RCFGL: Rapid Condition adaptive Fused Graphical Lasso and application to modeling brain region co-expression networks

Souvik Seal^{1*}, Qunhua Li², Elle Butler Basner², Laura M. Saba³, Katerina Kechris¹

1 Department of Biostatistics and Informatics, Colorado School of Public Health, University of Colorado Anschutz Medical Campus, Aurora, Colorado, USA

2 Department of Statistics, Pennsylvania State University, Centre County, Pennsylvania, USA

3 Skaggs School of Pharmacy and Pharmaceutical Sciences, University of Colorado Anschutz Medical Campus, Aurora, Colorado, USA

*souvik.seal@cuanschutz.edu

Abstract

Inferring gene co-expression networks is a useful process for understanding gene regulation and pathway activity. The networks are usually undirected graphs where genes are represented as nodes and an edge represents a significant co-expression relationship. When gene-expression data from multiple conditions (e.g., treatments, tissues, strains) are available, joint estimation of networks harnessing shared information across them can significantly increase the power of analysis. In addition, examining condition-specific patterns of co-expression can provide insights into the underlying cellular processes activated in a particular condition. Condition adaptive fused graphical lasso (CFGL) is an existing method that incorporates condition specificity in a fused graphical lasso (FGL) model for estimating multiple co-expression networks. However, the current implementation of CFGL is prohibitively slow even for a moderate number of genes and can only be used for a maximum of three conditions. In this paper, we propose a fast alternative of CFGL known as rapid condition adaptive fused graphical lasso (RCFGL). In RCFGL, we incorporate the condition specificity into another popular model for joint network estimation, known as fused multiple graphical lasso (FMGL). We use a more efficient algorithm in the iterative steps compared to CFGL, enabling faster computation and making it easily generalizable for more than three conditions. We also present a novel screening rule to determine if the full network estimation problem can be broken down into estimation of smaller disjoint sub-networks, thereby reducing the complexity further. We demonstrate the computational advantage and superior performance of our method compared to two non-condition adaptive methods, FGL and FMGL, and one condition adaptive method, CFGL in several simulation scenarios. We use RCFGL to jointly estimate the gene co-expression networks of different brain regions (conditions) using a cohort of heterogeneous stock rats. We also provide an accommodating *C* and *Python* based package that implements RCFGL.

Author summary

Inferring gene co-expression networks can be useful for understanding pathway activity and gene regulation. While jointly estimating co-expression networks of multiple

conditions, taking into account condition specificity such as information about an edge being present only in a specific condition, substantially increases the power. In this paper, a computationally rapid condition adaptive method for jointly estimating gene co-expression networks of multiple conditions is proposed. The novelty of the method is demonstrated through a broad range of simulation studies and a real data analysis with multiple brain regions from a genetically diverse cohort of rats.

Introduction

A gene co-expression network is an undirected graph, where each node corresponds to a gene, and gene pairs are connected with an edge if they share a significant co-expression relationship [1–3]. Gene co-expression network analysis is a useful tool for uncovering the complex molecular interplay in biological processes [4–7]. Fitting Gaussian graphical models (GGM) is a popular approach for constructing biological networks in various applications [8–13]. In the context of gene co-expression network analysis, GGM assumes a multivariate normal distribution between the expression profiles of a set of genes [14]. The estimate of the inverse of the covariance matrix (also known as “precision matrix”) is then examined to find which pairs of genes have significant conditional dependence and the co-expression network is constructed based on the dependence structure. The nonzero off-diagonal elements of the estimated precision matrix represent edges in the network.

Numerous approaches [15–21] have focused on the estimation of the aforementioned precision matrix. In most realistic scenarios, the number of genes (p) is much larger than the number of samples (n). It compels the researchers to use some form of regularization to induce sparsity in the estimation of the p -dimensional precision matrix. Yuan and Lin [16], Banerjee et al. [17], Friedman et al. [18], considered a penalized maximum likelihood model with ℓ_1 regularization, known as graphical lasso (GL). Solving the GL model is a constrained convex optimization problem. Alternating direction method of multipliers (ADMM) [22–28] is a widely popular algorithm for solving constrained convex optimization problems. Different variations of ADMM have been used to solve the GL problem [29–34].

In a multi-condition gene co-expression study, the co-expression profiles across K conditions are available [35–37]. It is of interest to find out how similar or dissimilar the co-expression networks are across different conditions. For example, a particular co-expression network module can be present in a tumor tissue but not in a healthy tissue and thus, can serve as a key tool in identification of the tissue type. A joint analysis of co-expression networks (more generally, any graphical network) harnessing shared information across different conditions can be significantly more powerful than individual analyses [38, 39]. Fused graphical lasso (FGL) [40] is one of the most popular approaches for joint estimation of multiple graphical networks. As the name suggests, FGL is an extension of the GL model in the context of multiple conditions. It simultaneously estimates multiple precision matrices (corresponding to multiple conditions) by considering the sum of multiple GL likelihoods and further employs a pair-wise fused lasso penalty [41] across the conditions. A similar method named fused multiple graphical lasso (FMGL) was proposed by Yang et. al. [42]. FMGL considers a sequential fused lasso penalty across the conditions instead of the pairwise penalty considered in FGL. FGL and FMGL are equivalent when there are only two conditions. Both the methods use the iterative ADMM algorithm [22] for estimating the parameters. However, FMGL makes use of a very efficient intermediate step originally described in [43] that substantially speeds up the computation (from $O(K \log K)$ to $O(K)$).

The fused lasso penalties (both pair-wise and sequential) inherently assume that the precision matrices (consequently, the co-expression networks) in all conditions are

equally similar to each other. This assumption is rigid and may easily be violated in most real life scenarios. For example, tissues with two different tumor sub-types are expected to be more similar to each other than a healthy tissue. To account for such condition specific similarities and dissimilarities into the FGL framework, Lyu et. al. [44] developed condition adaptive fused graphical lasso (CFGL). The penalty term considered in CFGL is a modification of the pair-wise fused lasso penalty by incorporating binary weight matrices capturing condition-specificity. CFGL uses an iterative ADMM algorithm for estimating the parameters. However, the CFGL *R* package is limited because it can only accommodate a maximum of three conditions and is prohibitively slow even for a moderate number of genes ($p \approx 1000$). Thus, in a dataset with more than three conditions and a large number of genes, the CFGL *R* package is not scalable.

In this paper, we propose a new method named rapid condition adaptive fused graphical lasso (RCFGL) for jointly estimating multiple co-expression networks that takes into account condition specificity, is computationally rapid, and can handle more than three conditions. Similar to CFGL, we compute the binary weight matrices that capture pair-wise condition specificity. Instead of considering a pair-wise fused lasso penalty, as considered in CFGL, we incorporate the computed weight matrices with a sequential fused lasso penalty. In that sense, RCFGL is a condition adaptive extension of the FMGL algorithm. We use iterative ADMM algorithm for estimation of the parameters. As in FMGL, using a sequential fused lasso penalty enables us to solve an intermediate step efficiently using fast algorithms [43, 45, 46]. This particular step is one of the main reasons behind the computational hurdle faced in CFGL. The authors of FGL and FMGL both had proposed a set of necessary conditions that can be investigated prior to fitting the models to evaluate the existence of a block diagonal structure in the precision matrices to be estimated. It can drastically reduce the computational time since all the matrix operations of order $O(p^3)$ reduce to $\sum_{l=1}^M O(p_l^3)$ (where M is the total number of blocks with l -th block having size p_l). We have theoretically shown that the same set of conditions can also be used in the context of RCFGL further facilitating the computation. Through extensive simulation studies, we verified the robustness of our proposed method and demonstrated the computational advantage. We also analyzed the gene co-expression networks of three different brain regions from a dataset of heterogeneous stock rats. We have built a *C* and *Python* based package implementing RCFGL which is available here, <https://github.com/sealx017/RCFGL>.

Materials and methods

Suppose there are p genes with expression profiles available across K conditions and there are n_k individuals under each condition k . Let y_i^k denote the $p \times 1$ expression vector corresponding to the i -th individual under condition k . $\mathbf{Y}^{(k)} = (y_1^k, \dots, y_{n_k}^k)^T$ is the $n_k \times p$ matrix of expression vectors under condition k and $\bar{\mathbf{Y}}^{(k)}$ is the corresponding $1 \times p$ column mean vector. It is assumed that $y_1^k, \dots, y_{n_k}^k \in \mathbb{R}^p$, are independently and identically drawn from $N_p(\mu_k, \Sigma_k)$ where $\mu_k \in \mathbb{R}^p$ and $\Sigma_k \succ \mathbf{0}$ (the notation $\succ \mathbf{0}$ denotes positive-definiteness). Let $\Theta^{(k)} = \Sigma_k^{-1}$ denote the precision matrix under condition k . Upon estimating $\Theta^{(k)}$, the gene co-expression network would be constructed by representing the genes as nodes and conditional dependencies as edges in a graph. To be more specific, two genes i, j under condition k , will only be connected in the graph if and only if $\Theta_{ij}^{(k)} \neq 0$. Throughout the paper we use $\mathbf{1}_n$ to denote $n \times 1$ vector of all 1's. Next, we discuss the existing methods for estimating $\Theta^{(k)}$'s.

Review of methods

Fused graphical lasso and fused multiple graphical lasso

Fused graphical lasso (FGL) [40] and fused multiple graphical lasso (FMGL) [42] maximize the following penalized log-likelihood function,

$$\underset{\Theta^{(k)} \succ 0, k=1, \dots, K}{\text{maximize}} \sum_{k=1}^K n_k [\log(\det(\Theta^{(k)})) - \text{tr}(\mathbf{S}^{(k)} \Theta^{(k)})] - P(\Theta); \quad (1)$$

where $\mathbf{S}^{(k)} = (\mathbf{Y}^{(k)} - \mathbf{1}_{n_k} \bar{\mathbf{Y}}^{(k)})^T (\mathbf{Y}^{(k)} - \mathbf{1}_{n_k} \bar{\mathbf{Y}}^{(k)}) / n_k$ is the sample covariance matrix and $P(\Theta)$ is the penalty term with $\Theta = \{\Theta^{(1)}, \dots, \Theta^{(K)}\}$. As mentioned earlier, the only difference between FGL and FMGL is in the penalty term, $P(\Theta)$. FGL considers a pair-wise fused lasso penalty and FMGL considers a sequential fused lasso penalty which have the following forms,

$$P^{\text{FGL}}(\Theta) = \lambda_1 \sum_{i \neq j} \sum_{k=1}^K |\Theta_{ij}^{(k)}| + \lambda_2 \sum_{i \neq j} \sum_{k < k'}^K |\Theta_{ij}^{(k)} - \Theta_{ij}^{(k')}|;$$

$$P^{\text{FMGL}}(\Theta) = \lambda_1 \sum_{i \neq j} \sum_{k=1}^K |\Theta_{ij}^{(k)}| + \lambda_2 \sum_{i \neq j} \sum_{k=1}^{K-1} |\Theta_{ij}^{(k)} - \Theta_{ij}^{(k+1)}|;$$

where λ_1, λ_2 are non-negative tuning parameters. The first term of both $P^{\text{FGL}}(\Theta)$ and $P^{\text{FMGL}}(\Theta)$ is the lasso penalty used in the GL model [18] that controls the overall sparsity. The second term of both the penalties controls the similarity of the off-diagonal elements of the precision matrices between conditions. Note that the second term of $P^{\text{FMGL}}(\Theta)$ is different from that of $P^{\text{FGL}}(\Theta)$ since it only focuses on differences between two consecutive conditions. If there are only two conditions i.e., $K = 2$, $P^{\text{FGL}}(\Theta) = P^{\text{FMGL}}(\Theta)$. For $K = 3$, writing the penalties as functions of λ_1, λ_2 , we show that $P^{\text{FGL}}(\Theta, \lambda_1, \lambda_2) \leq P^{\text{FMGL}}(\Theta, \lambda_1, 2\lambda_2)$. For $K > 3$, we are able to establish a crude connection: $P^{\text{FGL}}(\Theta, \lambda_1, \lambda_2) \leq P^{\text{FMGL}}(\Theta, \lambda_1, \lfloor \frac{K^2}{4} \rfloor \lambda_2)$ (S1 Supplementary).

$P^{\text{FGL}}(\Theta)$ encourages the same level of similarity between all the pairs of conditions and $P^{\text{FMGL}}(\Theta)$ encourages the same level of similarity between each consecutive pair of conditions. However, these assumptions may be violated in practical scenarios. For example, two different subtypes of tumor tissues can be more similar to each other than to a healthy tissue. Therefore, ideally the penalty term should be such that it penalizes the difference between the tumor subtypes more than it penalizes the difference between one of the tumor subtypes and the healthy tissue. Lyu et. al. [44] addressed this issue by incorporating a special weight term into $P^{\text{FGL}}(\Theta)$ which is discussed in the next section.

Condition adaptive fused graphical lasso

Lyu et. al. [44] introduced binary screening matrices: $\mathbf{W}^{(kk')} = [[\mathbf{w}_{ij}^{(kk')}]]$ for $k \neq k'$ defined as,

$$\mathbf{w}_{ij}^{(kk')} = \begin{cases} 1 & \Theta_{ij}^{(k)} \text{ and } \Theta_{ij}^{(k')} \text{ are non-differential between conditions } k \text{ and } k' \\ 0 & \Theta_{ij}^{(k)} \text{ and } \Theta_{ij}^{(k')} \text{ are differential between conditions } k \text{ and } k'. \end{cases}$$

The weight matrices are included in $P^{\text{FGL}}(\Theta)$ to define a penalty function that takes into account condition-specificity,

$$P^{\text{CFGL}}(\Theta) = \lambda_1 \sum_{i \neq j} \sum_{k=1}^K |\Theta_{ij}^{(k)}| + \lambda_2 \sum_{i \neq j} \sum_{k < k'}^K \mathbf{w}_{ij}^{(kk')} |\Theta_{ij}^{(k)} - \Theta_{ij}^{(k')}|.$$

The weight matrices in real data are unknown. Lyu et. al. [44] estimated them by performing a hypothesis test [47] for evaluating differences between the conditions. The test determines if the ij -th entry of the precision matrices: $\Theta^{(k)}$ and $\Theta^{(k')}$ is differential. If the test is rejected, $w_{ij}^{kk'}$ is set to 0, otherwise, it is set to 1. Going back to the example of two tumor subtypes and a healthy tissue, suppose the ij -th element is non-differential between the tumor subtypes (let's denote them as condition 1 and 2) but is differential between each of the tumor subtypes and the healthy tissue (let's denote it as condition 3). The weight terms in this case will be, $w_{ij}^{12} = 1$, $w_{ij}^{23} = 0$, and $w_{ij}^{13} = 0$. As a consequence, $P^{\text{CFGL}}(\Theta)$ will penalize the difference between the tumor subtypes but not the difference between one of the tumor subtypes and the healthy tissue for the ij -th element.

Proposed method

Model

We propose to maximize the penalized log-likelihood from (1) with a new penalty term. We consider the binary weight matrices: $\mathbf{W}^{(kk')}$ discussed in the last section and infuse them to $P^{\text{FMGL}}(\Theta)$ instead of $P^{\text{FGL}}(\Theta)$ as in CFGL. Thus, the penalty term we propose has the following form,

$$P^{\text{RCFGL}}(\Theta) = \lambda_1 \sum_{i \neq j} \sum_{k=1}^K |\Theta_{ij}^{(k)}| + \lambda_2 \sum_{i \neq j} \sum_{k=1}^{K-1} w_{ij}^{(k,k+1)} |\Theta_{ij}^{(k)} - \Theta_{ij}^{(k+1)}|.$$

We name the method rapid condition adaptive fused graphical lasso (RCFGL) due to the computational rapidness it offers over CFGL. Note that when $K = 2$, RCFGL is equivalent to CFGL (since, $P^{\text{CFGL}}(\Theta) = P^{\text{RCFGL}}(\Theta)$). Denote the set of all weight matrices as, $\mathbf{W} = \{\mathbf{W}^{kk'} : k < k'\}$. For $K > 2$, writing the penalties as functions of $\lambda_1, \lambda_2, \mathbf{W}$, we show that $P^{\text{CFGL}}(\Theta, \lambda_1, \lambda_2, \mathbf{W}) \leq P^{\text{RCFGL}}(\Theta, \lambda_1, \lambda_2, \mathbf{W}^*)$, where $\mathbf{W}^* = \{\mathbf{W}^{*(k,k+1)} : k = 1, \dots, K-1\}$ is a set of slightly modified weight matrices (S1 Supplementary).

All the methods discussed so far consider penalty functions which are sums of two individual penalties: the first one being the standard lasso penalty controlling overall sparsity and the second one controlling similarity between conditions. The methods differ from each other only in terms of the second penalty term. In the second penalty term, FGL and CFGL consider all possible pairwise differences between the conditions, whereas FMGL and RCFGL consider only sequential differences. CFGL and RCFGL take into account condition specificity by incorporating weights, whereas FGL and FMGL are not condition adaptive (Table 1).

ADMM algorithm for solving the model

We use an iterative ADMM algorithm [22] to maximize the penalized log-likelihood. Our algorithm is very similar to that used in FGL [40] and CFGL [44] with a few key modifications. The algorithm requires several intermediate variables such as $\mathbf{Z}$, $\mathbf{U}$ that need not have any physical interpretation. We rewrite the problem as,

$$\underset{\Theta, \mathbf{Z}}{\text{minimize}} - \sum_{k=1}^K n_k [\log(\det(\Theta^{(k)}) - \text{tr}(\mathbf{S}^{(k)} \Theta^{(k)}))] + P^{\text{RCFGL}}(\mathbf{Z});$$

incorporating the constraint of positive-definiteness: $\Theta^{(k)} \succ 0$ for $k = 1, \dots, K$ and the constraint that $\mathbf{Z}^{(k)} = \Theta^{(k)}$ for $k = 1, \dots, K$, where $\mathbf{Z} = \{\mathbf{Z}^{(1)}, \dots, \mathbf{Z}^{(K)}\}$. The

Table 1. Penalty functions used in different methods. The different methods consider penalty functions which are sum of two individual penalties and they differ only in the second term. The table categories the second penalty term for each of the methods by whether the method is condition adaptive and whether it uses the sequential difference.

Condition Adaptive \ Sequential Difference	No	Yes
No	FGL: $\sum_{i \neq j} \sum_{k < k'} \Theta_{ij}^{(k)} - \Theta_{ij}^{(k')} $	FMGL (RFGL*): $\sum_{i \neq j} \sum_{k=1}^{K-1} \Theta_{ij}^{(k)} - \Theta_{ij}^{(k+1)} $
Yes	CFGL: $\sum_{i \neq j} \sum_{k < k'} \mathbf{w}_{ij}^{kk'} \Theta_{ij}^{(k)} - \Theta_{ij}^{(k')} $	RCFGL: $\sum_{i \neq j} \sum_{k=1}^{K-1} \mathbf{w}_{ij}^{(k,k+1)} \Theta_{ij}^{(k)} - \Theta_{ij}^{(k+1)} $

*In the developed *Python* package, we provide an implementation of FMGL that we refer to as RFGL.

corresponding scaled augmented Lagrangian [22] can be written as,

$$L_{\rho}(\Theta, \mathbf{Z}, \mathbf{U}) = - \sum_{k=1}^K n_k [\log(\det(\Theta^{(k)}) - \text{tr}(\mathbf{S}^{(k)} \Theta^{(k)}))] + \mathbf{P}^{\text{RCFGL}}(\mathbf{Z}) + \frac{\rho}{2} \sum_{k=1}^K \|\Theta^{(k)} - \mathbf{Z}^{(k)} + \mathbf{U}^{(k)}\|_F^2 - \frac{\rho}{2} \sum_{k=1}^K \|\mathbf{U}^{(k)}\|_F^2 \quad (2)$$

where $\mathbf{U} = \{\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(K)}\}$ are dual variables, ρ is a penalty parameter and $\|\cdot\|_F$ denotes the Frobenius norm.

The algorithm is as follows,

1. Initialize the variables: $\Theta^{(k)} = \mathbf{I}, \mathbf{Z}^{(k)} = \mathbf{0}, \mathbf{U}^{(k)} = \mathbf{0}$ for $k = 1, \dots, K$.
2. Select a constant $\rho > 0$.
3. For $i = 1, 2, 3, \dots$ until convergence:
 - i. For $k = 1, \dots, K$, update $\Theta_{(i)}^{(k)}$ as the minimizer (w.r.t $\Theta^{(k)}$) of $-n_k [\log(\det(\Theta^{(k)}) - \text{tr}(\mathbf{S}^{(k)} \Theta^{(k)}))] + \frac{\rho}{2} \|\Theta^{(k)} - \mathbf{Z}_{(i-1)}^{(k)} + \mathbf{U}_{(i-1)}^{(k)}\|_F^2$.
Let $\mathbf{VDV}^T$ denote the eigen-decomposition of $\mathbf{S}^{(k)} - \rho/n_k (\mathbf{Z}_{(i-1)}^{(k)} + \mathbf{U}_{(i-1)}^{(k)})$.
The solution of the above minimization [48] is given by $\mathbf{V}\tilde{\mathbf{D}}\mathbf{V}^T$, where $\tilde{\mathbf{D}}_{jj}$ is the diagonal matrix with j -th diagonal element being

$$\rho/n_k (-\mathbf{D}_{jj} + \sqrt{\mathbf{D}_{jj}^2 + 4\rho/n_k}).$$

- ii. Update $\mathbf{Z}_{(i)}$ as the minimizer (w.r.t $\mathbf{Z}$) of

$$\mathbf{P}^{\text{RCFGL}}(\mathbf{Z}) + \frac{\rho}{2} \sum_{k=1}^K \|\Theta_{(i)}^{(k)} - \mathbf{Z}^{(k)} + \mathbf{U}_{(i-1)}^{(k)}\|_F^2$$

The problem can be rewritten as,

$$\underset{\mathbf{Z}}{\text{minimize}} \left\{ \mathbf{P}^{\text{RCFGL}}(\mathbf{Z}) + \frac{\rho}{2} \sum_{k=1}^K \|\mathbf{Z}^{(k)} - \mathbf{A}^{(k)}\|_F^2 \right\}; \quad \mathbf{A}^{(k)} = \Theta_{(i)}^{(k)} + \mathbf{U}_{(i-1)}^{(k)}$$

With the actual expression of $P^{RCFGL}(\mathbf{Z})$ the above problem takes the form,

$$\underset{\mathbf{Z}}{\text{minimize}} \left\{ \lambda_1 \sum_{i \neq j} \sum_{k=1}^K |\mathbf{Z}_{ij}^{(k)}| + \lambda_2 \sum_{i \neq j} \sum_{k=1}^{K-1} \mathbf{w}_{ij}^{(kk+1)} |\mathbf{Z}_{ij}^{(k)} - \mathbf{Z}_{ij}^{(k+1)}| + \frac{\rho}{2} \sum_{k=1}^K \|\mathbf{Z}^{(k)} - \mathbf{A}^{(k)}\|_F^2 \right\}$$

The above problem is completely separable w.r.t each pair of matrix elements (i, j) , where $i \neq j$. It means that one can independently solve, for each pair (i, j) , the following minimization problem:

$$\underset{\mathbf{Z}_{ij}^{(1)}, \dots, \mathbf{Z}_{ij}^{(K)}}{\text{minimize}} \left\{ \lambda_1 \sum_{k=1}^K |\mathbf{Z}_{ij}^{(k)}| + \lambda_2 \sum_{k=1}^{K-1} \mathbf{w}_{ij}^{(kk+1)} |\mathbf{Z}_{ij}^{(k)} - \mathbf{Z}_{ij}^{(k+1)}| + \frac{\rho}{2} \sum_{k=1}^K |\mathbf{Z}_{ij}^{(k)} - \mathbf{A}_{ij}^{(k)}|^2 \right\} \quad (3)$$

The problem is known as the weighted 1-D fused lasso signal approximator, which can be solved very efficiently.

iii. For $k = 1, \dots, K$, update $\mathbf{U}_{(i)}^k$ as $\mathbf{U}_{(i-1)}^k + (\Theta_{(i)}^k - \mathbf{Z}_{(i)}^k)$.

The step where using a sequential fused lasso penalty instead of a pair-wise fused lasso penalty is beneficial is in Equation (3). When $\mathbf{w}_{ij}^{(kk+1)} = 1$ for all $k = 1, \dots, K-1$, the problem of Equation 3 becomes the 1-D fused lasso signal approximator [41, 49] for which an efficient and exact solution is available by the algorithm of Condat et al. [43]. The *MATLAB* package of FMGL [42] also uses this particular algorithm. The algorithm of Condat et al. [43] treats the fused lasso signal approximator as a 1-D total variation denoising problem [50]. When $\mathbf{w}_{ij}^{(kk+1)} = 0$ for at least one k , the problem of Equation 3 can be thought of as a special case of a weighted 1-D total variation problem (where weights are 1 or 0). There is an efficient ‘Taut-String’ algorithm [45, 46] for solving weighted 1-D total variation denoising problems.

The algorithm of Condat et al. [43] and the ‘Taut-String’ algorithm [45, 46] both have computational complexity of $O(K)$ in most practical scenarios. Recall that FGL [40] uses a pair-wise fused lasso penalty which results in the general fused lasso approximator [41, 49] in the $\mathbf{Z}$ updating step. FGL follows a path algorithm [49] for solving the above step which has computational complexity of $O(K \log K)$. In CFGL [44], the authors solve the $\mathbf{Z}$ updating step exactly for $K = 2$ and 3, but, do not provide any solution for $K > 3$. For details about the computation of the weight matrices: $\mathbf{W}^{kk+1}$ for $k = 1, \dots, K-1$, we refer to the CFGL paper [44].

Detecting block diagonal structure in the precision matrices

Here, we present a theorem involving a set of sufficient conditions that can be checked prior to fitting the ADMM algorithm and can potentially result in substantial computational benefit. A similar theorem has been used in the context of FGL [40] and FMGL [42]. Using the theorem, one would inspect the sample covariance matrices $\mathbf{S}^{(1)}, \dots, \mathbf{S}^{(K)}$ to determine if the solution to the RCFGL problem i.e., the estimates of the precision matrices: $\hat{\Theta}^{(k)}$ for $k = 1, \dots, K$, are block-diagonal after some permutation of the genes. The inspection is based on comparing the absolute values of $\mathbf{S}_{ij}^{(k)}$ ’s with the tuning parameter λ_1 .

Theorem 1 Denote the set of p genes by C . Suppose, there are M many disjoint subsets of C s.t. $C_1 \sqcup C_2 \sqcup \dots \sqcup C_M = C$. For the genes in C_l to be completely disconnected from those in $C_{l'}$ in each of the resulting estimates, it will be sufficient to have $|n_k \mathbf{S}_{ij}^{(k)}| < \lambda_1$ for $k = 1, 2, \dots, K, \forall i \in C_l, j \in C_{l'}$.

Using Theorem 1, for a given value of λ_1 , suppose we find out that the estimated precision matrices: $\hat{\Theta}^{(k)}$ for $k = 1, \dots, K$, will be block-diagonal with M blocks i.e., they will have the following form,

$$\hat{\Theta}^{(k)} = \begin{pmatrix} \hat{\Theta}_1^{(k)} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \hat{\Theta}_2^{(k)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \hat{\Theta}_M^{(k)} \end{pmatrix} \quad (4)$$

where $\hat{\Theta}_l^{(k)}$ for $k = 1, \dots, K$ have the same dimensions and correspond to the same subset of genes: C_l . It would imply that instead of solving the RCFGL problem for full $\Theta^{(k)}$, one can solve the RCFGL problems for $\Theta_l^{(k)}$ for $l = 1, \dots, M$ independently. This drastically reduces the computational complexity. Let the dimension of each block $\Theta_l^{(k)}$ be $p_l \times p_l$ (the size of the subset C_l is p_l), where $\sum_{l=1}^M p_l = p$. The ADMM algorithm discussed in the last section, involves eigen-decomposition of K many $p \times p$ matrices which takes up computational complexity of $O(Kp^3)$. Whereas, solving block RCFGL problems will only have the computational complexity of $K \sum_{l=1}^M O(p_l^3)$. The proof of Theorem 1 can be found in S1 Supplementary.

Tuning parameter selection

Following the suggestion of [40, 44] for selecting the tuning parameters λ_1, λ_2 , we use an approximation of the Akaike information criterion (AIC),

$$AIC(\lambda_1, \lambda_2) = \sum_{k=1}^K \left[n_k \text{tr}(\mathbf{S}^{(k)} \hat{\Theta}_{\lambda_1, \lambda_2}^{(k)}) - n_k \log(\det(\hat{\Theta}_{\lambda_1, \lambda_2}^{(k)}) + 2E_k \right]$$

$\hat{\Theta}_{\lambda_1, \lambda_2}^{(k)}$ is the precision matrix estimated for the k -th condition using the tuning parameters λ_1 and λ_2 , and E_k is the number of unique non-zero elements in $\hat{\Theta}_{\lambda_1, \lambda_2}^{(k)}$. A grid search can then be performed to select λ_1 and λ_2 that minimize the $AIC(\lambda_1, \lambda_2)$ score. However, as pointed out by [40], such an approach may tend to choose models that are too large to be useful. Thus, in many cases, model selection is better guided by practical considerations, such as network interpretability and stability.

Effect of ordering of the conditions

Our penalty term, $P^{\text{RCFGL}}(\Theta)$ only considers sequential differences between the conditions. It implies that different ordering of the conditions would yield different levels of penalty. For example, suppose there three conditions: 1, 2 and 3, where the network of 1 is same as that of 3 but the network of 2 is totally different ($\Theta^{(1)} = \Theta^{(3)} \neq \Theta^{(2)}$). If we consider the sequence (1, 3, 2), $P^{\text{RCFGL}}(\Theta)$ will include the terms: $w_{ij}^{13} |\Theta_{ij}^{(1)} - \Theta_{ij}^{(3)}|$ that encourage similarity in the estimated networks of 1 and 3. But, if we consider the sequence (1, 2, 3), $P^{\text{RCFGL}}(\Theta)$ will not include those terms, thereby not encouraging similarity between $\Theta^{(1)}, \Theta^{(3)}$. Thus, it can be potentially more

powerful to use a particular ordering of conditions that places more similar conditions closer. We study the effect of misspecified ordering in our simulation studies.

The ordering can be based on biological information available about the degree of similarity across the conditions, for instance, relationship in cell lineages. Alternatively, we can use hierarchical clustering based on the gene expression data, $\mathbf{Y}^{(k)}$, $k = 1, \dots, K$. We recommend comparing the sample covariance matrices across the conditions. More specifically, we compute the sample covariance matrix for every condition k , $\mathbf{S}^{(k)} = (\mathbf{Y}^{(k)} - \mathbf{1}_{n_k} \bar{\mathbf{Y}}^{(k)})^T (\mathbf{Y}^{(k)} - \mathbf{1}_{n_k} \bar{\mathbf{Y}}^{(k)}) / n_k$. Then, we consider the Euclidean distance between a pair of conditions (k, k') as,

$$d(k, k') = \sqrt{\sum_{i=1}^{n_k} \sum_{j=1}^{n_{k'}} (\mathbf{s}_{i,j}^{(k)} - \mathbf{s}_{i,j}^{(k')})^2}.$$

We subject the generated distance matrix (between all the conditions) to hierarchical clustering to identify conditions that are closer or farther from each other and use these relationships to order the conditions in RCFGL. In our simulation studies, this procedure was able to detect the right ordering every time.

Software implementation

We mainly make use of Condat et. al.'s algorithm [43] available as a *C* code and a *Python* module named proxTV to build our package named RCFGL available here. We provide a *Jupyter* notebook [51] with detailed guidance for fitting the RCFGL model. Additionally, we provide an implementation of the FMGL model that we refer to as RFGL, an acronym for rapid fused graphical lasso. We also provide functions for visualizing the estimated networks and compare them across conditions.

Simulation setup

We considered seven different simulation scenarios, (S1), (S2), ..., and (S7), with varying levels of differentiation across conditions. In each of the scenarios, we considered 500 genes and 100 subjects. Each gene co-expression network consisted of 5 equally sized sub-networks, made of 100 genes. The aim and details of each of the scenarios are described next.

1. In the first four simulation scenarios, our goal was to compare RCFGL with other three methods i.e., FGL, FMGL (referred to as, RFGL), and CFGL in terms of both estimation accuracy and computational time.
 - (a) In both (S1) and (S2), three conditions were considered i.e., $K = 3$. In (S1), the first two networks were exactly the same, whereas the third network shared only three sub-networks common with the first two and the other two sub-networks were generated independently. In (S2), the first two networks were again exactly the same but the third one did not share any sub-network common with the first two i.e., all 5 of its sub-networks were generated independently.
 - (b) In both (S3) and (S4), four conditions were considered i.e., $K = 4$. In (S3), the first two and the last two networks were the same as each other. In (S4), only the first two networks were the same and the other two were different.
2. In the last three simulation scenarios, we studied the effect of the ordering of the conditions on RCFGL's performance for three and four conditions. In (S5), three conditions were considered. The first and third conditions had the same networks,

whereas the second network was entirely different. In (S6) and (S7), four conditions were considered. In (S6), the first and third networks were the same and the other two were different. In (S7), the first and fourth networks were the same and the other two were different.

All of the scenarios are summarized in terms of the precision matrices of different conditions in Figure 1. Next, we describe how exactly the above networks and corresponding edge weights were simulated. To mimic real-world biological network

Fig 1. The relationship between the precision matrices across conditions in different simulation scenarios. Notice that in (S1), only the first three blocks of the precision matrices were common across all the three conditions.

$$\begin{aligned} \text{(S1)} : K = 3, \Theta^{(1)} = \Theta^{(2)} = \Theta^{(3)} &= \begin{pmatrix} \Theta_1 & 0 & 0 & 0 & 0 \\ 0 & \Theta_2 & 0 & 0 & 0 \\ 0 & 0 & \Theta_3 & 0 & 0 \\ 0 & 0 & 0 & \Theta_4 & 0 \\ 0 & 0 & 0 & 0 & \Theta_5 \end{pmatrix}; \Theta^{(3)} = \begin{pmatrix} \Theta_1 & 0 & 0 & 0 & 0 \\ 0 & \Theta_2 & 0 & 0 & 0 \\ 0 & 0 & \Theta_3 & 0 & 0 \\ 0 & 0 & 0 & \Theta_4^{(3)} & 0 \\ 0 & 0 & 0 & 0 & \Theta_5^{(3)} \end{pmatrix}. \\ \text{(S2)} : K = 3, \Theta^{(1)} = \Theta^{(2)} \neq \Theta^{(3)}. \\ \text{(S3)} : K = 4, \Theta^{(1)} = \Theta^{(2)} \neq \Theta^{(3)} = \Theta^{(4)}. \\ \text{(S4)} : K = 4, \Theta^{(1)} = \Theta^{(2)} \neq \Theta^{(3)} \neq \Theta^{(4)}. \\ \text{(S5)} : K = 3, \Theta^{(1)} = \Theta^{(3)} \neq \Theta^{(2)}. \\ \text{(S6)} : K = 4, \Theta^{(1)} = \Theta^{(3)} \neq \Theta^{(2)} \neq \Theta^{(4)}. \\ \text{(S7)} : K = 4, \Theta^{(1)} = \Theta^{(4)} \neq \Theta^{(2)} \neq \Theta^{(3)}. \end{aligned}$$

structures [44,52], we used the Barabasi-Albert model [53] to simulate the unweighted network topology, i.e., the adjacency matrix with indicator elements, 1 if an edge was present between a pair of genes and 0 otherwise. Next, the k -th weighted network, $A^{(k)}$ was generated as,

$$A_{ij}^{(k)} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } (i, j)\text{-th element of the } k\text{-th adjacency matrix was 0} \\ \sim \text{Unif}(D) & \text{if } (i, j)\text{-th element of the } k\text{-th adjacency matrix was 1} \end{cases}$$

where $\text{Unif}(D)$ refers to a uniform distribution with $D = [-0.9, -0.6] \cup [0.6, 0.9]$. To ensure that the weighted network was positive definite, an eigen-value adjustment was performed as, $A^{*(k)} = A^{(k)} + |\delta^{(k)}| \mathbf{I}$, where $\delta^{(k)}$ was the smallest eigen-value of $A^{(k)}$. Based on $A^{*(k)}$, covariance matrix $\Sigma_k = [\Sigma_{k(i,j)}]$ was constructed as,

$$\Sigma_{k(i,j)} = \frac{[A^{*(k)}]_{(i,j)}^{-1}}{\sqrt{[A^{*(k)}]_{(i,i)}^{-1} [A^{*(k)}]_{(j,j)}^{-1}}}.$$

With the covariance matrix Σ_k (and, consequently $\Theta^{(k)}$), the gene expression vector of the i -th subject under condition k was simulated as $y_i^{(k)} \sim N_p(0, \Sigma_k)$. Ten replications were considered in every simulation scenario and average findings were reported.

Real data

RNA expression from three brain regions was measured using high-throughput RNA sequencing on the Illumina HiSeq 4000 platform and a poly-A selection protocol (GSE173141). The original data set included tissue from 88 alcohol and drug naive heterogeneous stock rats and most rats had RNA-Seq libraries from all the regions. The

average number of raw reads per rat and brain region was 26.7 million. After extensive quality control, 83 rats remained with RNA-Seq from the lateral habenula (LHB) core, 84 rats with data from the infralimbic (IL) cortex, and 82 rats with data from the prelimbic (PL) cortex [54, 55]. Reads were trimmed to remove adaptors and low-quality base calls using cutAdapt [56]. They were then aligned to the Ensembl Rat transcriptome using RSEM (RNA-Seq Expectation-Maximization; [57]). An upper quantile scaling was initially applied to the estimated read counts for individual genes using the betweenLaneNormalization function from the EDASeq package in R [58]. A regularized (r)log was then used to transform the read counts using the DESeq2 package in R [59]. Finally, a batch effects adjustment was made using the ComBat function in the sva R package [60]. For this manuscript, we focused on 15,421 protein-coding genes common across all three brain regions and 64 rats having data available for all of those genes in all three brain regions.

Measures for evaluating performance

In the simulation studies, the estimation performance of the methods were assessed based on both network topology and edge weights. Denote the true precision matrices as $\Theta^{(k)}$ and the estimated precision matrices by $\hat{\Theta}^{(k)}$ for $k = 1, \dots, K$. We computed the true positive rate (TPR) and false positive rate (FPR) of edge detection in the following way. If the ij -th edge was present in the true network of k -th condition (i.e., $\Theta_{ij}^{(k)} \neq 0$), and was also identified in the estimated precision matrix (i.e., $\hat{\Theta}_{i,j}^{(k)} \neq 0$), it was counted as a true positive (TP). If the edge was absent in the true network (i.e., $\Theta_{ij}^{(k)} = 0$) but was identified in the estimated one (i.e., $\hat{\Theta}_{i,j}^{(k)} \neq 0$), it was counted as a false positive (FP). TPR and FPR were then jointly displayed in a ROC curve. Note that if $\Theta_{i,j}^{(k)}$ and $\hat{\Theta}_{i,j}^{(k)}$ were both non-zero, but the sign of $\hat{\Theta}_{i,j}^{(k)}$ did not match the sign of $\Theta_{i,j}^{(k)}$, the ij -th edge would still be counted as a TP. Hence, to judge the accuracy of edge weight estimation, we computed the sum of squared error (SSE) between the estimated and the true precision matrices: $\sum_{k=1}^K \sum_{i=1}^{n_k} \sum_{j=1}^{n_k} (\Theta_{i,j}^{(k)} - \hat{\Theta}_{i,j}^{(k)})^2$. We compared the run-times of the methods based on a *MacOS* system with 32 GB RAM and *intel i9* CPU with 8 cores.

In the real data analysis, first we compared the run-times of the different methods. Next, we compared the estimation performance of RFGL and RCFGL to assess the advantages of condition adaptive estimation in this context. To demonstrate how similar the results from RCFGL and CFGL were, we inspected the top Z edges of every brain region based on the absolute value of the estimated precision matrices. Let the sets of the top Z edges detected by CFGL for regions LHB, IL and PL be respectively denoted by M_1, M_2 and M_3 and those by RCFGL be denoted as N_1, N_2 and N_3 . In mathematical terms, we looked at the following proportion,

$$\text{prop}(Z) = \frac{|\bigcup_{i=1}^3 M_i \cap \bigcup_{i=1}^3 N_i|}{|\bigcup_{i=1}^3 M_i \cup \bigcup_{i=1}^3 N_i|} \quad (5)$$

for different values of Z . A value close to 1 for $\text{prop}(Z)$ would imply both the methods produced the same top Z edges. As discussed earlier, the penalty terms of RCFGL and CFGL share a special inequality, $P^{\text{CFGL}}(\Theta, \lambda_1, \lambda_2, \mathbf{W}) \leq P^{\text{RCFGL}}(\Theta, \lambda_1, \lambda_2, \mathbf{W}^*)$, where $\mathbf{W}^*$ is a set of modified weight matrices defined as, $\mathbf{W}^* = (\mathbf{W}^{*12}, \mathbf{W}^{*23})$, where $\mathbf{W}^{*12} = \mathbf{W}^{12} + \mathbf{W}^{13}$ and $\mathbf{W}^{*23} = \mathbf{W}^{23} + \mathbf{W}^{13}$ (see Proposed method section and S1 Supplementary). Thus, to achieve better agreement with CFGL, we fitted RCFGL with the modified set of weight matrices $\mathbf{W}^*$. We also performed enrichment analysis on the hub-genes detected by RCFGL using the application ShinyGO [61] and summarised the results in a table listing enrichment false discovery rate (FDR), number of hub-genes

found in the pathway, number of background genes found in the pathway and fold
enrichment scores. Fold enrichment is defined as the percentage of detected genes
belonging to a pathway, divided by the corresponding percentage of the background
genes. FDR tells us the likelihood of the enrichment being by chance and fold
enrichment tells us how many genes of a certain pathway are over-represented.

Results

Simulation study

We evaluated the performance of RCFGL in seven simulation scenarios described earlier.
For fitting FGL and CFGL, we used the corresponding *R* packages and for fitting
FMGL (referred to as, RFGL) and RCFGL, we used our package. The methods RFGL,
FGL, RCFGL, CFGL were denoted by different colored lines in all the figures. They
were run with the same sets of hyperparameters, (λ_1, λ_2) . Different λ_1 's resulted in
different numbers of edges detected, and different values of λ_2 modulated the similarity
penalty from low to high.

Simulation with three conditions

The difference between (S1) and (S2) lied in the level of similarity across the networks.
As discussed in Review of methods, FGL and RFGL both assume that networks of all
the conditions share same level of similarity. Scenario (S1) was close to that assumption,
whereas (S2) violated it since the third network did not share any similarity with the
first two. Figure 2 and 3 respectively show the ROC curves for edge detection and the
SSE of the methods. Figure 4 shows the comparison of the run-times of different
methods. The time-comparison is demonstrated across low to high values of λ_1 since it
controls how dense the networks will be and a denser network may take more time to
estimate.

Since CFGL is a condition adaptive extension of FGL and RCFGL is a condition
adaptive extension of RFGL, it would be sensible to compare the methods pair-wise. In
scenario (S1), all the methods showed nearly identical ROC curves for edge detection
(Figure 2). However, in scenario (S2) where the assumption of same level of similarity
across all the pairs of conditions was violated, CFGL and RCFGL respectively
performed better than their non-condition adaptive counterparts, FGL and RFGL. In
addition CFGL and RCFGL showed significantly lower SSE compared to FGL and
RFGL in both the scenarios for all three values of λ_2 (Figure 3). This illustrates the
advantage of the condition adaptive methods over the simpler ones especially when
some pairs of conditions share different levels of similarity. CFGL had significantly
higher run-time compared to all the other methods, whereas RFGL and RCFGL took
just fractions of that time (Figure 4). RFGL was notably faster than FGL. So, when
there are a lot of genes and a large number of conditions, RFGL can be used over FGL
for a much faster network exploration. RCFGL was also faster than FGL, such that one
can perform a condition adaptive network estimation in a similar amount of time taken
by a non-condition adaptive network estimation model such as FGL.

Simulation with four conditions

Next, we evaluated the performance of RCFGL in scenarios with four conditions. In this
case the CFGL *R* package was not usable and was omitted from comparison. Scenario
(S3) is close to the assumption of FGL and RFGL that all the networks share same level
of similarity, whereas (S4) violates that assumption. RCFGL showed consistently better
TPR compared to the other methods in all the scenarios (Figure 5). When λ_2 was low,

Fig 2. Comparison of edge detection performance for simulations with three conditions. Top and bottom rows respectively correspond to the ROC curves in scenario (S1) and scenario (S2). The x and y axes respectively correspond to false positive rate (FPR) and true positive rate (TPR) for different values of λ_1 . Three different values of λ_2 are considered.

RFGL had a ROC curve close to RCFGL, and when λ_2 was high, RFGL had a ROC curve close to FGL. RCFGL also had significantly lower SSE compared to RFGL and FGL in both the scenarios for all the values of λ_2 (Figure 6). RCFGL took significantly lower run-time compared to FGL, especially for smaller λ_1 's (Figure 4). It reaffirmed our earlier point that using RCFGL one can perform a condition adaptive network estimation even faster than a non-condition adaptive network estimation model such as FGL.

Sensitivity with respect to ordering of the conditions

Next, we studied the impact of different orderings of the conditions on RCFGL. Note that in (S7), two conditions 'far' from each other (conditions 1 and 4) had the same networks, whereas in (S6), two conditions relatively closer (conditions 1 and 3) had the same networks. We compared the performance of RCFGL with 'incorrect' ordering of conditions ((1, 2, 3) for (S5) and (1, 2, 3, 4) for (S6) and (S7)) with RCFGL with 'correct' ordering of conditions ((1, 3, 2) for (S5), (1, 3, 2, 4) for (S6), and (1, 4, 2, 3) for (S7)). In the plots, we referred to the latter as RCFGL-C and it was expected to

Fig 3. Comparison of edge weight estimation performance for simulations with three conditions. Top and bottom rows respectively correspond to the SSE of methods in scenario (S1) and scenario (S2). The x and y axes respectively correspond to the total number of edges detected and SSE for different values of λ_1 . Three different values of λ_2 are considered.

perform the best. We compared regular RCFGL and RCFGL-C with FGL because it was the only method that would not be affected by the ordering (CFGL would also not be affected but could not be used with 4 conditions). Figure 7 displays the SSE for edge weight estimation. Refer to S1 Fig for the ROC curves for edge detection. In (S5), RCFGL and RCFGL-C had similar SSE values except for the smallest λ_2 . In both (S6) and (S7), for smaller values of λ_2 , RCFGL-C had noticeably better SSE compared to RCFGL. However, RCFGL had better SSE compared to FGL for all the values of λ_2 . Therefore, it could be concluded that the effect of ‘incorrect’ ordering will have more of an impact with more than three conditions, particularly when two conditions ‘far’ from each other are similar.

The procedure we discussed in Effect of ordering of the conditions section to identify the ordering was able to detect the correct order in every scenario i.e., it placed the similar conditions side by side, the conditions (1, 3) in (S5), (S6) and (1, 4) in (S7). Therefore, using the proposed order-detection procedure, we achieved the best possible performance of RCFGL-C.

Fig 4. Comparison of run-time for simulations with three and four conditions. Top row corresponds to the run-times in seconds of different methods in scenario (S1) and scenario (S2). Bottom row corresponds to the run-times in scenario (S3) and scenario (S4). For the x -axis λ_1 is varied from low to high generating increasingly sparser networks. For each value of λ_1 , the average run-time over three values of λ_2 is reported.

Real data analysis

Because in the real data the true networks are unknown, we focused on checking the consistency of the estimated networks by different methods and their run-times. Regions IL and PL are anatomically closer and are similar in terms of overall structure and regulatory function in many studies [62–64]. In our dataset as well, IL and PL were found to be similar in terms of the gene-expression based on hierarchical clustering (S2 Fig) compared to LHB. The condition adaptive methods, such as CFGL and RCFGL, were expected to detect more edges common between IL and PL compared to FGL and RFGL.

Time comparison

FGL and CFGL both would be computationally infeasible to run on all of 15,421 genes. So, we focused on smaller sets of genes obtained by pruning based on coefficient of

Fig 5. Comparison of edge detection performance for simulations with four conditions. Top and bottom rows respectively correspond to the ROC curves in scenario (S3) and scenario (S4). The x and y axes respectively correspond to false positive rate (FPR) and true positive rate (TPR) for different values of λ_1 . Three different values of λ_2 are considered.

variation (CV) [65]. To prune, we concatenated the gene expression data from all the regions and computed the CV (ratio of mean to SD) of every gene. Next, we removed the genes which had CV less than a certain cut-off from the analysis. For example, removing the genes with $CV < 0.02$ left us with 1,106 genes, whereas removing the genes with $CV < 0.04$ left us with only 201 genes. We considered five such CV cut-offs, 0.015, 0.02, 0.025, 0.03 and 0.04. RFGL and RCFGL consistently took just fractions of the time taken by FGL and CFGL (Table 2). For the CV cut-off of 0.015, there were 4706 genes in the sample. In that case, we only reported the time taken by RFGL and RCFGL since both FGL and CFGL would be taking an exorbitant amount of time (more than 10 hours) to converge. It should also be mentioned that we had applied RFGL and RCFGL on the full dataset with 15,421 genes using a much more powerful Dell PowerEdge R740XD server with Intel Xeon Gold 6152 2.1G X (2) CPU having 44 cores, and they respectively took around 4 and 7 hours.

Fig 6. Comparison of edge weight estimation performance for simulations with three conditions. Top and bottom rows respectively correspond to the SSE of methods in scenario (S3) and scenario (S4). The x and y axes respectively correspond to the total number of edges detected and SSE for different values of λ_1 . Three different values of λ_2 are considered.

Table 2. The run-times of different methods (in seconds) with the genes left after pruning based on different CV cut-offs. λ_1 and λ_2 were respectively kept at 0.01 and 0.02. The mark “X” means that we could not run those methods due to inordinate amount of time required.

CV cut-off (# genes left)	RFGL	FGL	RCFGL	CFGL
0.040 (201)	2	20	14	36
0.030 (376)	7	102	27	145
0.025 (557)	15	335	64	470
0.020 (1106)	62	1468	172	2548
0.015 (4674)	1511	X	3237	X

Comparison of RFGL and RCFGL

Next, we compared the networks estimated by RFGL and RCFGL. We considered the set of 557 genes, obtained by pruning the full set of genes based on the CV cut-off of 0.025. To address the variability in the estimated networks, we repeated the following procedure 50 times. Each time, we randomly selected 500 genes from the set of 557

Fig 7. Comparison of edge weight estimation performance for simulations studying effect of ordering. Top row corresponds to scenario (S5) which has three conditions, and the next two rows correspond to scenario (S6) and (S7) each of which has four conditions. The x and y axes respectively correspond to the total number of edges detected and SSE for different values of λ_1 .

genes and estimated the networks using RFGL and RCFGL for $\lambda_1 = 0.01$ and three values of λ_2 . To investigate brain-region specificity of the edges detected by the two methods, we partitioned the identified edges into seven mutually exclusive categories: LHB region only, IL region only, PL region only, LHB-IL shared, IL-PL shared, PL-LHB shared, and common between all regions. Figure 8 displays the box-plot of the proportion of edges detected by the two methods. As mentioned earlier, we expected the regions IL and PL to share more edges compared to region LHB and condition adaptive methods should be better at capturing that. Consistent with the expectation, we noticed that RCFGL detected more IL-PL specific edges compared to RFGL with the difference becoming increasingly apparent as λ_2 increased. RCFGL also detected more LHB specific edges. RFGL produced more edges common between all regions. A large value of λ_2 implies imposing a very high similarity penalty that would force the estimated networks of the three regions to be very close to each other. Thus, as we increased λ_2 , both the methods produced more edges common between all three regions and fewer edges specific to a single region or a pair of regions.

Fig 8. Comparison of edge detection by RFGL and RCFGL in real data.
The y axis corresponds to the proportion of edges in seven mutually exclusive categories out of all the edges. The categories are coupled and displayed in three columns. The first column has edges specific to different regions. The second column has edges specific to different pairs of regions, and the third has edges common to all the regions. The rows from top to bottom respectively correspond to three different values of λ_2 , 0.02, 0.03 and 0.04.

Comparison of RCFGL and CFGL

Next, we compared the performance of RCFGL to CFGL on the set of 557 genes (CV cutoff < 0.025). We kept λ_1 at 0.01 and varied λ_2 from low to high. We compared the top Z edges detected by the two methods to investigate the degree of agreement. We used the measure $\text{prop}(Z)$ from Equation 5 for several values of Z . The top edges detected by RCFGL and CFGL matched by a great degree ($\text{prop}(Z) > 0.85$) in all the cases (Table 3). The agreement expectedly increased as λ_2 increased because for large values of λ_2 , the difference between the penalty terms of RCFGL and CFGL becomes minimal, making them theoretically very close.

Gene set enrichment analysis

Our next goal was to identify biological functions of the hub-genes of the estimated networks by RCFGL. We applied RCFGL on the set of 557 genes (CV cutoff < 0.025)

Table 3. Proportion of overlap of the top edges detected by RCFGL and CFGL. The cells correspond to $\text{prop}(Z)$ for varying values of Z and λ_2 .

Z	$\lambda_2 = 0.01$	$\lambda_2 = 0.015$	$\lambda_2 = 0.02$
100	0.86	0.94	0.95
300	0.89	0.93	0.96
500	0.92	0.95	0.97
700	0.94	0.94	0.97

with $\lambda_1 = 0.01$ and $\lambda_2 = 0.01, 0.02, 0.03$ and 0.04 . The lowest value of AIC was observed for $\lambda_2 = 0.03$ and we interpreted the corresponding estimates. First, we investigated the network structure common across all the three regions i.e., the edges which appeared in all the regions and identified the hub-genes with more than five edges. We annotated the biological functions of these genes using the application ShinyGO [61]. Table 3 displays the top pathways detected and lists corresponding enrichment FDR and fold enrichment scores. The pathways were primarily associated with gene expression regulation and transcription factors. There also were pathways influencing tissue, organic development and different metabolic processes (see S1 Supplementary for the full table). There were twenty six genes in total which appeared in these pathways

Table 4. Top pathways detected by the enrichment analysis of the hub-genes found in the common network of the three brain regions.

Enrichment FDR	# Hub-genes in Pathway	# Background Genes in Pathway	Fold Enrichment	Pathway
0.00002	18	40	3.69	Positive regulation of RNA biosynthetic process
0.00002	18	40	3.69	Positive regulation of nucleic acid-templated transcription
0.00002	18	40	3.69	Positive regulation of transcription, DNA-templated
0.00002	19	44	3.54	Positive regulation of nucleobase-containing compound metabolic process
0.00002	19	42	3.71	Positive regulation of RNA metabolic process
0.00005	15	30	4.10	Positive regulation of transcription by RNA polymerase II
0.00009	18	44	3.35	Positive regulation of macromolecule biosynthetic process
0.00019	18	46	3.21	Positive regulation of cellular biosynthetic process
0.00024	18	47	3.14	Positive regulation of biosynthetic process
0.00130	22	73	2.47	Cellular response to chemical stimulus
0.00136	7	9	6.37	Skeletal muscle tissue development
0.00136	7	9	6.37	Skeletal muscle organ development
0.00137	11	22	4.10	Response to organonitrogen compound
0.00158	17	49	2.84	Regulation of transcription by RNA polymerase II
0.00158	19	59	2.64	Positive regulation of gene expression
0.00192	11	23	3.92	Response to nitrogen compound
0.00192	17	50	2.79	Transcription by RNA polymerase II

(union of the genes from each pathway). Figure 9 shows the sub-network between these genes in each of the three regions. We noticed that the sub-networks of IL and PL were exactly the same, whereas LHB had three edges missing and two new edges. The finding was quite consistent with our expectation that the regions, IL and PL should be more similar to each other than LHB.

Fig 9. The sub-networks corresponding to the hub-genes found in the reported pathways. The figure displays the edges shared between the genes found in the pathway analyses for each of the three brain regions, LHB, IL and PL (from left to right). The grey edges are the edges present in all the regions, whereas the green edges are the edges missing at least in one of the regions.

Discussion

We present a method, named rapid condition adaptive fused graphical lasso (RCFGL), for estimating gene co-expression networks of multiple conditions jointly. Similar to an existing method named condition adaptive fused graphical lasso (CFGL), we compute data-driven weight terms between every pair of conditions storing information about pair-specific co-expression patterns. We infuse the weight terms to a sequential fused lasso penalty, a penalty earlier considered in a method named fused multiple graphical lasso (FMGL). As CFGL is interpreted as a condition adaptive extension of the method fused graphical lasso (FGL), RCFGL can be interpreted as a condition adaptive extension of FMGL. Unlike CFGL, RCFGL is computationally much faster and can be used to analyze more than three conditions together. As we have seen in the simulation studies and real data analysis, the performance of RCFGL and CFGL are very comparable. Both the methods outperform non-condition adaptive methods FMGL (referred to as RFGL in the figures) and FGL. We have demonstrated how fast RCFGL is compared to CFGL and even FGL in most of the cases.

We considered simulation scenarios with both three and four conditions. With three conditions, RCFGL and CFGL both achieved better true positive rate (TPR) and smaller sum of squared error (SSE) than the non-condition adaptive methods FMGL and FGL, especially when there was a different level of similarity between the conditions. Furthermore, RCFGL took just a fraction of time taken by CFGL. With four conditions as well, RCFGL achieved superior performance than both FMGL and FGL. It was computationally much faster than FGL. For an example real data, we considered gene expression data from three brain regions of a heterogeneous stock panel of rats. We first compared the time taken by different methods to estimate the co-expression networks with varying sets of genes, showing again the computational feasibility of RCFGL. Then, we compared the performance of FMGL (RFGL) and RCFGL. The results demonstrated that network estimation was likely superior in the latter since it could detect more edges shared only between IL and PL, two regions that are expected to be more similar compared to the other region LHB. Finally, we performed enrichment analysis with the hub-genes of the network structure (detected by RCFGL) common across the three brain regions, finding them to be influential in regulating gene expression and transcription factors.

Even though our method is developed for the purpose of estimating gene co-expression networks, it can be applied to any dataset that requires joint estimation of multiple networks and would benefit from taking into account condition specificity. In this paper, we have considered a maximum of four conditions. But, the run-time of RCFGL is approximately linear with respect to the number of conditions which makes it scalable for any number of conditions as long as the results remain interpretable. However, one limitation of both RCFGL and CFGL is that the weight-terms which capture information about pair-specific co-expression patterns, are binary. That is the weight for an edge between a pair of conditions takes value 1 if the edge is expected to be present in both the conditions and 0 otherwise. Future extensions will allow for continuous valued weight terms that will allow for more flexibility and can potentially improve performance.

RCFGL is implemented in the form of an open-source software package based on *C* and *Python*, available with a detailed *Jupyter* notebook at this link, <https://github.com/sealx017/RCFGL>. The package also implements the non-condition adaptive method, FMGL (RFGL). Note that the authors of FMGL provide a package that requires *MATLAB* and thus, it is not entirely open-source. Our package can be used as an open-source alternative of their package. Along with the above functions, the package provides several tools for downstream analyses such as comparing networks across conditions and visualizing common or pair-specific networks as well.

Supporting information

S1 Fig. Comparison of edge detection performance for simulations studying effect of ordering. Top row corresponds to scenario (S5) which has three conditions, and the next two rows correspond to scenario (S6) and (S7) each of which have four conditions. The x and y axes respectively correspond to FPR and TPR for different values of λ_1 .

S2 Fig. Hierarchical clustering based on the gene-expression data of three brain regions. We concatenated the expression data of the 557 genes (left after pruning based coefficient of variation (CV) cut-off of 0.025) and computed the Euclidean distance between each pair of brain regions. Next, hierarchical clustering was performed on the distance matrix revealing the order of similarity.

S1 Supplementary. Proof of the theorem, connection between the penalty terms and the full table with enrichment results. We provide the proof of the theorem for detecting block-diagonal structure in the precision matrices and derive the connections between the penalty terms used in different methods. We also provide the full table of the significant pathways detected by the gene-set enrichment analysis of the real data.

Acknowledgements

Research reported in this publication was supported by National Institute of General Medical Sciences (NIGMS) of the National Institutes of Health (NIH) under award numbers T32GM102057 and R01HL152735, National Institute on Drug Abuse (NIDA) of the NIH under award numbers P30DA044223 and P50DA037844, National Institute on Alcohol Abuse and Alcoholism (NIAAA) of the NIH under award number R24AA013162, and by the National Heart, Lung, and Blood Institute under award number R01HL152735. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Author contributions

- **Conceptualization:** Souvik Seal, Katerina Kechris.
- **Formal analysis:** Souvik Seal.
- **Funding acquisition:** Laura M. Saba, Qunhua Li, Katerina Kechris.
- **Investigation:** Souvik Seal, Katerina Kechris.
- **Methodology:** Souvik Seal, Qunhua Li, Elle Butler Basner, Laura M. Saba, Katerina Kechris.
- **Project administration:** Katerina Kechris.

- **Resources:** Laura M. Saba, Katerina Kechris. 564
- **Software:** Souvik Seal. 565
- **Supervision:** Katerina Kechris. 566
- **Validation:** Souvik Seal, Elle Butler Basner. 567
- **Visualization:** Souvik Seal, Katerina Kechris. 568
- **Writing – original draft:** Souvik Seal, Katerina Kechris. 569
- **Writing – review and editing:** Souvik Seal, Qunhua Li, Elle Butler Basner, 570
Laura M. Saba, Katerina Kechris. 571

References 572

1. Stuart JM, Segal E, Koller D, Kim SK. A gene-coexpression network for global discovery of conserved genetic modules. *science*. 2003;302(5643):249–255. 573
574
2. Yang Y, Han L, Yuan Y, Li J, Hei N, Liang H. Gene co-expression network analysis reveals common system-level properties of prognostic genes across cancer types. *Nature communications*. 2014;5(1):1–9. 575
576
577
3. Van Dam S, Vosa U, van der Graaf A, Franke L, de Magalhaes JP. Gene co-expression analysis for functional classification and gene–disease predictions. *Briefings in bioinformatics*. 2018;19(4):575–592. 578
579
580
4. Vanderlinden LA, Saba LM, Kechris K, Miles MF, Hoffman PL, Tabakoff B. Whole brain and brain regional coexpression network interactions associated with predisposition to alcohol consumption. *PloS one*. 2013;8(7):e68878. 581
582
583
5. Saba LM, Flink SC, Vanderlinden LA, Israel Y, Tampier L, Colombo G, et al. The sequenced rat brain transcriptome–its use in identifying networks predisposing alcohol consumption. *The FEBS journal*. 2015;282(18):3556–3578. 584
585
586
6. Harrall KK, Kechris KJ, Tabakoff B, Hoffman PL, Hines LM, Tsukamoto H, et al. Uncovering the liver’s role in immunity through RNA co-expression networks. *Mammalian Genome*. 2016;27(9):469–484. 587
588
589
7. Saba LM, Hoffman PL, Homanics GE, Mahaffey S, Daulatabad SV, Janga SC, et al. A long non-coding RNA (Lrap) modulates brain gene expression and levels of alcohol consumption in rats. *Genes, Brain and Behavior*. 2021;20(2):e12698. 590
591
592
8. Ma S, Gong Q, Bohnert HJ. An Arabidopsis gene network based on the graphical Gaussian model. *Genome research*. 2007;17(11):1614–1625. 593
594
9. López-Kleine L, Leal L, López C. Biostatistical approaches for the reconstruction of gene co-expression networks based on transcriptomic data. *Briefings in functional genomics*. 2013;12(5):457–467. 595
596
597
10. Wang T, Ren Z, Ding Y, Fang Z, Sun Z, MacDonald ML, et al. FastGGM: an efficient algorithm for the inference of gaussian graphical model in biological networks. *PLoS computational biology*. 2016;12(2):e1004755. 598
599
600
11. Zhao H, Duan ZH. Cancer genetic network inference using gaussian graphical models. *Bioinformatics and biology insights*. 2019;13:1177932219839402. 601
602

12. Yi H, Zhang Q, Lin C, Ma S. Information-incorporated Gaussian graphical model for gene expression data. *Biometrics*. 2021;. 603
604
13. Li T, Qian C, Levina E, Zhu J. High-dimensional Gaussian graphical models on network-linked data. *Journal of Machine Learning Research*. 2020;21(74):1–45. 605
606
14. Pena JM. Learning gaussian graphical models of gene networks with false discovery rate control. In: *European conference on evolutionary computation, machine learning and data mining in bioinformatics*. Springer; 2008. p. 165–176. 607
608
609
15. Meinshausen N, Bühlmann P, et al. High-dimensional graphs and variable selection with the lasso. *Annals of statistics*. 2006;34(3):1436–1462. 610
611
16. Yuan M, Lin Y. Model selection and estimation in the Gaussian graphical model. *Biometrika*. 2007;94(1):19–35. 612
613
17. Banerjee O, El Ghaoui L, d’Aspremont A. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *The Journal of Machine Learning Research*. 2008;9:485–516. 614
615
616
18. Friedman J, Hastie T, Tibshirani R. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*. 2008;9(3):432–441. 617
618
19. Hsieh CJ, Sustik MA, Dhillon IS, Ravikumar P. Sparse inverse covariance matrix estimation using quadratic approximation. *arXiv preprint arXiv:13063212*. 2013;. 619
620
20. Cai TT, Liu W, Zhou HH, et al. Estimating sparse precision matrix: Optimal rates of convergence and adaptive estimation. *Annals of Statistics*. 2016;44(2):455–488. 621
622
623
21. Wang L, Ren X, Gu Q. Precision matrix estimation in high dimensional gaussian graphical models with faster rates. In: *Artificial Intelligence and Statistics*. PMLR; 2016. p. 177–185. 624
625
626
22. Boyd S, Parikh N, Chu E. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc; 2011. 627
628
23. He B, Yuan X. On the $O(1/n)$ convergence rate of the Douglas–Rachford alternating direction method. *SIAM Journal on Numerical Analysis*. 2012;50(2):700–709. 629
630
631
24. Wahlberg B, Boyd S, Annergren M, Wang Y. An ADMM algorithm for a class of total variation regularized estimation problems. *IFAC Proceedings Volumes*. 2012;45(16):83–88. 632
633
634
25. Kadhodaie M, Christakopoulou K, Sanjabi M, Banerjee A. Accelerated alternating direction method of multipliers. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*; 2015. p. 497–506. 635
636
637
638
26. Chen C, He B, Ye Y, Yuan X. The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*. 2016;155(1-2):57–79. 639
640
641
27. Debbabi I, Le Gal B, Khouja N, Tlili F, Jegou C. Fast converging ADMM-penalized algorithm for LDPC decoding. *IEEE Communications Letters*. 2016;20(4):648–651. 642
643
644

28. Huang F, Chen S, Huang H. Faster stochastic alternating direction method of multipliers for nonconvex optimization. In: International Conference on Machine Learning. PMLR; 2019. p. 2839–2848.
29. Scheinberg K, Ma S, Goldfarb D. Sparse inverse covariance selection via alternating linearization methods. arXiv preprint arXiv:10110097. 2010;.
30. Yuan X. Alternating direction method for covariance selection models. Journal of Scientific Computing. 2012;51(2):261–273.
31. Wang H, Banerjee A, Hsieh CJ, Ravikumar P, Dhillon IS. Large Scale Distributed Sparse Precision Estimation. In: NIPS. vol. 13; 2013. p. 584–592.
32. Zare A, Jovanović MR, Georgiou TT. Alternating direction optimization algorithms for covariance completion problems. In: 2015 American Control Conference (ACC). IEEE; 2015. p. 515–520.
33. Li P, Xiao Y. An efficient algorithm for sparse inverse covariance matrix estimation based on dual formulation. Computational Statistics & Data Analysis. 2018;128:292–307.
34. Wang C, Jiang B. An efficient ADMM algorithm for high dimensional precision matrix estimation via penalized quadratic loss. Computational Statistics & Data Analysis. 2020;142:106812.
35. Ficklin SP, Dunwoodie LJ, Poehlman WL, Watson C, Roche KE, Feltus FA. Discovering condition-specific gene co-expression patterns using gaussian mixture models: a cancer case study. Scientific reports. 2017;7(1):1–11.
36. Gov E, Arga KY. Differential co-expression analysis reveals a novel prognostic gene module in ovarian cancer. Scientific reports. 2017;7(1):1–10.
37. Xiang S, Huang Z, Wang T, Han Z, Christina YY, Ni D, et al. Condition-specific gene co-expression network mining identifies key pathways and regulators in the brain tissue of Alzheimer’s disease patients. BMC medical genomics. 2018;11(6):39–51.
38. Guo J, Levina E, Michailidis G, Zhu J. Joint estimation of multiple graphical models. Biometrika. 2011;98(1):1–15.
39. Cai TT, Li H, Liu W, Xie J. Joint estimation of multiple high-dimensional precision matrices. Statistica Sinica. 2016;26(2):445.
40. Danaher P, Wang P, Witten DM. The joint graphical lasso for inverse covariance estimation across multiple classes. Journal of the Royal Statistical Society Series B, Statistical methodology. 2014;76(2):373.
41. Tibshirani R, Saunders M, Rosset S, Zhu J, Knight K. Sparsity and smoothness via the fused lasso. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2005;67(1):91–108.
42. Yang S, Lu Z, Shen X, Wonka P, Ye J. Fused multiple graphical lasso. SIAM Journal on Optimization. 2015;25(2):916–943.
43. Condat L. A direct algorithm for 1-D total variation denoising. IEEE Signal Processing Letters. 2013;20(11):1054–1057.

44. Lyu Y, Xue L, Zhang F, Koch H, Saba L, Kechris K, et al. Condition-adaptive fused graphical lasso (CFGL): An adaptive procedure for inferring condition-specific gene co-expression network. *PLoS computational biology*. 2018;14(9):e1006436. 686 687 688 689
45. Jiménez ÁB, Sra S. Fast Newton-type methods for total variation regularization. In: *ICML*; 2011. 690 691
46. Barbero A, Sra S. Modular Proximal Optimization for Multidimensional Total-Variation Regularization. *Journal of Machine Learning Research*. 2018;19(56):1–82. 692 693 694
47. Xia Y, Cai T, Cai TT. Testing differential networks with applications to the detection of gene-gene interactions. *Biometrika*. 2015;102(2):247–266. 695 696
48. Witten DM, Tibshirani R. Covariance-regularized regression and classification for high dimensional problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*. 2009;71(3):615–636. 697 698 699
49. Hoeffling H. A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*. 2010;19(4):984–1006. 700 701
50. Rudin LI, Osher S, Fatemi E. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*. 1992;60(1-4):259–268. 702 703
51. Perkel JM. Why Jupyter is data scientists’ computational notebook of choice. *Nature*. 2018;563(7732):145–147. 704 705
52. Newman ME. The structure and function of complex networks. *SIAM review*. 2003;45(2):167–256. 706 707
53. Yook SH, Jeong H, Barabási AL. Modeling the Internet’s large-scale topology. *Proceedings of the National Academy of Sciences*. 2002;99(21):13382–13386. 708 709
54. Salgado S, Kaplitt MG. The nucleus accumbens: a comprehensive review. *Stereotactic and functional neurosurgery*. 2015;93(2):75–93. 710 711
55. Baker PM, Jhou T, Li B, Matsumoto M, Mizumori SJ, Stephenson-Jones M, et al. The lateral habenula circuitry: reward processing and cognitive control. *Journal of Neuroscience*. 2016;36(45):11482–11488. 712 713 714
56. Martin M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet journal*. 2011;17(1):10–12. 715 716
57. Li B, Dewey CN. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC bioinformatics*. 2011;12(1):1–16. 717 718
58. Risso D, Schwartz K, Sherlock G, Dudoit S. GC-content normalization for RNA-Seq data. *BMC bioinformatics*. 2011;12(1):1–17. 719 720
59. Love MI, Huber W, Anders S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome biology*. 2014;15(12):1–21. 721 722
60. Leek JT, Johnson WE, Parker HS, Jaffe AE, Storey JD. The sva package for removing batch effects and other unwanted variation in high-throughput experiments. *Bioinformatics*. 2012;28(6):882–883. 723 724 725
61. Ge SX, Jung D, Yao R. ShinyGO: a graphical gene-set enrichment tool for animals and plants. *Bioinformatics*. 2020;36(8):2628–2629. 726 727

62. Vertes RP. Analysis of projections from the medial prefrontal cortex to the thalamus in the rat, with emphasis on nucleus reuniens. *Journal of Comparative Neurology*. 2002;442(2):163–187. 728
729
730
63. Giustino TF, Maren S. The role of the medial prefrontal cortex in the conditioning and extinction of fear. *Frontiers in behavioral neuroscience*. 2015;9:298. 731
732
733
64. Capuzzo G, Floresco SB. Prelimbic and infralimbic prefrontal regulation of active and inhibitory avoidance and reward-seeking. *Journal of Neuroscience*. 2020;40(24):4773–4787. 734
735
736
65. Reed GF, Lynn F, Meade BD. Use of coefficient of variation in assessing variability of quantitative assays. *Clinical and Vaccine Immunology*. 2002;9(6):1235–1239. 737
738
739