Universitat
Pompeu Fabra
Barcelona

# Lecture 14 · Communities

## Networks, Crowds and Markets

# What is coming next

1. What do we mean by a community?
2. Zachary's Karate Club
3. Hypotheses and definitions (H1–H4)
4. The Girvan–Newman algorithm
5. The Stochastic Block Model (SBM)

# Communities in graphs

# What are communities?

**Definition ( Informal )**

Groups of nodes with more connections inside than outside.

Examples:

- Social networks: circles of friends, political communities.
- Scientific collaboration: fields or subdisciplines.
- Biology: protein complexes in interaction networks.
- Infrastructure: airline networks with hubs and regional groups.

There is no single formal definition.

# Communities in Economics

**Trade Blocs:** Countries cluster into EU, NAFTA, ASEAN.

**Political Polarization:** Twitter users cluster into left vs. right.

**Firms:** Industry supply chains reveal modular communities.

- Detecting communities = identifying hidden structure in markets.

A famous example: Communication structure in Belgium.

# Zachary's Karate Club

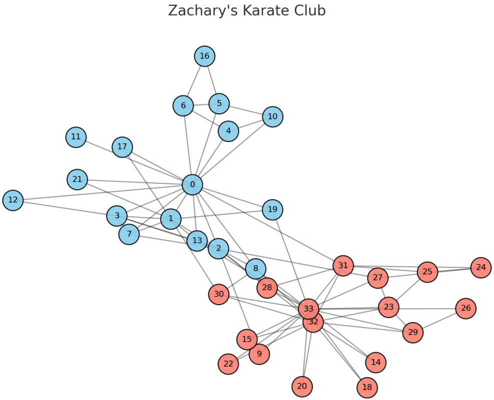
This is the first famous example of community structure in a network.

Initially analysed by Wayne W. Zachary, 1977.

Became the most classic network for community analysis.

- 34 members of a karate club, edges = friendships outside the club.
- Conflict between instructor ("Mr. Hi") and administrator ("John A") led to a split of the club.
- Based on the friendship network, predict how the club splits.
- Zachary's analysis correctly predicted all but one member's side.

# Karate Club Network

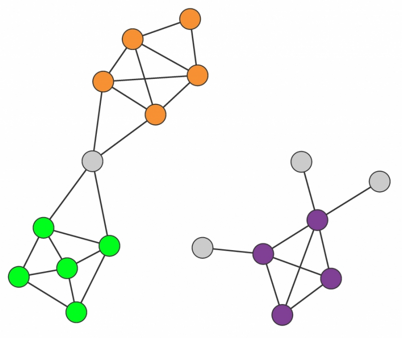



- Mr. Hi corresponds to node 0,
- John A corresponds to node 33.

# Communities: Defining principles

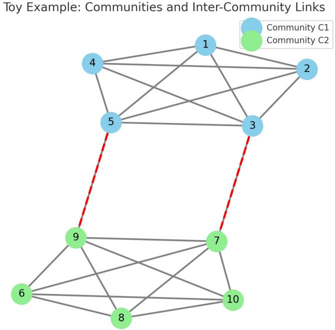
# Principle 1 – Fundamental

A network's community structure is encoded in its wiring diagram. That is, communities can in principle be discovered by looking only at the graph structure.



# Principle 2 – Connectedness & Density

A **community** should be a connected subgraph. Links inside a community should be denser than links going outside.



# Strong vs Weak Communities

$$\deg_C(v) = \#\{\text{edges from } v \text{ to nodes in } C\}$$
$$\deg_{\overline{C}}(v) = \#\{\text{edges from } v \text{ to nodes outside } C\}.$$

# Strong vs Weak Communities

$$\deg_C(v) = \#\{\text{edges from } v \text{ to nodes in } C\}$$
$$\deg_{\overline{C}}(v) = \#\{\text{edges from } v \text{ to nodes outside } C\}.$$

**Strong community (restrictive):** every node $v \in C$ satisfies

$$\deg_C(v) \; > \; \deg_{\overline{C}}(v).$$

# Strong vs Weak Communities

$$\deg_C(v) = \#\{\text{edges from } v \text{ to nodes in } C\}$$
$$\deg_{\overline{C}}(v) = \#\{\text{edges from } v \text{ to nodes outside } C\}.$$

**Strong community (restrictive):** every node $v \in C$ satisfies

$$\deg_C(v) \; > \; \deg_{\overline{C}}(v).$$

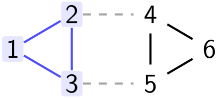
**Weak community:** total internal degree exceeds external degree:

$$\sum_{v \in C} \deg_C(v) \; > \; \sum_{v \in C} \deg_{\overline{C}}(v).$$

(the average in-community degree larger than out-community degree)

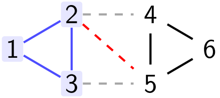
# Example · Strong vs Weak Community

Original graph (strong)



$C = \{1, 2, 3\}$ $\overline{C} = \{4, 5, 6\}$

After adding edge $(2, 5)$



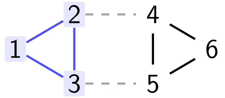
$C = \{1, 2, 3\}$ $\overline{C} = \{4, 5, 6\}$

| $v$ | $\deg_C(v)$ | $\deg_{\overline{C}}(v)$ | Strong? |
|---|---|---|---|
| 1 | 2 | 0 | ✓ |
| 2 | 2 | 1 | ✓ |
| 3 | 2 | 1 | ✓ |

$\Rightarrow$ $C$ is a strong community.

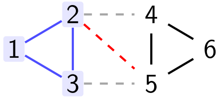
# Example · Strong vs Weak Community

Original graph (strong)




$C = \{1,2,3\}$ $\overline{C} = \{4,5,6\}$

After adding edge $(2,5)$




$C = \{1,2,3\}$ $\overline{C} = \{4,5,6\}$

| $v$ | $\deg_C(v)$ | $\deg_{\overline{C}}(v)$ | Strong? |
|---|---|---|---|
| 1 | 2 | 0 | ✓ |
| 2 | 2 | 1 | ✓ |
| 3 | 2 | 1 | ✓ |

$\Rightarrow$ $C$ is a strong community.

Now add edge $(2,5)$: $\deg_C(2) = 2$, $\deg_{\overline{C}}(2) = 2 \Rightarrow$ not strong. But

$$\sum_{v \in C} \deg_C(v) = 6, \qquad \sum_{v \in C} \deg_{\overline{C}}(v) = 4,$$

so $C$ is still a weak community.

# Principle 3 – Random Baseline

Erdős–Rényi graphs do not have meaningful community structure.

- Communities are detected when the observed structure **deviates significantly from total randomness**.
- Useful for benchmarking algorithms for community detection.

# Modularity

# Modularity Maximization

Connections *within communities* denser than expected by random chance.

**Measured by modularity (Newman-Girvan 2004):**

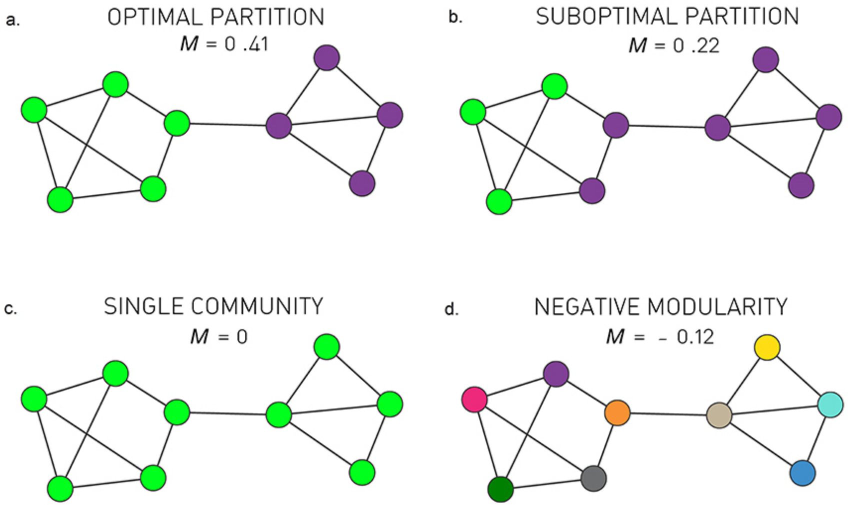
$$M = \frac{1}{2L}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2L}\right)\delta(c_i, c_j),$$

where

- $A_{ij}$ - adjacency matrix (1 if edge $i$–$j$ exists, else 0),
- $k_i$ - degree of node $i$, $2L = \sum_i k_i$ (total edge ends),
- $\delta(c_i, c_j) = 1$ if $i, j$ lie in the same community, 0 otherwise.

$M$ compares the *observed* edges ($A_{ij}$) to what we would *expect at random* ($k_i k_j / 2L$; see configuration model).

# The Girvan-Newman Algorithm



# Identifying communities

# Community detection via hierarchical clustering

Find a computationally efficient community detection procedure.

Let $S = [\delta_{ij}]$ be a similarity matrix:

- $S$ symmetric;
- $s_{ij} \geq 0$ for all $i \neq j$.

If $i, j$ are "close", $s_{ij}$ is higher.

- Build a *similarity matrix* $s_{ij}$ from the network.
- Iteratively group (or split) nodes using $s_{ij}$.
- Output is a dendrogram; cutting it gives a partition.

# An agglomerative algorithm

# Ravasz agglomerative algorithm

Let $B_i := \{j : d(i,j) \leq 1\}$. Let $A$ be the adjacency matrix.

We define node similarity using the *topological overlap*:

$$s_{ij} \;=\; \frac{|B_i \cap B_j|}{\min\{|B_i|, |B_j|\}} \;\in\; [0,1], \qquad (i \neq j).$$

- $s_{ij} = 0$ iff $i, j$ are not connected and they share no neighbors.
- $s_{ij} = 1$ iff $B_i \subseteq B_j$ or $B_j \subseteq B_j$.

# Ravasz agglomerative algorithm

Let $B_i := \{j : d(i,j) \leq 1\}$. Let $A$ be the adjacency matrix.

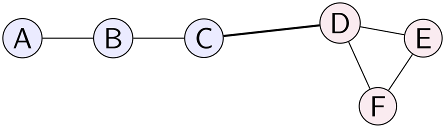
We define node similarity using the *topological overlap*:

$$s_{ij} \;=\; \frac{|B_i \cap B_j|}{\min\{|B_i|, |B_j|\}} \;\in\; [0,1], \qquad (i \neq j).$$

- $s_{ij} = 0$ iff $i, j$ are not connected and they share no neighbors.
- $s_{ij} = 1$ iff $B_i \subseteq B_j$ or $B_j \subseteq B_j$.

Many authors subtract $A_{ij}$ in both the nominator and denominator to slightly down-weight the direct edge when $i$ and $j$ are linked ($\tilde{s}_{ij}$).

# Toy example




| Pair | $\lvert B_i \rvert$ | $\lvert B_j \rvert$ | $\lvert B_i \cap B_j \rvert$ | $s_{ij}$ | $\tilde{s}_{ij}$ | $A_{ij}$ |
|---|---|---|---|---|---|---|
| A–B | 2 | 3 | 2 | 1.00 | 1.00 | 1 |
| B–C | 3 | 3 | 2 | 0.67 | 0.50 | 1 |
| C–D (bridge) | 3 | 4 | 2 | 0.67 | 0.33 | 1 |
| D–E | 4 | 3 | 3 | 1.00 | 1.00 | 1 |
| D–F | 4 | 3 | 3 | 1.00 | 1.00 | 1 |
| E–F | 3 | 3 | 3 | 1.00 | 1.00 | 1 |
| A–C | 2 | 3 | 1 | 0.50 | 0.50 | 0 |
| B–D | 3 | 4 | 1 | 0.33 | 0.33 | 0 |
| C–E | 3 | 3 | 1 | 0.33 | 0.33 | 0 |
| A–D | 2 | 4 | 0 | 0.00 | 0.00 | 0 |
| A–E | 2 | 3 | 0 | 0.00 | 0.00 | 0 |
| A–F | 2 | 3 | 0 | 0.00 | 0.00 | 0 |

# Ravasz agglomerative algorithm: hierarchical clustering

We apply standard hierarchical clustering to $S$.

**Algorithm.**

1. Compute the similarity matrix $s_{ij} = \frac{|B_i \cap B_j|}{\min(|B_i|, |B_j|)}$ for $i, j$.
2. Treat each node as a separate cluster.

# Ravasz agglomerative algorithm: hierarchical clustering

We apply standard hierarchical clustering to $S$.

**Algorithm.**

1. Compute the similarity matrix $s_{ij} = \frac{|B_i \cap B_j|}{\min(|B_i|,|B_j|)}$ for $i, j$.
2. Treat each node as a separate cluster.
3. Find the two clusters with the largest average pairwise $s_{ij}$.
4. Merge them into a new cluster. Compute similarities between this new cluster and every other cluster using a chosen linkage rule:

$$s_{A,B} = \begin{cases} \max_{i \in A, j \in B} s_{ij}, & \text{(single linkage)}, \\ \min_{i \in A, j \in B} s_{ij}, & \text{(complete linkage)}, \\ \text{average}_{i \in A, j \in B} s_{ij}, & \text{(average linkage)}. \end{cases}$$

# Ravasz agglomerative algorithm: hierarchical clustering

We apply standard hierarchical clustering to $S$.

**Algorithm.**

1. Compute the similarity matrix $s_{ij} = \frac{|B_i \cap B_j|}{\min(|B_i|,|B_j|)}$ for $i, j$.
2. Treat each node as a separate cluster.
3. Find the two clusters with the largest average pairwise $s_{ij}$.
4. Merge them into a new cluster. Compute similarities between this new cluster and every other cluster using a chosen linkage rule:

$$s_{A,B} = \begin{cases} \max_{i \in A, j \in B} s_{ij}, & \text{(single linkage)}, \\ \min_{i \in A, j \in B} s_{ij}, & \text{(complete linkage)}, \\ \text{average}_{i \in A, j \in B} s_{ij}, & \text{(average linkage)}. \end{cases}$$

5. Repeat Step 3–4 until all nodes are merged into one cluster.

**Output.** The sequence of merges defines a *dendrogram*. Cutting it at a chosen similarity threshold yields the community partition.

# Average-linkage: merge order and dendrogram

**Cluster similarities $s(A, B)$ (averages):**

Within right triangle: $s(\{D, E\}, \{F\}) = \frac{s_{DF} + s_{EF}}{2} = 1$,
Within left path: $s(\{A\}, \{B\}) = s_{AB} = 1$,
Left vs. $C$: $s(\{A, B\}, \{C\}) = \frac{s_{AC} + s_{BC}}{2} = \frac{0.50+0.67}{2} = 0.583$,
$C$ vs. triangle: $s(\{C\}, \{D, E, F\}) = \frac{2/3+1/3+1/3}{3} = 4/9 \approx 0.444$,
Left vs. triangle: $s(\{A, B\}, \{D, E, F\}) = \frac{0+0+0+1/3+0+0}{6} = 1/18 \approx 0.056$,
Final: $s(\{A, B, C\}, \{D, E, F\}) = \frac{1.666...}{9} \approx 0.185$.

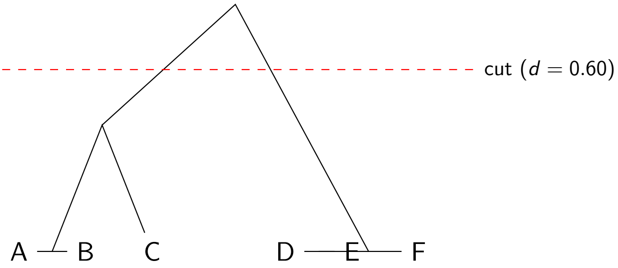
**Merge order (average linkage):**

$$\{D\}, \{E\}, \{F\} \rightarrow \{D, E, F\} \; (s = 1), \qquad \{A\}, \{B\} \rightarrow \{A, B\} \; (s = 1),$$

$$\{A, B\}, \{C\} \rightarrow \{A, B, C\} \; (s = 0.583), \quad \{A, B, C\}, \{D, E, F\} \rightarrow \text{all} \; (s = 0.185).$$

# Resulting dendrogram

**Dendrogram (heights $d = 1 - s$):**




Cutting with $0.417 < d_{\text{cut}} < 0.815$ yields two communities: $\{A, B, C\}$ and $\{D, E, F\}$.
Cutting below 0.417 (e.g., $d = 0.30$) gives three: $\{A, B\}$, $\{C\}$, $\{D, E, F\}$.

Modularity check: $M(\{A, B, C\}, \{D, E, F\}) \approx 0.32$, $M(\{A, B\}, \{C\}, \{D, E, F\}) \approx 0.24 \Rightarrow$ two-community cut gives the highest modularity.

# A divisive algorithm

# Divisive community detection: Girvan–Newman algorithm

**Goal**: detect communities by removing *bridges* between them.

For each edge $(i, j)$, define its edge betweenness

$$b_{ij} = \sum_{s \neq t} \frac{\sigma_{st}(i, j)}{\sigma_{st}},$$

where $\sigma_{st}$ is the number of shortest paths between nodes $s$ and $t$, and $\sigma_{st}(i, j)$ counts how many of them go through edge $(i, j)$.

- Edges with high $b_{ij}$ tend to connect different communities.

# Divisive community detection: Girvan–Newman algorithm

**Goal**: detect communities by removing *bridges* between them.

For each edge $(i, j)$, define its edge betweenness

$$b_{ij} \;=\; \sum_{s \neq t} \frac{\sigma_{st}(i,j)}{\sigma_{st}},$$

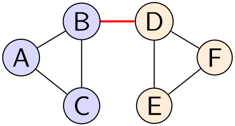
where $\sigma_{st}$ is the number of shortest paths between nodes $s$ and $t$, and $\sigma_{st}(i,j)$ counts how many of them go through edge $(i, j)$.

- Edges with high $b_{ij}$ tend to connect different communities.

**Algorithm (top–down / divisive):**

1. Compute $b_{ij}$ for all edges.
2. Remove the edge with the highest $b_{ij}$.
3. Recompute betweenness on the updated graph.
4. Repeat until the graph splits.

# Girvan–Newman: toy example (manual computation)




**Step 1. Compute edge betweenness**

- Shortest paths inside each triangle use only local edges.
- All shortest paths between left and right triangles must go through $(B, D)$.
- $\Rightarrow$ edge $(B, D)$ has the highest betweenness.

**Step 2. Remove** $(B, D)$
Graph splits into two dense components:

$$\{A, B, C\}, \qquad \{D, E, F\}.$$

**Communities recovered!**

# Girvan–Newman: selecting the best split

As in the Ravasz algorithm, the splits define a dendrogram.

The tree is constructed from top to bottom.

Cutting it defines a split.

Which split is best? Again, we can decide based on modularity.

# Agglomerative vs. Divisive (summary)

| | Agglomerative (Ravasz) | Divisive (Girvan–Newman) |
|---|---|---|
| Direction | Bottom–up merges | Top–down edge removals |
| Input | Node similarity $s_{ij}$ | Edge centrality $s_{ij}$ |
| Linkage | Single / complete / average | Not applicable |
| Output | Dendrogram of merges | Dendrogram of splits |
| Cost | $O(N^2)$ typical | $O(LN)$–$O(N^3)$ (implementation) |

Both yield a dendrogram; choose the cut to get communities.

# Stochastic Block Model

# The Stochastic Block Model (SBM)

The SBM provides a simple generative model for networks with community structure.

**Definition (Stochastic Block Model)**

- $N$ nodes, each assigned to one of $K$ groups: $g_i \in \{1, \ldots, K\}$.
- Edge between $i$ and $j$ appears independently with probability

$$\Pr[(i,j) \in E] = p_{g_i, g_j}.$$

- The $K \times K$ matrix $P = (p_{ab})$ specifies connection probabilities between groups.

# The Stochastic Block Model (SBM)

The SBM provides a simple generative model for networks with community structure.

**Definition (Stochastic Block Model)**

- $N$ nodes, each assigned to one of $K$ groups: $g_i \in \{1, \ldots, K\}$.
- Edge between $i$ and $j$ appears independently with probability

$$\Pr[(i,j) \in E] = p_{g_i, g_j}.$$

- The $K \times K$ matrix $P = (p_{ab})$ specifies connection probabilities between groups.

**Applications:**

- Benchmark for testing community detection algorithms.
- Statistical inference: given the observed network, estimate group labels $\{g_i\}$ and/or $P$.

# SBM (2 groups): signal in the spectrum

**Model.** Two communities of sizes $N_1, N_2$ ($N = N_1 + N_2$).

$$\Pr[(i,j) \in E] = \begin{cases} p, & g_i = g_j, \\ q, & g_i \neq g_j. \end{cases} \qquad \text{with } p > q.$$

Let $A$ be the adjacency matrix (random) and $B = \mathbb{E}[A]$ its expectation.

**Block structure of** $B$:

$$B = \begin{pmatrix} p\mathbf{1}_{N_1 \times N_1} & q\mathbf{1}_{N_1 \times N_2} \\ q\mathbf{1}_{N_2 \times N_1} & p\mathbf{1}_{N_2 \times N_2} \end{pmatrix},$$

a rank-2 matrix.

**Key eigenvectors:**

- $\mathbf{1}$ (all-ones) $\Rightarrow$ "size/degree" direction.
- Community indicator $s$ ($s_i = +1$ if $g_i = 1$, $s_i = -1$ if $g_i = 2$).

**Eigenvalues (balanced case $N_1 = N_2 = N/2$):**

$$\lambda_1(B) \approx \tfrac{N}{2}\big(p+q\big),$$

$$\lambda_2(B) \approx \tfrac{N}{2}\big(p-q\big).$$

$\lambda_2(B)$ carries the community signal:

$$\underbrace{p-q}_{\text{separation}} \quad \uparrow \Rightarrow \text{ easier recovery.}$$

**Idea:** If we compute the leading eigenvectors of the *observed* $A$, they should align with those of $B$ and reveal the partition.

## Note

Spectral clustering works because $A$ concentrates around its block-structured mean and the "community" eigenvector survives the noise when separation is large enough.

# Spectral Clustering for Communities (practical)

**Input:** Graph $G = (V, E)$ with $|V| = N$ and number of groups $K$.

**Matrix choice:**

- *Adjacency* $A$ for reasonably dense graphs.
- *(Normalized) Laplacian* $L_{\mathrm{sym}} = I - D^{-1/2} A D^{-1/2}$ for sparse graphs or degree heterogeneity.

**Algorithm:**

1. Compute $K$ eigenvectors:
   - top-$K$ of $A$, or
   - bottom-$K$ (smallest) of $L_{\mathrm{sym}}$.
2. Stack them into $V \in \mathbb{R}^{N \times K}$ (row $i$ = embedding of node $i$). (Option: row-normalize $V$.)
3. Run $k$-means on rows of $V$ to obtain labels $\hat{g}_1, \ldots, \hat{g}_N$.

**Model selection tips:**

- Choose $K$ via eigengap heuristic or cross-validation on modularity.
- Use degree-corrected variants (e.g., regularized Laplacian) if degrees are very skewed.

# Hands-on: SBM + Spectral Clustering in NetworkX

```
import networkx as nx
import numpy as np
from sklearn.cluster import KMeans
from scipy.sparse.linalg import eigsh

# 1) Generate a 2-block SBM
N1, N2 = 80, 70
sizes = [N1, N2]
pin, pout = 0.12, 0.02
P = [[pin, pout], [pout, pin]]
G = nx.stochastic_block_model(sizes, P, seed=7)
A = nx.to_scipy_sparse_array(G, format="csr", dtype=float)
N = A.shape[0]

# 2) Use normalized Laplacian for robustness on sparse graphs
L = nx.normalized_laplacian_matrix(G)      # scipy sparse CSR
# take K=2 smallest eigenpairs of L (skip the zero vector if graph disconn
K = 2
vals, vecs = eigsh(L, k=K, which='SM')     # smallest magnitude eigenvalue
```

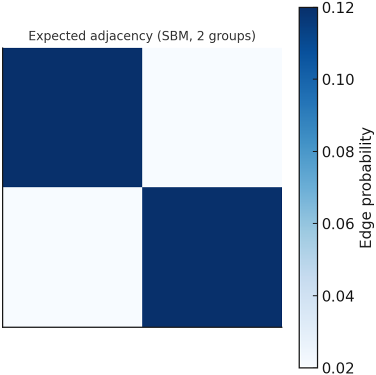
```
# 3) Row-embed nodes and k-means
X = vecs                                          # N x K embedding
X = X / (np.linalg.norm(X, axis=1, keepdims=True) + 1e-12)  # row-normaliz
labels = KMeans(n_clusters=2, n_init=20, random_state=7).fit_predict(X)

# 4) Compare to planted labels (0 for first block, 1 for second)
true = np.array([0]*N1 + [1]*N2)
acc = max((labels == true).mean(), (1-labels == true).mean())
print(f"Accuracy vs planted partition: {acc:.3f}")
```

**Notes:** For very sparse graphs, prefer Laplacian; for denser ones, adjacency often suffices. Replace $K$ by eigengap-based choice when unknown.

# What the eigenvectors "see": a geometric picture

- $B = \mathbb{E}[A]$ has two constant blocks.
- Its top eigenvector is roughly constant on all nodes (size/degree axis).
- Its second eigenvector is roughly $+c$ on group 1 and $-c$ on group 2.
- The observed $A$ is a noisy version of $B$; eigenvectors of $A$ wobble around those of $B$.




Expected adjacency $B$ (2 blocks)

**Consequence.** Embedding nodes by the top eigenvectors separates communities along the "$+/-$" direction; $k$-means then recovers the groups.

# Open Questions

- Do all networks really have communities?
- Are "communities" well-defined or context-dependent?
- Must all nodes belong to some community?
- How do communities influence dynamics (diffusion, epidemics)?