Appendix for "Covariance-based sample selection for heterogeneous data: Applications to gene expression and autism risk gene detection"

Kevin Z. Lin

Carnegie Mellon University, Department of Statistics & Data Science, Pittsburgh, PA

Han Liu

Northwestern University, Department of Electrical Engineering and Computer Science, Evanston, IL

Kathryn Roeder

Carnegie Mellon University, Department of Statistics & Data Science, Pittsburgh, PA

A Code and dataset

The `R` code for replicating all analyses and figures in this article are hosted on GitHub in the repository `https://github.com/linnylin92/covarianceSelection`. The three major datasets used in this article are also included in the repository. The first dataset is the BrainSpan microarray samples collected by (Kang et al., 2011). While the original dataset is publicly available on GEO (`https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE25219`), we provide a locally preprocessed dataset, which was created to be amendable for our analysis in `R`. The second dataset is the older TADA scores (De Rubeis et al., 2014). The third dataset is the list of 102 risk genes detected using the newer TADA scores (Satterstrom et al., 2020).

B Brain region details

There are four primary brain regions, each containing smaller subregions.

- **PFC-MSC**: The prefrontal cortex (PFC) and primary motor-somatosensory cortex (MSC) consist of six smaller regions: primary motor cortex, primary somatosensory cortex, ventral prefrontal cortex, medial prefrontal cortex, dorsal prefrontal cortex and orbital prefrontal cortex.
- **V1C, ITC, IPC, A1C, STC**: A region consisting of the primary visual cortex (V1C), inferior temporal cortex (ITC), primary auditory cortex (A1C), and superior temporal cortex (STC).

- **STR, HIP, AMY**: A region consisting of the stratum (STR), hippocampal anlage or hippocampus (HIP) and amygdala (AMY).
- **MD, CBC**: A region consisting of the mediodorsal nucleus of the thalamus (MD) and the cerebellar cortex (CD).

C Extension to the Stepdown method

One of the largest drawbacks of the Stepdown method lies in its intensive computational cost. For r partitions, at most $\binom{r}{2}$ bootstrap statistics need to be computed in each bootstrap trial, each requiring a computational cost of $O(d^2 \cdot n_p)$. In this section, we develop a computational extension to the Stepdown method that yields a more computationally efficient algorithm as long as the test statistic $\widehat{T}$ satisfies the *triangle inequality*. That is, for any bootstrap trial b and for any partitions i, j and k, we require that the bootstrap statistics satisfy

$$\widehat{T}^{(b)}_{(i,k)} \leq \widehat{T}^{(b)}_{(i,j)} + \widehat{T}^{(b)}_{(j,k)}. \tag{C.1}$$

This property can potentially save expensive calculations when calculating (4.1) in Algorithm 2 by reducing the number of bootstrap statistics we need to explicitly calculate. Since we only care about the maximum bootstrap statistic $\widehat{T}^{(b)}$ in each trial, the triangle inequality gives an upper bound on the bootstrap statistic $\widehat{T}^{(b)}_{(i,k)}$ between partitions i and k, leveraging bootstrap statistics already calculated within a specific bootstrap trial. As we sequentially iterate through all pairs of partitions (i, k), if the upper bound for $\widehat{T}^{(b)}_{(i,k)}$ is smaller than the current maximum bootstrap statistic within a specific bootstrap trial b, we do not need to explicitly compute $\widehat{T}^{(b)}_{(i,k)}$.

Unfortunately, the test statistic (3.1) described in Section 3.1 originally from Chang et al. (2017) does not satisfy the triangle inequality (C.1). Hence, we consider a new test statistic defined as

$$\widehat{T} = \max_{ij} \left(\widehat{t}_{ij}\right) \quad \text{where } \widehat{t}_{ij} = \left|\widehat{\sigma}_{X,ij} - \widehat{\sigma}_{Y,ij}\right|, \quad i, j \in 1, \ldots, d, \tag{C.2}$$

and we make a similar modification for its bootstrap counterpart, $\widehat{T}^{(b)}$. It can easily be shown that the above bootstrap statistics satisfies the desired triangle inequality. Additionally, using the techniques in Chernozhukov et al. (2013), it can be proven that this test statistic will still yield a hypothesis test with asymptotic $1 - \alpha$ coverage under the null, analogous to (4.2). We will call the Stepdown procedure that uses (C.2) the "Accelerated Stepdown" procedure.

To formalize how to take advantage of this triangle inequality property, we describe a subroutine that leverages this property to compute $\widehat{T}^{(b)}$ in (4.1) by representing the individual bootstrap statistics $\widehat{T}^{(b)}_{(i,j)}$ as weighted edges in a graph. The algorithm uses Dijsktra's algorithm to find the shortest path between vertices. This implicitly computes the upper bound in the bootstrap statistic between two partitions using the triangle inequality. This algorithm can provide substantial improvement in computational speed by leveraging the fact that determining the shortest path on a fully-dense graph has a computational complexity of $O(r^2)$, whereas computing $T^{(b)}_{(i,j)}$ has a computational complexity of $O(d^2 \cdot n_p)$.

Algorithm 3: Distance metric-based procedure to compute $\widehat{T}^{(b)}$

1. Form graph $G = (V, E)$ with r nodes and all $\binom{r}{2}$ edges, and initialize each edge to have a weight of infinity.
2. Arbitrarily construct a spanning tree $\mathcal{T}$ and compute all $\widehat{T}^{(b)}_{(i,j)}$'s corresponding to edges $(i, j) \in \mathcal{T}$. Record $z = \max_{(i,j)\in\mathcal{T}} \widehat{T}^{(b)}_{(i,j)}$.
3. Construct a set of edges $\mathcal{S} = \mathcal{L}\backslash\mathcal{T}$ which represents the bootstrap statistics between specific pairs of partitions that have yet to be computed.
4. While $\mathcal{S}$ is not empty:
   (a) Arbitrarily select an edge $(i, j) \in \mathcal{S}$ and remove it from $\mathcal{S}$. Compute the shortest-path distance from vertex i to j in G.
   (b) If the shortest-path distance is larger than z, update the edge (i, j) to have weight $\widehat{T}^{(b)}_{(i,j)}$, and update z to be $\max(z, \widehat{T}^{(b)}_{(i,j)})$.
5. Return z.

As we will see in Section F, while the new test statistic (C.2) can take advantage of this computational speedup, it yields a much less powerful test when compared to test using (3.1). This is intuitive, as (C.2) does not normalize by the sum of the empirical variances, unlike (3.1). Hence, we do not use the Accelerated Stepdown procedure within COBS when analyzing the BrainSpan dataset in this paper. However, we believe there are potentially other settings outside of covariance testing where this computational speedup idea can be used more effectively. We leave this as direction for future work.

D Details of algorithms to find quasi-cliques

The first subsection remarks on possible extensions to the clique-based selection method described in Algorithm 4. The second subsection describes the three other algorithm used in Section 5 that we compare against. Throughout this section, for a generic graph G, we use V to denote the set of vertices in G, G_S to denote a subgraph formed by a vertex set $S \subseteq V$, and $E(G)$ to denote the number of edges in G.

D.1 Extensions to the clique-based selection method

We mention two extensions to clique-based selection method (Algorithm 4) that can be useful in practice.

- **Initializing the algorithm around a desired set of vertices**: In certain cases, the user would want the γ-quasi-clique to be initialized around a desired subset of vertices in G. For instance, in our setting, since Liu et al. (2015) applies DAWN to the 10 partitions in Window 1B, it is natural for us to encourage COBS to select as many partitions in Window 1B as possible to enable a meaningful comparison.

To achieve this, first, we run Algorithm 4 at the desired level γ on G_S. This would output a subset of vertices $S_{\text{core}} \subseteq S$ that form the largest γ-quasi-clique in G_S. Then, we run Algorithm 4 at the same level γ on the full graph G but perform an additional operation after Step 2 – after $\mathcal{Q}$ is initialized with all maximal cliques in G, we check each vertex set $A \in \mathcal{Q}$ if $A \cup S_{\text{core}}$ forms a γ-quasi-clique. If yes, we replace A with $A \cup S_{\text{core}}$ in $\mathcal{Q}$. If not, we remove A from $\mathcal{Q}$. The algorithm then proceeds to Step 3 as usual. By applying this simple change, we are ensured the returned vertex set by Algorithm 4 contains S_{core}.

- **Post-processing the returned vertex set**: In certain cases, the returned vertex set of Algorithm 4 has a few vertices with a very low degree when compared to the other vertices. To resolve this, we post-process this vertex set by removing vertices that are connected to less than half the other vertices in the returned set.

In this paper, we use the initialization extension only when analyzing the BrainSpan dataset in Section 6, where we initialize the largest quasi-clique around the 10 partitions in Window 1B.

D.2 Overview of other algorithms

We overview the three algorithms introduced in Section 5 that are designed to find large quasi-cliques.

- Chen and Saad (2010): This algorithm recursively splits a graph G into two in a hierarchical-clustering type approach with respect to a carefully constructed weight matrix. This forms a tree-type data structure, and then the algorithm scans the tree in a breath-first-search type fashion for the largest subgraph with an edge density larger than γ.

- Tsourakakis et al. (2013): This algorithm performs a local search by adding vertices mypoically and then removing vertices occasionally until no more myopic improvements can be made. Specifically, it first initializes the set S of vertices to contain a vertex that maximizes the ratio between the number of triangles and the degree, and includes all of the neighbors of said vertex. Then algorithm iteratively tries to incrementally improve the $f_\gamma(S) = E(G_S) - \gamma\binom{|S|}{2}$ as much as possible by adding neighbors of S. When it is no longer able to improve $f_\gamma(S)$, the algorithm tries removing a vertex from S to improve $f_\gamma(S)$. The algorithm then iterates between such adding and removing of vertices from S for a fixed number of iterations.

- Spectral clustering: While many different community detection methods for random graphs now exist (for example, see Abbe (2017) and Athreya et al. (2017) and the references within), we choose spectral clustering as described in Lei and Rinaldo (2015) as a prototypical example of how many of such methods fail to demonstrate the monotone property as described in Subsection 4.2. Specifically, this method applies K-means clustering to the top K eigenvectors of the adjacency matrix, where K is a tuning parameter to specify. To find large quasi-cliques, we iteratively try spectral clustering

for a range of K's (i.e., $K = 2, \ldots, 5$), and for each detected cluster in any of the estimated clusterings, we compute if the corresponding vertices of said cluster forms a γ-quasi-clique. If any γ-quasi-clique is found, we return the largest γ-quasi-clique discovered in this fashion.

E Formal description of simulation setup

We say a multivariate vector $\boldsymbol{X} \in \mathbb{R}^d$ is distributed based a nonparanormal distribution with proxy mean vector $\boldsymbol{\mu}$, proxy covariance matrix $\boldsymbol{\Sigma}$, and monotonic and differentiable functions $f_1, \ldots, f_d$ if the density of $\boldsymbol{X}$ is

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\Big\{ -\frac{1}{2}\big(f(\boldsymbol{x}) - \boldsymbol{\mu}\big)^\top \boldsymbol{\Sigma}\big(f(\boldsymbol{x}) - \boldsymbol{\mu}\big)\Big\} \prod_{j=1}^{d} \big|f_j'(x_j)\big|, \tag{E.1}$$

where $f(\boldsymbol{x}) = (f_1(x_1), \ldots, f_d(x_d))$. This is defined in Liu et al. (2009). In our simulation suite, we set $\boldsymbol{\mu} = 0$. Let this distribution be denoted as $\mathrm{NPN}(\mathbf{0}, \boldsymbol{\Sigma}, f)$. In the next two subsections, we formalize the details of $\boldsymbol{\Sigma}$ and $f_1, \ldots, f_d$.

E.1 Details on proxy covariance matrices $\boldsymbol{\Sigma}$

The following three bullet points detail the construction of $\boldsymbol{\Sigma}^{(1)}$, $\boldsymbol{\Sigma}^{(2)}$ and $\boldsymbol{\Sigma}^{(3)}$ respectively. As mentioned in Section 5, $\beta \in [0, 1]$ is a user-defined parameter that controls the dissimilarity among these three matrices.

- **Construction of $\boldsymbol{\Sigma}^{(1)}$**: As mentioned in Section 5, $\boldsymbol{\Sigma}^{(1)} \in \mathbb{R}_+^{d\times d}$ follows an SBM with two equally-sized clusters. Specifically, the first cluster contains indices $1, \ldots, \lfloor d/2 \rfloor$ and the second cluster contains indices $\lfloor d/2 \rfloor + 1, \ldots, d$. Then, we construct $\boldsymbol{\Sigma}^{(1)}$ where

$$\Sigma_{ij}^{(1)} = \begin{cases} 1 & \text{if } i = j, \\ a & \text{if } i \neq j,\ i \text{ is in the same cluster as } j, \\ b & \text{if } i \neq j,\ i \text{ is not in the same cluster as } j, \end{cases} \tag{E.2}$$

  for all $i, j \in 1, \ldots, d$ and $a = 0.9$ and $b = 0.1$.

- **Construction of $\boldsymbol{\Sigma}^{(2)}$**: $\boldsymbol{\Sigma}^{(2)}$ is constructed the same as $\boldsymbol{\Sigma}^{(1)}$, except

$$a = 0.9 - \beta \cdot 0.4, \quad \text{and} \quad b = 0.1 + \beta \cdot 0.4.$$

  When $\beta = 1$, this means that $\boldsymbol{\Sigma}^{(2)}$ is a matrix with 0.5 everywhere along the off-diagonal.

- **Construction of $\boldsymbol{\Sigma}^{(3)}$**: $\boldsymbol{\Sigma}^{(3)}$ is constructed in a similar way to $\boldsymbol{\Sigma}^{(1)}$, except there are three clusters. The first cluster contains indices $1, \ldots, \lfloor \beta \cdot d/6 \rfloor, \lfloor d/2 \rfloor + 1, \ldots, \lfloor d/2 + \beta \cdot d/6 \rfloor$. The second cluster contains indices $\lfloor \beta \cdot d/6 \rfloor + 1, \ldots, \lfloor d/2 \rfloor$. The third cluster contains indices $\lfloor d/2 + \beta \cdot d/6 \rfloor + 1, \ldots, d$. Observe that this partitions $1, \ldots, d$, and

when $\beta = 1$, this results in three clusters of roughly the same size. We then construct $\mathbf{\Sigma}^{(3)}$ like in (E.2) but using these three clusters.

E.2 Details on functions $f_1, \ldots, f_d$

At a high-level, the functions $f_1, \ldots, f_d$ ensure that these marginal distributions of our sampled nonparanormal random variables are similar to the marginal distributions of the BrainSpan data. These marginal distributions are constructed in the following way. We first randomly sample d variables (i.e., genes) uniformly from the BrainSpan dataset, $g_1, \ldots, g_d$. Next, for each j, let $\widehat{p}_{g_j}$ denote the kernel density estimate of variable g_j in the BrainSpan dataset, using the default bandwidth selection used by the `stats::density` function in `R`.

We now formalize how to construct $f_1, \ldots, f_d$. As described in Liu et al. (2009), we actually construct the inverse of these functions $f_1^{-1}, \ldots, f_d^{-1}$ as they are more amendable for sampling, which must exist since $f_1, \ldots, f_d$ are monotonic and differentiable. Recall that $\boldsymbol{\mu} = \mathbf{0}$. We first sample a vector $\boldsymbol{z} = (z_1, \ldots, z_d)$ from a Gaussian distribution $N(\mathbf{0}, \mathbf{\Sigma})$. Let $\Phi(t; P)$ denote the cumulative distribution function evaluated at t for a univariate density P. For any $j \in \{1, \ldots, d\}$, we construct f_j^{-1} such that

$$\Phi\big(t; N(0, \Sigma_{jj})\big) = \Phi\big(f_j^{-1}(t); \widehat{p}_{g_j}\big), \quad \forall t \in \mathbb{R}.$$

That is, we construct f_j^{-1} so that z_j is at the same quantile with respect to $N(0, \Sigma_{jj})$ as $f_j^{-1}(z_j)$ is with respect to the kernel density estimate $\widehat{p}_{g_j}$. Notice that by constructing $f_1^{-1}, \ldots, f_d^{-1}$ in this fashion, each function is monotone and differentiable. We then set

$$\boldsymbol{x} = \big(x_1, \ldots, x_d\big) = \big(f_1^{-1}(z_1), \ldots, f_d^{-1}(z_d)\big)$$

as one sample from the nonparanormal distribution $\text{NPN}(\mathbf{0}, \mathbf{\Sigma}, f)$.

Notice that by introducing non-Gaussianity into our simulation suite in this fashion, we ensure that the marginal distribution of all r partitions resemble the BrainSpan dataset, and also ensure that the first r_1 partitions still are drawn from the same population covariance matrix. Also, by generating data in this fashion, we are able to obtain complicated dependencies between the mean and variance, as well as observe multi-modal distributions and heavier-tailed distributions compared to the Gaussian. See Liu et al. (2009) for a more detailed discussion.

E.3 Example of sampled nonparanormal distribution

We provide a visual illustration of what the sampled nonparanormal distribution could look like. We sample 375 samples from $\text{NPN}(\mathbf{0}, \mathbf{\Sigma}^{(1)}, f)$ when $\beta = 0$, and plot two of the resulting pairwise scatterplots in Figure S.1. We can think of the 375 samples as equivalent to aggregating all $r = 25$ partitions together, each having $n = 15$ samples. These two scatterplots show that the nonparanormal can display multiple modes marginally or heavier tails.

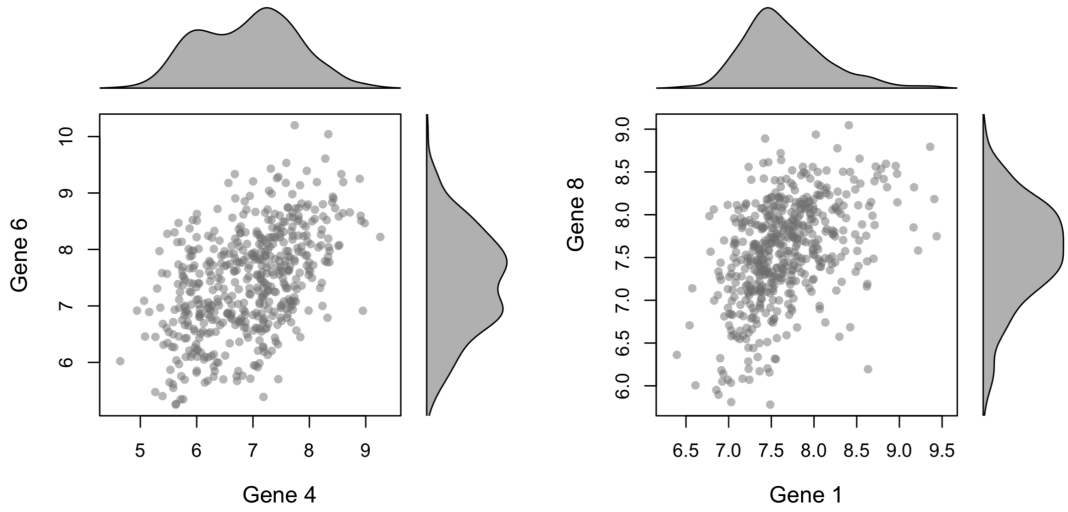

Figure S.1: Two scatter plots of bivariate distributions sampled from the nonparanormal for $\beta = 0$. The densities shown on the top and the right of each plot represents the targeted kernel density estimates from the BrainSpan data that the nonparanormal is sampling from, captured by $f_1, \ldots, f_d$.

F Additional simulation results

F.1 Covariance homogeneity diagnostic in simulation

In this subsection, we apply the diagnostic developed in Section 3 to the simulation suite described in Section 5. Our goal is to determine how the QQ-plots behave as the selected partitions become less homogeneous. As done in Section 5, we consider four partition selection strategies: COBS (using $\alpha = 0.1$ and $\gamma = 0.95$), Base (which selects 3 partitions containing samples drawn from the nonparanormal distribution with proxy covariance $\mathbf{\Sigma}^{(1)}$, and other 2 partitions containing samples from each of the remaining two distributions), All (which selects all r partitions) and Oracle (which selects exactly the r_1 partitions containing samples drawn the nonparanormal distribution with proxy covariance $\mathbf{\Sigma}^{(1)}$).

We see in Figure S.2 and Figure S.3 that the QQ-plot is a reasonable diagnostic in this simulation suite. Between these two figures, we vary β among 0, 0.3, 0.6 and 1. We notice that as β increases, the QQ-plot derived from COBS remains relative uniform, similar to that of the Oracle. When $\beta = 1$, COBS selects one erroneous partition in this particular trial shown, which results in the QQ-plot showing a deviation away from uniform. The QQ-plots derived from the Base procedure looks relative uniform when $\beta = 0$ (which is to be expected, as all r partitions share the same covariance matrix when $\beta = 0$), but quickly has QQ-plots that deviate from uniform as β increases. Note that the since the Base procedure selects only 5 partitions, there are a limited number of ways to split the partitions into two groups, which yields a limited number of points in the QQ-plot. The QQ-plots derived from the All procedure follow a similar trend as the Base procedure, but not as severe. These plots match the findings shown in Figure 8.

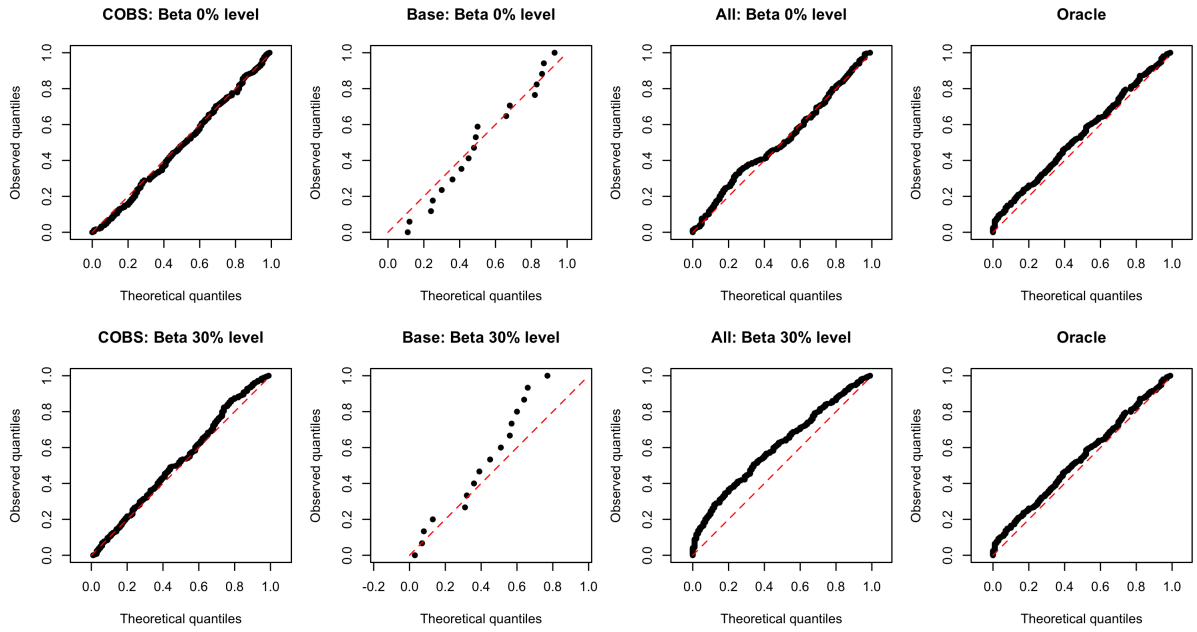

Figure S.2: QQ-plots from the covariance homogeneity diagnostic using four different selection procedures: COBS (left-most), Base (center left), All (center right) and Oracle (right-most). The top row represents the simulation setting where $\beta = 0$, while the second row represents the simulation setting where $\beta = 0.3$. The plots are created from one instance of COBS, Base, All and Oracle procedures, and 250 trials are used within the covariance homogeneity diagnostic.

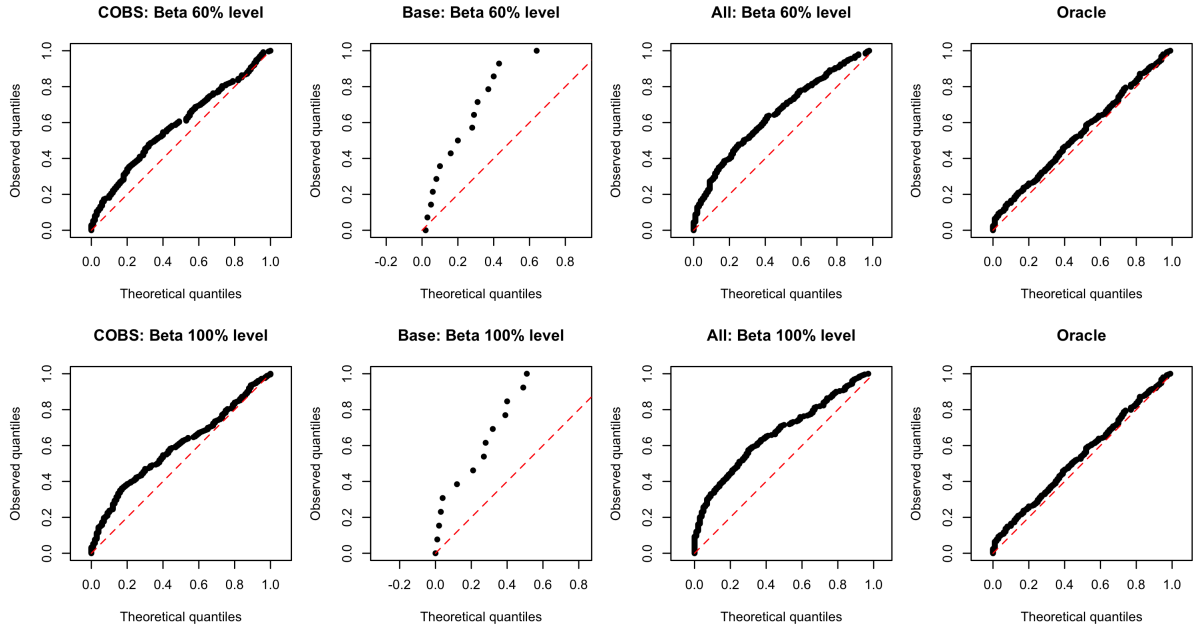

Figure S.3: QQ-plots derived in a similar way as in Figure S.3. However, in this plot, the top row represents the simulation setting where $\beta = 0.6$, while the second row represents the simulation setting where $\beta = 1$.

F.2 Simulation under Gaussian setting

While the simulations in Section 5 use nonparanormal distributions, we demonstrate that similar results hold for Gaussian distributions. This demonstrates that there is nothing particularly special about the nonparanormal or the Gaussian distribution that enable COBS to work well, and suggests COBS can work in much more general settings. Specifically, in this simuation suite, everything is the same as in Section 5, except all the functions $f_1, \ldots, f_d$ are set to be the identity function. Hence, this means that the first r_1 partitions are drawn from Gaussian distributions with covariance $\mathbf{\Sigma}^{(1)}$, the next r_2 partitions are drawn from Gaussian distributions with covariance $\mathbf{\Sigma}^{(2)}$, and so on.

When we use Bonferroni or the Stepdown method in this Gaussian setting, we observe ROC curves for the individual hypotheses that strongly resemble Figure 6. This is shown in Figure S.4.

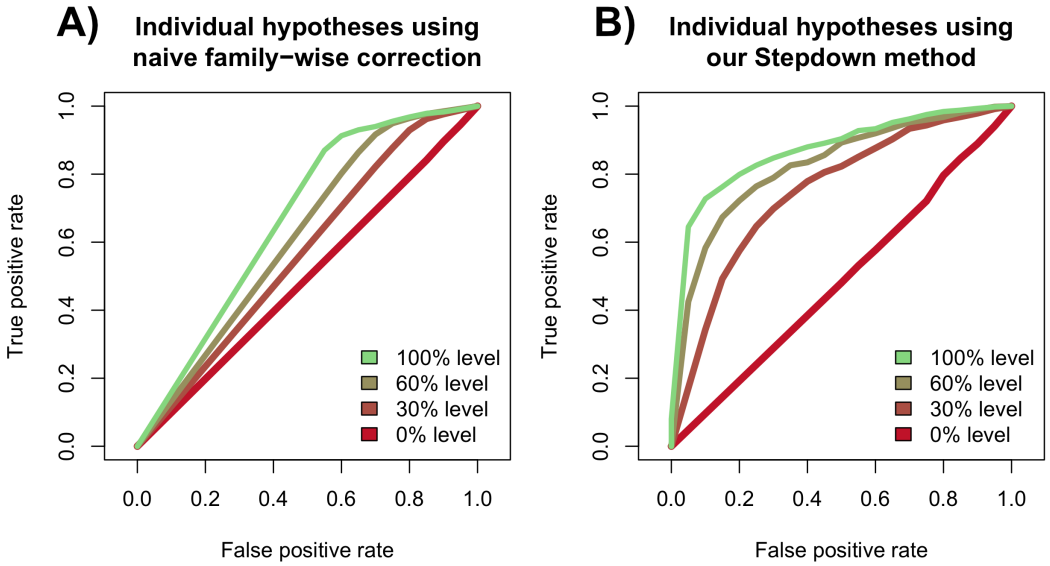

Figure S.4: ROC curves for the hypothesis under the Gaussian setting. These plots are set up in the same way as in Figure 6.

Similarly, when we use COBS to select partitions, the ROC curves as well as the spectral error curves strongly resemble Figure 8A and B. This is shown in Figure S.5.

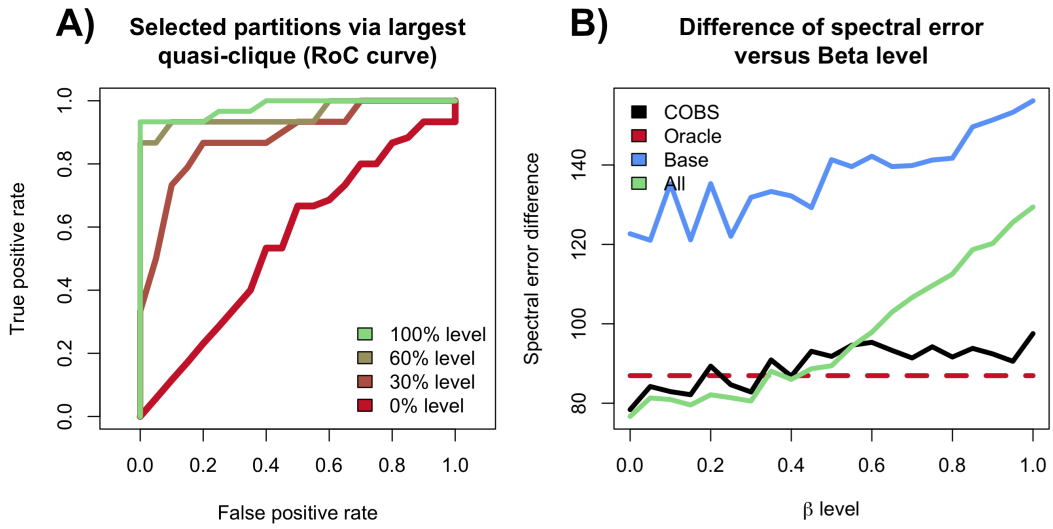

Figure S.5: A) ROC curves for the partitions selected by COBS under the Gaussian setting. This plot is set up in the same way as in Figure 8A. B) Mean spectral error of the estimated covariance matrix for varying β under the Gaussian setting. This plot is set up in the same way as in Figure 8B.

F.3 Simulation using Accelerated Stepdown

In this subsection, we apply the Accelerated Stepdown procedure described in Section C within the COBS procedure in the simulation setting described in Section 5. Specifically, we use the test statistic (C.2) and analogous bootstrap statistics, but keep all other parts of the simulation suite the same.

When we plot the ROC curve for the individual hypotheses in Figure S.6, we already notice a dramatic loss of power when compared to its original counterpart using the test statistic (3.1) shown in Figure 6. In fact, it seems like the Bonferroni procedure has almost no power at all, even when $\beta = 1$.

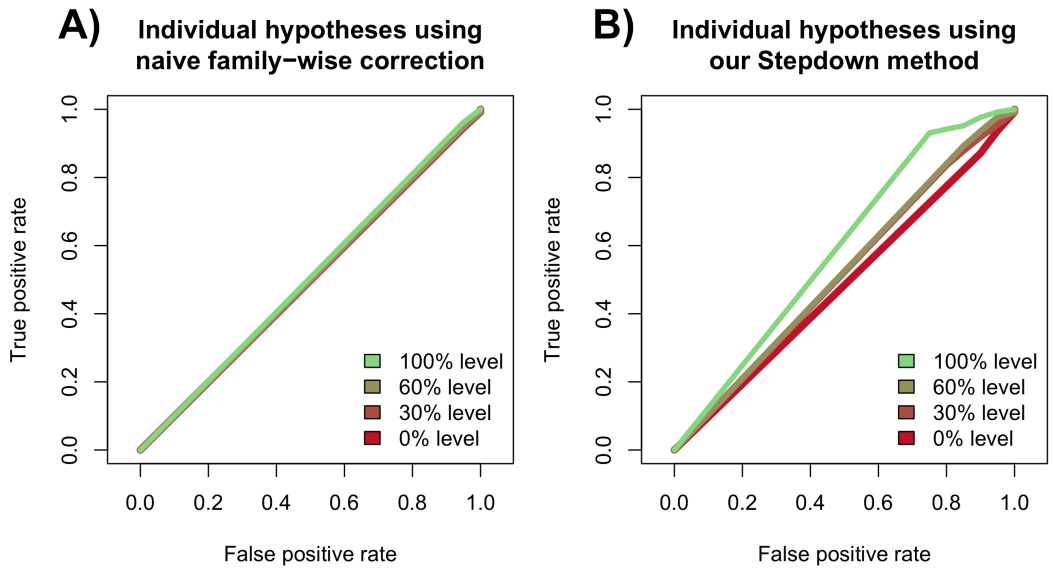

Figure S.6: ROC curves for the hypothesis using the Accelerated Stepdown procedure described in Section C in the nonparanormal setting. These plots are set up in the same as in Figure 6.

Due to the loss of power for the individual hypotheses, we observe a loss of power for the selected partitions as well (Figure S.7A) and spectral errors that strongly resemble selecting all the partitions (Figure S.7B).

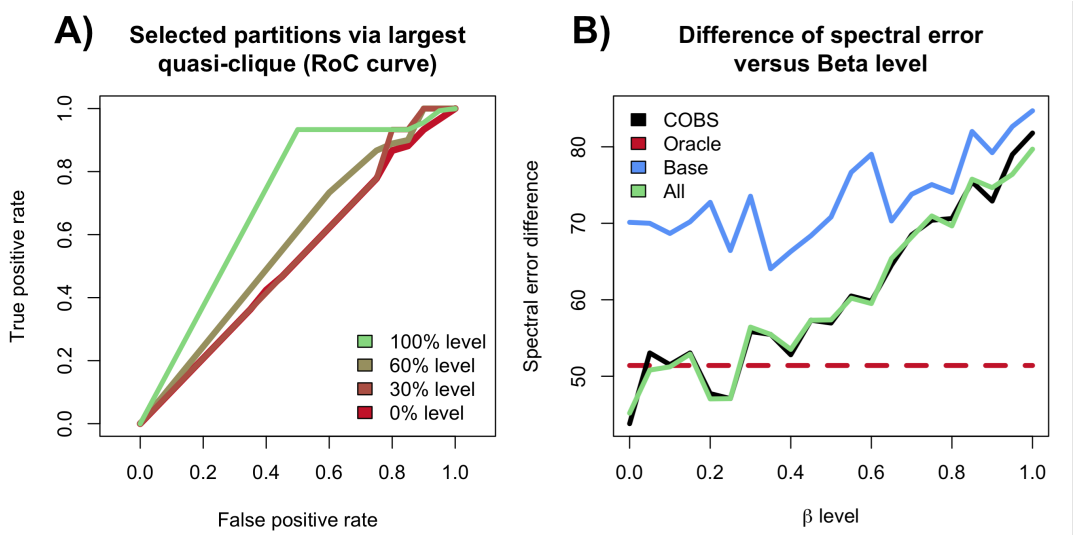

Figure S.7: A) ROC curves for the partitions selected by COBS using the Accelerated Stepdown procedure described in Section C in the nonparanormal setting. This plot is set up in the same way as in Figure 8A. B) Mean spectral error of the estimated covariance matrix for varying β using the Accelerated Stepdown procedure in the nonparanormal setting. This plot is set up in the same way as in Figure 8B.

G Additional details on BrainSpan analysis

The first subsection describes the analysis pipeline we used throughout Section 6 in more detail. The second subsection describes the two distance metrics used in Subsection 6.4. The third subsection describes additional results alluded to in Subsection 6.4.

G.1 Description of analysis pipeline

We now summarize the pipeline used in Section 6 for clarity.

1. **Screening of genes**: This is the step described in Subsection 6.1, derived from Liu et al. (2015). We first select all genes whose p-value in the older TADA dataset (De Rubeis et al., 2014) is less than 0.01. Then, we rank all remaining genes by their maximum Pearson correlation in magnitude with any of the formerly selected genes in decreasing order based on the BrainSpan partitions within Window 1B aggregated. We select genes based on this ranking in order until we have selected a combined total of $d = 3500$ genes. We analyze all 125 partitions (i.e., all partitions with 5 or more samples) using only these d genes for the remainder of the analysis.

2. **Applying COBS**: This is the two-staged procedure we developed in this paper, detailed in Section 4. In the first stage, we apply the Stepdown procedure using $\alpha = 0.1$. In the second stage, we select the clique-based selection method where $\gamma = 0.95$, as well as using both extensions discussed in Subsection D.1. This results in 24 selected partitions within the BrainSpan dataset, as detailed in Subsection 6.2. We then combine all the 24 selected partitions to form a dataset $\mathbb{X} \in \mathbb{R}^{n \times d}$ with $n = 272$ microarray samples and d genes to be used for the remainder of the analysis.

3. **Estimating the Gaussian graphical model**: This step is described in Subsection 6.3 and is the same as in Liu et al. (2015). We fit a Gaussian graphical model using neighborhood selection (Meinshausen and Bühlmann, 2006) based on $\mathbb{X}$, where the tuning parameter λ (which controls the sparsity of the graphical model) is chosen such that the resulting graph has high scale-free index as well as a comparable number of edges to the estimated graph when COBS is not used. This choice of λ is detailed at the end of this subsection. We defer the remaining estimation and computation details to Liu et al. (2015). We denote the adjacency matrix of the estimated graphical model as $\widehat{\boldsymbol{A}} \in \{0,1\}^{d \times d}$.

4. **Estimating the HMRF**: This step is also described in Subsection 6.3 and is the same as in Liu et al. (2015). We briefly summarize this step here, as it is less common in the statistical literature. Let $\boldsymbol{Z} \in \mathbb{R}^d$ denote the Z-scores for the selected genes, derived from the TADA scores in De Rubeis et al. (2014). We model $\boldsymbol{Z}$ using a HMRF, where for each gene in $j \in \{1, \ldots, d\}$, Z_j is an i.i.d. random variable drawn from a mixture of two Gaussians,

$$Z_j \sim \mathbb{P}(I_j = 0)N(0, \sigma^2) + \mathbb{P}(I_j = 1)N(\mu, \sigma^2),$$

where $I_j \in \{0,1\}$ is an unobserved Bernoulli random variable and $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}_+$ are two unknown scalars to be estimated. The first Gaussian distribution represents the Z-scores for genes that are not associated with ASD, and the second Gaussian distribution represents the Z-scores for risk genes. The distribution of entire vector $\boldsymbol{I} \in \{0,1\}^d$ follows an Ising model with probability mass function,

$$\mathbb{P}(\boldsymbol{I} = \boldsymbol{\eta}) \propto \exp\left(b \cdot \sum_{j=1}^{d} \eta_j + c \cdot \boldsymbol{\eta}^T \widehat{\boldsymbol{A}} \boldsymbol{\eta}\right),$$

for any $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_d) \in \{0,1\}^d$ and two unknown scalars $b, c \in \mathbb{R}$ to be estimated. An EM algorithm is used to fit this HMRF model, and we obtain the estimated posterior probability $\widehat{p}_j = \mathbb{P}(I_j = 0 | \boldsymbol{Z})$, representing the probability gene j is not a risk gene given the risk scores. We defer the estimation and computation details to Liu et al. (2015).

5. **Applying Bayesian FDR**: This step is also described in Subsection 6.3 and is the same as in Liu et al. (2015). We apply a procedure (Muller et al., 2006) to $\widehat{\boldsymbol{p}} = (\widehat{p}_1, \ldots, \widehat{p}_d)$ to select a set of genes where the Bayesian FDR is controlled at level 10%. We defer the computation details to Liu et al. (2015). This results in the set of 209 detected risk genes detailed in Subsection 6.4.

Usage of De Rubeis et al. (2014). We note that the older risk scores dataset (De Rubeis et al., 2014) is used twice, once in the screening stage (Step 1 above) and again to estimate the parameters of the HMRF (Step 4 above). As argued by Liu et al. (2015), it is important for this dataset to be the same in both steps, as the goal of DAWN is to boost the power of the risk scores by a "guilt-by-association" strategy. Hence, it is important to ensure the genes with low TADA scores remain in the analysis after screening, so they can implicate genes with TADA scores that are not as low.

Choice of λ. We use the following procedure to tune λ when estimating the Gaussian graphical model using only the 10 partitions from Window 1B as well as when using the 24 partitions selected by COBS. We tune λ on a grid between 0.05 and 0.1, equally spaced into 15 values, for both graphical models. Our criteria for selecting λ within this grid is inspired by Liu et al. (2015), who use a *scale-free index*, a number between 0 and 1 that measures how well the graph follows a power law. Specifically, we ensure the scale-free indices from both graphical models are approximately comparable as well as that both estimated graphical models have about 10,000 edges. Our focus on this number of edges comes from Liu et al. (2015), which estimated a graphical model with 10,065 edges. By ensuring both of our estimated graphical models have around 10,000 edges, we are able to ensure that both graphical models pass roughly the same amount of information into the HMRF stage of DAWN.

Using this procedure, we set $\lambda = 0.05$ when estimating the graphical model using only the 10 partitions from Window 1B (for 9990 edges and a scale-free index of 0.77) and $\lambda = 0.064$ when estimating the graphical model using the 24 partitions selected by COBS (for 9142

edges and scale-free index of 0.83).

G.2 Methods to measure distance of two nodes in a graph

As alluded to in Subsection 6.4, the shortest path distance and the commute distance do not seem like appropriate candidates to measure the distance between two genes (i.e., vertices) in a gene co-expression network (i.e., graph) due to the fact that the network estimated in the Window 1B analysis has more edges than in the COBS analysis (9990 and 9142 edges respectively). Hence, both of these distance metrics would naturally favor the denser graph.

To overcome this problem, we use two distance metrics that we believe enable a more fair comparison.

- **Minimal spanning tree (MST) distance**: This is a natural alternative to measure the distance between two vertices. Given a graph $G = (V, E)$, we first find the MST $G_{(\text{MST})} \subseteq G$, and then compute the path distance between the two vertices in $G_{(\text{MST})}$.
- **Graph root embedding distance**: A more statistically motivated way to measure the distance between two vertices is to first embed all vertices V into a latent space. As shown in Lei (2018), the graph root embedding is a natural candidate to do this, as it can theoretically represent a wide range of random graphs. This is essentially a more sophisticated spectral embedding. We first represent the graph G as an adjacency matrix $\boldsymbol{A}$, and compute the top-k eigenvectors (corresponding both the largest k positive eigenvalues and largest k negative eigenvalues in magnitude). Each vertex is then represented as a latent vector of length $2k$. The distance between two vertices is then defined as the Euclidean distance between their corresponding latent vectors. We defer the remaining details to Lei (2018).

  It is important to use both positive and negative eigenvalues since a scree plot reveals there are almost the same number of positive and negative eigenvalues for the adjacency matrices estimated in both the COBS and Window 1B analyses.

G.3 Additional results about closeness of genes

We provide more details that the 102 genes detected by the newer TADA scores (Satterstrom et al., 2020) are roughly 10%-30% closer to the 33 genes detected in the older TADA scores (De Rubeis et al., 2014) in the gene network estimated in the COBS analysis than in the Window 1B analysis. We call the 33 genes detected in De Rubeis et al. (2014) as the De Rubeis genes, and the 102 genes detected in Satterstrom et al. (2020) that are not part of the former 33 genes as the Satterstrom genes.

We use the MST distance defined above to ask: how far away are the closest k De Rubeis genes from any Satterstrom gene on average (mean). Figure S.8A plots this average distance against k. We use the graph root embedding distance to ask: how close is the nearest De Rubeis gene from any Satterstrom gene on average (mean) when using an embedding of latent dimension $2k$. Figure S.8B plots this average distance against k. In both instances, regardless of how the parameter k is chosen, the plot shows that the Satterstrom genes are

closer to the De Rubeis genes on average. Both metrics show that the red curve is roughly 10%-30% lower than the pale curve across all values of k, hence giving our stated result.

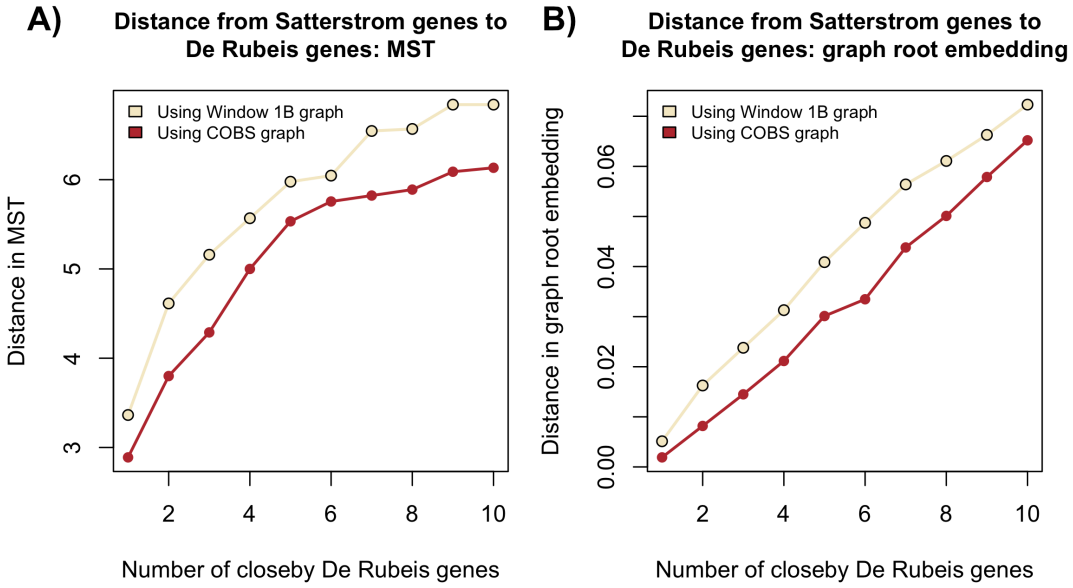

Figure S.8: A) Average MST distance from a Satterstrom gene to the closest k De Rubeis genes against k. B) Average graph root embedding distance from a Satterstrom gene to the closest De Rubeis genes against the half of the embedding dimension k.

References

Abbe, E. (2017). Community detection and stochastic block models: Recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531.

Athreya, A., Fishkind, D. E., Tang, M., Priebe, C. E., Park, Y., Vogelstein, J. T., Levin, K., Lyzinski, V., and Qin, Y. (2017). Statistical inference on random dot product graphs: A survey. *The Journal of Machine Learning Research*, 18(1):8393–8484.

Chang, J., Zhou, W., Zhou, W.-X., and Wang, L. (2017). Comparing large covariance matrices under weak conditions on the dependence structure and its application to gene clustering. *Biometrics*, 73(1):31–41.

Chen, J. and Saad, Y. (2010). Dense subgraph extraction with application to community detection. *IEEE Transactions on knowledge and data engineering*, 24(7):1216–1230.

Chernozhukov, V., Chetverikov, D., Kato, K., et al. (2013). Gaussian approximations and multiplier bootstrap for maxima of sums of high-dimensional random vectors. *The Annals of Statistics*, 41(6):2786–2819.

De Rubeis, S., He, X., Goldberg, A. P., Poultney, C. S., Samocha, K., Cicek, A. E., Kou, Y., Liu, L., Fromer, M., Walker, S., et al. (2014). Synaptic, transcriptional and chromatin genes disrupted in autism. *Nature*, 515(7526):209–215.

Kang, H. J., Kawasawa, Y. I., Cheng, F., Zhu, Y., Xu, X., Li, M., Sousa, A. M., Pletikos, M., Meyer, K. A., Sedmak, G., et al. (2011). Spatio-temporal transcriptome of the human brain. *Nature*, 478(7370):483–489.

Lei, J. (2018). Network representation using graph root distributions. *arXiv preprint arXiv:1802.09684*.

Lei, J. and Rinaldo, A. (2015). Consistency of spectral clustering in stochastic block models. *The Annals of Statistics*, 43(1):215–237.

Liu, H., Lafferty, J., and Wasserman, L. (2009). The Nonparanormal: Semiparametric estimation of high-dimensional undirected graphs. *The Journal of Machine Learning Research*, 10:2295–2328.

Liu, L., Lei, J., and Roeder, K. (2015). Network assisted analysis to reveal the genetic basis of autism. *The Annals of Applied Statistics*, 9(3):1571–1600.

Meinshausen, N. and Bühlmann, P. (2006). High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, pages 1436–1462.

Muller, P., Parmigiani, G., and Rice, K. (2006). FDR and Bayesian multiple comparisons rules. In *Bayesian Statistics*, volume 8. Oxford University Press.

Satterstrom, F. K., Kosmicki, J. A., Wang, J., Breen, M. S., De Rubeis, S., An, J.-Y., Peng, M., Collins, R., Grove, J., Klei, L., et al. (2020). Large-scale exome sequencing study implicates both developmental and functional changes in the neurobiology of autism. *Cell*, 180(3):568–584.

Tsourakakis, C., Bonchi, F., Gionis, A., Gullo, F., and Tsiarli, M. (2013). Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 104–112. ACM.