NETWORK SCIENCE OF ONLINE INTERACTIONS

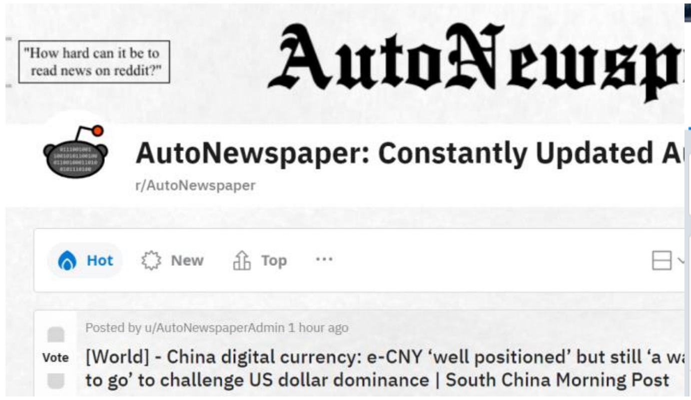
Reddit Inorganic communities +
Chapter 6 exercises +
Descriptive vs Inference methods

Joao Neto

31/May/2023

INORGANIC COMMUNITIES

- Inorganic subreddits
  - Botted subreddits
  - Purpose-built subreddits
- Can we find observables to identify those?
- Cascade match
  - Match on cascade size, look into other tree observables
  - If they collapse, differences are due to size
  - If not, some other difference
  - Used in [1] to show that fake news in the Vosoughi paper did not spread further, just more



"How hard can it be to read news on reddit?"

AutoNewsp

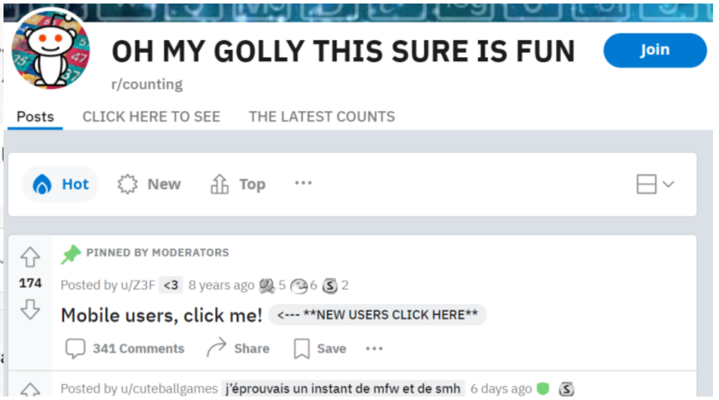
AutoNewspaper: Constantly Updated A

r/AutoNewspaper

Posted by u/AutoNewspaperAdmin 1 hour ago

Vote

[World] - China digital currency: e-CNY 'well positioned' but still 'a wa to go' to challenge US dollar dominance | South China Morning Post



OH MY GOLLY THIS SURE IS FUN

r/counting

PINNED BY MODERATORS

174

Posted by u/Z3F <3 8 years ago 5 6 2

Mobile users, click me! <--- **NEW USERS CLICK HERE**

341 Comments Share Save

Posted by u/cuteballgames j'éprouvais un instant de mfw et de smh 6 days ago

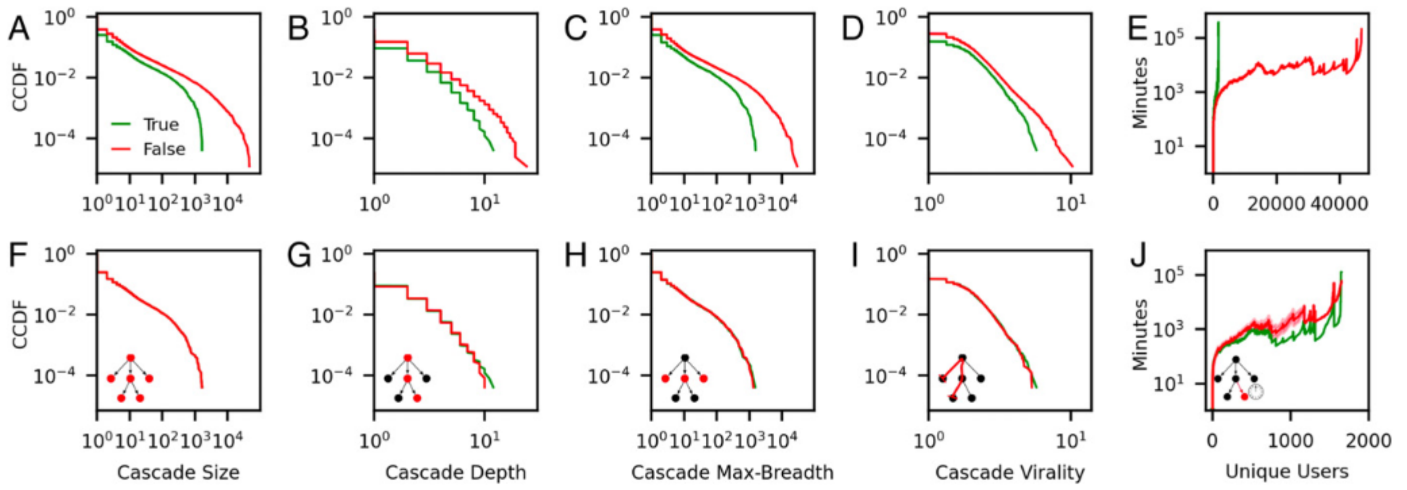
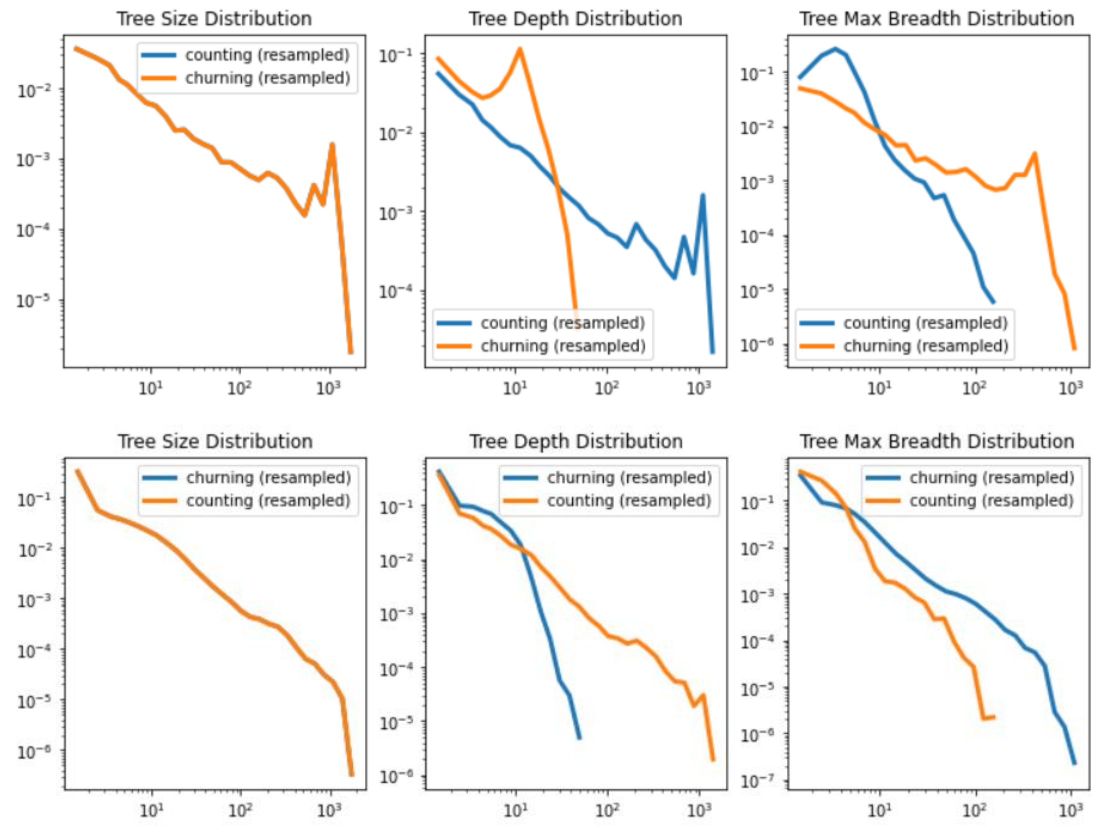


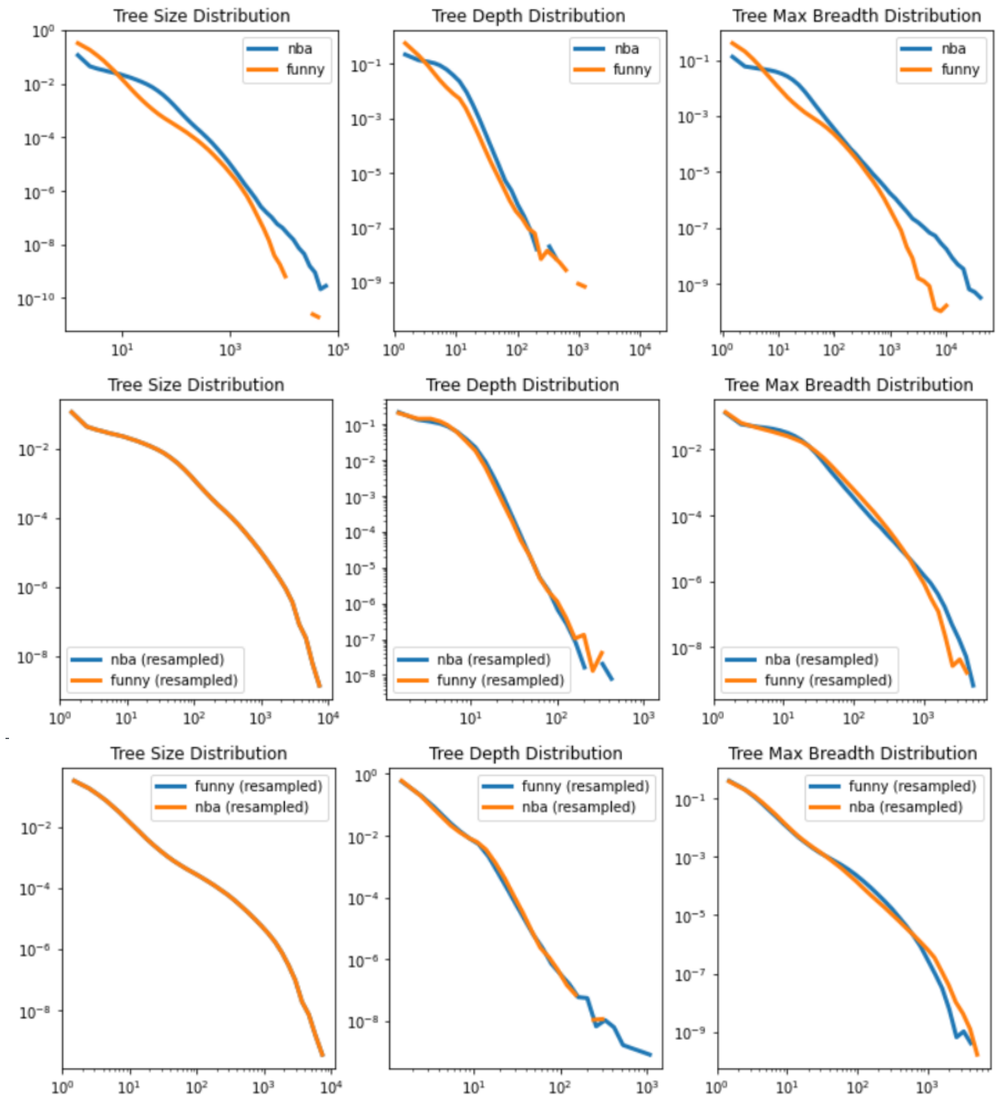
Fig. 1. (*A–E*) Structural and temporal statistics of false-news and true-news cascades diffusing on Twitter, as presented in ref. 11. Cascades in the tw datasets have different size distributions (*A*). (*F–J*) The same analyses as the plots directly above, carried out for two subsampled datasets with matched siz distributions. Controlling for size collapses statistical differences in these properties. *Insets* depict each statistic on a simple cascade.

[1] Juul, J. L. & Ugander, J. *Proc Natl Acad Sci USA* **118**, e2100786118 (2021).

INORGANIC COMMUNITIES

- Distribution collapse
  - Do different subreddits collapse?
  - Yes, but not all:

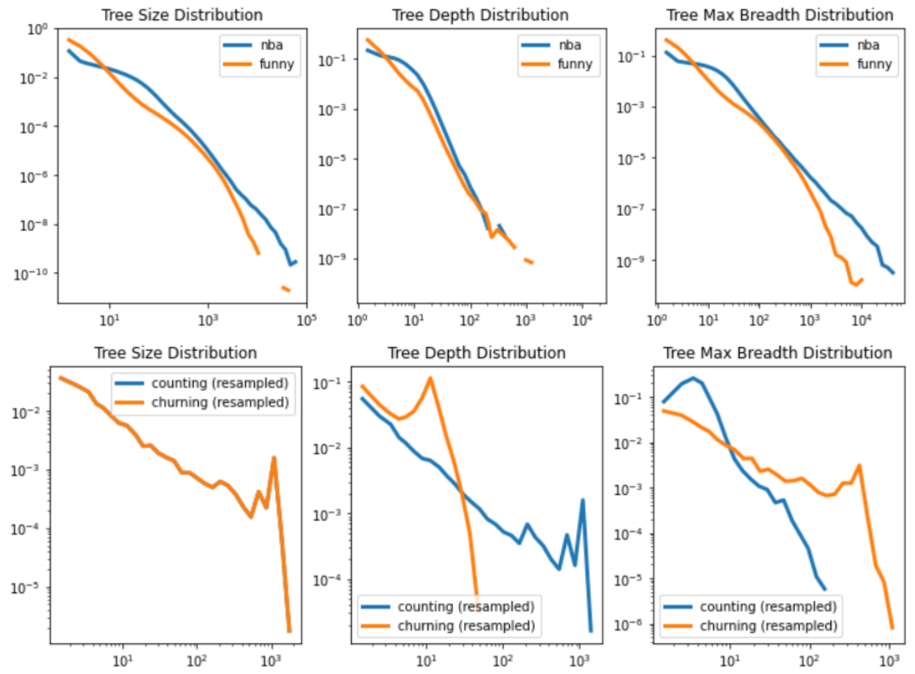


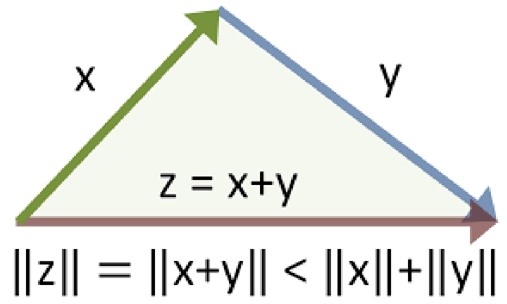


INORGANIC COMMUNITIES

- How to measure the statistical distance?
  - Various statistical measures, with different assumptions
- Definition of statistical metric $d(x,y)$
  1. Non-negative: $d(x,y) \geq 0$
  2. $d(x,y) = 0$ iff $\mathrm{x} = \mathrm{y}$
  3. Symmetric: $d(x,y) = d(y,x)$
  4. Triangle inequality: $d(x,z) \leq d(x,y) + d(x,z)$

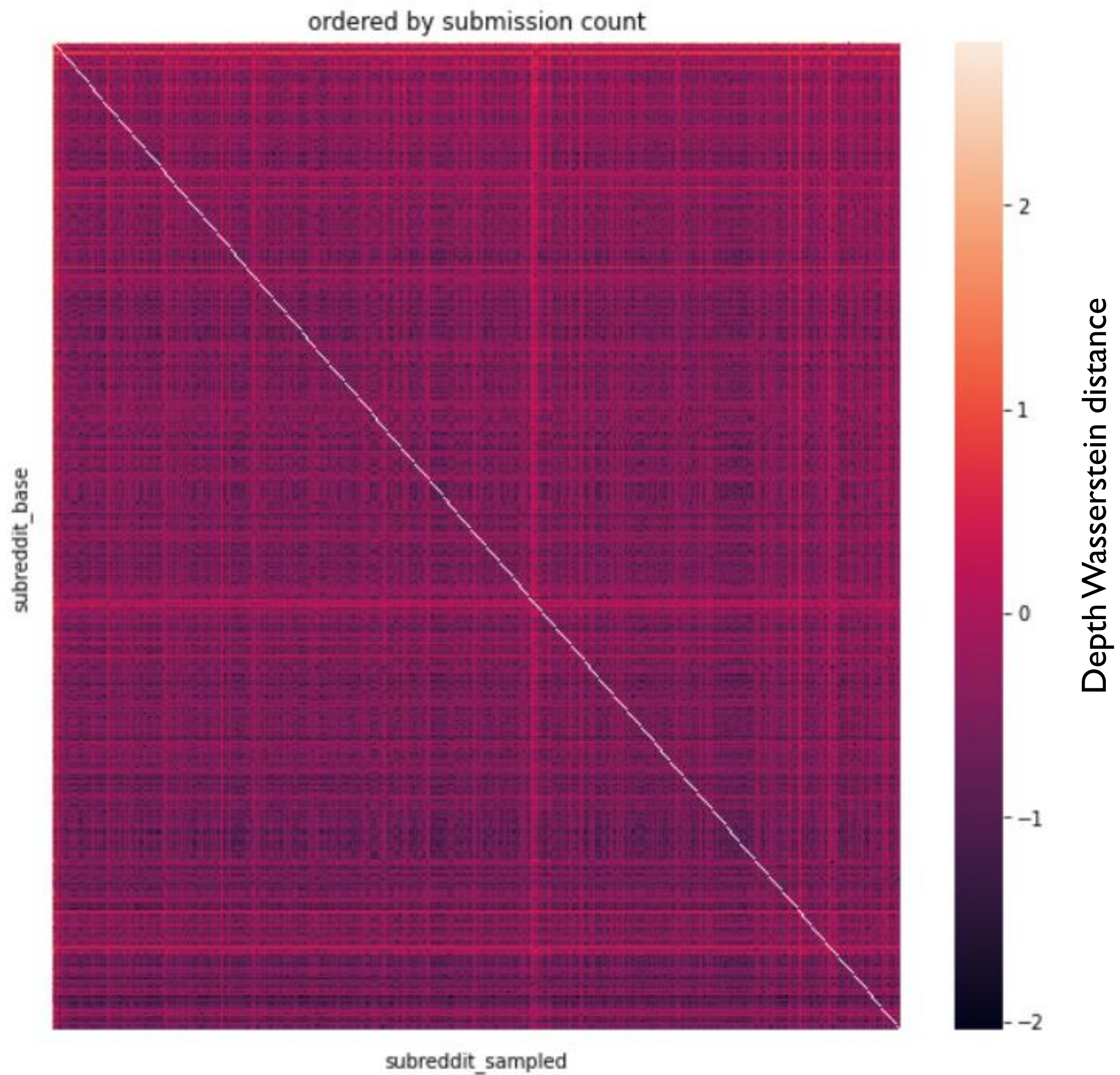
- Measures:
  - **Kullback–Leibler divergence** $D_{KL}(P||Q)$
    - Not symmetric
  - **Jensen-Shannon divergence** $JSD(P||Q)$
    - Symmetric, but doesn't satisfy triangle inequality
  - **Wasserstein distance** $W(P,Q)$
    - True metric





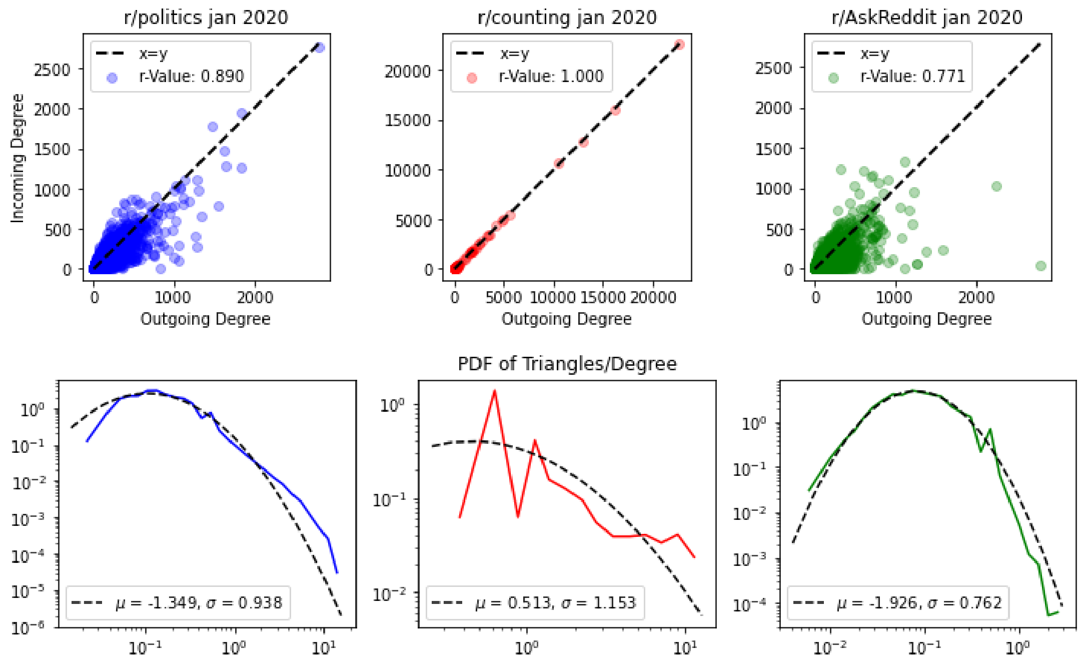
INORGANIC COMMUNITIES

- TOP 500 subreddits
  - No clear effect of size
  - Subreddits that don’t collapse with some don’t collapse with any → inorganic subreddits
  - TOP 10 largest median distance
    - r/counting
    - r/DebateReligion
    - r/PurplePillDebate
    - r/pan_media
    - r/MLPLounge
    - r/SVExchange
    - r/AskOuija
    - r/pokemontrades
    - r/changemyview
    - r/RoastMe

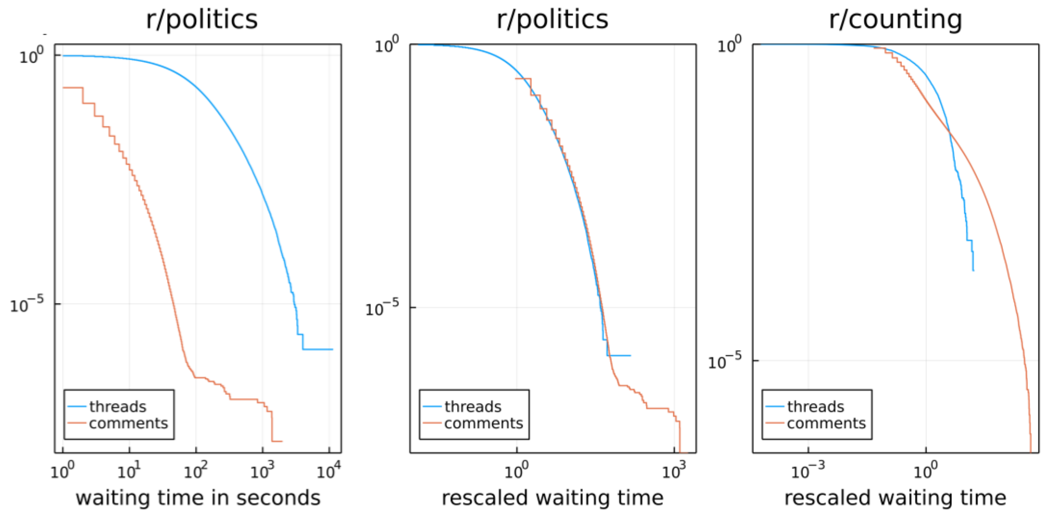


INORGANIC COMMUNITIES

- User-interaction network
  - Can network metrics spot differences?
  - r/counting differs from normal subreddits

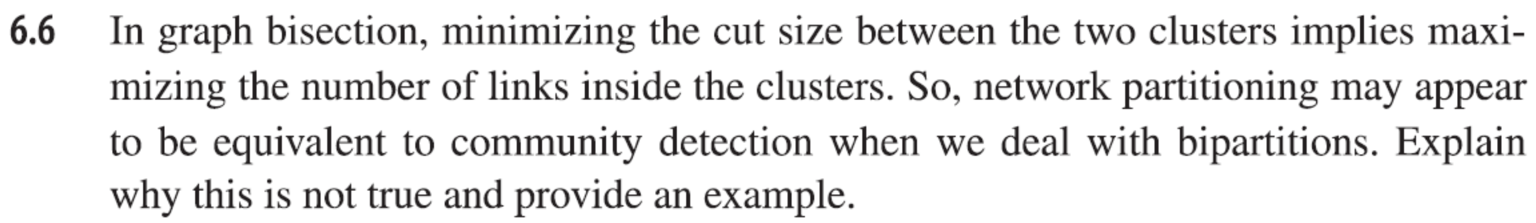


- Timestamp analysis
  - Waiting times differ between submissions and comments
  - Collapse when rescaled by the mean
  - Interval distribution doesn't collapse for r/counting



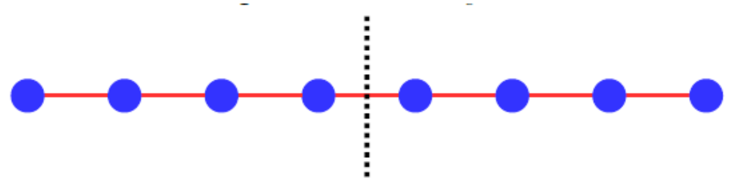
CHAPTER 6 EXERCISES

- Exercises 6.6, 6.10, 6.22



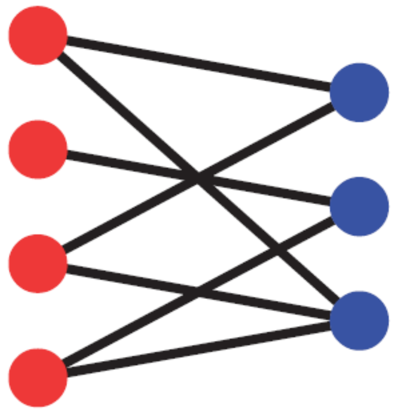
6.6 In graph bisection, minimizing the cut size between the two clusters implies maximizing the number of links inside the clusters. So, network partitioning may appear to be equivalent to community detection when we deal with bipartitions. Explain why this is not true and provide an example.

- The two clusters may not be communities due to low internal link density



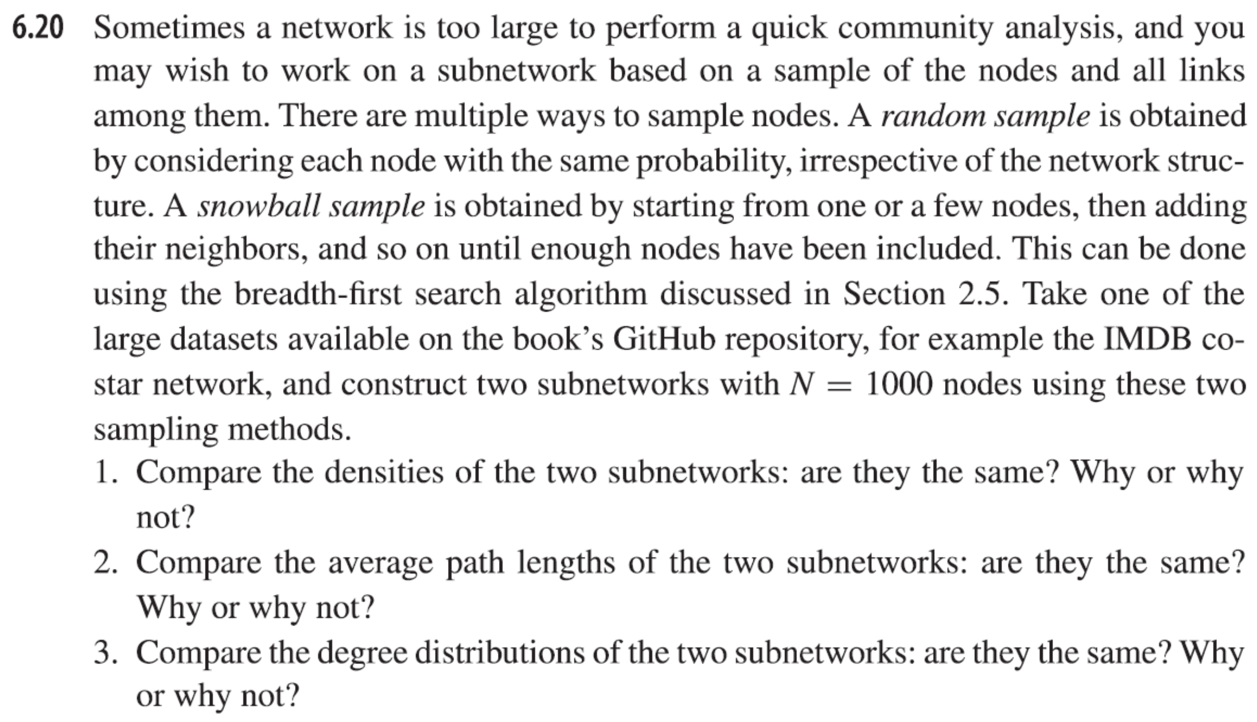
CHAPTER 6 EXERCISES

6.10 Recall that a bipartite network consists of two classes of nodes, say A and B, and links join only nodes in A with nodes in B. See the example in Figure 6.18, where the classes are colored in red and blue. Show that the modularity of the partition of a bipartite network in the two groups A and B is $-1/2$. This is also the lowest value that modularity can reach.



- Modularity $Q = \frac{1}{L}\sum_C \left(L_C - \frac{k_C^2}{4L}\right)$
- Bipartite network
  - $L_C = 0$
  - $k_C = L$
  - $Q = \frac{1}{L}\left(-\frac{L^2}{4L^2} - \frac{L^2}{4L^2}\right) = -\frac{1}{2}$

CHAPTER 6 EXERCISES



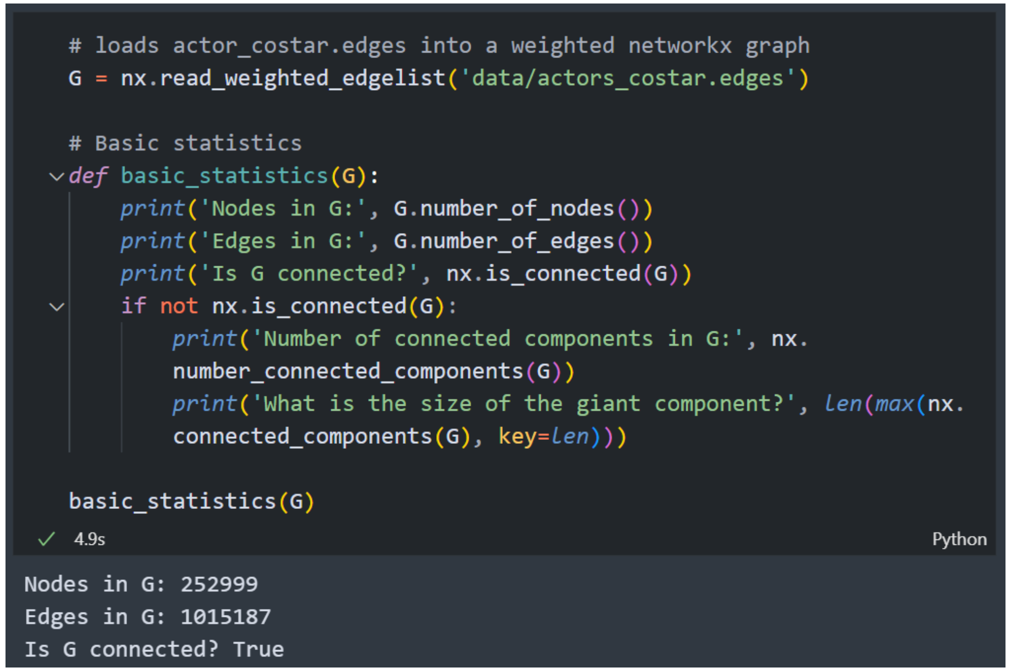
6.20 Sometimes a network is too large to perform a quick community analysis, and you may wish to work on a subnetwork based on a sample of the nodes and all links among them. There are multiple ways to sample nodes. A *random sample* is obtained by considering each node with the same probability, irrespective of the network structure. A *snowball sample* is obtained by starting from one or a few nodes, then adding their neighbors, and so on until enough nodes have been included. This can be done using the breadth-first search algorithm discussed in Section 2.5. Take one of the large datasets available on the book's GitHub repository, for example the IMDB co-star network, and construct two subnetworks with $N = 1000$ nodes using these two sampling methods.

1. Compare the densities of the two subnetworks: are they the same? Why or why not?
2. Compare the average path lengths of the two subnetworks: are they the same? Why or why not?
3. Compare the degree distributions of the two subnetworks: are they the same? Why or why not?

- About sampling (more in the future)

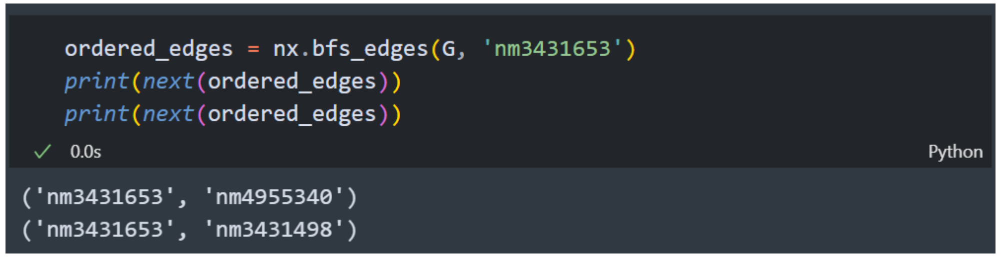
CHAPTER 6 EXERCISES

- 253K nodes, 1M edges, weighted, connected



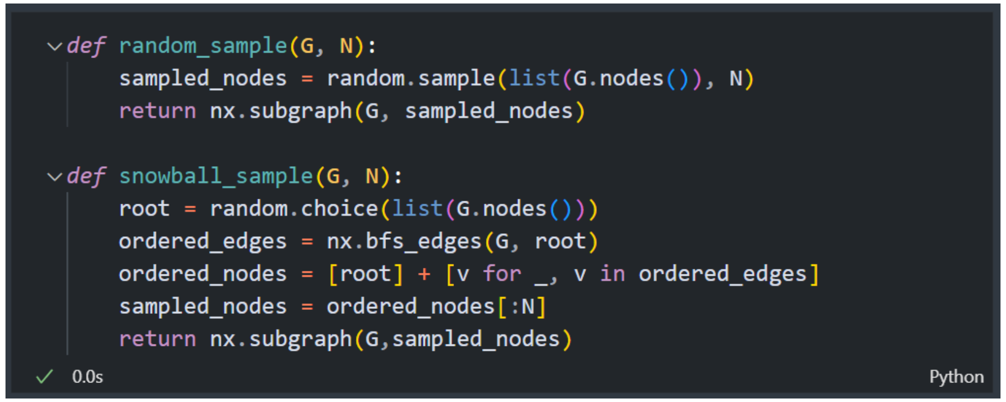
```
# loads actor_costar.edges into a weighted networkx graph
G = nx.read_weighted_edgelist('data/actors_costar.edges')

# Basic statistics
def basic_statistics(G):
    print('Nodes in G:', G.number_of_nodes())
    print('Edges in G:', G.number_of_edges())
    print('Is G connected?', nx.is_connected(G))
    if not nx.is_connected(G):
        print('Number of connected components in G:', nx.
        number_connected_components(G))
        print('What is the size of the giant component?', len(max(nx.
        connected_components(G), key=len)))

basic_statistics(G)
```

```
Nodes in G: 252999
Edges in G: 1015187
Is G connected? True
```

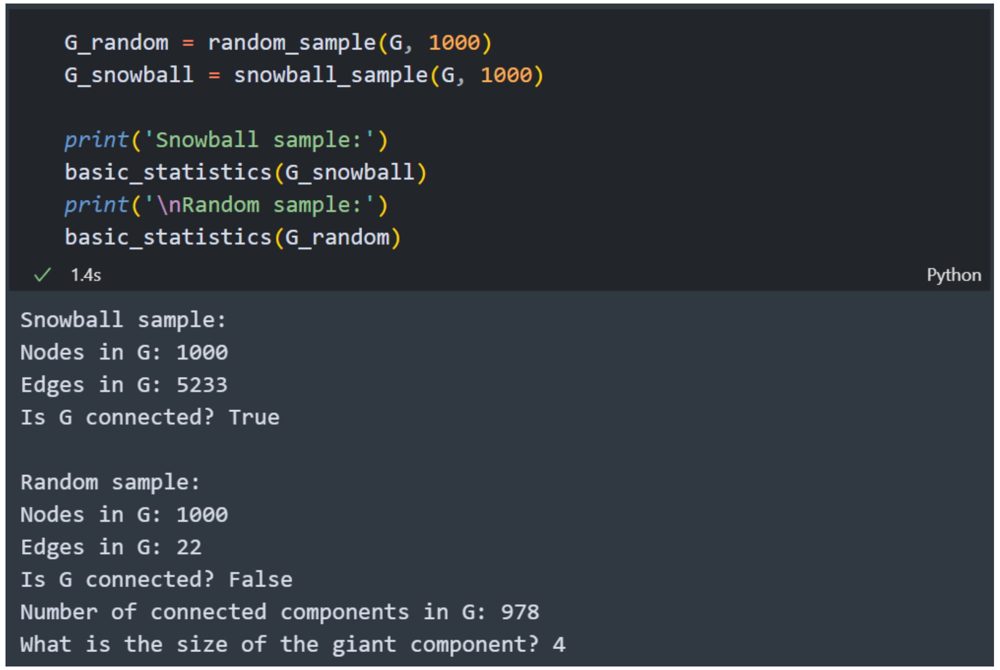


```
ordered_edges = nx.bfs_edges(G, 'nm3431653')
print(next(ordered_edges))
print(next(ordered_edges))
```

```
('nm3431653', 'nm4955340')
('nm3431653', 'nm3431498')
```



```
def random_sample(G, N):
    sampled_nodes = random.sample(list(G.nodes()), N)
    return nx.subgraph(G, sampled_nodes)

def snowball_sample(G, N):
    root = random.choice(list(G.nodes()))
    ordered_edges = nx.bfs_edges(G, root)
    ordered_nodes = [root] + [v for _, v in ordered_edges]
    sampled_nodes = ordered_nodes[:N]
    return nx.subgraph(G,sampled_nodes)
```



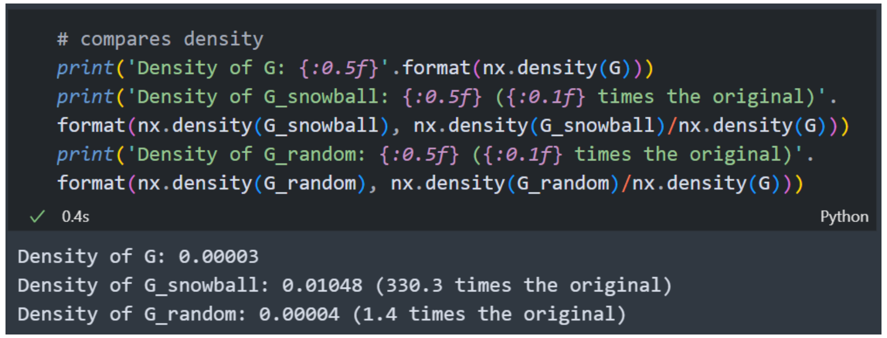
```
G_random = random_sample(G, 1000)
G_snowball = snowball_sample(G, 1000)

print('Snowball sample:')
basic_statistics(G_snowball)
print('\nRandom sample:')
basic_statistics(G_random)
```

```
Snowball sample:
Nodes in G: 1000
Edges in G: 5233
Is G connected? True

Random sample:
Nodes in G: 1000
Edges in G: 22
Is G connected? False
Number of connected components in G: 978
What is the size of the giant component? 4
```

CHAPTER 6 EXERCISES

1. Compare the densities of the two subnetworks: are they the same? Why or why not?
2. Compare the average path lengths of the two subnetworks: are they the same? Why or why not?
3. Compare the degree distributions of the two subnetworks: are they the same? Why or why not?

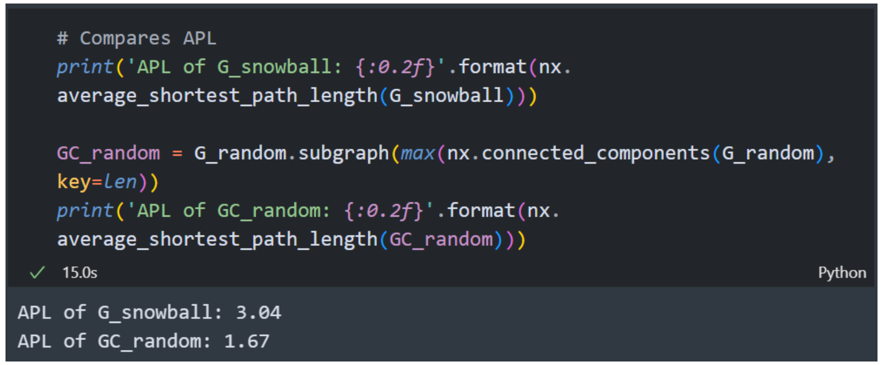
1. Snowball sampling massively overestimates density
2. APL not useful for random sampling due to small GC
3. Sampling biases degree distribution p_K
   - Random sampling: p_K not useful due to small GC
   - Snowball sampling: still heavy-tailed, but biased and shorter



```
# compares density
print('Density of G: {:0.5f}'.format(nx.density(G)))
print('Density of G_snowball: {:0.5f} ({:0.1f} times the original)'.
format(nx.density(G_snowball), nx.density(G_snowball)/nx.density(G)))
print('Density of G_random: {:0.5f} ({:0.1f} times the original)'.
format(nx.density(G_random), nx.density(G_random)/nx.density(G)))
```

✓ 0.4s Python

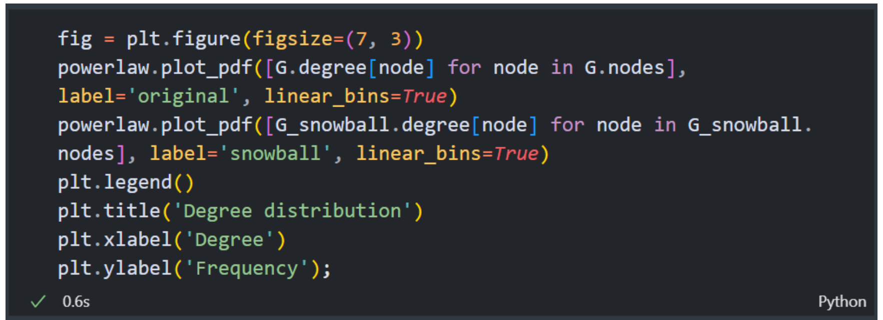
```
Density of G: 0.00003
Density of G_snowball: 0.01048 (330.3 times the original)
Density of G_random: 0.00004 (1.4 times the original)
```



```
# Compares APL
print('APL of G_snowball: {:0.2f}'.format(nx.
average_shortest_path_length(G_snowball)))

GC_random = G_random.subgraph(max(nx.connected_components(G_random),
key=len))
print('APL of GC_random: {:0.2f}'.format(nx.
average_shortest_path_length(GC_random)))
```

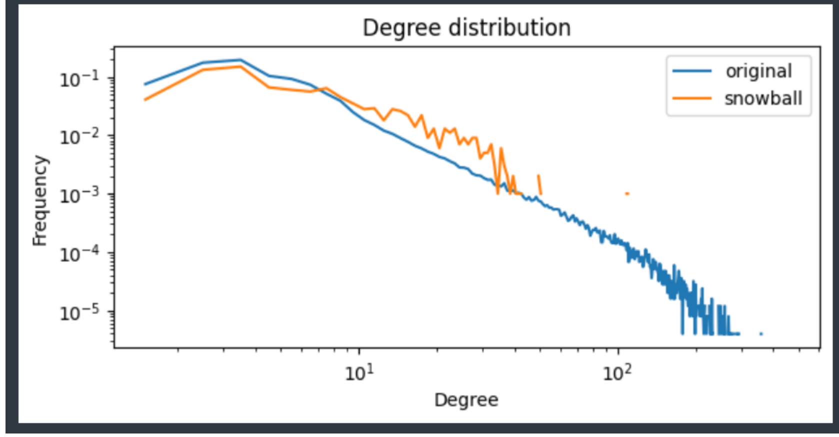
✓ 15.0s Python

```
APL of G_snowball: 3.04
APL of GC_random: 1.67
```



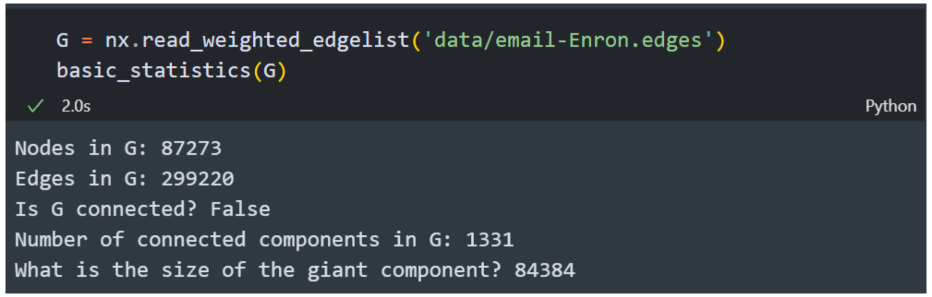
```
fig = plt.figure(figsize=(7, 3))
powerlaw.plot_pdf([G.degree[node] for node in G.nodes],
label='original', linear_bins=True)
powerlaw.plot_pdf([G_snowball.degree[node] for node in G_snowball.
nodes], label='snowball', linear_bins=True)
plt.legend()
plt.title('Degree distribution')
plt.xlabel('Degree')
plt.ylabel('Frequency');
```

✓ 0.6s Python



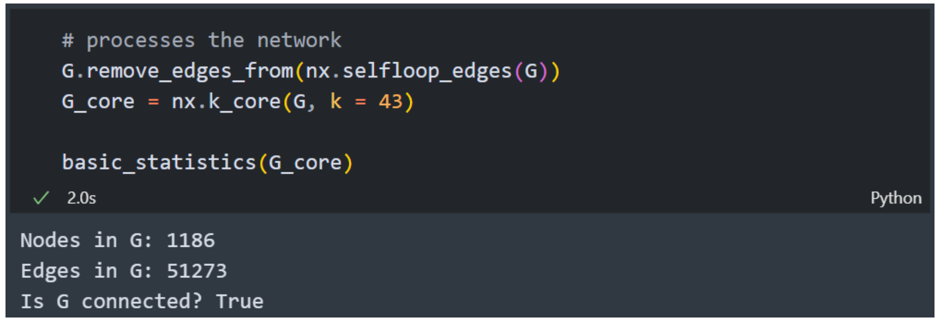
CHAPTER 6 EXERCISES

6.22 Analyze the community structure of the Enron email network (the dataset is available on the book's GitHub repository, in the email-Enron folder). Can you identify the module at the right-hand side in Figure 0.4? (*Hint*: Treat the network as undirected. It is large, so you should focus on the core as shown in Figure 0.4; use the NetworkX function `k_core()` from Section 3.6 with $k = 43$. You will first need to remove self-loops.)



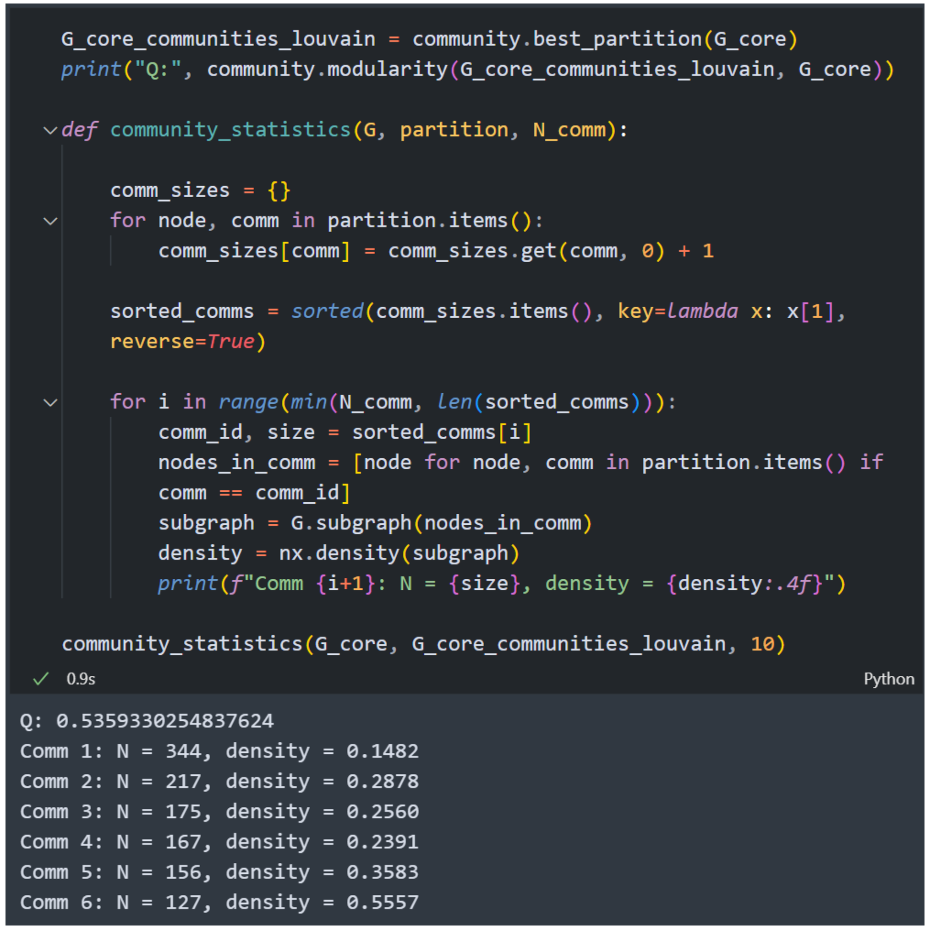
```python
G = nx.read_weighted_edgelist('data/email-Enron.edges')
basic_statistics(G)
```

```
Nodes in G: 87273
Edges in G: 299220
Is G connected? False
Number of connected components in G: 1331
What is the size of the giant component? 84384
```



```python
# processes the network
G.remove_edges_from(nx.selfloop_edges(G))
G_core = nx.k_core(G, k = 43)

basic_statistics(G_core)
```

```
Nodes in G: 1186
Edges in G: 51273
Is G connected? True
```



```python
G_core_communities_louvain = community.best_partition(G_core)
print("Q:", community.modularity(G_core_communities_louvain, G_core))

def community_statistics(G, partition, N_comm):

    comm_sizes = {}
    for node, comm in partition.items():
        comm_sizes[comm] = comm_sizes.get(comm, 0) + 1

    sorted_comms = sorted(comm_sizes.items(), key=lambda x: x[1],
    reverse=True)

    for i in range(min(N_comm, len(sorted_comms))):
        comm_id, size = sorted_comms[i]
        nodes_in_comm = [node for node, comm in partition.items() if
        comm == comm_id]
        subgraph = G.subgraph(nodes_in_comm)
        density = nx.density(subgraph)
        print(f"Comm {i+1}: N = {size}, density = {density:.4f}")

community_statistics(G_core, G_core_communities_louvain, 10)
```

```
Q: 0.5359330254837624
Comm 1: N = 344, density = 0.1482
Comm 2: N = 217, density = 0.2878
Comm 3: N = 175, density = 0.2560
Comm 4: N = 167, density = 0.2391
Comm 5: N = 156, density = 0.3583
Comm 6: N = 127, density = 0.5557
```

CHAPTER 6 EXERCISES

6.22 Analyze the community structure of the Enron email network (the dataset is available on the book's GitHub repository, in the email-Enron folder). Can you identify the module at the right-hand side in Figure 0.4? (*Hint*: Treat the network as undirected. It is large, so you should focus on the core as shown in Figure 0.4; use the NetworkX function `k_core()` from Section 3.6 with $k = 43$. You will first need to remove self-loops.)

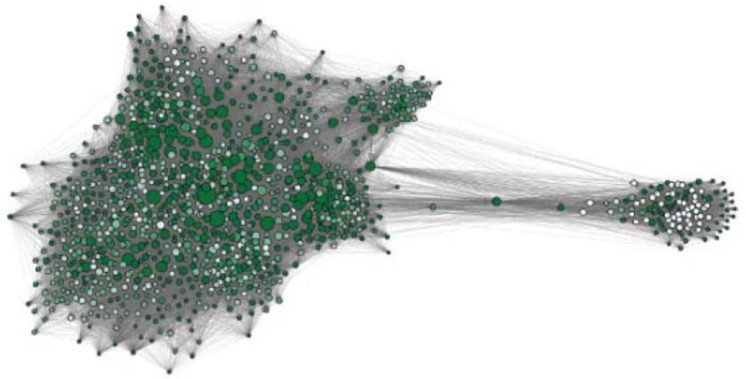
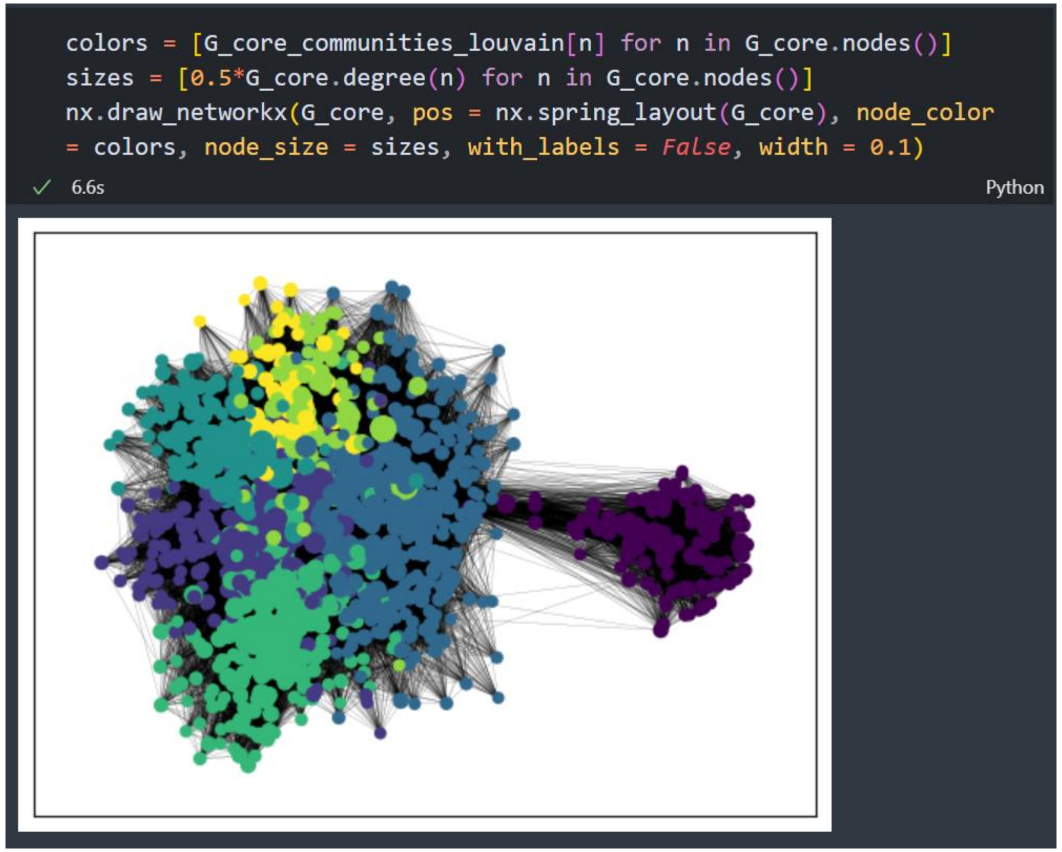
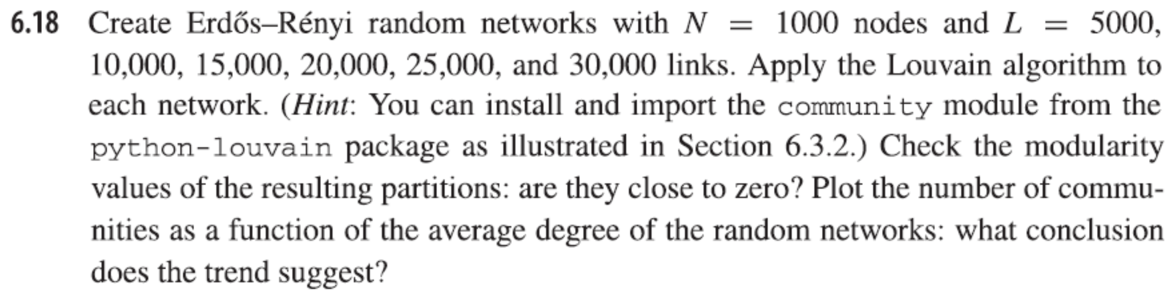


Fig. 0.4 A network based on a database of emails generated by employees of the Enron energy company. The data was acquired by the US Federal Energy Regulatory Commission during its investigation after the company's collapse in 2001. At the conclusion of the investigation, the emails were deemed to be in the public domain and made publicly available for historical research and academic purposes. Only a small portion of the central core of the network is shown. The direction of the links is shown by arrows.



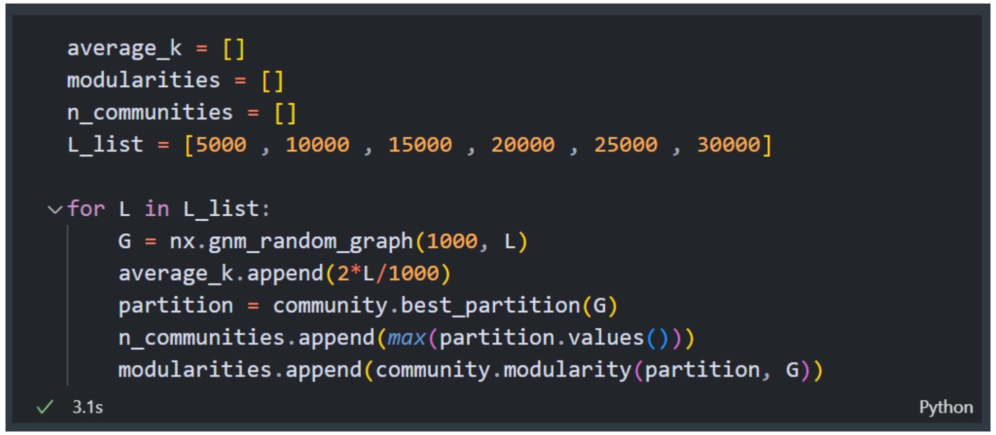
```
colors = [G_core_communities_louvain[n] for n in G_core.nodes()]
sizes = [0.5*G_core.degree(n) for n in G_core.nodes()]
nx.draw_networkx(G_core, pos = nx.spring_layout(G_core), node_color
= colors, node_size = sizes, with_labels = False, width = 0.1)
```

DESCRIPTIVE VS INFERENTIAL METHODS



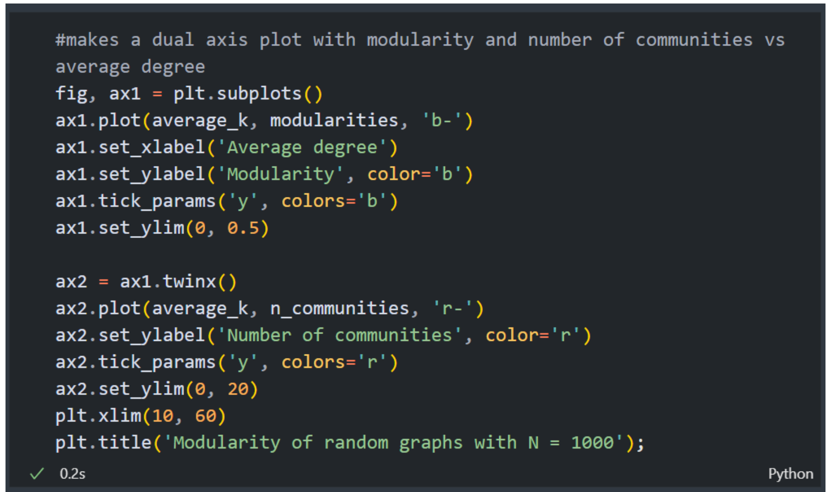
6.18 Create Erdős–Rényi random networks with $N = 1000$ nodes and $L = 5000$, 10,000, 15,000, 20,000, 25,000, and 30,000 links. Apply the Louvain algorithm to each network. (*Hint*: You can install and import the `community` module from the `python-louvain` package as illustrated in Section 6.3.2.) Check the modularity values of the resulting partitions: are they close to zero? Plot the number of communities as a function of the average degree of the random networks: what conclusion does the trend suggest?

- Basic test: community detection should find no community in ER
- Modularity maximization fails the test



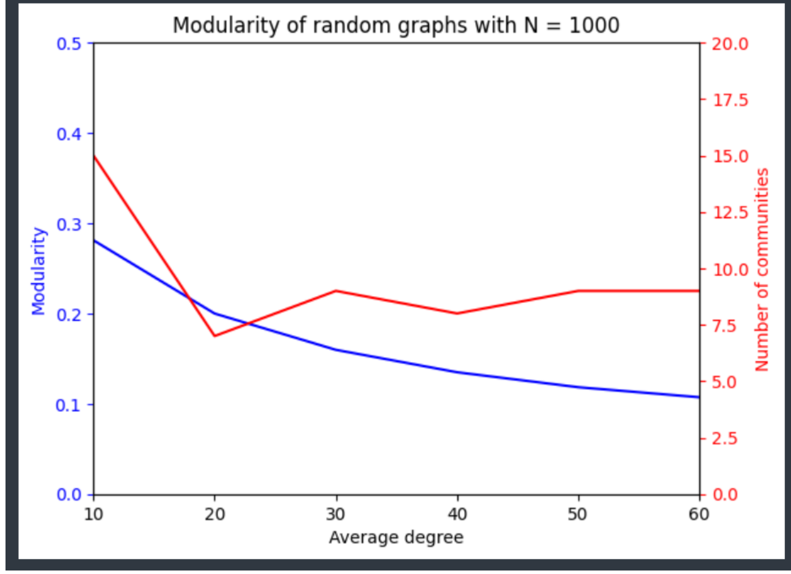
```
average_k = []
modularities = []
n_communities = []
L_list = [5000 , 10000 , 15000 , 20000 , 25000 , 30000]

for L in L_list:
    G = nx.gnm_random_graph(1000, L)
    average_k.append(2*L/1000)
    partition = community.best_partition(G)
    n_communities.append(max(partition.values()))
    modularities.append(community.modularity(partition, G))
```

✓ 3.1s Python



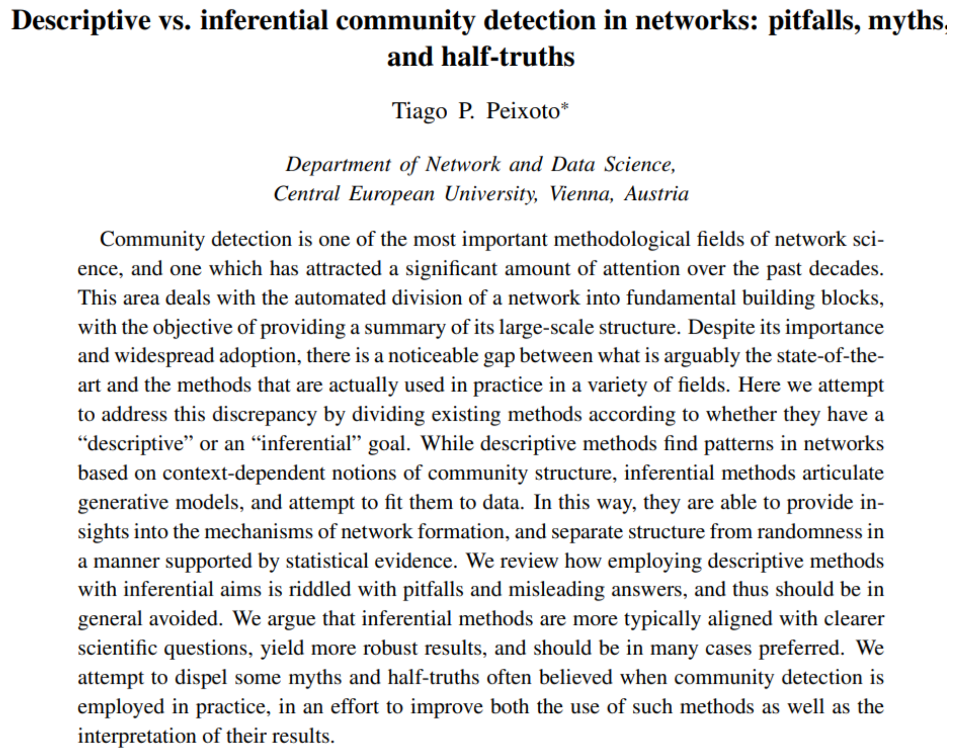
```
#makes a dual axis plot with modularity and number of communities vs
average degree
fig, ax1 = plt.subplots()
ax1.plot(average_k, modularities, 'b-')
ax1.set_xlabel('Average degree')
ax1.set_ylabel('Modularity', color='b')
ax1.tick_params('y', colors='b')
ax1.set_ylim(0, 0.5)

ax2 = ax1.twinx()
ax2.plot(average_k, n_communities, 'r-')
ax2.set_ylabel('Number of communities', color='r')
ax2.tick_params('y', colors='r')
ax2.set_ylim(0, 20)
plt.xlim(10, 60)
plt.title('Modularity of random graphs with N = 1000');
```

✓ 0.2s Python



DESCRIPTIVE VS INFERENTIAL METHODS

- Descriptive methods
  - Implicit or explicit definition of communities
  - Describes the patterns found algorithmically
  - Can produce spurious results
- Inferential methods
  - Starts from a generative, explicit network model
  - Infers how likely that the model could have created the data
  - Can find model parameters with **maximum likelihood estimation** (MLE)
  - Fitting to a model can offer *more insight* into the network than finding partitions



Descriptive vs. inferential community detection in networks: pitfalls, myths, and half-truths

Tiago P. Peixoto*

Department of Network and Data Science, Central European University, Vienna, Austria

Community detection is one of the most important methodological fields of network science, and one which has attracted a significant amount of attention over the past decades. This area deals with the automated division of a network into fundamental building blocks, with the objective of providing a summary of its large-scale structure. Despite its importance and widespread adoption, there is a noticeable gap between what is arguably the state-of-the-art and the methods that are actually used in practice in a variety of fields. Here we attempt to address this discrepancy by dividing existing methods according to whether they have a "descriptive" or an "inferential" goal. While descriptive methods find patterns in networks based on context-dependent notions of community structure, inferential methods articulate generative models, and attempt to fit them to data. In this way, they are able to provide insights into the mechanisms of network formation, and separate structure from randomness in a manner supported by statistical evidence. We review how employing descriptive methods with inferential aims is riddled with pitfalls and misleading answers, and thus should be in general avoided. We argue that inferential methods are more typically aligned with clearer scientific questions, yield more robust results, and should be in many cases preferred. We attempt to dispel some myths and half-truths often believed when community detection is employed in practice, in an effort to improve both the use of such methods as well as the interpretation of their results.

DESCRIPTIVE VS INFERENTIAL METHODS

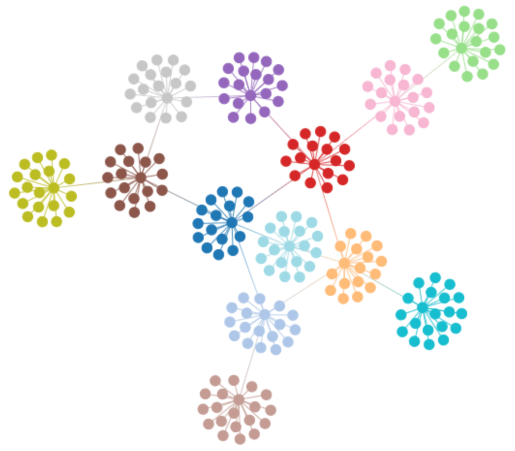
- Example: network with specific degree sequence
  - Descriptive methods: 13 communities
  - Inferential methods: no communities
    - Randomly wired nodes in one *block*

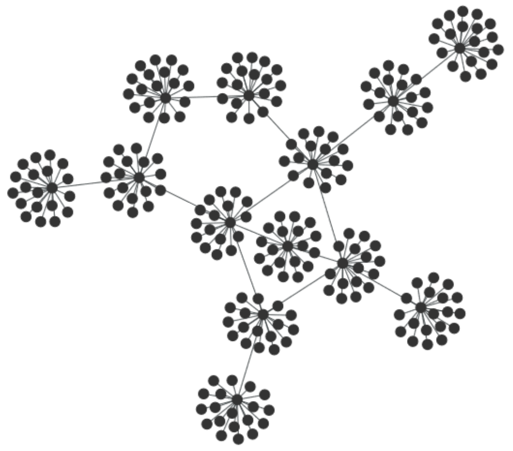
Litmus test: to infer or to describe?

Q: "Would the usefulness of our conclusions change if we learn, after obtaining the communities, that the network being analyzed is maximally random?"

If the answer is "yes," then an inferential approach is needed.

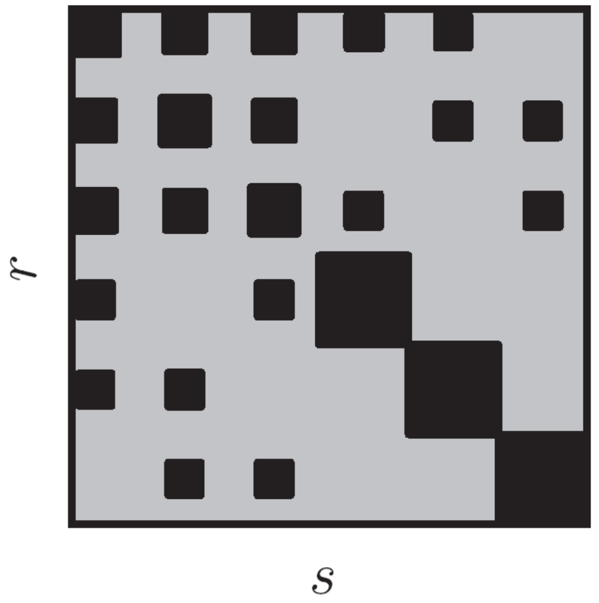
If the answer is "no," then an inferential approach is not required.

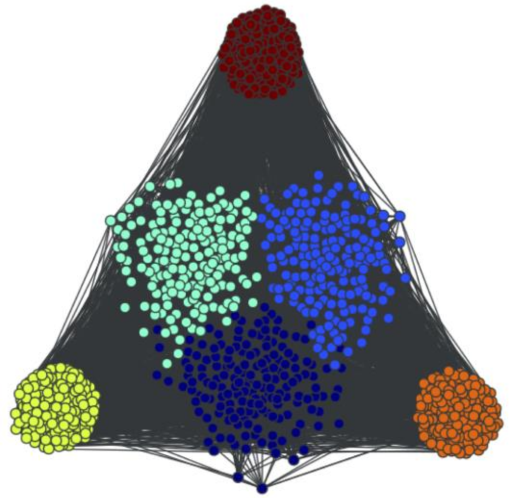

A network with 13 communities


A random network with a prescribed degree sequence, and no community structure.

DESCRIPTIVE VS INFERENTIAL METHODS

- **Stochastic Block Models**
  - Most common inferential method
  - **The idea:** nodes are divided into *blocks* and the probability that two nodes are connected is determined by the blocks to which they belong
  - q groups, g_i label of group of node i
  - Connection probability $p_{ij} = p_{g_i g_j}$
  - Stochastic block matrix: $q \times q$ matrix with the connection probabilities

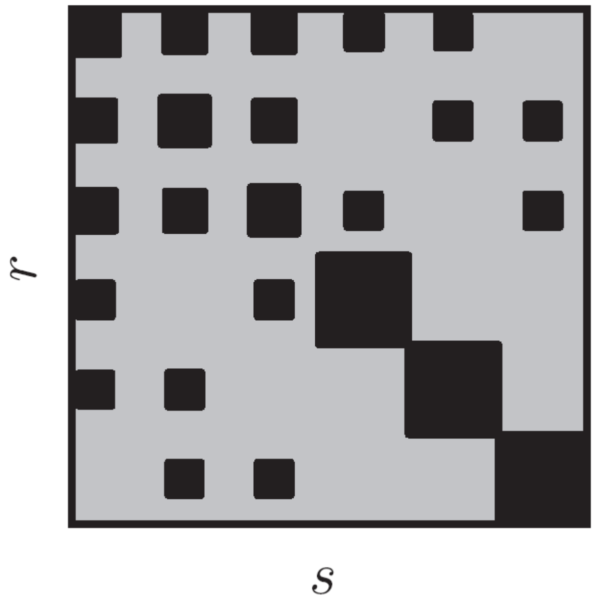


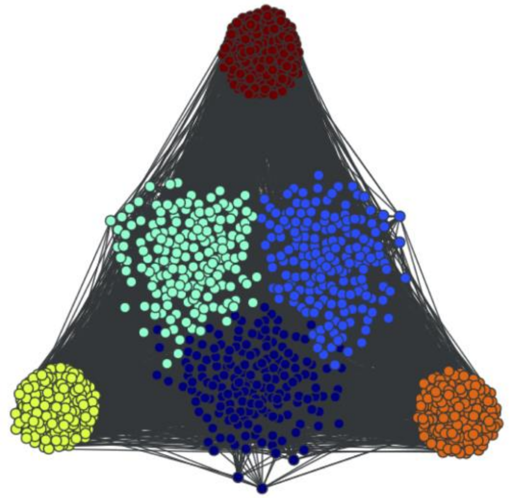


DESCRIPTIVE VS INFERENTIAL METHODS

- For groups $r \neq s$, if
  - $p_{rr} > p_{rs}$: **community structure**
  - $p_{rr} < p_{rs}$: **multipartite structure**
  - $\mathrm{q} = 2$ and $p_{11} \gg p_{12} \gg p_{22}$: **core-periphery structure**
  - $\mathrm{p_{rs}} = \mathrm{p}\ \forall r, s$: **random network**

- Operationalizing it
  - Given the model parameters $\boldsymbol{\theta}$, calculate the likelihood that it created the data.
    - In 1D: $Likelihood(\theta|x) = P_\theta(X = x)$
  - Use an algorithm to maximize the likelihood as a function of $\boldsymbol{\theta}$
  - Best partition comes from maximum value





DESCRIPTIVE VS INFERENTIAL METHODS

- Problem: SBM ignores degree heterogeneity
- Solution: add degree sequence to to SBM
  - **Degree-corrected SBM** (DCSBM)

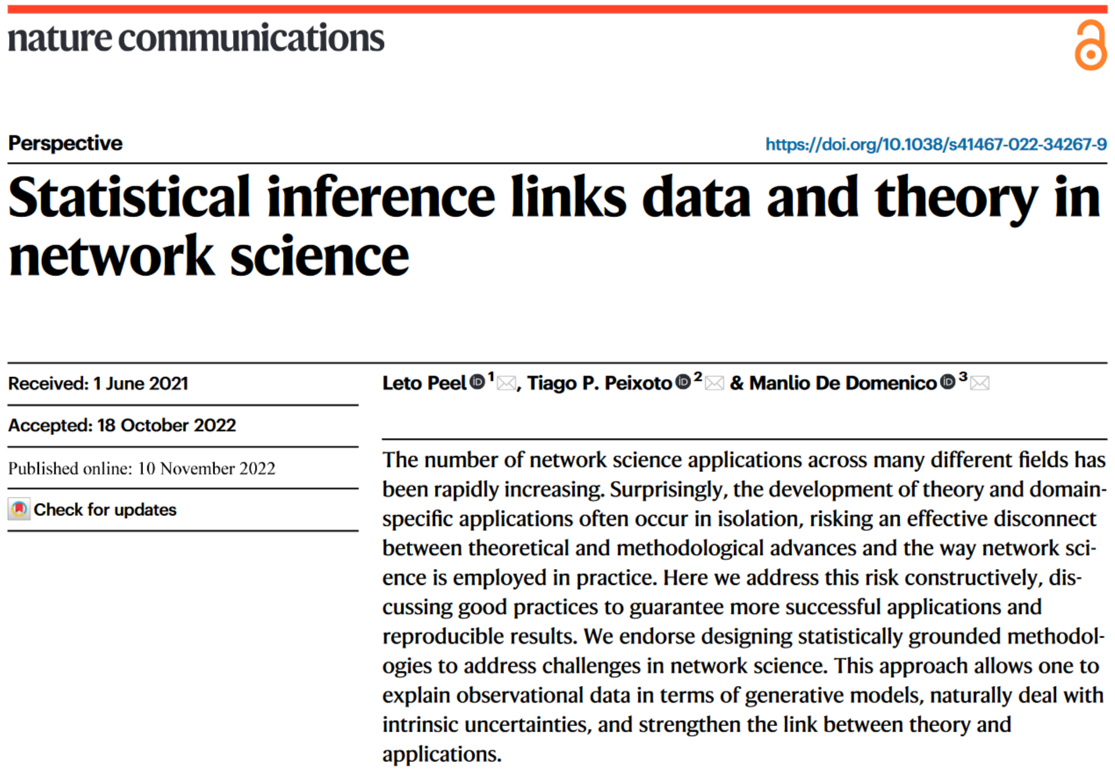
- Explicitly:
  - The probability that a network *G* is reproduced by the DCSBM based on a given partition *g* of *G*'s nodes into *q* groups is the **log-likelihood:**

$$\mathcal{L}(G|g) = \sum_{r,s=1}^{q} L_{rs} \log\left(\frac{L_{rs}}{k_r k_s}\right)$$

- Procedure:
  1. Start from random partition in q clusters
  2. Repeatedly move a node from one group to another, selecting at each step the move that will most increase the likelihood under the constraint that each node may be moved only once
  3. When all nodes have been moved, inspect the partitions through which the system passed from start to end of the procedure in step 2 and select the one with the highest likelihood
  4. Stop if likelihood cannot be increased

DESCRIPTIVE VS INFERENTIAL METHODS

- Limitations of SBMs
  - Need to input number of clusters
    - Scanning over it results in singletons
    - Array of techniques to estimate it
  - Greedy algorithm may get stuck at local minima
    - Usual set of solutions apply (multiple runs, random moves, etc)



nature communications

Perspective https://doi.org/10.1038/s41467-022-34267-9

Statistical inference links data and theory in network science

Received: 1 June 2021

Accepted: 18 October 2022

Published online: 10 November 2022

Check for updates

Leto Peel[1], Tiago P. Peixoto[2] & Manlio De Domenico[3]

The number of network science applications across many different fields has been rapidly increasing. Surprisingly, the development of theory and domain-specific applications often occur in isolation, risking an effective disconnect between theoretical and methodological advances and the way network science is employed in practice. Here we address this risk constructively, discussing good practices to guarantee more successful applications and reproducible results. We endorse designing statistically grounded methodologies to address challenges in network science. This approach allows one to explain observational data in terms of generative models, naturally deal with intrinsic uncertainties, and strengthen the link between theory and applications.

- Questions?