Supplemental Technical Appendix to "Block Dense Weighted Networks with Augmented Degree Correction"

June 1, 2022

This supplement discusses several topics related to the main paper, and can be treated as an extended Appendix. Appendix A proposes an alternative estimation approach for Normal LSM and LSM models using Nonnegative Matrix Factorization. Appendix B contains two concentration inequalities for LSM networks using this alternative estimation approach, followed by auxiliary lemmas. Appendix C discusses estimation while accounting for certain diagonal 0's for within community subnetworks, along with how to adapt the estimation procedures when there are repeated values. Appendix D.1 explains how modularity may fail to characterize the kinds of communities discussed in the main paper. Appendix D.2 introduces a greedy, agglomerative approach to community detection by trying to maximize the measure L. Appendix D.3 considers community detection as a spatial clustering problem, and Appendix D.4 includes an algorithm built upon spectral clustering to try to maximize L. Appendix D.5 shows that the real parts of the first several eigenvectors of a normalized version of the network also appear to capture community information. Appendix E displays additional simulated networks and results. Finally, Appendix F extends both H-functions and the kinds of "errors" which may be observed in LSMs or NSMs.

A Estimation for Normal LSM and LSM

A.1 Estimation for Normal LSM

Recall from Section 2 that a Normal LSM has the form

$$\begin{aligned} f(W_{uv}) &= \gamma + \alpha\Phi_1^{-1}(\Psi_u) + \beta\Phi_1^{-1}(\Psi_v) + \sigma\epsilon_{uv} \\ &= \gamma + \alpha Z_u + \beta Z_v + \sigma\epsilon_{uv}. \end{aligned}$$

It is for now assumed that no transformation is needed, that is, $f(W_{uv}) = W_{uv}$. We also assume for simplicity that $\gamma = 0$. Then, the model (12) can be expressed after exponentiation as

$$e^{W_{uv}} = e^{\alpha Z_u} e^{\beta Z_v} e^{\sigma\epsilon_{uv}} = e^{\alpha Z_u} e^{\beta Z_v} e^{\frac{\sigma^2}{2}} \left(\frac{e^{\sigma\epsilon_{uv}}}{\mathbb{E}e^{\sigma\epsilon_{uv}}} \right), \tag{A.1}$$

where the last term within the parentheses has an expected value of 1. The structure of (A.1) enables the use of rank-one Nonnegative Matrix Factorization (NMF) to estimate the parameters of interest. NMF approximates the matrix represented by $e^{W_{uv}}$ as the decomposition ab' where $a = (a_u)$ and $b = (b_v)$ are column vectors with positive entries. Taking the log of this approximation yields the approximate identity

$$\log(a_u) + \log(b_v) \approx \alpha Z_u + \beta Z_v + \frac{\sigma^2}{2}, \tag{A.2}$$

which suggests the following estimators for the model parameters of interest:

$$\widehat{\alpha} = \mathrm{SD}(\log(a)), \quad \widehat{\beta} = \mathrm{SD}(\log(b)),$$

$$\widehat{Z}_u = \frac{\log(a_u) - \overline{\log(a)}}{\widehat{\alpha}}, \quad \widehat{\Psi}_u = \Phi_1(\widehat{Z}_u), \quad \widehat{Z}_v = \frac{\log(b_v) - \overline{\log(b)}}{\widehat{\beta}}, \quad \widehat{\Psi}_v = \Phi_1(\widehat{Z}_v),$$

where SD stands for the standard deviation, and $\overline{\log(x)}$ indicates the sample mean of $\log(x)$. In light of (12), we also set $\widehat{\sigma} = \mathrm{SD}(W_{uv} - \widehat{\alpha}\widehat{Z}_u - \widehat{\beta}\widehat{Z}_v)$. An adjustment for subnetworks where all nodes are in the same community is given in Appendix C below.

A.2 Extension to LSM

The difference between (12) and (7), ignoring extra subscripts, is that rather than having standard normal random variables ascribed to each node, (7) includes random variables $h_1(\Psi_u)$ and $h_2(\Psi_v)$ with possibly different forms, albeit with identical first 2 moments. The procedure described in Appendix A.1 will still apply with one exception. The relation (A.2) cannot directly estimate $\widehat{Z}_u$ or $\widehat{Z}_v$, but rather $\widehat{h_1(\Psi_u)}$ and $\widehat{h_2(\Psi_v)}$. After getting these estimates, a distribution can be fit to the data points while assuming that Ψ_u and Ψ_v are truly distributed uniformly over the unit interval. The best fitting distribution can then be inverted to estimate $\widehat{\Psi}_u$ and $\widehat{\Psi}_v$. Finally, the parameter σ is estimated to be the standard deviation of $W_{uv} - \widehat{\alpha}\widehat{h_1(\Psi_u)} - \widehat{\beta}\widehat{h_2(\Psi_v)}$.

B Concentration inequality for Normal LSM

In Section 4, we introduced methods for estimating sociability parameters for certain kinds of network generating models. It would be helpful to ensure these estimates are actually capturing the underlying "error-free" network generating mechanisms, what we refer to as the network's SC. As there are several introduced models, the accuracy of the estimation results may depend on the specific functional forms of a given network. We present below a result on estimation accuracy of the SC for a Normal LSM network (12). The estimation procedure is formulated through theoretical means; it remains to be seen how the procedure compares to the practical estimation approach taken in Sections 4 and 6. The proof of the result adapts ideas from Noroozi et al. (2021).

In the theorems that follow, we assume there are true communities (with number of communities K_*) but the estimates $\hat{P}$ do not necessarily use these true communities, but rather chooses a partition that minimizes the resulting estimate's Frobenius distance to the observed matrix with an added penalty term based on the number of nodes and estimated number of communities, $\hat{K}$. The penalty term is necessary in these proofs, because if no penalty were included, we could simply let each node be its own community (setting $\hat{K} = n$), then perfectly find a formula for each single edge weight that connects each pair of communities/nodes. Having each node serve as a different community defeats the purpose of clustering nodes into communities. If we did so, we would have 2 free parameters to fit each edge. Though this procedure would make the Frobenius norm 0, this would surely be overfitting.

Including the penalty leads to a different clustering more in the spirit of the models introduced in the paper, which still yields a concentration inequality. We do not use the measure of clustering quality discussed in the paper as a means of estimating $\hat{P}$, instead comparing the minimizing estimate to the observed network. Establishing the theoretical consistency of the community detection approach, as well as the non-asymptotic upper bound on the proportion of the misclassified nodes are outside the scope of this paper, but are topics which merit further investigation.

B.1 Concentration for Normal LSM with known σ^2_{max}

Theorem B.1 *Let A be a Normal LSM network such that every edge weight is generated as in (12), and P_* be the network such that each edge weight has the same generating process as the corresponding edge weight in A, but with every σ value set to 0. In particular, all $\{\epsilon_{uv}\}$ are i.i.d. standard normal random variables that are independent of all other model parameters, and all $\{\Psi\} \sim U(0,1)$. Also let $\hat{P}$ be the estimated network (of the form (12) with $\sigma = 0$) induced by the clustering of nodes that minimizes*

$$||A - \hat{P}||_F^2 + Pen(n, \hat{K}), \tag{B.1}$$

where $\hat{K}$ is the number of different communities in this "best" clustering, and

$$Pen(n, K) = 6\sigma^2_{max}\left(C_1 nK + C_2 K^2 \log(n)\right) + (6C_3 + \frac{2}{c})\sigma^2_{max}(\log(n) + n\log(K)), \tag{B.2}$$

where n is the number of nodes in the network, K is the number of communities in the clustering, c is some constant such that $0 < c < 1$, $\sigma^2_{max} < \infty$ is the largest variance parameter of any generating function for edges in network A, and C_1 and C_2 are as given in Lemma B.1. Then, for some constant C_3 and any $t > 0$,

$$\mathbb{P}\left(||\hat{P} - P_*||_F^2 \leq (1-c)^{-1} Pen(n, K_*) + \frac{C_3}{c}\sigma^2_{max} t\right) \geq 1 - 3e^{-t}. \tag{B.3}$$

Dividing by the number of edges n^2, the distance from the estimate to the truth continues to shrink on a per edge basis. As an immediate consequence of the theorem, if $K\sqrt{log(n)}/n \to 0$, the estimates are consistent.

Proof: It is first useful to define the $\hat{P}$ induced by a particular clustering. Assuming we have clustered all nodes of A into $\hat{K}$ estimated communities, we denote the subnetwork including only edges connecting nodes in estimated community $\hat{k}$ to nodes in estimated community $\hat{l}$ as $A^{(\hat{k},\hat{l})}$. Letting $\mathbb{1}_n$ be a length n column vector of 1's, define $\Pi(A^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})$ as the projection of $A^{(\hat{k},\hat{l})}$ of the form (12) with $\sigma = 0$ which minimizes the Frobenius norm to the observed subnetwork, which can be written as

$$\Pi(A^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})}) = \hat{\gamma}^{(\hat{k},\hat{l})}\mathbb{1}_{n_{\hat{k}}}\mathbb{1}'_{n_{\hat{l}}} + \hat{\alpha}^{(\hat{k},\hat{l})}\hat{Z}^{(\hat{k},\hat{l})}\mathbb{1}'_{n_{\hat{l}}} + \hat{\beta}^{(\hat{k},\hat{l})}\mathbb{1}_{n_{\hat{k}}}\hat{Z}^{(\hat{l},\hat{k})\prime}, \tag{B.4}$$

such that

$$\hat{\gamma}^{(\hat{k},\hat{l})}, \hat{\alpha}^{(\hat{k},\hat{l})}, \hat{\beta}^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{l},\hat{k})} = \underset{\gamma,\alpha,\beta,Z_1,Z_2}{\text{argmin}} ||A^{(\hat{k},\hat{l})} - (\gamma\mathbb{1}_{n_{\hat{k}}}\mathbb{1}'_{n_{\hat{l}}} + \alpha Z_1\mathbb{1}'_{n_{\hat{l}}} + \beta\mathbb{1}_{n_{\hat{k}}}Z'_2)||_F. \tag{B.5}$$

For identifiability purposes, we also mandate the following constraints:

$$\sum \hat{Z}^{(\hat{k},\hat{l})} = 0, \qquad \sum \hat{Z}^{(\hat{l},\hat{k})} = 0, \qquad SD(\hat{Z}_1) = 1, \qquad SD(\hat{Z}_2) = 1, \qquad \hat{\alpha}^{(\hat{k},\hat{l})} \geq 0, \qquad \hat{\beta}^{(\hat{l},\hat{k})} \geq 0. \tag{B.6}$$

For this proof, we need not explicitly express every component quantity in (B.5), but note that $\Pi(A^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})$ has rank ≤ 3 by construction. In the between estimated community subnetworks where $\hat{k} \neq \hat{l}$, the estimates $\hat{P}^{(\hat{k},\hat{l})} = \Pi(A^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})$.

However, for within estimated community subnetworks where $\hat{k} = \hat{l}$, we must define $\Pi(A^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})$ slightly differently, so as to ignore any influence from certain 0's on the diagonal of $A^{(\hat{k},\hat{k})}$. In that case, we keep the pertinent constraints while modifying (B.4) and (B.5) as follows:

$$\Pi(A^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})}) = \hat{\gamma}^{(\hat{k},\hat{k})} \mathbb{1}_{n_{\hat{k}}} \mathbb{1}'_{n_{\hat{k}}} + \hat{\alpha}^{(\hat{k},\hat{k})} \hat{Z}^{(\hat{k},\hat{k})} \mathbb{1}'_{n_{\hat{k}}} + \hat{\alpha}^{(\hat{k},\hat{k})} \mathbb{1}_{n_{\hat{k}}} \hat{Z}^{(\hat{k},\hat{k})\prime}, \tag{B.7}$$

such that

$$\hat{\gamma}^{(\hat{k},\hat{k})}, \hat{\alpha}^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k},\hat{k})} = \underset{\gamma,\alpha,\beta,Z}{\text{argmin}} || \big(A^{(\hat{k},\hat{k})} - (\gamma \mathbb{1}_{n_{\hat{k}}} \mathbb{1}'_{n_{\hat{k}}} + \alpha Z \mathbb{1}'_{n_{\hat{k}}} + \alpha \mathbb{1}_{n_{\hat{k}}} Z') \big)_{u>v} ||_F. \tag{B.8}$$

This is the projection which minimizes the Frobenius norm to the original subnetwork only with respect to off-diagonal entries, i.e. where $u \neq v$. $\Pi(A^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})$ also has rank ≤ 3, but this projection does not necessarily have 0's on the diagonal. This gives rise to within community edge estimates, $\hat{P}^{(\hat{k},\hat{k})}$, which is equal to $\Pi(A^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})$ in the off diagonal entries, but replaces the diagonal entries with 0's (which are correct by construction). We also represent this estimate as $\Pi_0(A^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})$, where the 0 subscript indicates the diagonals are forced to 0. $\hat{P}^{(\hat{k},\hat{k})}$ is not a projection of $A^{(\hat{k},\hat{k})}$, but is closer to $A^{(\hat{k},\hat{k})}$ in Frobenius norm than $\Pi(A^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})$, which is itself the best projection of $A^{(\hat{k},\hat{k})}$ with respect to the Frobenius norm for off diagonal entries.

The projections defined on the observed network and the estimated communities differ from the projections we define on P_* and the estimated communities. In defining $\Pi(P_*^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})$, or $\Pi_0(P_*^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})$, we treat the $\hat{Z}$ values as fixed, using the estimated values from $\Pi(A^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})$, and only allowing $\hat{\gamma}^{(\hat{k},\hat{l})}, \hat{\alpha}^{(\hat{k},\hat{l})}$, and $\hat{\beta}^{(\hat{k},\hat{l})}$ values to vary.

With these preliminaries in place, we follow the proof of Theorem 1 in Noroozi et al. (2021). By assumption,

$$||A - \hat{P}||_F^2 + Pen(n, \hat{K}) \leq ||A - P_*||_F^2 + Pen(n, K_*).$$

Letting

$$\Xi = A - P_*, \tag{B.9}$$

writing $Tr(M)$ for the trace of matrix M, and assuming the network has been rearranged into blocks by estimated communities, adding and subtracting P_* within $||A - \hat{P}||_F^2$ on the left-hand side gives

$$||\hat{P} - P_*||_F^2 \leq 2Tr(\Xi'(\hat{P} - P_*)) + Pen(n, K_*) - Pen(n, \hat{K}). \tag{B.10}$$

Noting $2Tr(\Xi'(\hat{P} - P_*)) = 2 \sum_{\hat{k},\hat{l}=1}^{\hat{K}} Tr\left(\Xi^{(\hat{k},\hat{l})\prime} (\hat{P}^{(\hat{k},\hat{l})} - P_*^{(\hat{k},\hat{l})}) \right)$, adding and subtracting $\Pi_0(P_*^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})$ to within estimated community subnetworks and $\Pi(P_*^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})$ to between estimated community subnetworks yields for the trace term in (B.10):

$$\begin{aligned} &2 \sum_{\hat{k} \neq \hat{l}} Tr\left(\Xi^{(\hat{k},\hat{l})\prime} \Pi(\Xi^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})}) \right) + 2 \sum_{\hat{k}=1}^{\hat{K}} Tr\left(\Xi^{(\hat{k},\hat{k})\prime} \Pi_0(\Xi^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})}) \right) \\ &+ 2 \sum_{\hat{k} \neq \hat{l}} Tr\left(\Xi^{(\hat{k},\hat{l})\prime} (\Pi(P_*^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})}) - P_*^{(\hat{k},\hat{l})}) \right) + 2 \sum_{\hat{k}=1}^{\hat{K}} Tr\left(\Xi^{(\hat{k},\hat{k})\prime} (\Pi_0(P_*^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})}) - P_*^{(\hat{k},\hat{k})}) \right). \end{aligned} \tag{B.11}$$

We handle the four sums in (B.11) separately. Note that in the between community subnetworks, $Tr\left(\Xi^{(\hat{k},\hat{l})\prime}\Pi(\Xi^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})\right) = ||\Pi(\Xi^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})||_F^2$ since $\Pi(\Xi^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})$ is a projection. $\Pi(\Xi^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})$ has rank ≤ 3 as noted above, so

$$2||\Pi(\Xi^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})||_F^2 \leq 6||\Pi(\Xi^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})||_{op}^2 \leq 6||\Xi^{(\hat{k},\hat{l})}||_{op}^2, \tag{B.12}$$

where $||M||_{op}$ denotes the usual spectral norm of matrix M. The second inequality holds because the spectral norm is sub-multiplicative, and for a projection matrix P, $||P||_{op} \leq 1$.

When $\hat{k} = \hat{l}$,

$$Tr\left(\Xi^{(\hat{k},\hat{k})\prime}\Pi_0(\Xi^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})\right) = Tr\left(\Xi^{(\hat{k},\hat{k})\prime}\Pi(\Xi^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})\right),$$

as $Tr(M'N) = \sum_{u,v} M_{uv}N_{uv}$ and the diagonal of $\Xi^{(\hat{k},\hat{k})}$ is all 0's because there are no self loops in either A or P_*. With this equivalence, the same argument culminating in (B.12) also holds for within community subnetworks.

To derive a bound on the first two sums of (B.11), from Lemma B.2, we have

$$\mathbb{P}\left(\sum_{\hat{k},\hat{l}=1}^{\hat{K}} ||\Xi^{(\hat{k},\hat{l})}||_{op}^2 \leq \sigma_{max}^2(C_1 n\hat{K} + C_2\hat{K}^2\log(n) + C_3(t + \log(n) + n\log(\hat{K})))\right) \geq 1 - e^{-t}. \tag{B.13}$$

Moving to the last two sums in (B.11), these can be rewritten in terms of the whole network instead of estimated subnetworks. Slightly abusing notation, let $\Pi(P_*, \{\hat{k}\})$ represent the full network matrix where all entries are given by the value dictated by $\Pi(P_*^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})$ for between estimated community edges, and by $\Pi_0(P_*^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})$ for within estimated community edges. The last two sums in (B.11) can be represented as

$$2Tr\left(\Xi'(\Pi(P_*, \{\hat{k}\}) - P_*)\right) = 2||\Pi(P_*, \{\hat{k}\}) - P_*||_F\,|\langle\Xi, H(\{\hat{k}\})\rangle|, \tag{B.14}$$

where

$$H(\{\hat{k}\}) = \frac{\Pi(P_*, \{\hat{k}\}) - P_*}{||\Pi(P_*, \{\hat{k}\}) - P_*||_F}.$$

Since for any a, b and for $c > 0$, $2ab \leq ca^2 + \frac{b^2}{c}$,

$$2Tr\left(\Xi'(\Pi(P_*, \{\hat{k}\}) - P_*)\right) \leq c||\Pi(P_*, \{\hat{k}\}) - P_*||_F^2 + \frac{|\langle\Xi, H(\{\hat{k}\})\rangle|^2}{c}. \tag{B.15}$$

Denoting the set of partitions of the nodes into exactly K communities as $\mathcal{G}_K$, for any fixed partition $G \in \mathcal{G}_K$, $\sum_{u,v}(H(G)_{uv})^2 = 1$, and the matrix Ξ consists of independent normally distributed errors with finite variances. Representing the true communities of nodes u and v as i and j, respectively, since $|\langle\Xi, H(G)\rangle| = vec(\Xi)'vec(H(G))$, observe that,

$$\mathbb{P}\left(|\langle\Xi, H(G)\rangle|^2 > t\right) = \mathbb{P}\left((\sum_{u,v=1}^{n} \sigma_{ij}\epsilon_{uv}H(G)_{uv})^2 > t\right)$$

$$= 2\mathbb{P}\left(\sum_{u,v=1}^{n} \sigma_{ij}\epsilon_{uv}H(G)_{uv} > \sqrt{t}\right) \leq 2\mathbb{P}\left(\mathcal{N}(0, \sigma_{max}^2) > \sqrt{t}\right) \leq 2e^{-t/2\sigma_{max}^2}. \tag{B.16}$$

Applying the union bound,

$$\mathbb{P}\left(|\langle \Xi, H(\{\hat{k}\})\rangle|^2 - 2\sigma_{max}^2(\log(n) + n\log(\hat{K})) > 2\sigma_{max}^2 t\right)$$

$$\leq \mathbb{P}\left(\max_{1\leq K\leq N}\max_{G\in\mathcal{G}_K}\left(|\langle \Xi, H(G)\rangle|^2 - 2\sigma_{max}^2(\log(n) + n\log(K))\right) > 2\sigma_{max}^2 t\right)$$

$$\leq 2nK^n e^{-2\sigma_{max}^2(\log(n)+n\log(K)+t)/2\sigma_{max}^2} = 2e^{-t}. \tag{B.17}$$

The squared Frobenius norm matrix $||\Pi(P_*, \{\hat{k}\}) - P_*||_F^2$ in (B.15) can be written as

$$\sum_{\hat{k}\neq\hat{l}} ||\Pi(P_*^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})}) - P_*^{(\hat{k},\hat{l})}||_F^2 + \sum_{\hat{k}=1}^{\hat{K}} ||\Pi_0(P_*^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})}) - P_*^{(\hat{k},\hat{k})}||_F^2. \tag{B.18}$$

When $\hat{k}\neq\hat{l}$, $||\Pi(P_*^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})}) - P_*^{(\hat{k},\hat{l})}||_F \leq ||\hat{P}^{(\hat{k},\hat{l})} - P_*^{(\hat{k},\hat{l})}||_F$ because $\Pi(P_*^{(\hat{k},\hat{l})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{l})})$ is the *best* projection of $P_*^{(\hat{k},\hat{l})}$ onto the estimated $\hat{Z}^{(\hat{k},\hat{l})}$ and $\hat{Z}^{(\hat{l},\hat{k})}$ values. When $\hat{k}=\hat{l}$, $\Pi(P_*^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})$ is also the best projection of $P_*^{(\hat{k},\hat{k})}$ onto the $\hat{Z}^{(\hat{k},\hat{k})}$ values with respect to only the off diagonal entries, and the diagonal entries of both $\Pi_0(P_*^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})})$ and $P_*^{(\hat{k},\hat{k})}$ are all 0. Therefore, $||\Pi_0(P_*^{(\hat{k},\hat{k})}, \hat{Z}^{(\hat{k})}, \hat{Z}^{(\hat{k})}) - P_*^{(\hat{k},\hat{k})}||_F \leq ||\hat{P}^{(\hat{k},\hat{k})} - P_*^{(\hat{k},\hat{k})}||_F$. Combining this point with those given in (B.14)–(B.17),

$$\mathbb{P}\left(2Tr\left(\Xi'(\Pi(P_*, \{\hat{k}\}) - P_*)\right) \leq c||\hat{P} - P_*||_F^2 + \frac{2\sigma_{max}^2(\log(n) + n\log(\hat{K}) + t)}{c}\right) \geq 1 - 2e^{-t}. \tag{B.19}$$

Let Ω denote the set on which, for a particular t, both events in (B.13) and (B.19) occur. Then $P(\Omega) \geq 1 - 3e^{-t}$. Therefore on Ω, by using (B.10),

$$\begin{aligned}||\hat{P} - P_*||_F^2 \leq& 6\sigma_{max}^2(C_1 n\hat{K} + C_2\hat{K}^2\log(n) + C_3(\log(n) + n\log(\hat{K}) + t)) \\ &+ c||\hat{P} - P_*||_F^2 + \frac{2\sigma_{max}^2(\log(n) + n\log(\hat{K}) + t)}{c} + Pen(n, K_*) - Pen(n, \hat{K}).\end{aligned} \tag{B.20}$$

Letting $0 < c < 1$ and $Pen(n, K)$ as in (B.2), we derive (B.3), that is

$$\mathbb{P}\big(||\hat{P} - P_*||_F^2 \leq (1-c)^{-1}Pen(n, K_*) + \frac{C_3}{c}\sigma_{max}^2 t\big) \geq 1 - 3e^{-t}. \qquad \blacksquare$$

This theorem indicates that in the Normal LSM setting, by choosing an appropriate penalty, one can choose a clustering and the number of clusters such that, with high probability, $||\hat{P} - P||_F^2$, the squared Frobenius distance between the true generating process and the estimate thereof induced by this "best" clustering, is bounded above by a multiple of the penalty using the true number of communities, accounting for the scale of the σ values in the network. As might be expected, in a network with many nodes, many communities, and large σ values, the total distance between the estimate and generating process for the network enforced by this bound can become large. One drawback is this result relies on employing a penalty which includes σ_{max}^2, a value which cannot in all cases be assumed to be known.

B.2 Case with unknown σ^2

In practice, we do not know the value of σ^2_{max}. If instead we assume that throughout the whole network, there is a single value of σ^2, we can also estimate that value using the data. Furthermore, letting $\hat{K} = n$ and putting each node into its own community, the estimate $\hat{P}$ will appear to be error free. For any network of interest, this is overfitting, so it is reasonable to set some restriction on the number of communities relative to the number of nodes. The following theorem extends Theorem B.1 under these new assumptions, where we do not require prior knowledge of σ^2_{max}.

Theorem B.2 *In the same setting as Theorem B.1, assume the choice of $\hat{K}$ is restricted such that*

$$C_1 n\hat{K} + C_2\hat{K}^2\log(n) + C_3(\log(n) + n\log(\hat{K})) \leq \frac{n(n-1)}{4}, \tag{B.21}$$

where the constants C_1, C_2, C_3 are derived in the same manner as in Theorem B.1, but take on slightly different values here. Additionally, assume network A has a constant variance parameter σ^2. Let $\hat{P}$ be the estimated network induced by the clustering of nodes that minimizes

$$||A - \hat{P}||_F^2 + \hat{\sigma}^2 Pen(n, \hat{K}), \tag{B.22}$$

where

$$\hat{\sigma}^2 = \frac{||A - \hat{P}||_F^2}{n(n-1)}, \tag{B.23}$$

and the penalty is given by

$$Pen(n, K) = 4(C_1 nK + C_2 K^2 \log(n) + C_3(\log(n) + n\log(K))). \tag{B.24}$$

Then, for $t > 0, \epsilon \in [0, 1/2)$,

$$\mathbb{P}\left(||P_* - \hat{P}||_F^2 \leq \frac{\sigma^2}{1-c}\left(Ct + (1+\epsilon)Pen(n, K_*)\right)\right) \geq 1 - 3e^{-t} - e^{-\frac{3}{32}\epsilon^2 n(n-1)}. \tag{B.25}$$

Proof: By assumption,

$$||A - \hat{P}||_F^2 + \hat{\sigma}^2 Pen(n, \hat{K}) \leq ||A - P_*||_F^2 + \sigma_*^2 Pen(n, K_*),$$

where $\hat{\sigma}^2 = \frac{||A-\hat{P}||_F^2}{n(n-1)}$ as in (B.23), and $\sigma_*^2 = \frac{||A-P_*||_F^2}{n(n-1)}$. The above is equivalent to

$$||A - \hat{P}||_F^2\left(1 + \frac{Pen(n, \hat{K})}{n(n-1)}\right) \leq ||A - P_*||_F^2\left(1 + \frac{Pen(n, K_*)}{n(n-1)}\right).$$

Then, following the proof of Theorem B.1 leading to (B.10), we have

$$\begin{aligned} &||P_* - \hat{P}||_F^2\left(1 + \frac{Pen(n, \hat{K})}{n(n-1)}\right) \\ &\leq 2Tr(\Xi'(\hat{P} - P_*))\left(1 + \frac{Pen(n, \hat{K})}{n(n-1)}\right) + \frac{||A - P_*||_F^2}{n(n-1)}\left(Pen(n, K_*) - Pen(n, \hat{K})\right). \end{aligned}$$

Dividing both sides by $1+\frac{Pen(n,\hat{K})}{n(n-1)}$ and following the arguments culminating in (B.20) but adjusting constants as necessary, with probability $\geq 1-3e^{-t}$:

$$\begin{aligned}
&(1-c)||P_* - \hat{P}||_F^2 \\
&\leq \sigma^2(C_1 n\hat{K} + C_2\hat{K}^2\log(n) + C_3(\log(n) + n\log(\hat{K})) + Ct) + \frac{\sigma_*^2 Pen(n,K_*)}{1+\frac{Pen(n,\hat{K})}{n(n-1)}} - \frac{\sigma_*^2 Pen(n,\hat{K})}{1+\frac{Pen(n,\hat{K})}{n(n-1)}} \\
&\leq \sigma^2(C_1 n\hat{K} + C_2\hat{K}^2\log(n) + C_3(\log(n) + n\log(\hat{K})) + Ct) + \sigma_*^2 Pen(n,K_*) - \frac{\sigma_*^2 Pen(n,\hat{K})}{1+\frac{Pen(n,\hat{K})}{n(n-1)}}, \quad \text{(B.26)}
\end{aligned}$$

where the denominator from the second to last term was dropped in the last inequality.

Note that σ_*^2 is the average of $\frac{n(n-1)}{2}$ squared $\mathcal{N}(0,\sigma^2)$ random variables, so its distribution is $\frac{2\sigma^2}{n(n-1)}\chi^2_{\frac{n(n-1)}{2}}$. Using a result from Johnstone (2001), for $\epsilon \in [0,1/2)$,

$$\mathbb{P}\big((1-\epsilon)\sigma^2 \leq \sigma_*^2 \leq (1+\epsilon)\sigma^2\big) \geq 1 - e^{-\frac{3}{32}n(n-1)\epsilon^2}. \quad \text{(B.27)}$$

Putting together (B.26) and (B.27), for $\epsilon \in [0,.5)$, with probability $\geq 1-3e^{-t} - e^{-\frac{3}{32}\epsilon^2 n(n-1)}$,

$$\begin{aligned}
(1-c)||P_* - \hat{P}||_F^2 &\leq C\sigma^2 t + (1+\epsilon)\sigma^2 Pen(n,K_*) \\
&\quad + \sigma_*^2\left(\frac{C_1}{1-\epsilon}n\hat{K} + \frac{C_2}{1-\epsilon}\hat{K}^2\log(n) + \frac{C_3}{1-\epsilon}(\log(n)+n\log(\hat{K})) - \frac{Pen(n,\hat{K})}{1+\frac{Pen(n,\hat{K})}{n(n-1)}}\right).
\end{aligned} \quad \text{(B.28)}$$

Using (B.21), (B.24), and $\epsilon < 1/2$, the term in parentheses in (B.28) is bounded by

$$2(C_1 n\hat{K} + C_2\hat{K}^2\log(n) + C_3(\log(n)+n\log(\hat{K}))) - \frac{Pen(n,\hat{K})}{2} \leq 0.$$

Hence, with probability $\geq 1-3e^{-t} - e^{-\frac{3}{32}n(n-1)\epsilon^2}$,

$$||P_* - \hat{P}||_F^2 \leq \frac{C\sigma^2 t + (1+\epsilon)\sigma^2 Pen(n,K_*)}{(1-c)}. \qquad \blacksquare$$

B.3 Auxiliary results

The following results were used in the proof of Theorem B.1.

Lemma B.1 *Let Ξ be a symmetric $n \times n$ matrix with 0's on the diagonal and independent $\mathcal{N}(0,\sigma_{uv}^2)$ entries above the diagonal, where all $\sigma_{uv}^2 \leq \sigma_{max}^2$. Let Ξ be partitioned into K^2 submatrices $\Xi^{(k,l)}, k,l = 1,\ldots,K$ For some constants C_1, C_2, and $t > 0$,*

$$\mathbb{P}\left(\sum_{k,l=1}^{K} ||\Xi^{(k,l)}||_{op}^2 \leq \sigma_{max}^2(C_1 nK + C_2K^2\log(n) + C_3 t)\right) \geq 1 - e^{-t}. \quad \text{(B.29)}$$

PROOF: For a fixed partition, let ξ and μ be vectors with entries $\xi_{k,l} = ||\Xi^{(k,l)}||_{op}$ and $\mu_{k,l} = \mathbb{E}||\Xi^{k,l}||_{op}$ and $\eta = \xi - \mu$. Then,

$$\sum_{k,l=1}^{K} ||\Xi^{(k,l)}||_{op}^2 = ||\xi||^2 \leq 2||\eta||^2 + 2||\mu||^2.$$

We start by bounding $||\mu||^2$. Using Theorems 1.1 and 3.1 from Bandeira and van Handel (2016), letting n_k and n_l denote the number of rows and columns respectively in $\Xi^{(k,l)}$,

$$\mu_{k,l} = \mathbb{E}||\Xi^{(k,l)}||_{op} \leq (1+\epsilon)\left(\sqrt{n_k}\sigma_{max} + \sqrt{n_l}\sigma_{max} + \frac{6}{\sqrt{\log(1+\epsilon)}}\sigma_{max}\sqrt{\log(\min(n_k, n_l))}\right)$$

for any $0 < \epsilon \leq 1/2$. In other words,

$$\mu_{k,l} \leq C_0\sigma_{max}(\sqrt{n_k} + \sqrt{n_l} + \sqrt{\log(\min(n_k, n_l))}\,),$$

$$\mu_{k,l}^2 \leq 3C_0^2\sigma_{max}^2(n_k + n_l + \log(\min(n_k, n_l))),$$

$$||\mu||^2 \leq 3C_0^2\sigma_{max}^2 \sum_{k,l=1}^{K}(n_k + n_l + \log(\min(n_k, n_l))) \leq 6C_0^2\sigma_{max}^2 nK + 3C_0^2\sigma_{max}^2 K^2\log(n). \quad \text{(B.30)}$$

Next, for $1 \leq k \leq l \leq K$, $\eta_{k,l} = \xi_{k,l} - \mu_{k,l}$ are all independent random variables since all errors are assumed to be independent. By Theorem 5.8 of Boucheron et al. (2013),

$$\mathbb{P}(|\eta_{k,l}| \geq t) = \mathbb{P}(|\xi_{k,l} - \mu_{k,l}| \geq t) \leq 2e^{\frac{-t^2}{4\sigma_{max}^2}},$$

so $\eta_{k,l}$ is sub-gaussian. Since $\mathbb{E}(\eta_{k,l}) = 0$, using sub-gaussianity, from Proposition 2.5.2 of Vershynin (2018), there exists a constant $C \leq 288e$ such that

$$\mathbb{E}(e^{t\eta_{k,l}}) \leq e^{\frac{C\sigma_{max}^2 t^2}{2}}.$$

Let $\tilde{\eta}$ be the sub-vector of η which includes the $\eta_{k,l}$ values for $1 \leq k \leq l \leq K$. Then, Theorem 2.1 of Hsu et al. (2012) ensures that, for any square matrix M, using the same constant C,

$$\mathbb{P}\left(||M\tilde{\eta}||^2 > C\sigma_{max}^2(Tr(M'M) + 2\sqrt{Tr((M'M)^2)t} + 2||M'M||_{op}t)\right) \leq e^{-t}.$$

Letting $M = I_{K(K+1)/2}$, this becomes

$$\mathbb{P}\left(||\tilde{\eta}||^2 \geq C\sigma_{max}^2(K(K+1)/2 + \sqrt{2K(K+1)t} + 2t\right) \leq e^{-t},$$

and since $||\eta||^2 \leq 2||\tilde{\eta}||^2$,

$$\mathbb{P}\left(||\eta||^2 \leq 2C\sigma_{max}^2 K(K+1) + 6C\sigma_{max}^2 t\right) \geq 1 - e^{-t}. \quad \text{(B.31)}$$

Combining (B.30) and (B.31),

$$\mathbb{P}\left(|\xi||^2 \leq 12C_0^2\sigma_{max}^2 nK + 6C_0^2\sigma_{max}^2 K^2\log(n) + 4C\sigma_{max}^2 K(K+1) + 12C\sigma_{max}^2 t\right) \geq 1 - e^{-t}.$$

Collecting terms yields (B.29). ∎

The following lemma is nearly identical to Lemma 6 in Noroozi et al. (2021), and serves the same purpose, to translate the bound in Lemma B.1, which is conditional on a particular partition, into an unconditional bound.

Lemma B.2 *For any $t > 0$,*

$$\mathbb{P}\left(\sum_{\hat{k},\hat{l}=1}^{\hat{K}} ||\Xi^{(\hat{k},\hat{l})}||_{op}^2 \leq \sigma_{max}^2(C_1 n\hat{K} + C_2\hat{K}^2\log(n) + C_3(t + \log(n) + n\log(\hat{K})))\right) \geq 1 - e^{-t}. \quad \text{(B.32)}$$

Proof: Denoting the set of partitions of the nodes into K communities as $\mathcal{G}_K$, for any fixed partition $G \in \mathcal{G}_K$, from Lemma B.1,

$$\mathbb{P}\left(\sum_{k,l=1}^{K} ||\Xi^{(k,l)}||_{op}^2 \geq \sigma_{max}^2(C_1 nK + C_2K^2\log(n) + C_3x)\right) \leq e^{-x}.$$

Taking a union bound over all possible partitions and setting $x = t + \log(n) + n\log(K)$

$$\mathbb{P}\left(\sum_{\hat{k},\hat{l}=1}^{\hat{K}} ||\Xi^{(\hat{k},\hat{l})}||_{op}^2 - \sigma_{max}^2(C_1 n\hat{K} + C_2\hat{K}^2\log(n) + C_3(t + \log(n) + n\log(\hat{K}))) \geq 0\right)$$

$$\leq \mathbb{P}\left(\max_{1\leq K\leq n}\max_{G\in\mathcal{G}_K}\sum_{k,l=1}^{K} ||\Xi^{(k,l)}||_{op}^2 - \sigma_{max}^2(C_1 nK + C_2K^2\log(n) + C_3(\log(n) + n\log(K))) \geq \sigma_{max}^2 C_3 t\right)$$

$$\leq \sum_{K=1}^{n}\sum_{G\in\mathcal{G}_K}\mathbb{P}\left(\sum_{k,l=1}^{K} ||\Xi^{(k,l)}||_{op}^2 - \sigma_{max}^2(C_1 nK + C_2K^2\log(n) + C_3(\log n + n\log(K))) \geq \sigma_{max}^2 C_3 t\right)$$

$$\leq nK^n e^{-t-\log(n)-n\log(K)} = e^{-t} \qquad \blacksquare$$

C Additional estimation details

C.1 Pre-estimating the diagonal for within community NMF

When applying the methodology described in Appendix A.1 to a set of edges within the same community, symmetry ensures that $a = b$, so there are no conflicts among estimators. There is, however, a need to resolve the fact that no self loops ($W_{uu} = 0$) may impact the estimates of a and α for the within community setting $i = j$. To avoid this impact, the values of the diagonal entries W_{uu} should be imputed before computing the NMF. Were self loops allowed, the expected value a self loop would be $\mathbb{E}(W_{uu}|Z) = 2\alpha Z_u$, so we can estimate this value while accounting for all diagonal zeros and estimates of other Z's by setting

$$W_{uu} = \frac{\left(2\sum_{v\in i,u\neq v} W_{uv}\right) - \left(\frac{1}{n_i-2}\sum_{v\in i,u\neq v}\sum_{q\in i,q\neq u,v} W_{vq}\right)}{n_i - 1}, \quad \text{(C.1)}$$

where n_i is the number of nodes in community i. Essentially, this takes twice the average within community edge weight for a particular node u and subtracts out twice the average non-diagonal edge weight connecting the other nodes in community i. This approximates twice the impact node u while negating the collective impact of the other nodes.

C.2 Handling repeated edge weight values

In Section 4.1, when estimating the uniformly distributed values of $\widehat{G}(w)$ in (16) and $\widehat{\Psi}_u^{(j)}$ in (18), we make an adjustment if we observe the same value multiple times. Let $W_1 < W_2$ be two consecutive sorted values of W_{uv}'s in the same pair of communities, with $\widehat{G}(W_1) = \frac{k}{n+1}$ and $\widehat{G}(W_2) = \frac{k+m}{n+1}$, where $m > 1$. That is, there are m different edges in the set $\{W_{uv} : u \in i, v \in j, W_{uv} = W_2\}$. Then, we set

$$\widehat{G}(W_2) = \frac{k + \frac{m}{2} + \frac{1}{2m}}{n+1}. \tag{C.2}$$

The purpose of (C.2) is to make all the new $\{\widehat{G}(W_{uv}) : u \in i, v \in j, W_{uv} = W_2\}$ values to be at least halfway between $\widehat{G}(W_1)$ and the original $\widehat{G}(W_2)$ value, but where more duplicate values will bring this value down further. When there are no duplicate values (i.e. $W_1 < W_2 < W_3 < \ldots < W_n$), this formula leaves $\widehat{G}(W_2)$ intact. When duplicate values are present, there will still be duplicate values after applying (C.2), but they are moved to a different location between $\frac{k}{n+1}$ and $\frac{k+m}{n+1}$.

If the $D_j(u)$ values repeat in (17), we make an analogous adjustment to (18).

D Additional community detection details

In this appendix, more information is provided about community detection for the kinds of networks described in the main paper, including motivating ideas, algorithms, and some discussion of why existing community detection methods may be inappropriate in this setting.

D.1 Limitations of modularity

Many clustering algorithms on networks attempt to maximize modularity, but this may not be the appropriate measure to use for dense networks. To see this, consider the network shown in Figure 10. That figure begins on the left with a network where $\sigma = 0$ with 74 nodes in 2 communities exhibiting within community positive association and between community negative association. Within community edges are drawn from a uniform distribution with a maximum of 150, while between community edges are drawn from a uniform distribution with a maximum of 100. Node sociability parameters for both communities range from .05 to .95 in increments of .025. The H-function in the original network is the same as that used for the rightmost plot in Figure 2, though since the between community edges have negative association, the inputs to that H-function are $1 - \Psi_u$ and $1 - \Psi_v$. While the network is not strictly assortative, after accounting for node sociabilities and using the appropriate inputs, edges between nodes in the same communities have 50% greater weights than edges between nodes in different communities. In this sense there is some notion of homophily that is absent in other regimes where modularity fails. Ordered as in the figure, one can visually identify 2 distinct communities. The second plot from the left in the figure is the estimate of the left plot using the community assignments when clustering nodes using the walktrap algorithm of Pons and Latapy (2006) with four steps, which returns three communities, not two. The third plot is the estimate of the first plot using the community assignments by calculating the leading non-negative eigenvector of the modularity matrix of the graph. The fourth plot is the estimate of the first plot using the correct assignments. The modularity of the true communities on this network is in fact slightly negative. This failure of community detection algorithms will feed incorrect community labels to the estimation procedure, leading to estimates that don't preserve the structure of the original network, as seen in the figure.

The issue is modularity tries to identify highly interconnected nodes, where edge weights within communities are expected to be higher than edge weights between communities. The methodology

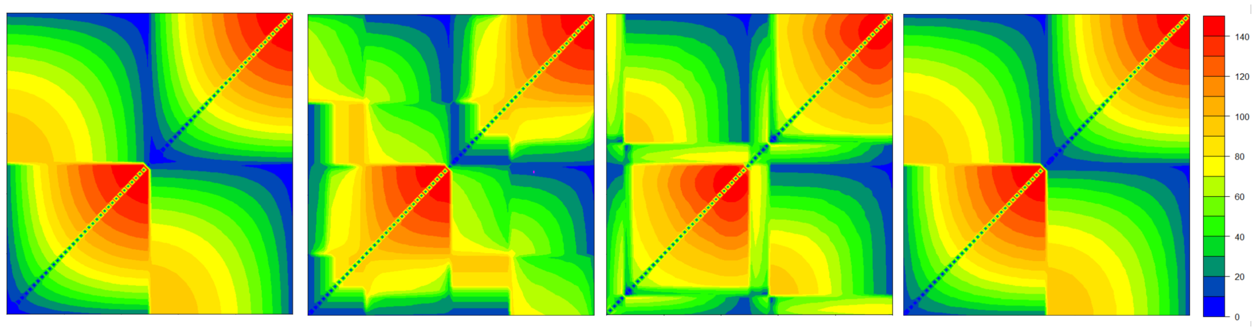

Figure 10: Reconstructing a network (left) based on different community assignments.

described above needs clusters to have a different property to work appropriately, namely that the nodes in each community should obey a kind of monotonicity. Nodes in the same community should have similar patterns of connecting to nodes in other communities, and their own community. In the network in Figure 10, all nodes in community 1 “prefer” other nodes in community 1 with high sociability (Ψ) values, but “prefer” nodes in community 2 with low sociability values. This shared ordering of preferences over nodes in each community is crucial for ensuring that the estimation procedure will get appropriate orderings of local sociability statistics.

It is not difficult to construct networks where it would be useful to combine the community detection using L with modularity. For example, assume communities i and j have positive association both within community and between the communities, but the within community edges are generally much larger than the between community edges. Also assume both i and j have negative association with community k. Clustering nodes to simply share ordering of preferences would separate nodes in community k but would not distinguish nodes in community i from nodes in community j. Subsequently employing a modularity maximization algorithm on the estimated community consisting of nodes in i and j would recover the true communities and lead to better estimates for all of the edges within and between i and j.

D.2 Greedy algorithm for community detection

A direct algorithm for maximizing L in (23) is to try to iteratively combine nodes into communities which will greedily make L larger. It is computationally impractical to test every possible partition of nodes to maximize L. However, based on the structure of L, Algorithm 2 tries to combine communities which are most correlated with one another.

At the start, each node is placed into its own estimated community. The “aggregate degree” of each estimated community is defined as the total weight of the edges emanating from any node in the estimated community, where edges are double counted if they connect two nodes in the same estimated community. In Algorithm 2, the aggregate degree of an estimated community is equal to the column sum of the estimated community’s corresponding column in the “communityAggregate” matrix. When each node is in its own estimated community, the aggregate degree of the community is the same as the degree of the node. In each “round,” the algorithm orders the estimated communities by their aggregate degrees at the beginning of the round from largest to smallest. Then the algorithm visits these estimated communities in order, and for each estimated community $\hat{i}$, selects a candidate estimated community $\hat{j} \neq \hat{i}$ which maximizes

$$\mathcal{C}_{\hat{i}\hat{j}} = corr\left\{ \big(\sum_{u:u\in\hat{i}} W_{uv}, \sum_{q:q\in\hat{j}} W_{qv} \big) : v \in V \right\},$$

where $\sum_{u:u\in\hat{i}} W_{uv}, v \in V$, is a length n vector where each entry has the total edge weight connecting nodes in estimated community $\hat{i}$ with each node in the network. Estimated communities $\hat{i}$ and $\hat{j}$ are merged if doing so does not decrease the measure L. If $\hat{i}$ and $\hat{j}$ are merged, and if $\hat{j}$ has not already been visited by the algorithm this round, the algorithm will visit the combined $\hat{i}$ and $\hat{j}$ when it would have visited $\hat{j}$. Each round completes after the algorithm has completed all scheduled visits.

If at least two estimated communities have been merged, the algorithm proceeds to the next round. If no communities have been merged, the algorithm does a sweep, calculating $\mathcal{C}_{\hat{i}\hat{j}}$ for all estimated communities, and attempts to merge pairs of estimated communities in decreasing order of $\mathcal{C}$ values. As soon as any pair of estimated communities are merged, the algorithm stops the sweep and proceeds to the next round. Algorithm 2 terminates when it goes through a full round and a sweep without merging any estimated communities, or when the number of estimated communities reaches 1.

Two details of the algorithm should be explained further. First, the algorithm does not require merging communities to increase L because at the outset, when all nodes are in their own estimated community, combining two communities *cannot* increase L. Therefore, requiring a merger to increase L would prevent the algorithm from gaining any traction. Second, why go through each round instead of constantly sweeping, or better yet just combining the estimated communities which would most increase L? In addition to this proposal being computationally costly, it may also lead to the initial formation of a single large but overly heterogeneous community since communities can only contribute to L once they contain three or more nodes. As this is already a greedy algorithm, we view our design as a means of not overlooking any estimated community, and as a conservative precaution against premature optimization.

D.3 Spatial clustering

For another perspective on community detection, we represent the original network as a matrix, and think of each row vector as a point in n-dimensional space corresponding to a particular node, where n is the number of nodes in the network. Spatial clustering of the points then gives us communities, which is motivated as follows. Consider two nodes u and v in the same community with similar Ψ values. For any third node $x \in j$, as $u \in i$ and $v \in i$, $H_{ij}(\Psi_u, \Psi_x)$ should be similar to $H_{ij}(\Psi_v, \Psi_x)$, thus W_{ux} should be close to W_{vx}. Therefore, nodes in the same community with similar Ψ values are expected to have similar edge weights, so entries in their corresponding rows should be similar in $n-2$ dimensions (the exceptions being due to no self loops). Therefore, in this n-dimensional space, two nodes with the same community and similar Ψ values should be neighbors.

Spatial clustering can be achieved using many existing clustering algorithms. For certain methods, such as those based on distances, one may need to account for the zeros on the diagonal by only calculating the distance between two nodes on the remaining $n-2$ dimensions. However, as with other spatial clustering problems, there is no algorithm which correctly clusters the nodes for every possible network.

Algorithm 2: Greedy algorithm using L

```
   Result: $\{\widehat{i}\}$
   Input: $W$
1  labels$^{(0)} = \vec{0}_n$
2  labels$^{(1)} = \{1, ..., n\}$
3  $q$=1
4  communityAggregate = $W$
5  while $labels^{(q)} \neq labels^{(q-1)}$ do
6      $q$++
7      labels$^{(q)}$ = labels$^{(q-1)}$
8      labelOrder = sort labels by decreasing column sum of communityAggregate
9      while $length(labelOrder) > 0$ do
10         labels$^{(q)}$, communityAggregate, labelOrder = attemptMerge(communityAggregate,
            labels$^{(q)}$, labelOrder)
11         dequeue(labelOrder(1))
       end
12     if $labels^{(q)} = labels^{(q-1)}$ then
13         labels$^{(q)}$, communityAggregate = sweep(communityAggregate, labels$^{(q)}$)
       end
   end
14 finalClustering = labels$^{(q)}$

   Procedure attemptMerge(communityAggregate, labels$^{(q)}$, labelOrder)
15     newLabels = labels$^{(q)}$
16     newOrder = labelOrder
17     A = newOrder(1)
18     B = $\underset{B' \neq A}{\text{argmax}}$ $corr$([communityAggregate(A)], [communityAggregate(B')])
       // [communityAggregate(A)] represents the Ath column in the
          communityAggregate matrix
19     mergedLabels = newLabels
20     mergedLabels(mergedLabels = B) = A
21     mergedOrder = newOrder
22     mergedOrder(mergedOrder = B) = A
23     if calculateMeasureL(W, mergedLabels) $\geq$ calculateMeasureL(W, newLabels) then
24         newLabels = mergedLabels
25         newOrder = mergedOrder
26         [communityAggregate(A)] += [communityAggregate(B)]
27         Remove [communityAggregate(B)] column from communityAggregate
       end
28     return newLabels, communityAggregate,
   Procedure sweep(communityAggregate, labels^(q))
29     correlationOrder = sort non-matching label pairs by decreasing correlations of columns
        in communityAggregate
30     while length(correlationOrder) > 0 do
31         newLabels = labels^(q)
32         A = correlationOrder(1, 1)
33         B = correlationOrder(1, 2)
34         mergedLabels = newLabels
35         mergedLabels(mergedLabels = B) = A
36         if calculateMeasureL(W, mergedLabels) ≥ calculateMeasureL(W, newLabels) then
37             newLabels = mergedLabels
38             [communityAggregate(A)] += [communityAggregate(B)]
39             Remove [communityAggregate(B)] column from communityAggregate
40             dequeueAll(correlationOrder)
           else
41             dequeue(correlationOrder(1))
           end
       end
42     return newLabels, communityAggregate
```

D.4 Spectral clustering algorithm incorporating L

Appendix D.3 justifies the application of spatial clustering techniques for our networks of interest. In this section, we focus on one spatial clustering algorithm, spectral clustering, which we employ in tandem with the measure L in (23), used to choose both a particular number of estimated communities as well as the best clustering for that number of communities. If the number of communities K is known, we can follow the methodology of Ng et al. (2001). Measure the distance between every column in our network using a radial basis function kernel. Then construct a neighbors graph based on these distances, and a Laplacian based on this neighbors graph. Finally, run K-means on the Laplacian to get clusterings. This can all be done using the existing `specc` function from the R package **kernlab** just by specifying the number of centers, as `specc` will automatically select a scale parameter for the kernel. However, as this is not a deterministic algorithm, it can be helpful to run several replicates and take clustering that maximizes L.

The remaining issue is how to choose the number of clusters, K, which is a priori unknown. However, we can use the introduced measure L as a measure of clustering success, so we can impose a simple stopping rule, which is shown in Algorithm 3.

D.5 Clustering using normalized network

Another approach to community detection which seems to return interesting results is as follows: take the network as a whole and normalize each row in the original matrix by taking $N_{uv} = \frac{A_{uv} - \bar{A}_{u\bullet}}{SD(A_{u\bullet})}$, where $A_{u\bullet}$ represents the row in A corresponding to node u. Next, calculate the eigenvalues $\lambda_1, \ldots, \lambda_n$ of the normalized network N, and note the index corresponding to the largest difference in the absolute values of the real parts of these successive eigenvalues, $\underset{1 \le i \le n-1}{\operatorname{argmax}} |Re(\lambda_i)| - |Re(\lambda_{i+1})|$. Keep only the real parts of the first several eigenvectors corresponding to eigenvalues $\lambda_1, \ldots, \lambda_i$, and look for clusters in this lower dimensional space.

Algorithm 3: Community detection using spectral clustering and stopping criterion

```
Result: $\{\widehat{i}\}$
Input: $W$, replicates
// replicates is the number of times to run specc for each value of K
1  $\Delta = 1$; $K$ =0;
2  while $\Delta > 0$ do
3      $K = K++$;
4      clusterGoodness($K$) = $-\infty$;
5      index = 0;
6      while index < replicates do
7          index++;
8          candidateClustering = specc($W$, centers = $K$);
9          candidateGoodness =calculateMeasureL($W$, candidateClustering);
10         if candidateGoodness > clusterGoodness($K$) then
11             clusterGoodness($K$) = candidateGoodness;
12             clustering($K$) = candidateClustering;
           end
       end
13     $\Delta$ = clusterGoodness($K$) $-$ clusterGoodness($K-1$)
   end
14 finalClustering = clustering($K-1$)
```

A normalized version of the network presented in Figure 10 is shown in the left plot of Figure 11. In the right plot of Figure 11, each point plots a row of the real parts of the first 2 eigenvectors of this row-normalized network, where the color of the point represents the true community of the corresponding node.

Plotting each row of these eigenvalues appears to give clearly separate clusters. Normalizing the network so all nodes have degree 1 doesn't give the same results. The first eigenvector of the original matrix often represents degree information, so the normalization to calculate N should discard that degree information, but in the process, it also seems have some kind of downstream effect on other eigenvectors which helps them to capture the underlying communities. This phenomenon is not specific to the constructed network in Figure 10. Figure 12 shows a network generated via an NSM with an H-function combining two exponential random variables, in the style of the third plot in Figure 2, where within community connections show positive association and between community connections show Simpson association, along with the eigenvectors from the normalized version of that network. Again, the real parts of the eigenvectors of the normalized network show a clear separating plane between the two communities, but it is not along one of the 2 dimensions, but rather along a combination of them.

There is some intuition for why clustering based on normalizing the network might work. Ignoring the diagonal, normalizing the matrix makes each column sum to 0, and have variance 1. Since this is a dense weighted network, there is no true concept of a "hub," but instead just a node that has greater edge weights. For this reason, we have no reason to value one node/dimension over another, and we can avoid having to find complicated kernels with different variances across different dimensions. If we don't normalize, the distance in only one dimension can totally dominate. In the case where all associations are positive, when doing community detection, we want to ignore degree in favor of preference, so it makes sense to divide by a standard deviation. In the case where

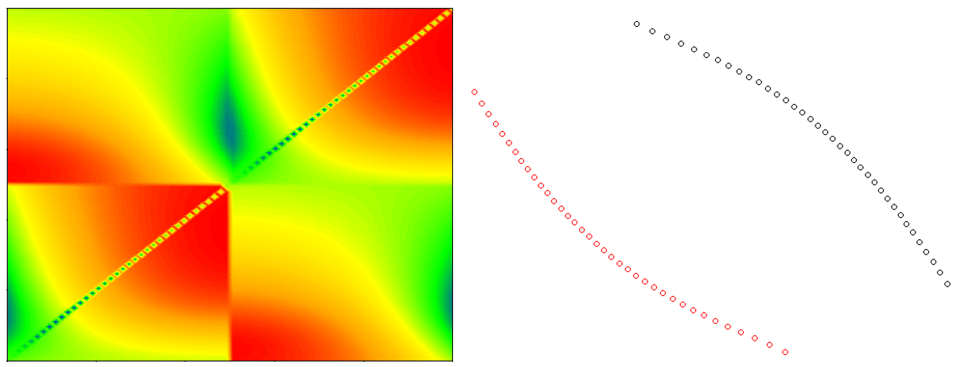

Figure 11: Normalized network with implied clustering.

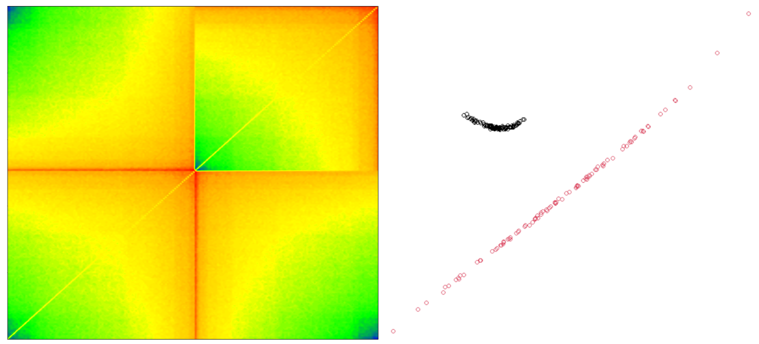

Figure 12: Network generated via NSM, with the clustering from its normalized version.

some associations are negative (or Simpson), extreme Ψ values will indicate greater variance, but again, we still want to ignore this for the purposes of community detection, so it still makes sense to normalize. The reason to normalize is since each node is connecting to all (or most) others, we want to ensure that we are accounting for a given node's average edge weight and the variance of the edge weights. In this way, we are still looking for patterns of preferences across other nodes. By normalizing the columns, we can compare them to one another on an apples-to-apples basis. A negative weight in the normalized matrix means that the edge between the reference node and another node is less than the average weight emanating from the reference node. For the purposes of community detection in this model, this is really what we care about, that is, patterns across nodes in a given community that show preferences across nodes throughout the whole network.

These last points also indicate the zeros on diagonal can bring up issues. Imagine a situation where every other edge weight emanating from a node is extremely large but low variance. The 0 will shrink the average and increase the standard deviation by a lot, but it's purely artificial. If we use a distance between nodes/columns to cluster, it is imperative to ignore both the rows corresponding to those nodes. If there is actually a relatively large edge weight between them, that edge weight will be subtracted and squared twice. This is also purely artificial due to no self loops.

E Additional simulations

E.1 H-Normal LSM with $\sigma = 0$

Figure 13 shows an H-Normal LSM with $\sigma = 0$, where the value for within community edges are equal to $5 + 3\Phi_1^{-1}(\Psi_u) + 3\Phi_1^{-1}(\Psi_v)$. The between community edges, assuming u is in community 1 and v is in community 2, are equal to $8 - 3\Phi_1^{-1}(\Psi_u) + 1.5\Phi_1^{-1}(\Psi_v)$. The original network is shown on the left. In the middle is the reconstructed estimated network, which looks nearly identical to the original network. $\widehat{\sigma}$ is essentially 0 for this network, so the MSE (as defined in (21)) for each subnetwork is shown on the right.

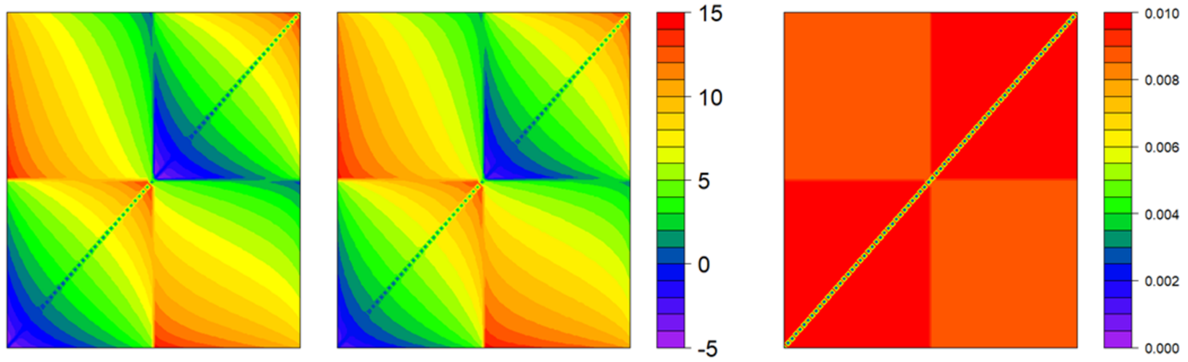

Figure 13: H-Normal LSM with $\sigma = 0$, its estimate, and the subnetwork level MSE (21).

E.2 Varied network

Figure 14 adds more complexity and departs even further from an H-Normal NSM, this time with 200 nodes and four communities of possibly different sizes. In this case, nodes are not assigned Ψ values but i.i.d. Gamma(shape = 5, scale = 10) random values $\Gamma_u^{(1)}$ and $\Gamma_u^{(2)} = 1/\Gamma_u^{(1)}$. However, within each community, $\Gamma_u^{(1)}$ and $\Gamma_u^{(2)}$ are normalized by dividing by $\sum_{u \in i} \Gamma_u^{(1)}$ and $\sum_{u \in i} \Gamma_u^{(2)}$, respectively. Each pair of communities is also assigned independently a random Gamma(shape $=50\sqrt{10}$, scale $= 50\sqrt{10}$) value Γ_{ij}. There is no mathematical significance to these parameters, they were chosen only to create a striking image. Within community edges are distributed according to a negative binomial distribution where the target number of successful trials is $2 \times \Gamma_{ii}$ and the probability of success in each trial is $(1-\Gamma_u^{(1)})(1-\Gamma_v^{(1)})$. "Adjacent" communities in the graph are distributed according to a negative binomial where the target number of successful trials is $1.5 \times \Gamma_{ij}$ and the probability of success in each trial is $(1-\Gamma_u^{(2)})(1-\Gamma_v^{(2)})$. Connections between communities 1 and 3 or between communities 2 and 4 are distributed according to a Poisson distribution with parameter $100 \times \Gamma_{ij}\Gamma_u^{(1)}\Gamma_v^{(1)}$. Finally, connections between 1 and 4 are normally distributed with mean $10000 \times (\Gamma_u^{(1)} + \Gamma_v^{(1)})$ and variance 1. Though the original network looks noisy, the estimate seems to capture a smooth approximation. In fact, in this network, $\hat{\sigma}$ is indistinguishable from 0 everywhere, so the MSE (21) of each subnetwork, shown in the right plot of Figure 14, would be used for the bootstrap.

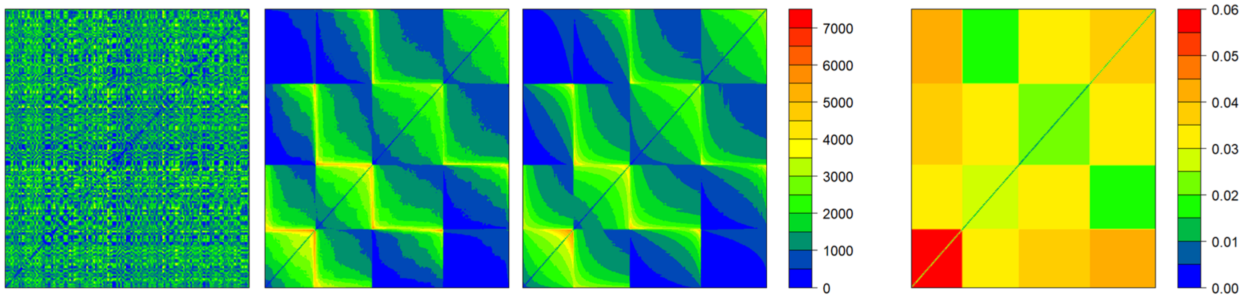

Figure 14: From left to right: original, reordered, estimate, MSE (21) of each subnetwork.

E.3 Correlation matrix

In this network, we first generate 3 "lodestar" series $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ of length 1000 composed entirely of i.i.d. $\mathrm{U}(0,1)$ random variables. We create a network with 200 nodes where the first 50 nodes each get a series which are (to different degrees) positively correlated with the first lodestar series, and negatively correlated with the second lodestar series. This is done by calculating the series for node u at time t as

$$u(t) = \beta_1(u)\mathcal{L}_1(t) + \beta_2(u)\mathcal{L}_2(t) + \beta_3(u)\mathcal{L}_3(t) + \left(1 - \sqrt{\beta_1^2(u) + \beta_2^2(u) + \beta_3^2(u)}\right)\epsilon_u(t),$$

where $\epsilon_u(t) \sim \mathrm{U}(0, .35)$ is an idiosyncratic term for each node at each time step. For the first 50 nodes, the $\beta_1(u)$ values are positive and increasing with u, while the $\beta_2(u)$ values are negative and getting more negative with u, and $\beta_3(u) = 0$. For the second 50 nodes, the $\beta_2(u)$ values are positive and increasing with u, while the $\beta_1(u)$ values are negative and getting more negative with u. The 101st to 150th nodes get series which are increasingly positively correlated to $\mathcal{L}_3$ series and increasingly negatively correlated to $\mathcal{L}_2$. Finally, the last 50 nodes have series which are increasingly negatively correlated to $\mathcal{L}_3$. Taking the correlations of the 200 series, we get the correlation matrix on the left of Figure 15. Though the subnetwork between the first and fourth communities may look disordered, because nodes within each community are relatively correlated with each other, there is a discernible ordering in that between community subnetwork. Even so, from a practical perspective, edge weights only take on a narrow range of values near 0 in that subnetwork.

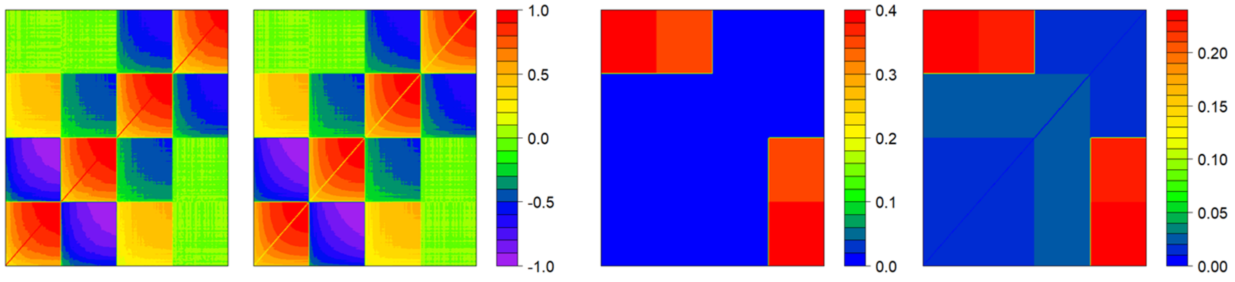

Figure 15: Correlation network. From left to right: original, estimate, $\hat{\sigma}$ for each subnetwork, MSE (21) of each subnetwork.

E.4 Edge weights with injected noise

In plot A of Figure 16, we start with the same underlying network as in Figure 3 but add Gaussian noise ζ_{uv} centered at 0 with a variance of 36 *to the final edge weights*, not in "normal" space, so the edge weights in the network can go below 0 and above 150. In plot B, the added Gaussian noise ζ_{uv} has variance 100 for within community edges and variance 225 for between community edges. In both cases, the reconstructed networks recover the underlying pattern, but the noisier network is estimated more coarsely.

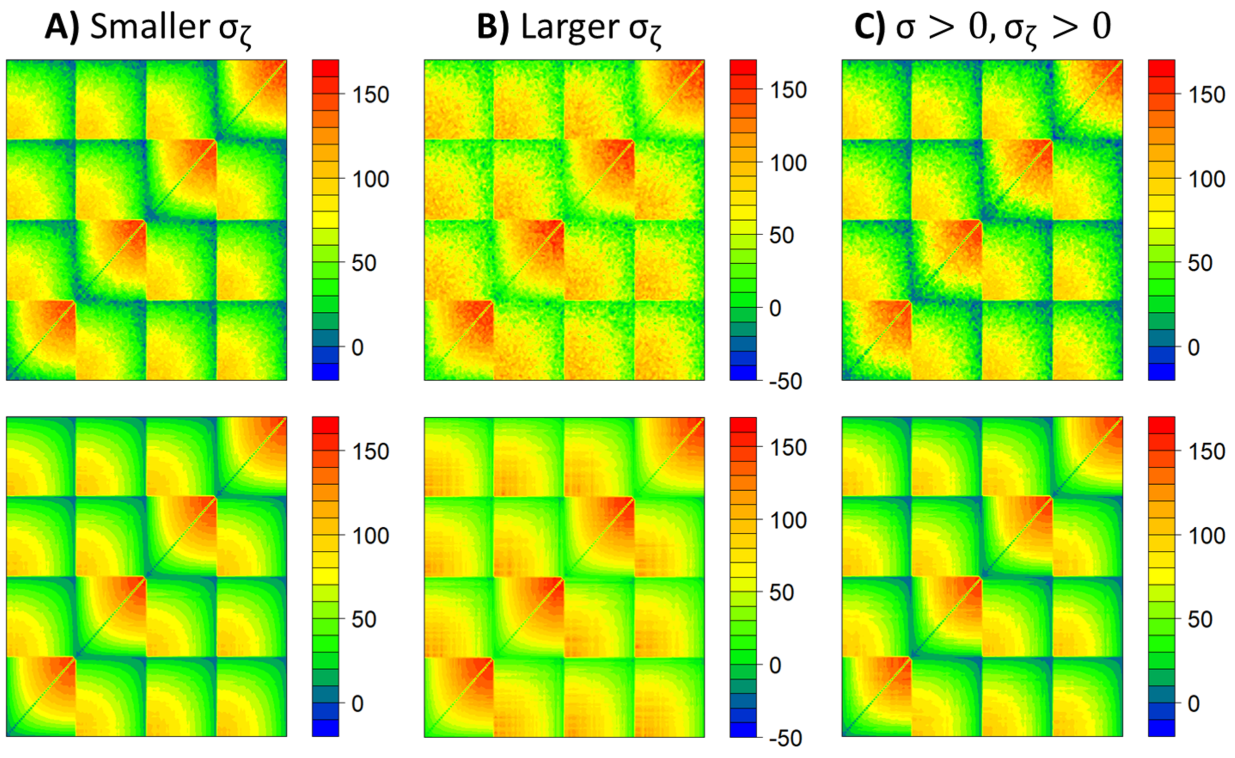

Figure 16: The network in plot A of Figure 3 with different values for σ_ζ^2

Finally, in plot C, the same network is taken with $\sigma = .05$ everywhere and external noise is included by adding ζ_{uv} with variance 36 to the final edge weights. Even in this last case, the underlying signal is broadly recovered.

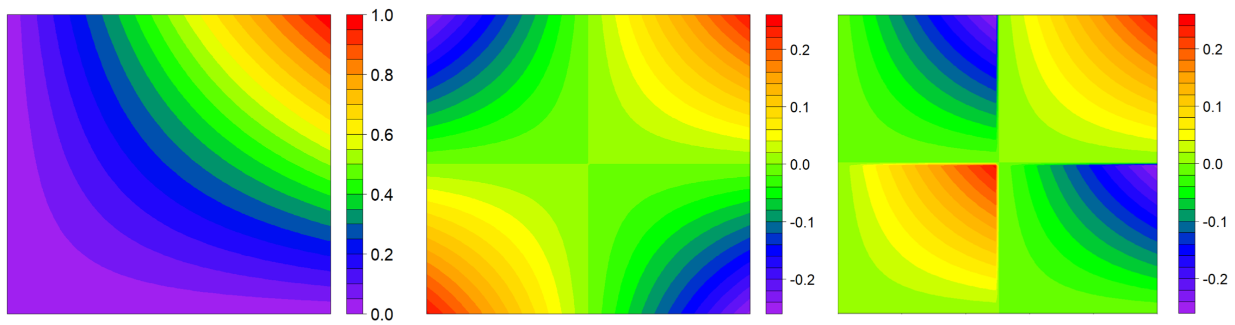

Figure 17: From left to right: Network generated by multiplying sociability parameters. Network generated by demeaning sociability parameters then multiplying. Reordering of second network based on within estimated community degree.

E.5 Multiplying sociability parameters

It may be worth considering alternative models which can generate networks similar to LSMs and NSMs. For example, if edge weights are generated by multiplying sociability parameters, some surprising things happen. Ignoring diagonal zeros, if we take the values from .01 to 1 by .01 on each axis and let the value in the matrix equal the product of the axes, that results in the leftmost plot in Figure 17. In other words, if all values are positive, the result looks like the 4th plot in Figure 2. But if we center each axis to be mean 0 and recalculate, that produces the second plot in Figure 17. This is not the intended setting for the models discussed in this paper, but if we were to treat these networks as such, in the first case, every node would be put into a single community. In the second case, those nodes with negative values would be put in a separate community from nodes with positive values. Using this split and reordering based on within community degree gives the rightmost plot in Figure 17.

In principle, were noise added to the edge weights of this network, knowing the true generating model type might improve estimation, as one may be able to smooth noise out over more observations by keeping all nodes in one large community. However, even using the "wrong" communities, our estimation procedure appears to replicate the underlying network. Even though the generating process for this network is the same across both estimated communities, separating the second network into two communities is therefore a reasonable choice, especially since the two communities can be so easily defined. This kind of pattern only arises when multiplying nodes with positive sociabilities and others with negative sociabilities, not when all sociabilities have the same sign. While multiplying sociabilities hints at the idea of negative association, networks generated from these models are still restricted to symmetric contours of the type seen in the 4th plot of Figure 2. However, simply multiplying sociability parameters can't generate networks that have contours of the type seen in the first 3 plots of Figure 2, nor can it give positive association patterns within a community but negative association patterns between communities when there are more than two communities.

F Higher dimensional H-functions and "failure"

Thus far, we have defined H-functions as a class of functions that take in two uniform random variables and output another uniform random variable. This model can be extended to include a broader class of functions.

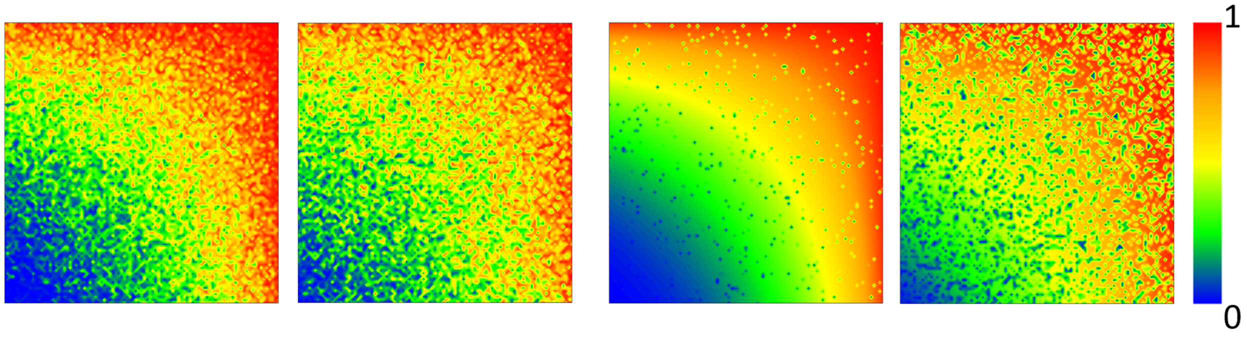

Figure 18: Discretized 3-dimensional H-functions where axes show x and y but η is drawn randomly at each (x, y) pair. The 2-dimensional H-function with arguments x and y is the same as in the third plot of Figure 2. From left to right: H-Normal NSM as in (9) with $\sigma = .05$. H-Normal NSM with $\sigma = .5$. Failure with $\alpha = .95$. Failure with $\alpha = .5$.

Definition F.1 (d-dimensional H-function.) A function $H : (0,1)^d \to (0,1)$ is a d-dimensional H-function if the inputs are d uniform random variables, and the output is a uniform random variable which is monotonic in each argument.

Note the H-Normal NSM (9) can be expressed in terms of a 3-dimensional H-function as:

$$W_{uv} = G^{-1}\left(H(\Psi_u, \Psi_v, \eta_{uv})\right) \tag{F.1}$$

with $\eta_{uv} = \Phi_1(\epsilon_{uv}) \sim U(0,1)$, where

$$H(x,y,\eta) = \Phi_1\left(\frac{1}{\sqrt{1+\sigma^2}}\Phi_1^{-1}(h(x,y)) + \frac{\sigma}{\sqrt{1+\sigma^2}}\Phi_1^{-1}(\eta)\right), \tag{F.2}$$

where $h(x,y)$ is a 2-dimensional H-function. The first two plots in Figure 18 show the values along the x and y axes of functions which have the form of (F.2) where $h(x,y)$ is in the schema of (15) such that F_1 and F_2 are Exponential distributions, and $F_{1,2}$ is a Gamma distribution. As "errors" are injected at each (x, y) pair, this can also be seen as a subnetwork generated via an H-Normal NSM where $H(x,y)$ in (9) is that of the third plot in Figure 2, and η is injected at the edge level.

The specific H-Normal NSM (14) is a simple example which can be expressed in closed form as a 3-dimensional H-function as in (F.2) where

$$H(x,y,\eta) = \Phi_1\left(\frac{1}{\sqrt{1+\sigma^2}\sqrt{1+\rho^2}}\Phi_1^{-1}(x) + \frac{\rho}{\sqrt{1+\sigma^2}\sqrt{1+\rho^2}}\Phi_1^{-1}(y) + \frac{\sigma}{\sqrt{1+\sigma^2}}\Phi_1^{-1}(\eta)\right). \tag{F.3}$$

In the network context, ρ defines the relative influence of each of the node sociabilities, while σ controls the "signal-to-noise" ratio of this 3-dimensional H-function. If one imagines observing several instances of the same network given by (F.1) with idiosyncratic η values, then increasing σ would increase the variance of the individual edge weights from one instance to another. The 3-dimensional H-function (F.3) can be viewed as a composition of two 2-dimensional H-functions as follows:

$$H(x,y,\eta) = H_{\sigma^2}(H_{\rho^2}(x,y),\eta),$$

where H_{ρ^2} and H_{σ^2} are both of the form (13) but with different variance parameters, as indicated by their subscripts. All the observations about (F.3) do not depend on using 3-dimensional H-functions built from normal distributions or even H-Normal NSMs, but rather one can use any

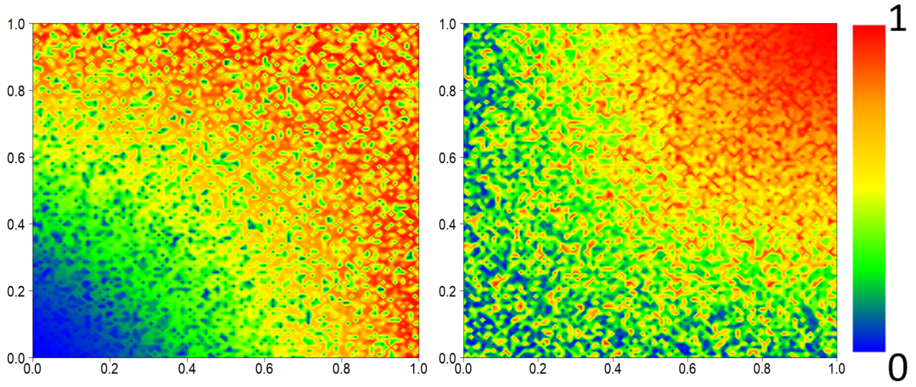

Figure 19: 3-dimensional H-functions of the form (F.4), where axes represent values of x and y.

d-dimensional functions similar to (15) where $d > 2$, with suitable adjustments based on the chosen distributions F_1, F_2, and $F_{1,2}$.

One general method of creating higher dimensional H-functions is by chaining together lower dimensional H-functions as follows:

$$H(x, y, \eta) = F_{1,2}\left(F_1^{-1}\left(F_{3,4}(F_3^{-1}(x), F_4^{-1}(y))\right), F_2^{-1}(\eta)\right). \tag{F.4}$$

In general, the inner functions F_3, F_4 do not need to share the same form as the outer functions F_1, F_2. In the left plot of Figure 19, F_3, F_4 and $F_{3,4}$ correspond to the third plot in Figure 2, $\eta \sim$ U$(0, 1)$ for each edge, and F_1, F_2 and $F_{1,2}$ correspond to the rightmost plot in Figure 2. The right plot of Figure 19 swaps the roles of the inner and outer 2-dimensional H-functions in the left plot of Figure 19. While building higher dimensional H-functions in this way provides a lot of flexibility, this method may not give simple closed form expressions, and may not guarantee identifiability.

While the H-Normal NSM is a model with additive "error," (F.1) is more general. The following can be used to generate a different kind of "error," one we shall call *failure*. In this context, let

$$H(x, y, \eta) = (h(x, y))^{\alpha}\delta^{1-\alpha}, \tag{F.5}$$

where α is a value between 0 and 1, h is a 2-dimensional H-function, and $\delta^{1-\alpha}$ is given by $F_\alpha^{-1}(\eta)$ where

$$F_\alpha(x) = \begin{cases} 0, & x < 0, \\ \frac{x}{1-\alpha}, & 0 \le x \le 1 - \alpha, \\ 1, & x > 1 - \alpha. \end{cases}$$

Seen another way,

$$\delta = \begin{cases} 1 & \text{with probability } \alpha, \\ \eta^{\frac{1}{1-\alpha}} & \text{with probability } 1 - \alpha. \end{cases}$$

One can see that (F.5) satisfies the definition of a 3-dimensional H-function by taking the Laplace-Stieltjes (LS) transform of the log of (F.5) with uniform inputs and recognizing that it matches the LS transform of the log of a uniform distribution. For $\eta \sim$ U$(0, 1)$, the model (F.5) can also be written as

$$H(x, y, \eta) \overset{d}{=} \begin{cases} h(x, y)^{\alpha}, & \text{with probability } \alpha, \\ h(x, y)^{\alpha}\varepsilon, & \text{with probability } 1 - \alpha, \end{cases}$$

where ε is also a uniform random variable. When $\alpha = 1$, there is no "error;" when $\alpha = 0$, there is no degree correction, and increasing α increases the "signal-to noise-ratio." As a contrast to the additive "error" regime, in the failure case, when α is large, several different instances of the same network would share many of the *exact* same edge weights, as in expectation, $100 \times (1-\alpha)\%$ of the edge weights are given precisely by $h(x,y)^\alpha$. The reason to call this kind of error "failure" is that rather than defining a distribution that is concentrated near $h(x,y)$ with relatively small variation, even when α is large, there are infrequent occasions where the value will fall far below the modal value of $h(x,y)^\alpha$. The injected error will never raise the value greater than $h(x,y)^\alpha$, which also accounts for why the modal value lies at $h(x,y)^\alpha$ rather than $h(x,y)$. This kind of variation is reminiscent of each component in a system possessing a particular capacity, but occasional component failures cause that capacity to not be met. Figure 18 depicts different levels of σ and α being injected into the third plot of Figure 2.

Consider the example of a road network, where the vertices are geographic locations, edges are roads, and edge weights are the number of cars that travel along the road each day. In this case, there should be degree correction, as there should be heavier traffic between certain locations than others. However, in addition to random variation for travelers along each road (which could be represented by additive "error"), on some days, whether due to accidents, construction, or some other issue, the traffic along certain roads may fall dramatically. This latter case would be an example of failure. In this case, it may be better to model the network using a 4-dimensional H-function of the form

$$H(x,y,\eta_1,\eta_2) = \left(F_{1,2,3}\left(F_1(x)+F_2(y)+F_3(\eta_1)\right)\right)^\alpha \delta_2^{1-\alpha},$$

which would incorporate additive "error" through η_1, and failure through $\delta_2^{1-\alpha}$, which is a function of η_2. While this would be a valid 4-dimensional H-function in theory, in practice there may be issues arising from dependence between inputs, such as the probability of failure being correlated with the additive "error." Furthermore, one may want to include other covariates in the H-function which are not completely random, but rather systematic features, like time, that are different from node sociabilities, for use with tensors rather than matrices. There are many other kinds of H-functions that can incorporate various covariates and errors to model specific phenomena. Our goal, however, is not to catalog these possibilities, but to illustrate the richness and generality of the class of H-functions, particularly for generating random networks.

References

Bandeira, A. S. and van Handel, R. (2016), 'Sharp nonasymptotic bounds on the norm of random matrices with independent entries', *The Annals of Probability* **44**(4), 2479–2506.

Boucheron, S., Lugosi, G. and Massart, P. (2013), *Concentration Inequalities: A Nonasymptotic Theory of Independence*, Oxford University Press.

Hsu, D., Kakade, S. and Zhang, T. (2012), 'A tail inequality for quadratic forms of subgaussian random vectors', *Electronic Communications in Probability* **17**(52), 1–6.

Johnstone, I. M. (2001), Chi-square oracle inequalities, *in* M. de Gunst, C. Klaasen and A. van der Vaart, eds, 'State of the Art in Probability and Statistics, Festschrift for Willem Van Zwet, Lecture Notes-Monograph Series', Vol. 36, Institute of Mathematical Statistics, Lecture Notes, Monograph Series, pp. 399–418.

Ng, A. Y., Jordan, M. I. and Weiss, Y. (2001), On spectral clustering: Analysis and an algorithm, *in* 'Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic', pp. 849–856.

Noroozi, M., Rimal, R. and Pensky, M. (2021), 'Estimation and clustering in popularity adjusted block model', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* .

Pons, P. and Latapy, M. (2006), Computing communities in large networks using random walks, *in* 'J. Graph Algorithms Appl', Citeseer.

Vershynin, R. (2018), *High-Dimensional Probability: An Introduction with Applications in Data Science*, Vol. 47, Cambridge University Press.