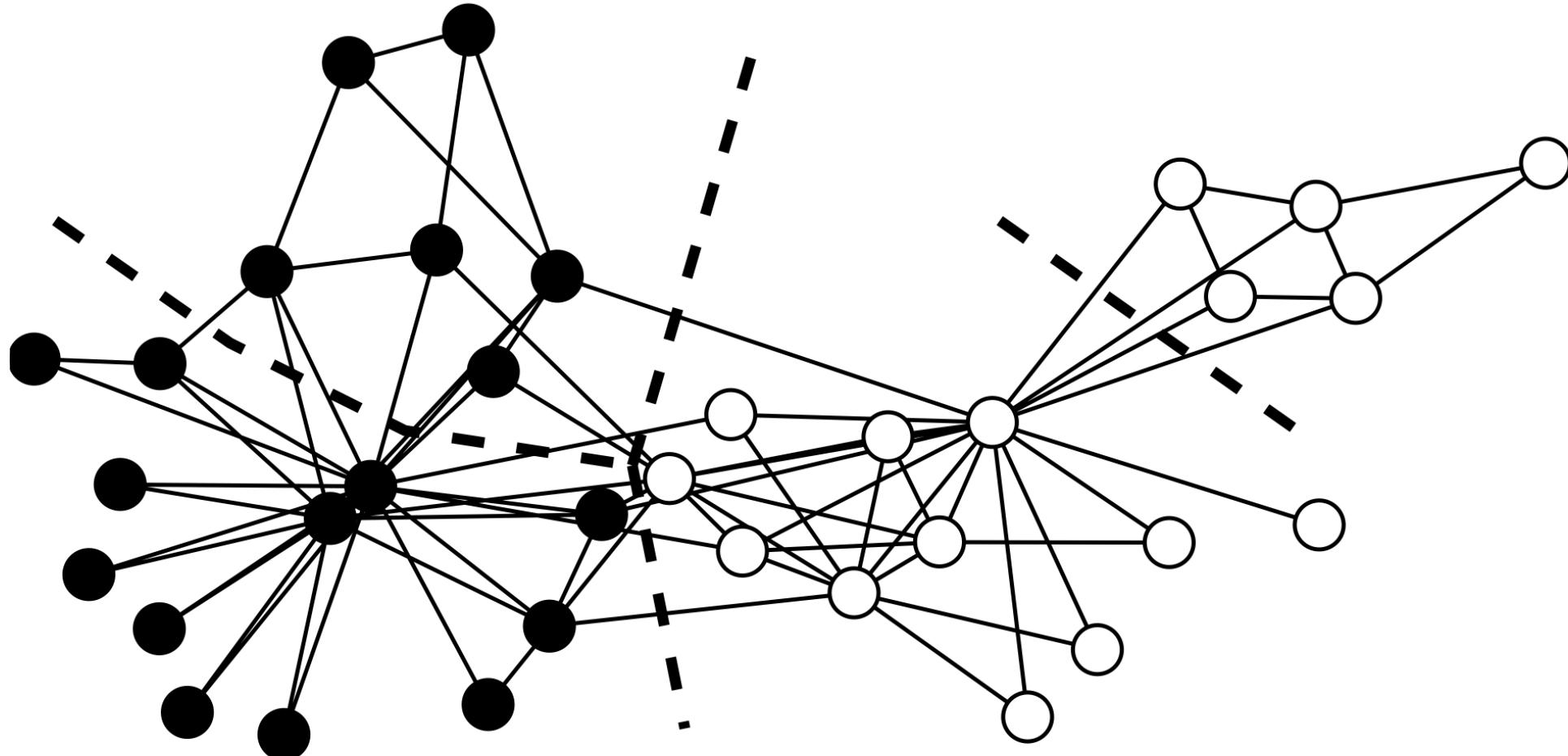


Network Scientists with Karate Trophies

<http://networkkarate.tumblr.com/>

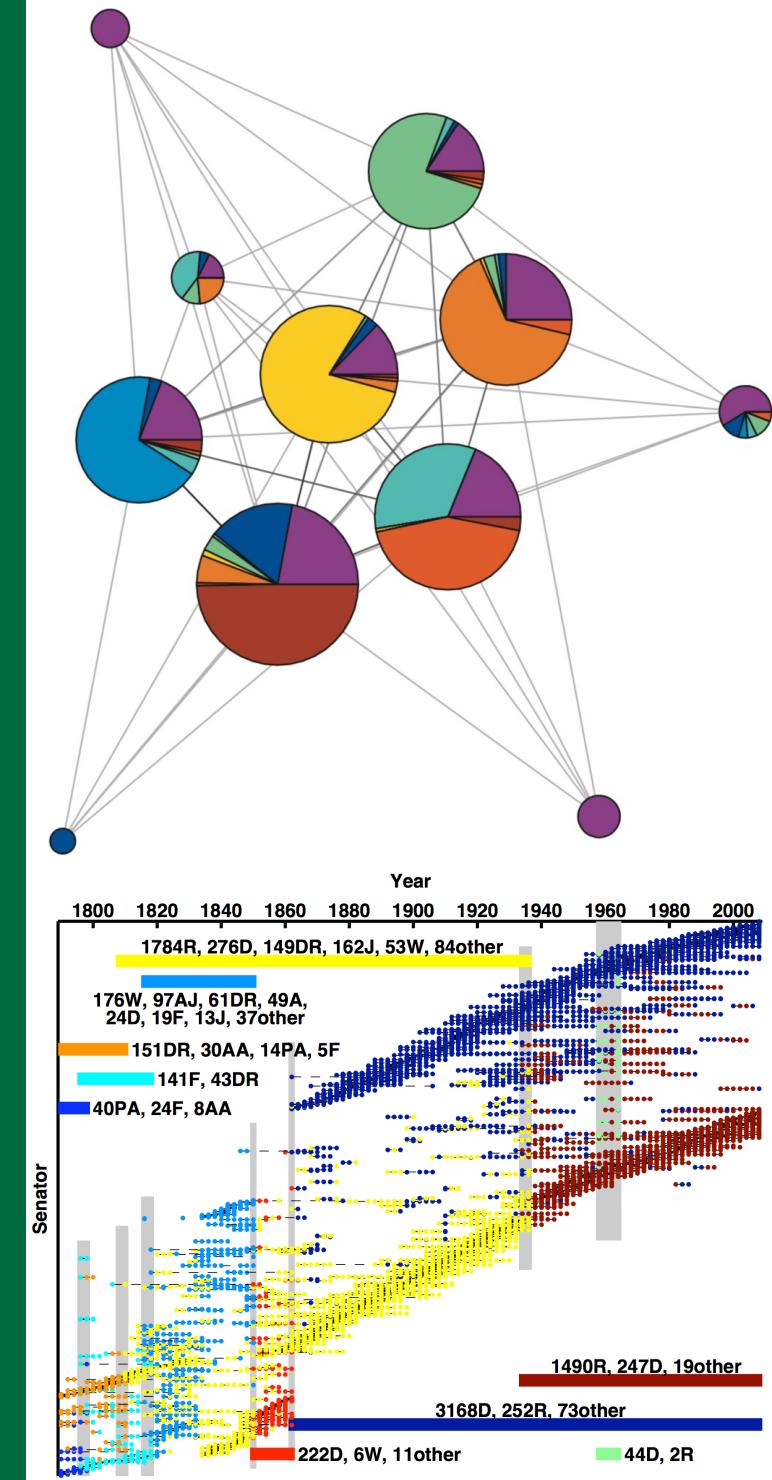


Zachary Karate Club



This partition optimizes **modularity** (at default resolution) which measures the number of intra-community ties (relative to a random model)
“If your method doesn’t work on this network, then go home.”

- Modularity is problematic
- Modularity maximization remains most used method of community detection
- Frequently misused
 - Heuristics → Stochastic
 - Pick resolution γ & coupling ω
- *If you're going to use modularity, we want to help you use it right*



Modularity with resolution parameter γ

(Newman & Girvan, 2004; Reichardt & Bornholdt, 2006)

$$Q_\sigma(\gamma) = \sum_{i,j} (A_{ij} - \gamma P_{ij}) \delta(c_{i\sigma}, c_{j\sigma})$$

“Modularity matrix”

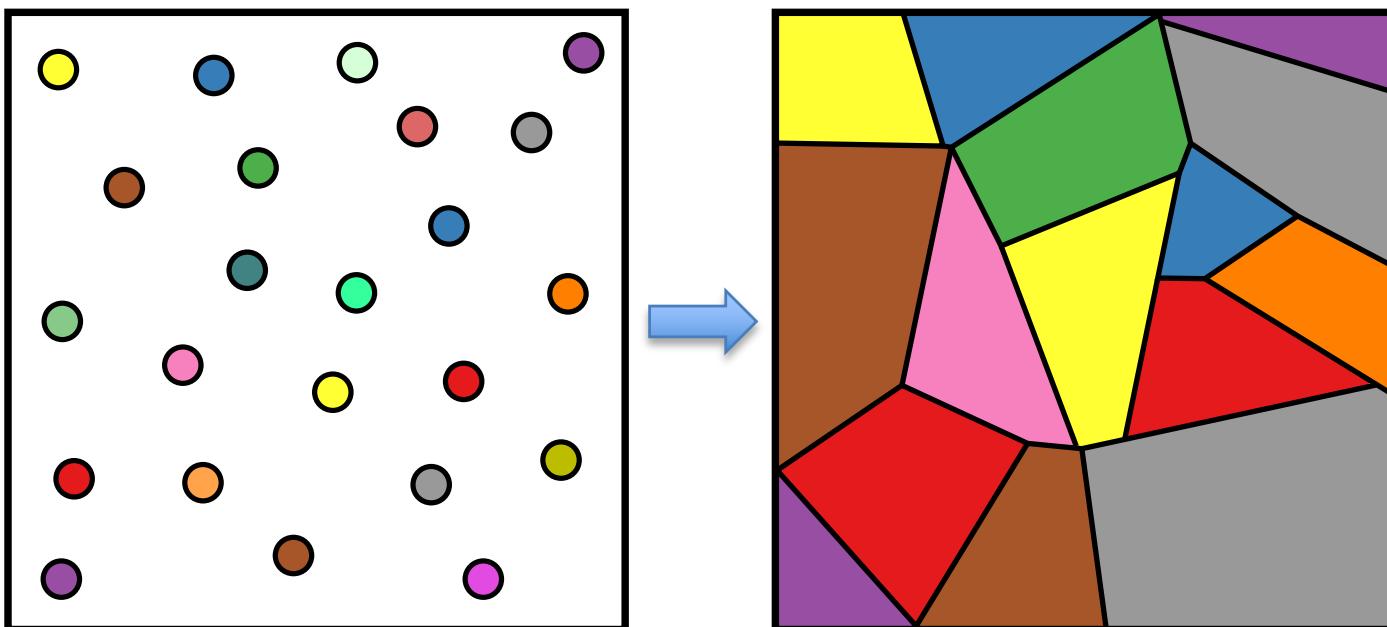
Adjacency data:
Edge/weight from i to j

“Null model” expected edge weight,
typically product of node degrees (undirected): $k_i k_j / (2m)$

Indicator on nodes i & j in
same community

Optimization: Find partition σ of nodes in communities to maximize Q
Resolution parameter γ indirectly controls number/sizes of communities

Convex Hull of Admissible Modularity Partitions



github.com/wweir827/champ



Article

Post-Processing Partitions to Identify Domains of Modularity Optimization

William H. Weir ^{1,2,*}, Scott Emmons ¹, Ryan Gibson ¹, Dane Taylor ¹ and Peter J. Mucha ^{1,2}

A screenshot of a GitHub repository page for "CHAMP (Convex Hull of Admissible Modularity Partitions)". The page includes a brief description, three network visualization examples (College Football, Human Reactome, and Caltech Facebook), and a section detailing the package's functionality and documentation.

The CHAMP python package provides two levels of functionality:

- Identifying the subset of partitions from a group of partitions (regardless of how they were discovered) with optimal modularity. See [Running](#).
- Parallelized implementation of modularity based community detection method, [louvain](#) with efficient filtering (*ala* CHAMP), management, and storage of the generated partitions. See [Louvain Parallel Extension](#).

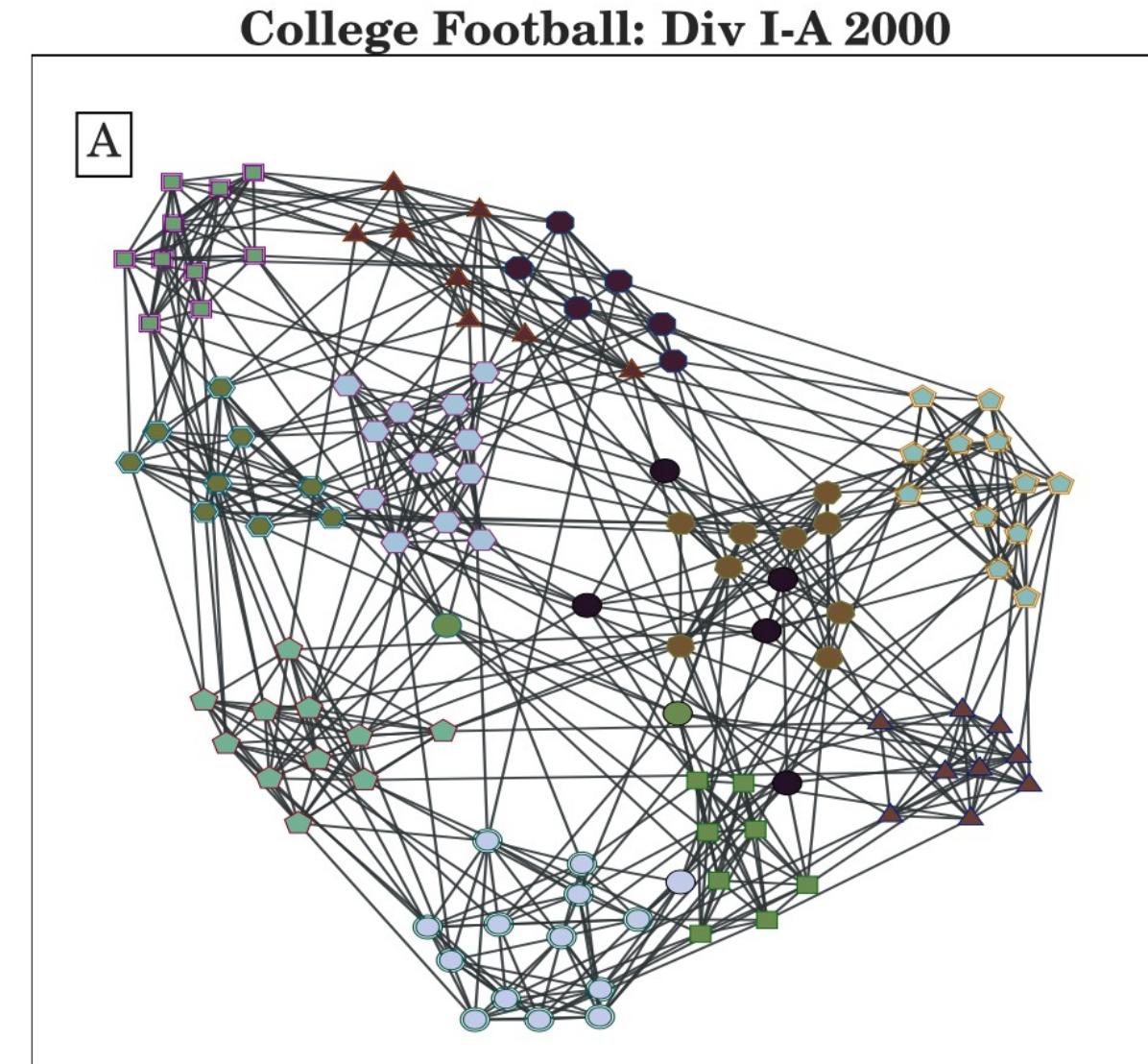
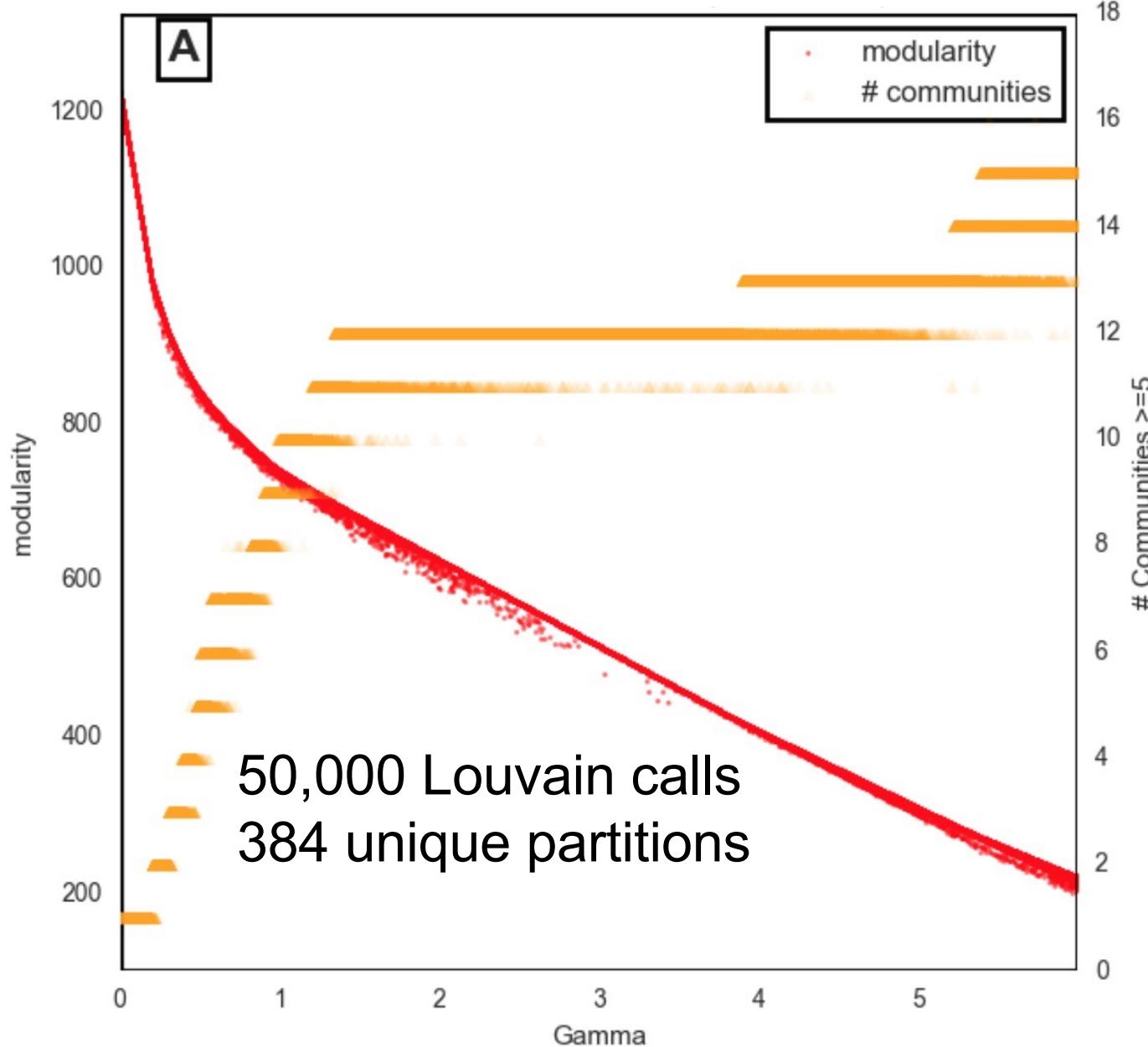
For complete documentation, please visit our [ReadTheDocs page](#):

<http://champ.readthedocs.io/en/latest/>

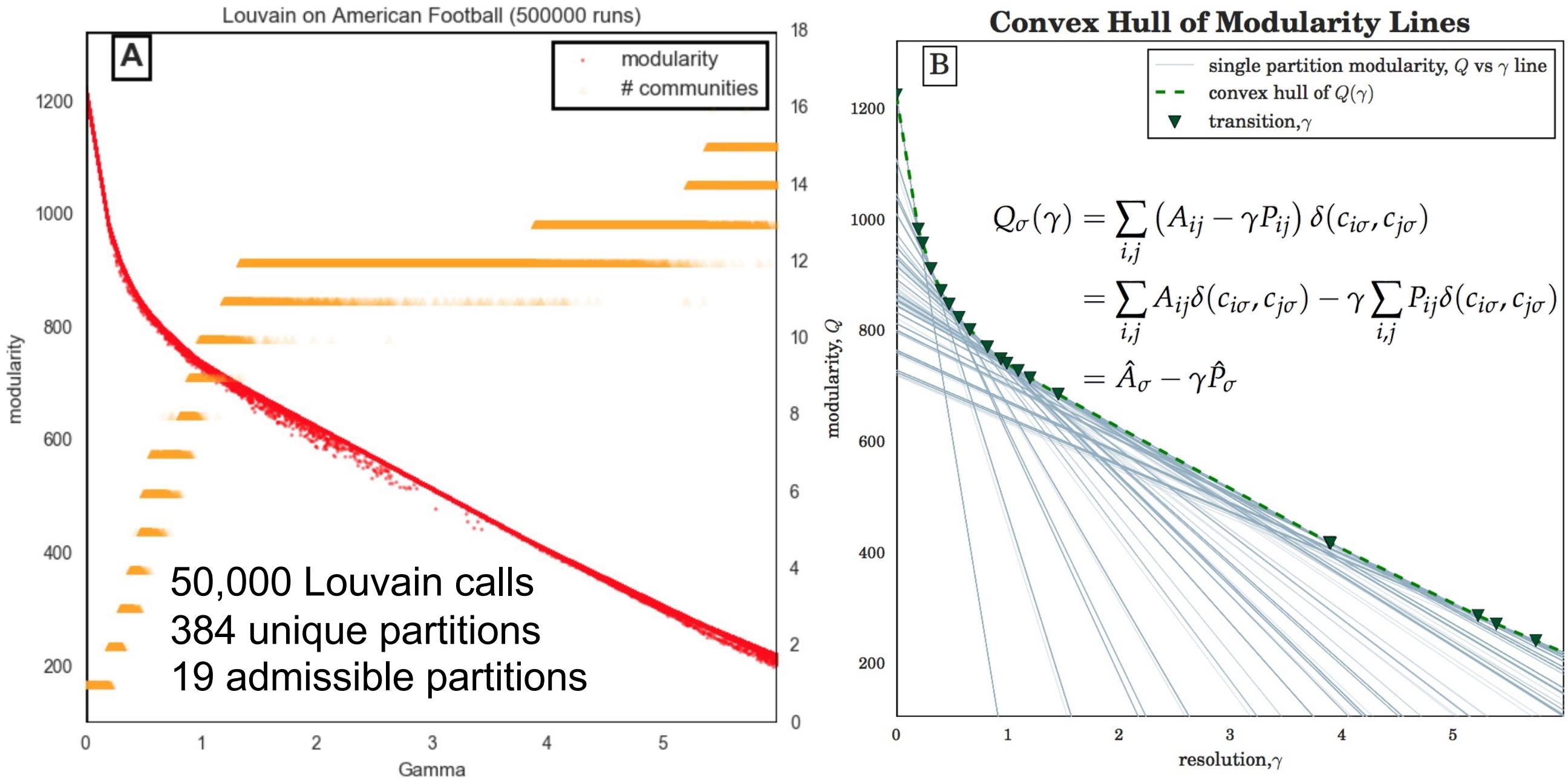
Download and Installation:

The CHAMP module is hosted on [PyPi](#). The easiest way to install is via the pip command:

How to pick γ ? (1) CHAMP

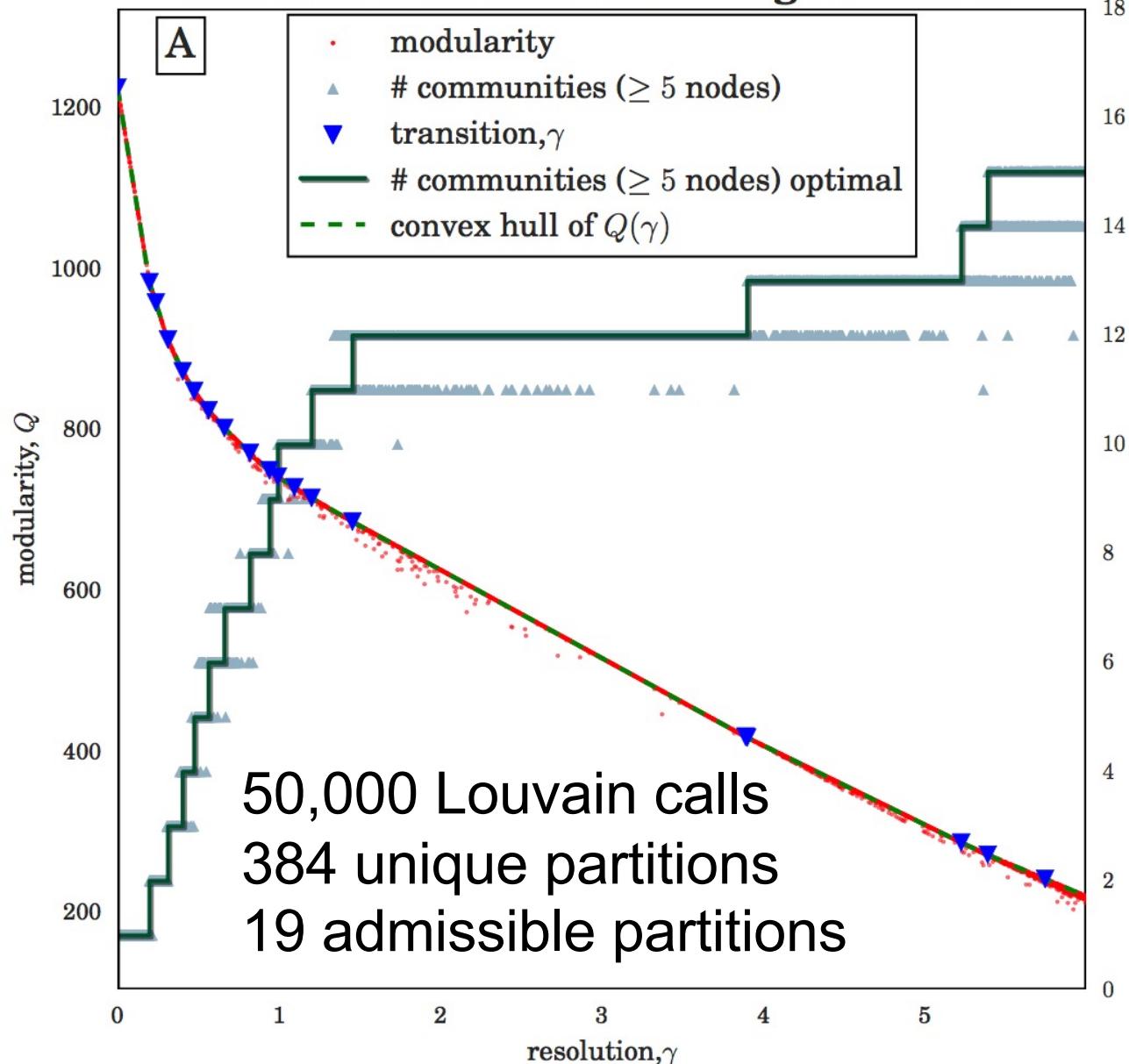


$Q_\sigma(\gamma)$ isn't a point. Each partition σ defines a line.

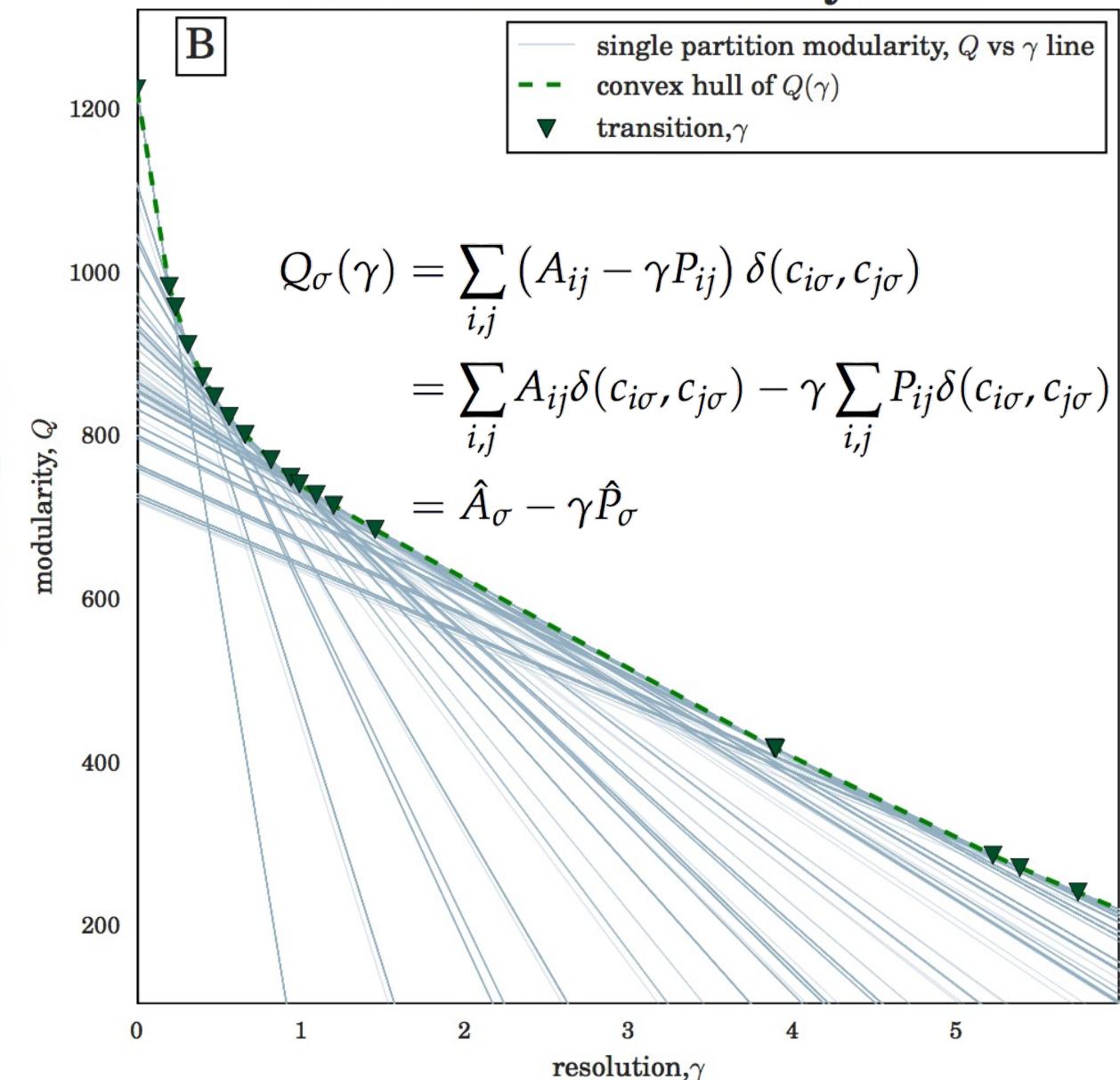


$Q_\sigma(\gamma)$ isn't a point. Each partition σ defines a line.

Louvain + CHAMP on College Football



Convex Hull of Modularity Lines



Modularity-SBM Equivalence (Newman, 2016)

- Degree-corrected “planted partition” SBM with K blocks, $\theta_{\text{in}}, \theta_{\text{out}}$

$$\ln P(\mathbf{A} \mid \theta_{\text{in}}, \theta_{\text{out}}, \mathbf{g}) = \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \cdot \frac{\theta_{\text{in}} - \theta_{\text{out}}}{\ln \theta_{\text{in}} - \ln \theta_{\text{out}}} \right] \delta(g_i, g_j)$$

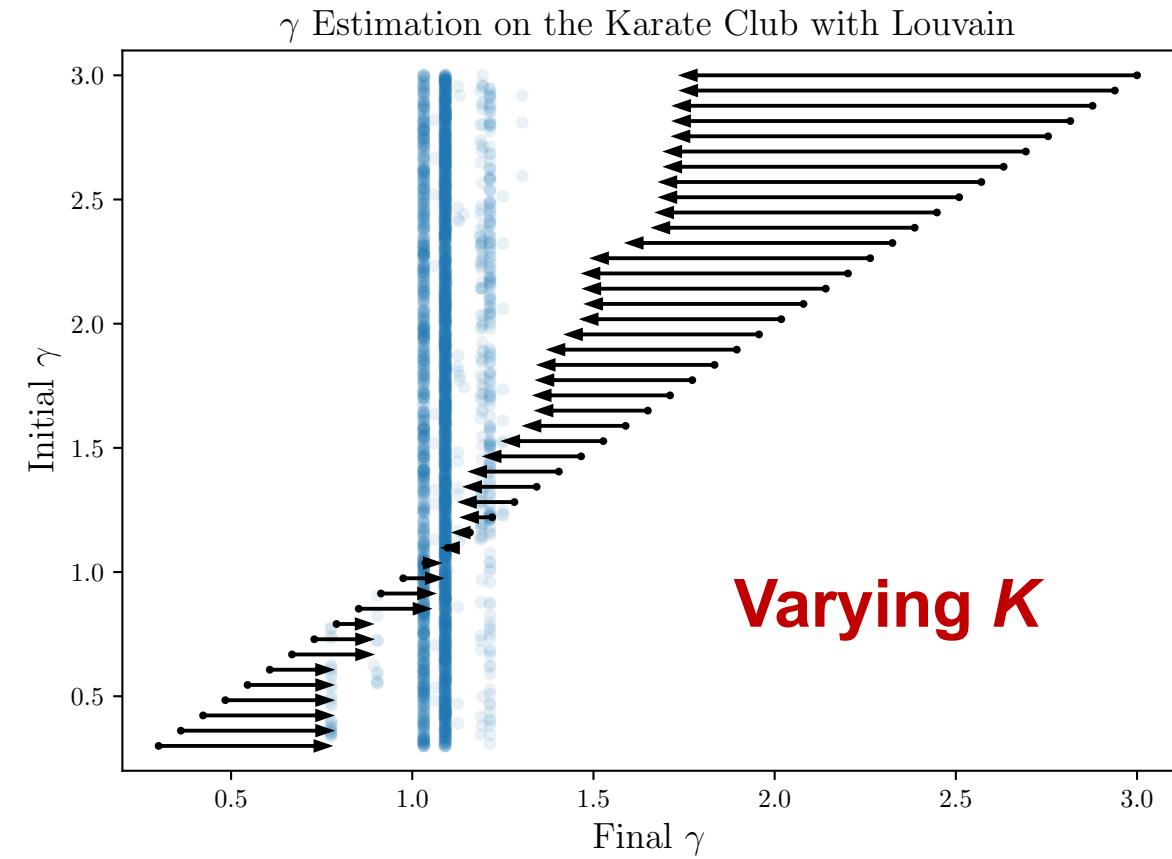
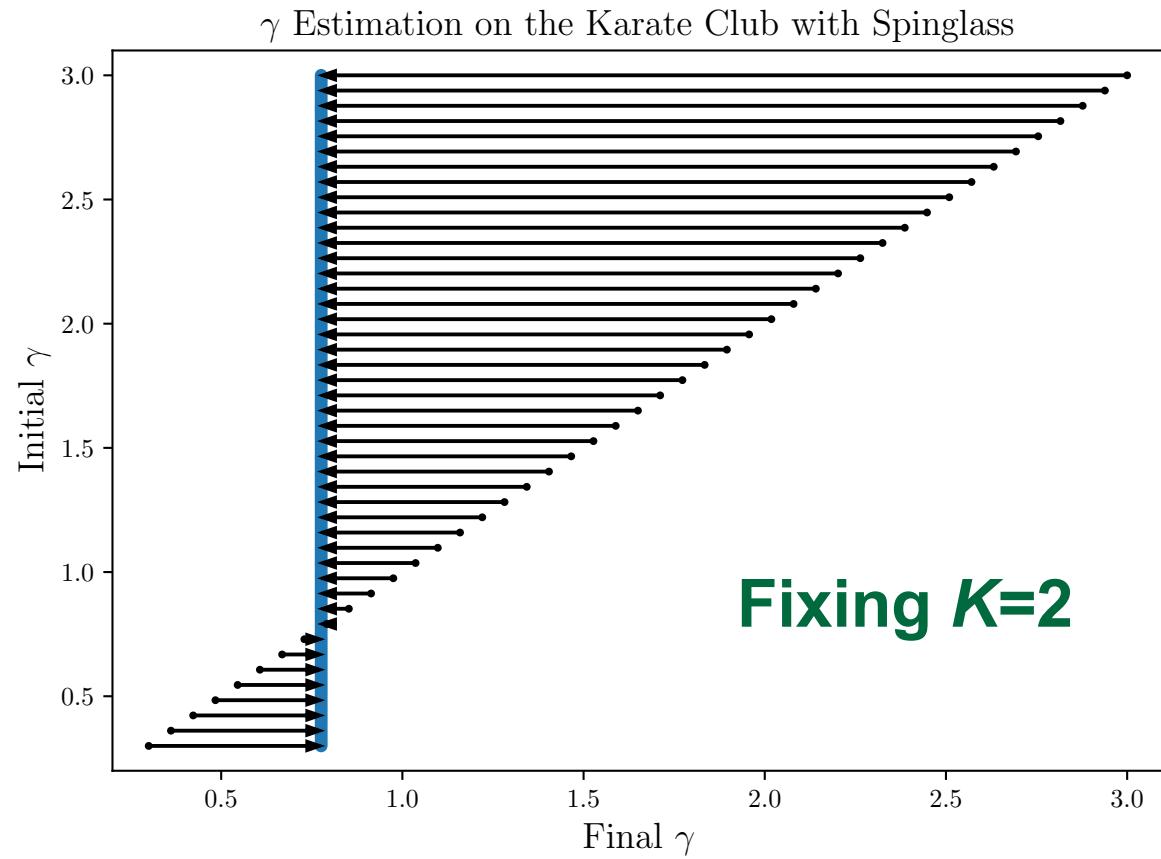
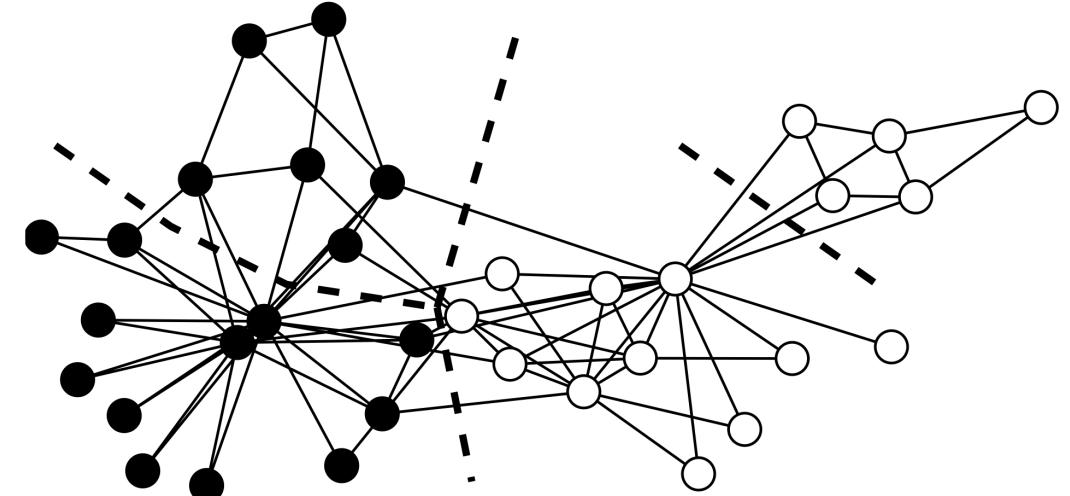
$$\gamma = \frac{\theta_{\text{in}} - \theta_{\text{out}}}{\ln \theta_{\text{in}} - \ln \theta_{\text{out}}}$$

$$\theta_{\text{in}} = \frac{2m_{\text{in}}}{\sum_r \kappa_r^2 / (2m)}, \quad \kappa_r = \sum_i k_i \delta(g_i, r)$$

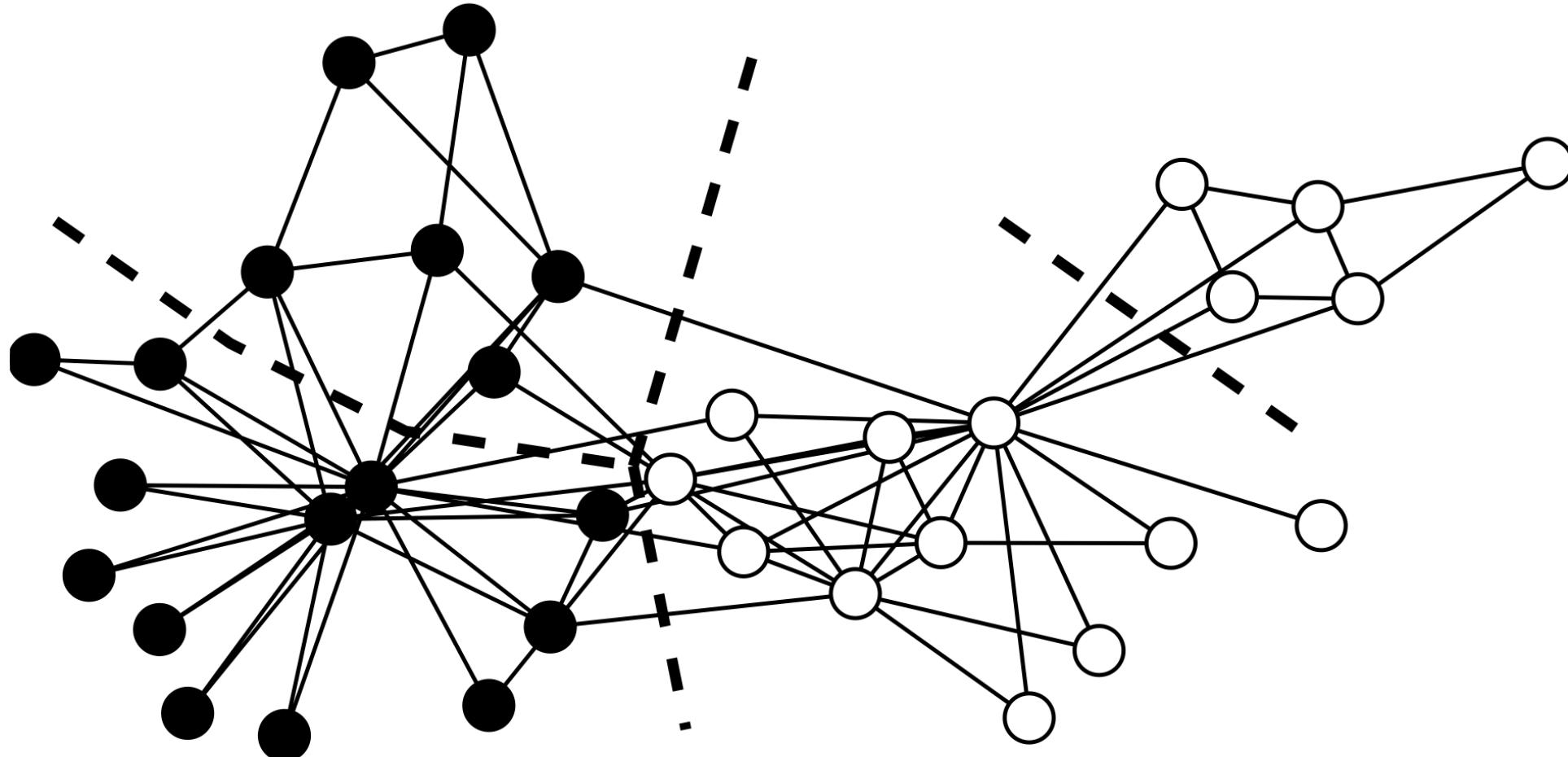
$$\theta_{\text{out}} = \frac{2m - 2m_{\text{in}}}{2m - \sum_r \kappa_r^2 / (2m)}$$

| Network | n | m | K | γ |
|---------------------------------------|------|-------|-----|----------|
| Karate club | 34 | 78 | 2 | 0.78 |
| Dolphin social network | 62 | 159 | 2 | 0.59 |
| Political blogs | 1225 | 16780 | 2 | 0.67 |
| Books about politics | 105 | 441 | 2 | 0.59 |
| Characters from <i>Les Misérables</i> | 77 | 254 | 6 | 1.36 |
| American college football | 115 | 614 | 11 | 2.27 |
| Jazz collaborations | 198 | 2742 | 16 | 1.19 |
| Email messages | 1133 | 5451 | 26 | 3.63 |

Fixing v. Varying K for Zachary's Karate Club

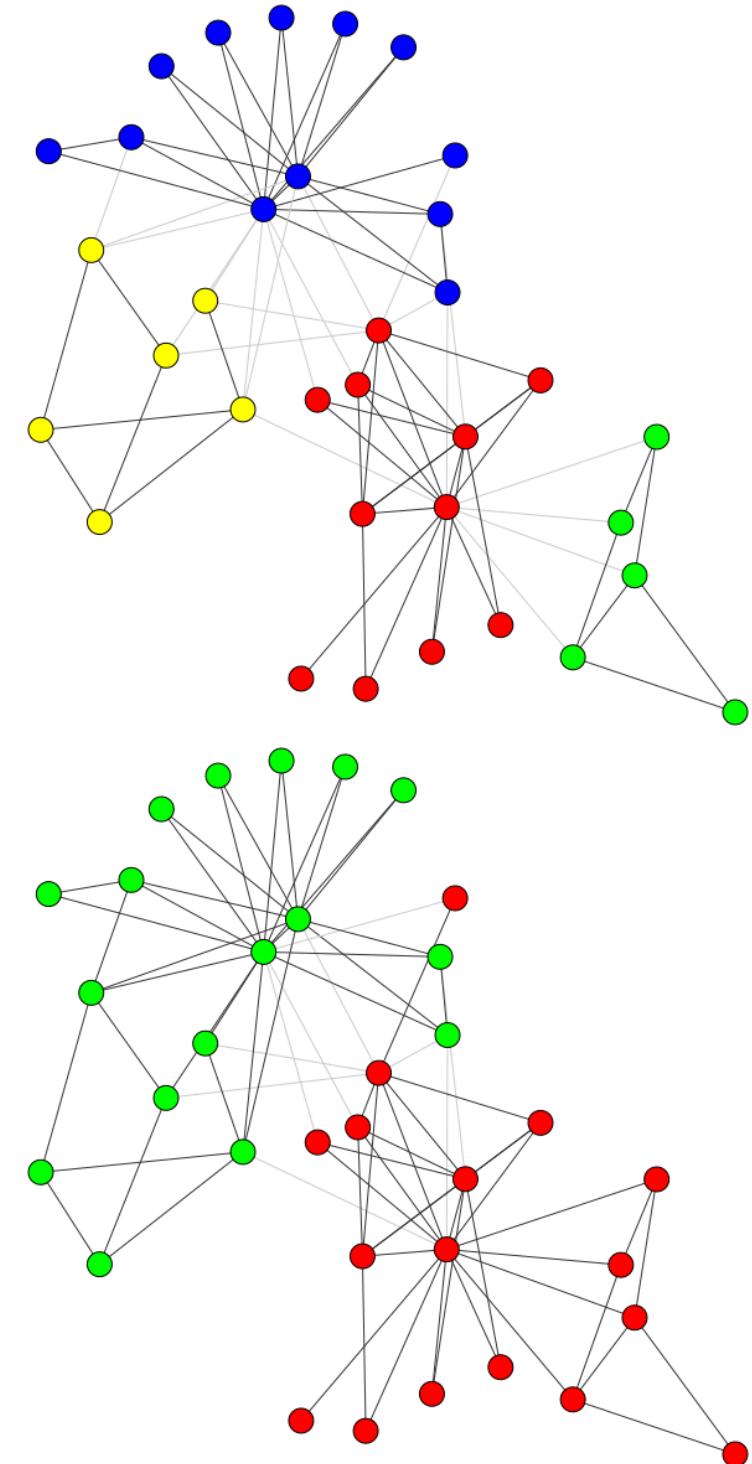
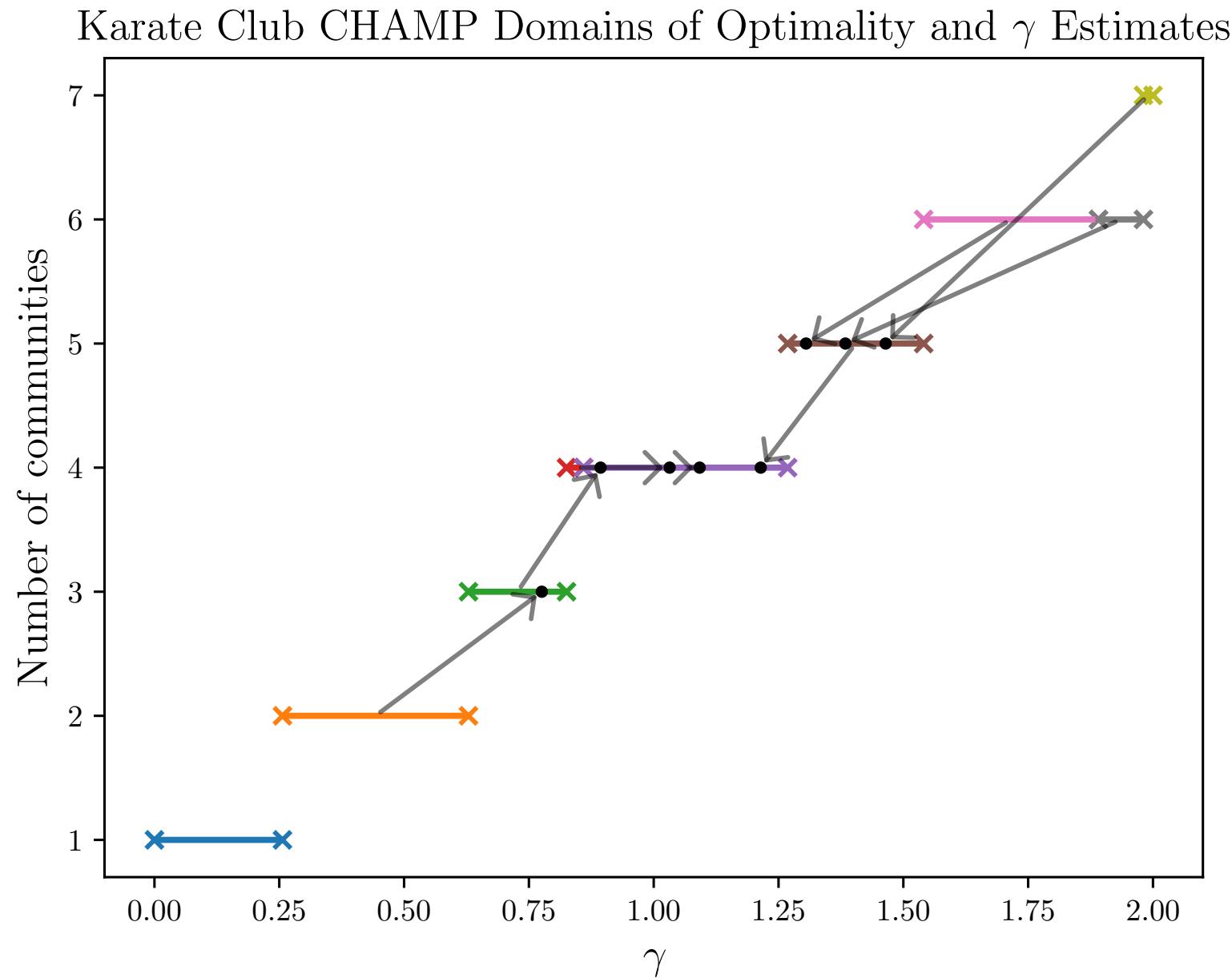


Zachary Karate Club



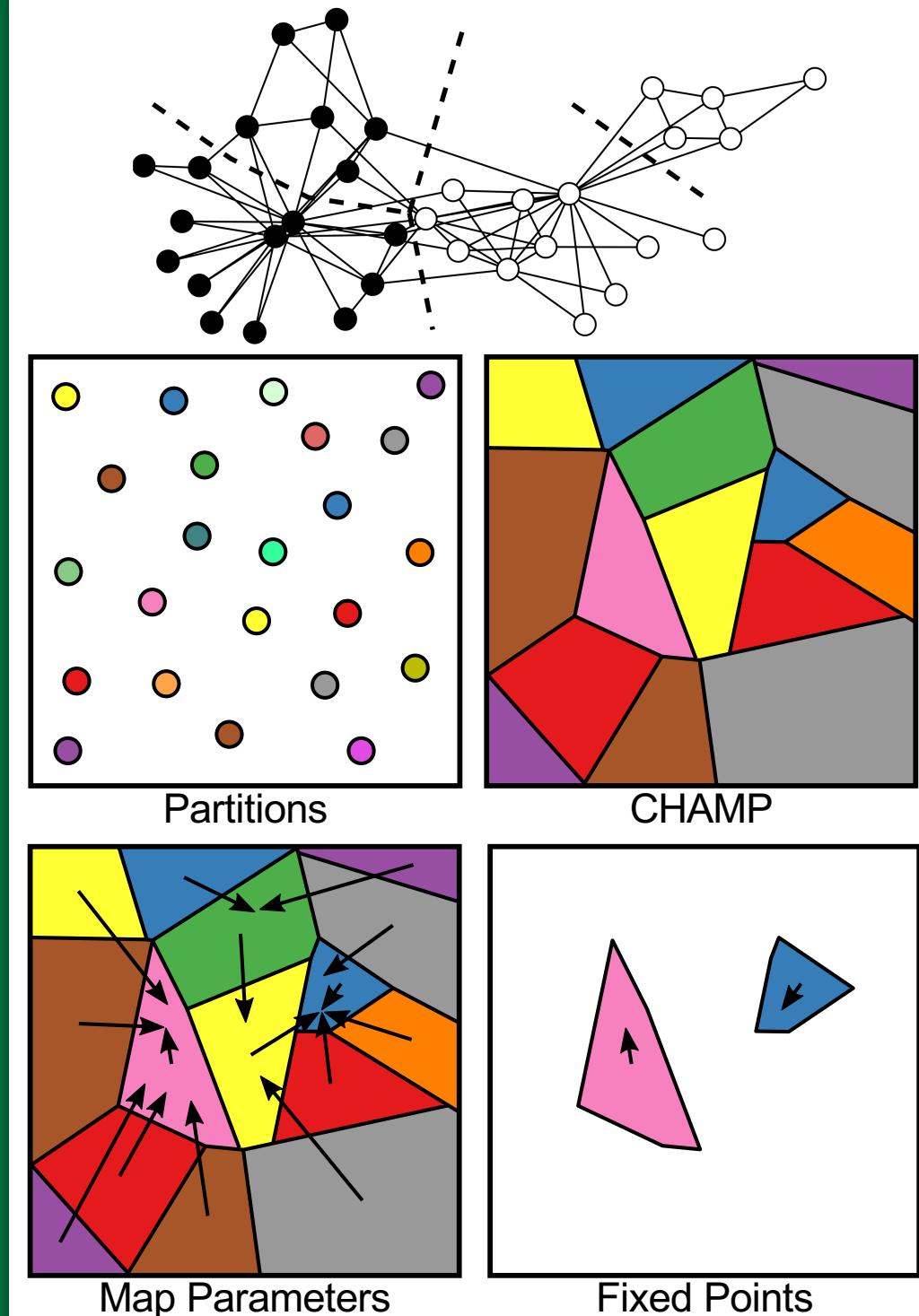
This partition optimizes ***modularity*** (at default resolution) which measures the number of intra-community ties (relative to a random model)
“If your method doesn’t work on this network, then go home.”

Karate Club: “ModularityPruning” Iterative Map on Parameter Space



Takeaway Messages

1. CommunitiesSNH2023.Rmd file designed to highlight upcoming IDEANet capabilities
2. Community detection is “just” unsupervised clustering of nodes in the network:
 - NFL: No single “right” way to cluster (different tools, knobs)
 - “Meaning” of identified groups depends on your application
3. CHAMP and Iterative Parameter Maps in R:
 - Multilayer modularity capabilities “coming soon”
github.com/wweir827/CHAMP
github.com/ragibson/ModularityPruning



Thank you!

github.com/GenLouvain

github.com/wweir827/champ

github.com/ragibson/ModularityPruning

<http://muchा.host.dartmouth.edu>

peter.j.mucha@dartmouth.edu