Assessing the Robustness of Cluster Solutions Obtained from Sparse Count Matrices

Kathleen M. Gates*, Zachary Fisher*, Cara Arizmendi*, Teague R.Henry*, Kelly Duffy*, & Peter J. Mucha**

*Department of Psychology & Neuroscience, **Department of Mathematics, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA

Author Note

This work was supported by Grant R01 EB022904-01 from the U.S. National Institutes of Health: National Institute for Biomedical Imaging and Bioengineering (PI: Gates). All of the data used herein were previously made publicly available. Some of the ideas presented here were presented during a talk given by the first author at the Duke Network Analysis Center symposium series.

Abstract

Psychological researchers often seek to obtain cluster solutions from sparse count matrices (e.g., social networks; counts of symptoms that are in common for two given individuals; structural brain imaging). Increasingly, community detection methods are being used to subset the data in a data-driven manner. While many of these approaches perform well in simulation studies and thus offer some improvement upon traditional clustering approaches, there is no readily available approach for evaluating the robustness of these solutions in empirical data. Researchers have no way of knowing if their results are due to noise. We describe here two approaches novel to the field of psychology that enable evaluation of cluster solution robustness. This tutorial also explains the use of an associated R package, `perturbR`, which provides researchers with the ability to use the methods described herein. In the first approach, the cluster assignment from the original matrix is compared against cluster assignments obtained by randomly perturbing the edges in the matrix. Stable cluster solutions should not demonstrate large changes in the presence of small perturbations. For the second approach, Monte Carlo simulations of random matrices that have the same properties as the original matrix are generated. The distribution of quality scores ("modularity") obtained from the cluster solutions from these matrices are then compared with the score obtained from the original matrix results. From this, one can assess if the results are better than what would be expected by chance. `perturbR` automates these two methods, providing an easy-to-use method for psychological researchers. We demonstrate the utility of this package using benchmark simulated data generated from a previous study and then apply the methods to publicly available empirical data obtained from social networks and structural neuroimaging.

Assessing the Robustness of Cluster Solutions Obtained from Sparse Count Matrices

Introduction

Cluster analysis has been widely used in social science research since at least 1943 (Cattell, 1943). Researchers have employed cluster analysis for achieving numerous ends, with objectives ranging from identifying clusters of behaviors that tend to co-occur in individuals (e.g., Newby & Tucker, 2004) to classifying subsets of brain regions that activate together across time (e.g., Mezer, Yovel, Pasternak, Gorfine, & Assaf, 2009). Increasingly, a complementary class of analyses called "community detection" has been used in psychological studies to identify clusters of individuals in matrices ranging from social networks (e.g., Papadopoulos, Kompatsiaris, Vakali, & Spyridonos, 2012) to constellations of symptoms (Bringmann, Lemmens, Huibers, Borsboom, & Tuerlinckx, 2015), much like traditional cluster analysis applications. One particular community detection algorithm, Walktrap (Pons & Latapy, 2006), has many similarities with cluster analysis while also bringing a number of additional benefits. Chief among these benefits, Walktrap and other modularity-based approaches can reliably recover clusters without *a priori* knowledge of the number of clusters in the data (Gates, Henry, Steinley, & Fair, 2016; Orman & Labatut, 2009), and in some cases performs better than factor analytic approaches (Golino & Epskamp, 2017). For these reasons, Walktrap is increasingly used by psychologists (e.g., Dalege, Borsboom, van Harreveld, & van der Maas, 2017; Jones, Mair, Riemann, Mugno, & McNally, 2018) and we expect this trend to continue given the emerging popularity of network psychometrics (Borsboom & Cramer, 2013).

Despite the demonstrated advantages, much like cluster analysis, there are no suitable fit indices or quality functions a researcher can use to ascertain if a solution is objectively good (Tonidandel & Overall, 2004). Walktrap performs well in terms of its ability to recovery the underlying cluster assignments in data when they exist (Gates et al., 2016; Orman & Labatut, 2009; Pons & Latapy, 2006). However, like all entirely data-driven approaches, it can sometimes provide cluster solutions even when strong cluster separation does not exist (for further discussion about this and community detection in general, see (Fortunato, 2010; Fortunato & Hric, 2016; Porter, Onnela, & Mucha, 2009; Shai, Stanley, Granell, Taylor, & Mucha, 2017) . Hence there is a great need for methods capable of determining if a solution obtained from Walktrap is stable and should be retained. The present tutorial introduces two general approaches, originally developed in the physics and neuroimaging literatures, for evaluating the robustness of cluster solutions. The paper also provides instruction on how to easily implement this approach for cluster solutions obtained from Walktrap conducted on undirected count matrices.

We begin by defining terminology in order to enable discussion of cluster analysis in general terms that cut across the diverse data types employed by psychological researchers. These terms are taken from graph theory literature (Rubinov & Sporns, 2011). At the foundation of community detection analysis is a symmetric, square matrix that contains information about how the units of analysis are related. Depending on the literature this matrix is referred to as a proximity, distance, similarity or adjacency matrix. In the present paper, this matrix is simply referred to as a "matrix" to facilitate a more general discussion[1]. Here, the unit of analysis is called a "node" (or "vertex"). Nodes may represent variables, people, or brain regions, for instance. The matrix elements themselves are "edges" and can be weighted (e.g., correlation matrices) or binary (e.g., social networks). Edges as defined here are equivalent to commonly used terms in other literatures such as "links", "ties", or "connections". The sum of the edge weights for a given node is the node "strength". "Undirected matrices" are bidirectional, for instance when two people nominate each other as friends in a social matrix study.

We are interested here in two main types of weighted matrices: dense weighted and sparse count. In dense weighted matrices, most or all of the edges have a weight, such as in correlation matrices. Sparse count matrices are different in that they contain only positive

[1]Please note that in the papers informing the present work, the matrix described here is referred to as a "graph" or "network". However, based on reviewer suggestions we use "matrix" throughout this paper so as to not confuse the intended audience of this journal who may not be familiar with these terms.

integers and the number of nonzero edges is less than the number of nodes (Newman & Girvan, 2004). While Walktrap performs well on both dense and sparse weighted matrices (Gates et al., 2016), the matrix evaluation methodology described herein is specifically tailored to sparse count matrices.

Some examples of symmetric sparse count matrices may help in solidifying these terms. One example would be data on co-authorship among academics. Here, the nodes are researchers and edges indicate the number of times each pair of researchers have co-authored a paper. Results from cluster analyses performed on co-authorship networks have identified clusters of individuals who commonly work together, with results indicating researchers have an overall tendency for researchers to publish with those who are in their own field (Girvan & Newman, 2002). Another type of matrix indexes the number of shared attributes among pairs of individuals. As with the prior example the nodes represent individuals. However in this case the edges may represent similarities in aspects of their dynamic processes as done in Subgrouping - Group Iterative Multiple Model Estimation (S-GIMME; Gates, Lane, Varangis, Giovanello, & Guskiewicz, 2017). Clustering of this type of matrix arrives at subgroups (or classifications) of individuals who have similar temporal multivariate relations (Lane, Gates, Pike, Beltz, & Wright, in press; Price, Gates, Kraynak, Thase, & Siegle, 2017). Clustering methods can be used on other types of count matrices as well. Structural neuroimaging data (specifically, diffusion tensor imaging or DTI) is often cast in terms of count matrices that can be clustered. In this case the nodes are brain regions and the edges are counts of physical tracts between brain regions (Bassett, Brown, Deshpande, Carlson, & Grafton, 2011). Clustering this data reveals subsets of brain regions that tend to be highly connected. As these examples demonstrate, undirected sparse count matrices are being used to uncover meaningful relations across a number of diverse data types and research questions.

Regardless which type of matrix used, “cluster solution” in this paper refers to the community partition obtained from a Walktrap analysis. This solution is a vector of labels containing the cluster (or “subgroup” or “community”) assignments for each individual. We use the term “cluster assignment” to describe the specific cluster a given node is assigned to. For a given cluster solution, each node may only be assigned to a single cluster. Good cluster solutions typically separate nodes into clusters with other nodes with whom they have high edge weights (Radicchi, Castellano, Cecconi, Loreto, & Parisi, 2004). Ideally, the average edge weight within clusters is higher than those between nodes in different clusters. Indeed, modularity directly quantifies the total weight of within-group edges relative to that expected under a particular network model (Newman, 2006). For social network matrices, that would mean that the individuals have more friends in common with individuals within their cluster than with individuals outside their cluster. In brain data, a good cluster solution would mean that brain regions have higher connectivity with other brain regions in their cluster when compared to their connectivity with brain regions outside their cluster. Cluster solutions can be said to be robust if the cluster assignments for the nodes are not due to chance and stable if the overall cluster solution does not change drastically from minor perturbations (Karrer, Levina, & Newman, 2008).

A cluster solution can only be as good as the data that are used to generate the matrices. Many things can contribute to a cluster solution not being robust. Perhaps the features chosen do not adequately separate nodes into subsets. As an example, Gates and colleagues (2017) investigated the impact of feature selection on the resulting community assignments when using data simulated to emulate human dynamic processes. Results from their tests showed that solutions varied greatly based on the features selected to generate the matrix. Another difficulty in community detection is that the underlying data simply does not contain clusters regardless of the features selected. Looking at a social network matrix with the intention of identifying subsets of individuals who tend to co-author together would be fruitless if all individuals in the sample co-author with similar frequency. In this way the edge weights would be uniform and random. Here, the best cluster solution is either one cluster or placing all individuals in their own cluster (both solutions indicate a lack of clusters). Due to chance, clustering approaches may still find a solution indicating clusters even if no clear separation exists in the data (Breckenridge, 2000; Clatworthy, Buick, Hankins, Weinman, & Horne, 2005). The underlying point is that the robustness of a solution is dependent on the data: some data simply might not have nodes that strongly separate into clusters. For these reasons, researchers should evaluate the robustness of their final solutions to ensure they are not due to chance.

The structure of the paper is as follows. First, a brief overview of common approaches for evaluating cluster solutions is provided. We then move to describing approaches that have emerged in other literatures: (1) comparison of the solution obtained from the original matrix to those obtained in matrices that have had their edge weights incrementally altered ("perturbed"); and (2) comparison of a relative quality score obtained from the original solution with the distribution of quality values ("modularity") obtained from solutions obtained from random matrices. We then provide instruction on how these approaches can be conducted via the R (R Core Team, 2017) package `perturbR` (Gates, Fisher, & Arizmendi, 2018). Publicly available simulated data is used for the interested reader. We end the demonstration with four empirical examples to illustrate a range of robust and non-robust solutions: two social network matrices where the nodes are individuals and two neuroimaging matrices where the nodes are brain regions. Code and data examples are available here: `https://osf.io/ucj9f/`.

Traditional Approaches for Evaluating Cluster Solutions

Unlike other types of analysis (e.g., structural equation modeling, regression), statistical tests and objective measures are not readily available for evaluating cluster solutions (Tonidandel & Overall, 2004). This makes it challenging to tell if a given cluster in a solution objectively contains nodes that are in some way more related to each other than they are to nodes outside their cluster. However, relative measures exist for evaluating the quality of cluster solutions as compared to other solutions based on the same underlying data (Milligan, 1981). One such measure, modularity, has recently emerged as a highly useful indication of cluster solution quality (Newman & Girvan, 2004). Modularity is a quality function that becomes higher in value as the clusters in a given matrix become more distinct (Newman, 2006). It quantifies the within-cluster edge weights minus the between-cluster edge weights after controlling for weights that may occur by chance. In practice researchers tend to consider modularity values above 0.30 to indicate strong communities (e.g., Meunier, Achard, Morcom, & Bullmore, 2009; Power, Fair, Schlaggar, & Petersen, 2010). Despite its merits in identifying the best solution from a subset of possible solutions, modularity has been shown to be unsuitable as an absolute measure of cluster solution robustness since high modularity can result even in data with poor cluster separation. Specifically, random matrices (which by definition have no cluster structure) can sometimes have high modularity in the absence of true clusters (Guimera, Sales-Pardo, & Amaral, 2004; Kehagias & Pitsoulis, 2013). In a more recent study, modularity was found to have no relation to the accuracy of cluster assignment recovery in Monte Carlo simulations (Gates et al., 2016, 2017). Like other metrics used for arriving at and evaluating optimal cluster solutions (see Mulligan & Cooper, 1985), these findings suggest that modularity is best used as a relative measure for comparing possible internal solutions but not as an absolute measure of the quality of cluster solutions.

To overcome this difficulty, researchers sometimes attempt to evaluate their solutions in a variety of ways. One option is to look at replicate solutions across samples. While this approach can help identify a solution that should be rejected, a successful replication in itself does not guarantee the validity of a solution (Aldenderfer & Blashfield, 1984). Furthermore, it is often unrealistic for researchers to obtain multiple samples. Another common approach is to analyze the same data with multiple clustering approaches (Henry, Tolan, & Gorman-Smith, 2005; Morey, Blashfield, & Skinner, 1983). This approach comes with problems since each method optimizes different criteria and clusters based on different features of the data (Clatworthy et al., 2005). Additionally, simulation studies indicate that clustering approaches have varying degrees of reliability in the methods themselves (e.g., Gates et al., 2016). As such, differences in cluster solutions could be due to poor recovery of communities from some of the approaches. The most common approach for evaluating cluster solutions is cross-validation (Clatworthy et al., 2005). Here, a sample is often split in half to ascertain if similar clusters are found in both sub-samples. This approach can be done with matrices by subsetting nodes into two or more matrices, and then conducting analysis on each of these matrices. A problem with this approach is that the sample size is reduced, and there are many different optimal cluster solutions that may occur in these cases (Barbaranelli, 2002; Thompson, 1994). In yet another set of approaches, researchers test for significant differences between the clusters on the features upon which they were clustered. For instance, one could ascertain if the within-cluster average weight is higher than the between-cluster average weight. While intuitive, this approach tends to always find significant differences and hence is not suitable for evaluating the robustness of solutions (Aldenderfer & Blashfield, 1984).

Two additional options involve perturbing the original data: jackknife and bootstrapping. In jackknife approaches (Crask & Perreault Jr, 1977; Snijders & Borgatti, 1999), one node is removed from analysis and the stability of results investigated. Bootstrapping approaches involve randomly sampling the nodes with replacement (Efron & Diaconis, 1983) to see how consistently the same cluster assignments are obtained. These approaches do not suffer from the issues seen in the previous approaches and can be very helpful in seeing if similar cluster solutions arise when some nodes are left out. A major drawback noted by Barbaranelli (2002) is that jackknife and bootstrapping techniques draw all of their data from the same sample (King, 1997). An additional critique is that the nodes are, by definition, thought to be interdependent in matrices; as such, removing (or replacing) an entire node will have a cascading effect on the resulting solutions while also changing the descriptive properties of the matrix.

A final option is to randomly permute cluster assignments and identify with some quality function, such as modularity, if the random assignments reflect the data poorer than the obtained assignment. The issue with this approach is that many methods for arriving at cluster solutions (appropriately) maximize a quality function. As such it is known that the solution provided to the user will have a higher value than any random solutions. While looking at the distribution of obtained modularity values could be informative, no clear guidelines exist with regards to thresholds or criteria for identifying poor solutions.

In the psychological literature, there has a been a dearth of new approaches for evaluating cluster assignment solutions, leaving researchers with very few options to evaluate the robustness of their findings. The present paper introduces a few methods taken from the graph theory literature. Before introducing these methods we introduce Walktrap (Pons & Latapy, 2006), a reliable method for clustering data that will be used as the basis for the evaluation methods to follow.

Arriving at Reliable Cluster Solutions

It is necessary that clustering approaches provide accurate results in order to evaluate the robustness of a given solution. Great strides have been made in methods that can accurately recover the cluster solution for a given set of data among a set of nested solutions. These advances were sparked by the introduction of the quality function modularity (Girvan & Newman, 2002; Newman, 2006; Newman & Girvan, 2004). A number of methods have been developed that maximizes modularity to find the best partition for a given matrix. These methods, which fall under the umbrella of community detection algorithms (for reviews, see Fortunato, 2010; Fortunato & Hric, 2016; Porter et al., 2009; Shai et al., 2017), provide benefit over traditional cluster analysis by not requiring researchers to specify *a priori* the number of clusters, or to arbitrarily decide where the best split occurred.

Simulation studies provide insight into which approaches tend to accurately recover the data-generating cluster assignments. Much variability exists in terms the reliability of clustering approaches. Some methods were developed for very large, binary matrices but do not always provide reliable results for smaller, weighted matrices. We additionally note that, for large networks, modularity suffers from a resolution limit (Fortunato & Barthelemy, 2007), motivating inclusion of a resolution parameter (Reichardt & Bornholdt, 2006) that may then be selected by various methods (see Weir, Emmons, Gibson, Taylor, & Mucha, 2017). However, we will ignore these issues for the small matrices considered here. Other methods are not deterministic and can return very different results on the same data. Still, a number of useful and reliable approaches exist for recovering community structures. To keep focus, the present paper utilizes one such approach: Walktrap (Pons & Latapy, 2006). Walktrap is available in the `igraph` package (Csardi & Nepusz, 2006) and is used within the `perturbR` package described here. Prior work on simulated sparse count matrices has suggested that Walktrap performs well for matrices with diverse data properties (Orman & Labatut, 2009) including small matrix sizes (Pons & Latapy, 2006) and when within-group edge weights are relatively small and cluster sizes unequal (Gates et al., 2016). Here we briefly discuss this approach and refer the reader to additional resources for more details.

Walktrap (Pons & Latapy, 2006) first recasts the matrix into a new (transition) matrix and then conducts standard cluster analysis. Specifically, Walktrap begins by generating a matrix of transition probabilities for each pair of nodes in the user-provided matrix. Each element of the transition matrix represents the probability of going from one node to another given node based on a random walk of a certain length (typically 4). The transition probabilities, which are based on node degrees or strength (i.e., sum of edge weights from a given node to other nodes), are used to arrive at this distance measure for each pair of nodes. This places the matrix into metric space much like cluster analytic methods. A traditional hierarchical clustering technique (Ward's method; Ward, 1963) is then applied to this distance matrix, forming communities by minimizing the sum of squared distances of each node to the other nodes within its own community. Conceptually, this approach aims to arrive at communities that have more transitions among nodes within the same cluster than with nodes outside their cluster. This agglomerative approach combines nodes into larger and larger clusters until they are all in the same cluster. After obtaining these partitions, the partition with the highest modularity is selected.

Results from even the most reliable approaches could prove not to be robust given the qualities of the data. This seemingly paradoxical scenario - that of a method being reliable yet results not being robust - is not so dissimilar to situations psychologists often encounter. For instance, conducting regression may be considered a reliable approach in a statistical sense, but perhaps the inferences change when an outlier in the data is removed from the sample. Concomitant with the development of community detection approaches has been the emergence of techniques for evaluating the robustness of results on a given data set. We now turn to these methods.

Theoretical Foundation of Current Approaches for Evaluating Robustness

Given the issues with previous methods for quantifying robustness of solutions, researchers in the fields of physics and neuroimaging have provided techniques to aid in assessing the stability of cluster assignments. These approaches echo the often unheeded advice of Aldenderfer and Blashfield (1984), who argue that the best approach for validation is to generate many samples of random data that have the same properties as the original data and then compare cluster solutions of the randomly generated data to that of the original.

The first approach to be described here does exactly this. As described by Karrer and colleagues (2008), the objective is to iteratively and randomly perturb, or alter the value of, the edges of the original matrix via random replacement of the edge values to arrive at rewired matrices with similar properties as the original matrix. The process essentially adds increasing levels of noise to the overall matrix, with a random matrix emerging at the highest degree of perturbation (see Figure 1). Solutions from the original and iteratively perturbed matrices are then compared, both visually and quantitatively.

By creating a random matrix one explicitly uses the notion of a null model in the evaluation. The second approach used also utilizes these random matrices. Given that modularity cannot be used in an absolute sense and can be highly influenced by the degree of sparsity, it is helpful to use the random matrices to arrive at a null distribution of expected modularity for matrices with similar qualities as the original but lacking a clear cluster solution. The modularity obtained from the original matrix solution can then be compared to the distribution found from a series of random matrices (from the null model) similar to the approach described by Bassett and colleagues (2013). This directly allows for null hypothesis testing: one can see if the modularity for the original matrix is higher than expected from a matrix with similar properties but no real clusters.

This set of methods is similar to the approaches described above in that (1) aspects of the sample are changed and (2) the similarity between the original and altered solutions are compared. Basing the comparison on random matrices rather than split-sample or bootstrapped matrices provides substantial benefit and marks the primary difference between prior approaches used in the psychology literature and the approaches introduced here. Namely, the approaches described here do not require additional samples for testing solution stability and keep the number of nodes constant. Furthermore, qualities from the original matrix are maintained during the subsequent perturbations. The driving heuristic is that cluster solutions can be said to be stable and robust if the cluster assignments for the nodes are not due to chance, and the overall cluster solution does not change drastically from minor perturbations (Karrer et al., 2008).

These approaches follow two guiding principles. First, stable cluster solutions should be robust to minor perturbations of the matrix edges. That is, if relatively few nodes have one or two different edge weights than the original matrix, then the cluster assignments for the nodes as a whole should not change drastically. One would expect, however, some change in the cluster assignment - particularly for the nodes for whom the edges were randomly switched. Perturbing edges leads to a rewired matrix that has some differences in the edges such that for some individuals their edge strengths are randomly generated. Importantly, the rewired matrices should be similar to the original matrix in terms of descriptive properties such as strength distribution. The original matrix solution is then compared to random matrices with increasing degrees of perturbation.

The second principal is that modularity obtained from the original matrix should be higher than that seen in random matrices if there truly are distinct clusters in the matrix (Bassett et al., 2013). As noted above, modularity is not an absolute measure of community solutions but still may be useful as a relative indicator of solution quality. By rewiring the matrix in a way that maintains the general features of the original matrix one can arrive at a distribution of modularity that would be expected from a random matrix with similar strength distribution as the original to identify if the modularity obtained from the original matrix is higher than what would be expected by chance.

Application of Current Approaches

Conducting these two approaches requires the generation of new matrices where edges are randomly perturbed. For approach one, there is a comparison of how different the community assignments from the rewired matrices are from that of the original matrix. For approach two, a comparison of modularity obtained from the original cluster solution is made with a set of solutions obtained from a random matrix. From these approaches, one can graphically depict how robust the cluster solution is to increasing levels of perturbation as well as statistically test the quality of the original solution.

The specific steps described in detail below are used for both approaches:

1. Arrive at a cluster solution for the original matrix using Walktrap.

2. Randomly perturb a specific proportion of edges (α) to rewire the original matrix.

3. Obtain a cluster solution for the rewired matrix.

.

These steps are depicted in Panels 1-3 of Figure 2. For the first approach described here, Steps 2 through 3 are repeated a pre-specified number of times in increasing increments of α until 100% of the edges have been perturbed. For the second approach, only the final matrix with all edges perturbed (i.e., a random matrix) is used.

Having obtained the set of increasingly perturbed matrices, one can plot the degree of similarity for the cluster solutions found in the original matrix and rewired matrices across all levels of perturbation. One can also see if the modularity from the original solution is higher than would be expected from solutions obtained from random matrices. We first describe the 3 steps common to both approaches before going into detail about the two approaches and distinct output. Both are provided here as they carry unique strengths. Thus it is recommended they be used in tandem.

Perturbing Matrices and Obtaining Solutions on Rewired Matrices

The second step requires generating rewired matrices in varying degrees of random perturbation denoted as α. The random perturbation approach must be appropriate for the qualities of the data on hand. The present paper (as well as the package `perturbR`) focuses on the evaluation of cluster solutions obtained for weighted matrices with positive integers (i.e., sparse count matrices). It must be strongly noted that the present approach is not intended for densely weighted, non-integer matrices (such as correlation matrices).

The method for arriving at new matrices with increasing degrees of randomness stems directly from the Erdos-Renyi (ER) binary random matrix. The ER random matrix is often used as a point of comparison as it provides a null matrix where edges are equiprobable across all nodes. It is particularly useful in the present context as these matrices have no defined clusters. The weighted extension of the ER matrix needs to provide a set of edges randomly drawn from a given distribution of node strengths. Considering a matrix $\mathbf{A}$ with element A_{ij} representing the edge weight for nodes i and j, node strength is the sum of weights for a given node:

$$s_i = \sum_{j \neq i} w_{ij} \tag{1}$$

where w_{ij} is the weight for the edge between nodes i and j. For weighted random matrices, all nodes need to be statistically equivalent, meaning here that the properties describing them need to have come from the same distribution. For this reason, Garlaschelli (2009) introduced an approach which ensures the total weight of the matrix remains constant across all generated weighted random matrices. This is preferred over methods that only ensure the same number of edges exist in the matrix since this would not provide similar properties across weighted random matrices. Note that the present approach for perturbing matrices differs from that used by Karrer et al. (2008), which retained node degree while perturbing edges in favor of emulating the null matrix used in modularity rather than a completely random matrix as used here. To arrive at random weighted matrices, we consider the weight distribution to geometric with parameter $1 - p$ and the node strengths to have a negative binomial distribution with parameters $N - 1$ and $1 - p$. The probability of a given edge weight occurring between any two nodes is given as:

$$q(w) = p^w(1 - p) \tag{2}$$

where p is a constant. Defining the total original matrix weight as $W = \sum_{i<j} w_{ij}$ (i.e., the sum of edge weights in the lower triangle of the matrix), we can now find the optimal criterion for tuning p:

$$p* = \frac{2W}{N(N-1) + 2W}. \tag{3}$$

One can randomly arrive at edge weights from this distribution using Bernoulli trials with success probability $p*$. Once an edge is randomly selected to be perturbed (at a given α) it is set to zero. If the first trial is not successful the edge weight remains at ‘0’. However, if the first trial is successful it is set to one and the Bernoulli trials continue for that edge. Each time a trial is successful a '1' is added to that edge weight, stopping once a failure occurs and moving to the next edge. To generate a completely random weighted matrix, one would iterate across the entire number of unique edges to arrive at a new set of weighted edges. This is what we refer to here as a rewired matrix that has had 100% of its edges perturbed: all possible edges have been perturbed using these Bernoulli trials.

However, since each edge is considered independently we can also use this approach to perturb only a fraction of edges and arrive at rewired networks with only a portion of randomly drawn edges replacing the original ones. For example, increasingly perturbing the number of edges by 1 percent ($\alpha = 0.01$) increments can be done using this Bernoulli trial approach. What follows is a brief description of this process intended to further elucidate what is meant by increasing perturbations.

Figure 1 depicts a matrix undergoing increasing perturbations. Panel A shows a count matrix where the edges indicate the degree to which the nodes are similar. Empty areas indicate nodes that have zero-weighted edges. (These nodes were each placed in their own communities.) The large square towards the center of the matrix represents nodes placed in the same community. The smaller square in the lower right hand corner contains edges for nodes placed in a different community. When 33% of the matrix is perturbed (Panel B) we see some of the individuals who had edge weights of zero in the original matrix now have higher counts. One clear change is that the edges in the center square have become lighter and there are more zeros. Additionally, the nodes that previously had no edges in common with other nodes now have some edge weights. With 66% of edges perturbed, the edges in the rewired matrix (Panel C) are becoming even more evenly distributed across nodes with no clear community structure. Finally, in Panel D a completely random matrix is depicted with 100% ($\alpha = 1$) of edge weights perturbed.

Step 3 involves conducting Walktrap on each rewired matrix. In summary, Steps 1 through 3 are used for (1) comparing the community assignments obtained from the original matrix with those obtained from the rewired matrices and (2) comparing the modularity obtained from the original matrix to the distribution obtained from random matrices that have similar properties as the original. We now discuss these two approaches in turn.

Approach 1: Comparisons of Cluster Assignments

Having perturbed the edges of the original matrix according to the increasingly large α values we can now compare the rewired matrices' community solutions with the original for each repetition at each α level. Two popular methods exist for measuring the degree to which two community solutions differ: Hubert-Arabie Adjusted Rand Index (ARI) and Variation of Information (VI).

The ARI (Hubert & Arabie, 1985) assesses the similarity of node cluster assignments while taking into account what would be expected by chance. Importantly, the ARI does not require that the two cluster assignments obtained from two solutions use the same labels. For instance, what is '1' in one solution might be '2' in another solution with both solutions indicating the identical set of individuals. Rather than quantify these assignments as being different, the ARI takes into account the proportions of nodes consistently clustered together in two given community assignment solutions regardless the arbitrary labels in the solutions. The ARI provides the ratio of similar nodal assignments to the total number of possible assignments. This can immediately be seen in the original Rand Index (RI):

$$RI = \frac{a+d}{a+b+c+d} \tag{4}$$

where each pair of nodes provides a count for either a, b, c, or d. The value a indicates the number of pairs placed in the same community for both community assignment solutions. d indicates the count of node pairs that have different cluster assignments in both solutions. Thus the numerator represents the number of nodes in which community solutions agreed in their assignments. Both b and c indicate different placement of nodes, with the former indicating the counts of node pairs having the same cluster assignment for the first cluster solution but different cluster assignments for the second cluster assignment and the latter indicating the opposite. From this equation it is clear the Rand Index is a ratio of similarities in node assignment patterns to the total possible node assignments. A drawback of the original RI is that it may be overinflated due to chance. Hubert and Arabie (1985) provide ARI as a solution by controlling for chance:

$$ARI = \frac{\binom{N}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{N}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]} \tag{5}$$

The ARI is particularly useful for comparing results across samples since it has an upper limit of 1.00, which indicates perfect similarity in the two cluster assignments. Lower values indicate incrementally worse recovery with negative values possible. An ARI of at least 0.80 is generally considered evidence of similarity in cluster solutions. Another important quality is that the ARI penalizes for combining two communities into one (i.e. requires high sensitivity and high specificity) or splitting a community into two smaller ones (Steinley, 2004). One criticism of the ARI is that non-locality exists. That is, the ARI for community assignments that differ only in one cluster depends on the cluster assignment pattern for the remaining nodes. This can result in downward biases in the ARI computation (Karrer et al., 2008; Meilă, 2007).

The VI (Meilă, 2007) has previously been used to compare cluster assignments obtained from different data sets such as those described here (Karrer et al., 2008). Drawing from concepts of information and entropy, the VI considers how much information each cluster solution has and how much information one cluster solution provides about the other. Given a total of K clusters in a given solution C, the probability of being in any one cluster k is:

$$P(k) = \frac{n_k}{N} \tag{6}$$

where n_k provides the total number of nodes in cluster k and N the total number of nodes in the matrix. The uncertainty or entropy of the vector of cluster assignment probabilities is:

$$H(C) = -\sum_{k=1}^{K} P(k) log P(k) \tag{7}$$

The entropy value depends on the relative proportions of the clusters assignments, with zero uncertainty occurring only when there is one cluster. The mutual information of the cluster solutions, or how much information one solution contains about the other solution, can be provided as follows:

$$P(k,k') = \frac{|C_k \cap C'_{k'}|}{N}. \tag{8}$$

The prime indicates results from the second community solution. Of note, much like the ARI the cluster in the second solution ($C'_{k'}$) need not have the same label as the first (C_k). Hence $P(k,k')$ provides the probability of overlap of assignments for any given pair of cluster labels $C_k, C_{k'}$ found in the two solutions. With this information the mutual information associated with the two clusterings C, C' can be computed:

$$I(C,C') = \sum_{k=1}^{K} \sum_{k'=1}^{K'} P(k,k') log \frac{P(k,k')}{P(k)P(k')} \tag{9}$$

Taking together the individual entropies associated with the two cluster solutions $H(C)$ and $H(C')$ and the mutual information that each provides for the other ($I(C,C')$) one can arrive at the VI:

$$VI(C,C') = H(C) + H(C') - 2I(C,C'). \tag{10}$$

The VI complements the ARI. Unlike the ARI the VI is not bound by any limits. While this may limit interpretability by requiring the solutions be compared in a relative rather than absolute sense, the VI has properties that make it superior. One, the VI has been proven to be a true metric by obeying the triangular inequality. This means that it has all the properties of a proper distance measure. Two, it does not suffer from the problem of non-locality noted in the ARI. In many cases both the ARI and VI will lead to similar conclusions.

The original cluster assignments are compared to solutions obtained from the rewired matrices using ARI and VI, and these comparisons are typically plotted. The figures depict the average degree to which the rewired matrix cluster solution matches the original for each α. Figure 2 provides a graphical overview of the steps taken to compare the cluster assignments from the original matrix with those from increasingly random matrices. Steps 1 through 3 are done for both approaches; Steps 4 through 6 are specific to this first approach. In addition to providing the final plot (Figure 2, Panel 6), `perturbR` also provides the values for ARI and VI at each level of α for each repetition. More details are provided below in the section on using the function and manipulating the output.

Approach 2: Comparison of Modularity Values

One can also conduct a statistical test of quality. The second approach uses modularity, which is an appropriate quality function for arriving at a relative measure of cluster solution quality. For weighted matrices, modularity is defined as:

$$Q_w = \frac{1}{2W}\sum_{ij}[Aij - \frac{s_i s_j}{2W}]\delta_{ij} \tag{11}$$

where δ_{ij} equals zero if the two nodes i and j are not in the same cluster and 1 if they are, $A_{ij} = w_{ij}$ and provides the weight for the given edge between i and j, s_i is the overall strength of node i, and W is the overall weight of the matrix. A noteworthy aspect of this equation is that it takes into account what would be expected by chance given the nodes' respective strengths (as seen in the subtraction of $\frac{s_i s_j}{2W}$).

Despite its benefits modularity lacks a known distribution and therefore cannot be used as an absolute measure of fit. As such additional methods are needed to assess the likelihood that the cluster solution is better than what would be found in a random matrix with properties similar to the original. As noted above, using Monte Carlo simulations of data to evaluate cluster solutions has historically been highlighted as a best practice

(Aldenderfer & Blashfield, 1984). The use of solutions from random matrices as a point of comparison for modularity when looking across numerous matrices has been used on neuroimaging data (e.g., Bassett et al., 2013) but has yet to become a readily available and consistently used approach in psychology at large.

In following this line of work, one can use Monte Carlo simulations to arrive at a distribution of modularity values obtained from cluster solutions using random matrices with similar overall weight as the original. Using the traditional probability cutoff of 0.05, the value in this distribution that corresponds to the 95th percentile would indicate a threshold for significance. If the modularity obtained from the cluster solution from the original matrix is higher than this value, then the value is higher than expected by chance given this distribution.

`perturbR`

Using the `perturbR` function

The two emerging approaches described above are available in the freely distributed `perturbR` package by using the function of the same name. When describing the use of the package we utilize publicly available benchmark data (https://dataverse.unc.edu/dataset.xhtml?persistentId=doi:10.15139/S3/1234) that have previously been thoroughly evaluated (see Gates et al., 2016, for details). For the purposes of demonstration here we selected an example that almost perfectly recovered the data-generating pattern: data set number 60 of Simulation 5. Here, there were 75 nodes, moderate average node strength, and approximately equally sized groups.

The `perturbR()` function arguments with defaults indicated are as follows:

```
>  perturbR(sym.matrix     = exampledata,
            plot           = TRUE,
            errbars        = FALSE,
            resolution     = 0.01,
            reps           = 100)
```

where `sym.matrix` is the argument where one indicates the symmetric count matrix to be used by the function. This matrix must be symmetric and can reflect a number of phenomenon studied in psychology. For instance, `sym.matrix` can be a similarity (or "attribute") matrix, containing counts of similar symptoms each pair of individuals have, or a matrix providing counts of fibers between pairs of brain regions, or counts of how often two pairs of individuals speak to each other.

`plot` and `errbars` both pertain to options regarding the figures. The `errbars` argument is a logical where users can indicate if they would like error bars in the plots. The error bars indicate a range of two standard errors above and below the mean values obtained across repetitions at that given α level.

```
> perturbR(sym.matrix     = exampledata,
           plot           = TRUE,
           errbars        = TRUE)
```

.

Figure 3 provides a demonstration of results on the simulated data with the error bar options invoked. The black circles in Figure 3 indicate the average value for the comparison between the cluster assignments from the original matrix and the cluster assignments from the rewired matrix at each level of edge perturbation α. The red triangles indicate the same comparisons made on a matrix that is completely random but has the same properties as the original matrix, thus providing an appropriate null matrix for comparison. This is important for scaling purposes for the VI comparisons (Figure 3B). However, comparison to random matrix solutions has less utility for the ARI results (Figure 3A) since the ARI values have absolute (not relative) cutoffs.

To further aid in interpreting the results two horizontal lines are provided. They indicate the values of similarity found between the original matrix and a matrix where 10% and 20% of nodes (not edges, as in the rewiring phase) were randomly assigned to different clusters. As noted in Karrer and colleague's work (2008) identifying at which α the rewired results cross these lines provides insight into interpretation. In a series of empirical examples, they considered cluster solutions to be robust if the matrix had 20% or more of its edges perturbed (i.e,. $\alpha \geq 0.20$) before intersecting with the line representing 20% of the nodes being in different clusters. They also note that looking at the distribution of results helps to guide inferences regarding robustness of the original community solution. This heuristic has become commonly acceptable and used to evaluate solutions (e.g., Fair et al., 2009; Gates, Molenaar, Iyer, Nigg, & Fair, 2014; Stevens, Tappon, Garg, & Fair, 2012).

Another option available to users is to manipulate the resolution for the incremental increases in the proportion of edges that are perturbed (α). Steps 2 through 3 (i.e., rewiring and arriving at new cluster solutions for the rewired matrices) are repeated across varying levels of perturbation ranging from default resolution of increments of $\alpha = 0.01$, starting from $\alpha = 0$ (the original matrix solution) to $\alpha = 1.00$ (a completely random matrix). The increases in α defaults to increments of 0.01, or 1 percent, but the user can indicate any number between 0.01 and 0.99 by using the `resolution` argument. The algorithm will run quicker with higher proportions, but the granularity of results will suffer. At the final α level the entire matrix is now random. These steps are iterated at a default of 100 times at each degree of α. The researcher can alter this option using the `reps` argument, which indicates the number of repetitions (i.e., matrices to generate) at each level of α.

Exploring Output

One can look directly at the `perturbR` output and obtain values related to what is seen visually. Table 1 provides description of the available output. We now demonstrate examples of how these output values can be used.

Approach 1: Comparison of Cluster Assignments from Original Matrix to Increasingly Perturbed Matrices. We begin by demonstrating how the first approach described above can be explored with the output. One might want to quantify the point at which the average ARI or VI crosses the 20% line described above. The following commands arrive at this point for the VI values:

```
# Run perturb on simulated data
 > install.packages(perturbR)
 > library(perturbR)
 > fort5_60 <- as.matrix(read.csv("~/Sparse Count/FortData/5_60_fort.csv",
    header = F))
 > fit5_60 <- perturbR(sym.matrix      = fort5_60,
                       plot            = FALSE)
# Obtain the VI value when 20% of the community assignments are randomly
   swapped:
 > fit5_60$vi20mark
   [1] 1.453757
# Identify the index for the alpha level for the first time the average VI
    is greater than this value:
 > min(which(colMeans(fit5_60$VI)>fit5_60$vi20mark))
  [1] 34
# Find alpha that corresponds with this index.
 > fit5_60$percent[34]
  [1] 0.332973
# About 0.33 of the edges need to be changed before the VI is as high as
   when 20% of the nodes have their cluster assignments randomly swapped.
```

In this example, about 33% of the edges need to be perturbed before the cluster solution for the rewired matrix is as different as when 20% of the nodes are randomly placed into different solutions. This can be seen both by looking at the point where the black circles intersect with the lower horizontal line in the VI figure and using the code above. By contrast, a random matrix drops far below this line with only 2% of the edges perturbed in the rewired matrix. The figures provide an immediate evaluation of cluster solutions whereas the code and output allow the user to further investigate the results.

It is also possible to examine the average VI and ARI at the 20% perturbation point and see if it is larger than the these marks. Here we provide an example for the ARI values:

```
> fit5_60$ari20mark
  [1] 0.648735

> mean(fit5_60$ARI[,which(round(fit5_60$percent, digits = 2) == .20)])
  [1] 0.7804633
```

These results indicate that the solutions at this point of perturbation ($\alpha = 0.20$), on average, are more similar to the original solution (average ARI = 0.78) than when 20% of the cluster assignments are randomly swapped (ARI = 0.65). Conducting a one-sampled t-test can evaluate if this difference is significant:

```
t.test(fit5_60$ARI[,which(round(fit5_60$percent, digits = 2) == .20)], mu=
   fit5_60$ari20mark)

 [1] One Sample t-test
 [2] data:  fit5_60$ARI[, which(round(fit5_60$percent, digits = 2) == 0.2)
    ]
 [3] t = 18.58, df = 99, p-value < 2.2e-16
```

We see that the distribution of ARI values at $\alpha = 0.20$ is significantly higher than the ARI value obtained with 20% of the cluster assignments for nodes are randomly changed.

Approach 2: Comparison of Modularity Values Obtained from Original and Random Matrices. The second approach complements the first approach, which has the drawback of having a rather arbitrary threshold (i.e., 20% of node assignments switched). The output also provides a value called `cutoff` which is the modularity value that marks the upper 5% percentile in the distribution of modularity results obtained from random matrices that have the same properties as the original matrix.

```
# The modularity value for which 95% of the solutions from random matrices
    are below:
 > fit5_60$cutoff
   [1] 0.31
# The modularity value from the solution on the original matrix:
 > fit5_60$modularity[1,1]
   [1] 0.70
```

In this example, the cutoff[2] was $Q_{.95} = 0.31$ and the modularity obtained in this simulated data set was $Q_{orig} = 0.70$. Hence the modularity in the original solution was well above the upper 5% threshold obtained from the random matrix simulation results.

The final panel in Figure 3 depicts a histogram of the modularity values obtained for solutions in the random matrices simulated to have properties similar to the original matrix. Researchers can easily obtain similar histograms from the output provided if they would like to explore how the distribution of modularity from the random matrices

[2]Note that values may differ slightly for each run do to the random generation of matrices.

compares to the modularity obtained in the original matrix.

```
> hist(fit5_60$modularity[,which(round(fit5_60$percent, digits = 2)
    ==1.00)], xlim = c(0,1))
> abline(v = (fit5_60$modularity[1,1]), col = "red")
```

Empirical examples

Social Networks

The often used Karate Club social network first introduced by Zachary (1977) is a classic example in network science literature[3]. It consists of counts of interactions between individuals belonging to a university karate club that was undergoing a schism after a disagreement between club president and karate instructor over the prices. In this matrix, the nodes represent 34 members of the karate club, including the president and instructor. The edges are weighted according to the number of interactions between each pair of members in the club over the course of three years. The dataset is regularly used as a benchmark with which to demonstrate recovery of true communities with community detection algorithms, with four clusters (or communities) being the expected outcome. On this weighted, sparse, undirected matrix, Walktrap recovered the expected four clusters of individuals (both here and in the original Walktrap paper, Pons & Latapy, 2006). Figure 4 visually depicts the solution obtained on the original matrix in Panel A. This depiction was obtained using the “plot” function in the `igraph` package and clearly demonstrates the four distinct clusters. The robustness to perturbation is depicted in Panels B and C. Here one can see evidence of robustness in the cluster solution. In both the ARI and VI figures, on average more that 20% of edges must be perturbed before the difference from the original solution crosses the line representing the cutoff described above for changes in 20% of the node assignments. The results from `perturbR` also provide the 95% cutoff point for the distribution of modularity obtained for the random matrices. In this case, the cutoff

[3]The data are publicly available here: http://vlado.fmf.uni-lj.si/pub/networks/data/UciNet/zachary.dat.

point was $Q_{.95} = 0.12$. The modularity obtained from the solution on the original matrix was $Q_{orig} = 0.35$. Thus according to both approaches - the visualization of differences for increasing degrees of perturbation and the distribution of modularity from the random matrices - the results appear to be robust.

The bottom half of Figure 4 depicts results obtained from a different social network: the Bernard and Killworth Fraternity Data (Killworth & Bernard, 1976)[4]. The data represents social interactions among 58 fraternity members who lived in the same house. The nodes are student members of the fraternity and each edge the number of times the pair of students were seen in conversation by an unobtrusive observer. The students were observed every fifteen minutes, 21 hours a day, for five days.

Figure 4D provides the network representation of the nodes. The distinction is not as clear as previously seen, but this may be because of the method used for plotting as well as the increase in nodes, so further exploration is warranted. Figures 4E and 4F provide clear indication that the clusters are not robust. According to both the ARI and VI comparisons, the cluster solution from matrices perturbed at an α of 0.10 differed from the original solution more so than if 20% of the nodes had their cluster assignments randomly changed. Furthermore, the modularity from the original matrix solution, $Q_{orig} = 0.02$ was much smaller than the $Q_{.95} = 0.23$ cutoff obtained from the random matrix simulations. In this example one can conclude that no clear clusters exist.

Diffusion Tensor Imaging (DTI) Data

Structural connectivity matrices for the diffusion tensor imaging (DTI) example were obtained from the UCLA Multimodal Connectivity Database (http:/umcd.humanconnectomeproject.org), a publicly available database of preprocessed connectivity matrices from multiple neuroimaging modalities. Example subjects were randomly selected from a set of 196 subjects from the Nathan Kline Institute

[4]Obtained from https://networkdata.ics.uci.edu/netdata/html/bkFrat.html

(NKI)/Rockland sample of the 1000 Functional Connectomes project[5]. Data were collected and preprocessed according to previous study protocol (see website for details), and deterministic tractomatrixy was performed to obtain connection matrices in the form of sparse count matrices. In deterministic tractomatrixy, the brain is parcellated into distinct regions of interest with tracts seeded in each voxel of each ROI and fibers counted if they intersected at least one voxel of the source ROI and one voxel of the target ROI. As is customary, edges between voxels are not counted if they had an angle greater than 45° as these typically denote different trajectories than the one being quantified. The nodes are brain regions ($N = 188$) and the edges the density of fiber tracts (i.e.,the number of tracts that connect region i with region j). DTI networks are count matrices of the number of white matter tracts, representing neural axons.

Figure 5 depicts the VI results from these two data sets. For this example, we used the argument `reps = 50` to increase processing speed given the size of these matrices. Acceptable solutions appear to be found for both matrices, with on average over 25% of the edges perturbed before the solution is as different as one would expect from randomly reassigning 20% of the nodes (i.e., the top line). That the results are robust is to be expected given physiological constraints of the brain and known clusters of brain regions that exist. Quantitative evidence also suggests robust solutions. The 95% cutoff from the distribution of modularity values obtained from the randomly generated matrices have $Q_{.95} = 0.03$ for both data sets. The results from the original matrices were $Q_{orig} = 0.46$ for both solutions. Hence all evidence suggests these results are robust.

Discussion

The present paper introduced a well-supported set of approaches for evaluating cluster solutions obtained from sparse count matrices (a type of weighted matrix). We provided an overview of approaches used in psychology for evaluating cluster solutions and

[5]Available for download from INDI here: `http://fcon_1000.projects.nitrc.org/indi/pro/nki.html`

highlighted the drawbacks. Next, we briefly described one clustering approach, Walktrap (Pons & Latapy, 2006), that provides accurate and reliable solutions in an entirely data-driven manner. Researchers do not need to provide the number of clusters *a priori* nor select the best cut on a dendrogram - Walktrap arrives at the number of clusters automatically. Having identified a clustering algorithm that provides reliable results, we then introduce methods for evaluating the robustness of these results. Robust cluster solutions should not change drastically with minor perturbations of the data. Finally, we demonstrate the utility of these approaches on benchmark and publicly available data.

The paper also describes how to easily conduct this analysis using the package `perturbR`. First, the algorithm arrives at a cluster solution for the original matrix using Walktrap. Next, the edges of the original matrix are iteratively perturbed in a manner that retains the overall matrix properties. Cluster solutions are then obtained for each rewired matrix using Walktrap. These rewired matrices are used as points of comparisons to identify: (1) how robust the original cluster solution is to minor perturbations of the network and (2) if the modularity value obtained on the original matrix is higher than expected by chance. This is determined by considering the distribution of modularity values obtained from random matrices lacking a community structure but otherwise having the same properties as the original matrix.

We demonstrate how to utilize these approaches with simulated data and present empirical data to highlight the utility of the methods. The simulated data examples indicate that our approach appropriately found the solutions to be robust and modularity to be higher than expected by chance for an example matrix that was indeed clustered correctly. For the empirical examples, we first demonstrate results using the Zachary Karate Club and Bernard-Killworth Fraternity social networks. Here we saw that small degrees of perturbation of the Zachary network did not lead to large differences in cluster solutions. Additionally, the modularity obtained from the cluster solution on the original matrix was much higher than obtained in random matrices with similar properties. A different pattern emerged for the Fraternity data set. Here, small amounts of edge perturbation led to large differences in cluster assignments. Additionally, the modularity obtained was smaller than the 95% cutoff seen in randomly generated matrices. We then demonstrate it's use on a brain networks, showing that robust cluster solutions were found for two individuals.

The modular approaches used herein have previously been evaluated. Still, more work can be done. Results will undoubtedly be influenced by the choice of random matrix used. As seen in work by Bassett and colleagues (2013), inferences made from comparing results to random matrices may differ based on the null used. A benefit of the null model used here is that the resulting matrices are true random matrices. Hence if one wishes to see if clusters exist it is appropriate to compare the solutions to those obtained from matrices where no clusters exist. Still, it is possible that other null matrices may be useful for some research questions. This is particularly true if the matrix being used does not have the same weight or strength distributions that guide the present approach. Arguments could also be made for using the degree-preserving approach in Karrer and colleagues' (2008) paper. This null matrix aligns with the configuration matrix used when calculating modularity.

Another area of development would be extensions of these methods for use with densely weighted matrices, such as correlation matrices. While methods for generating random correlation matrices exists (e.g., Joe, 2006), due to the interdependence of correlation values there does not exist, to our knowledge, a method for iteratively perturbing specific edges as described in the present paper. Specifically, should one correlation value be changed for nodes i and j, then this has implications for the values that can occur between i and j with a third node, k. Each edge perturbation done by simply replacing one edge with a value from a random distribution would render a matrix having different properties from correlation matrices. More work is needed for this and other types of densely weighted matrices.

In research environments the true cluster solutions are unknown. In addition to evaluating robustness, we suggest that researchers also validate their results using external variables (as also suggested by Aldenderfer & Blashfield, 1984) to ensure the clusters are meaningful in addition to being robust. `perturbR` can help researchers determine if their cluster solutions are robust. The set of information provided in the output and described herein provides researchers with additional tools for evaluating such robustness. The package is easy to use and provides interpretable output in addition to figures that are ready for publication.

References

Aldenderfer, M. S., & Blashfield, R. K. (1984). Cluster analysis: Quantitative applications in the social sciences. *Beverly Hills: Sage Publication*.

Barbaranelli, C. (2002). Evaluating cluster analysis solutions: An application to the italian neo personality inventory. *European Journal of Personality*, *16*(S1).

Bassett, D. S., Brown, J. A., Deshpande, V., Carlson, J. M., & Grafton, S. T. (2011). Conserved and variable architecture of human white matter connectivity. *Neuroimage*, *54*(2), 1262–1279.

Bassett, D. S., Porter, M. A., Wymbs, N. F., Grafton, S. T., Carlson, J. M., & Mucha, P. J. (2013). Robust detection of dynamic community structure in networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *23*(1), 013142.

Borsboom, D., & Cramer, A. O. (2013). Network analysis: an integrative approach to the structure of psychopathology. *Annual review of clinical psychology*, *9*, 91–121.

Breckenridge, J. N. (2000). Validating cluster analysis: Consistent replication and symmetry. *Multivariate Behavioral Research*, *35*(2), 261–285.

Bringmann, L., Lemmens, L., Huibers, M., Borsboom, D., & Tuerlinckx, F. (2015). Revealing the dynamic network structure of the beck depression inventory-ii. *Psychological medicine*, *45*(4), 747–757.

Cattell, R. B. (1943). The description of personality: basic traits resolved into clusters. *The journal of abnormal and social psychology*, *38*(4), 476.

Clatworthy, J., Buick, D., Hankins, M., Weinman, J., & Horne, R. (2005). The use and reporting of cluster analysis in health psychology: A review. *British journal of health psychology*, *10*(3), 329–358.

Crask, M. R., & Perreault Jr, W. D. (1977). Validation of discriminant analysis in marketing research. *Journal of Marketing Research*, 60–68.

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal, Complex Systems*, *1695*(5), 1–9.

Dalege, J., Borsboom, D., van Harreveld, F., & van der Maas, H. L. (2017). Network analysis on attitudes: A brief tutorial. *Social psychological and personality science*, *8*(5), 528–537.

Efron, B., & Diaconis, P. (1983). Computer intensive methods in statistics. *Scientific American*, *248*(5), 116–130.

Fair, D. A., Cohen, A. L., Power, J. D., Dosenbach, N. U., Church, J. A., Miezin, F. M., ... Petersen, S. E. (2009). Functional brain networks develop from a âĂIJlocal to distributedâĂİ organization. *PLoS computational biology*, *5*(5), e1000381.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*(3-5), 75–174. doi: 10.1016/j.physrep.2009.11.002

Fortunato, S., & Barthelemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, *104*(1), 36–41.

Fortunato, S., & Hric, D. (2016). Community detection in networks: A user guide. *Physics Reports*, *659*, 1–44.

Garlaschelli, D. (2009). The weighted random graph model. *New Journal of Physics*, *11*(7), 073005.

Gates, K. M., Fisher, Z. A., & Arizmendi, C. (2018). *perturbr: Random perturbation of count matrices.* Retrieved from `https://cran.r-project.org/web/packages/perturbR/index.html`

Gates, K. M., Henry, T., Steinley, D., & Fair, D. A. (2016). A monte carlo evaluatixon of weighted community detection algorithms. *Frontiers in neuroinformatics*, *10*.

Gates, K. M., Lane, S. T., Varangis, E., Giovanello, K., & Guskiewicz, K. (2017). Unsupervised classification during time-series model building. *Multivariate behavioral research*, *52*(2), 129–148.

Gates, K. M., Molenaar, P. C., Iyer, S. P., Nigg, J. T., & Fair, D. A. (2014). Organizing heterogeneous samples using community detection of gimme-derived resting state functional networks. *PloS one*, *9*(3), e91322.

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the national academy of sciences*, *99*(12), 7821–7826.

Golino, H. F., & Epskamp, S. (2017). Exploratory graph analysis: A new approach for estimating the number of dimensions in psychological research. *PloS one*, *12*(6), e0174035.

Guimera, R., Sales-Pardo, M., & Amaral, L. A. N. (2004). Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, *70*(2), 025101.

Henry, D. B., Tolan, P. H., & Gorman-Smith, D. (2005). Cluster analysis in family psychology research. *Journal of Family Psychology*, *19*(1), 121.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*(1), 193–218. doi: 10.1007/BF01908075

Joe, H. (2006). Generating random correlation matrices based on partial correlations. *Journal of Multivariate Analysis*, *97*(10), 2177–2189.

Jones, P. J., Mair, P., Riemann, B. C., Mugno, B. L., & McNally, R. J. (2018). A network perspective on comorbid depression in adolescents with obsessive-compulsive disorder. *Journal of anxiety disorders*, *53*, 1–8.

Karrer, B., Levina, E., & Newman, M. E. (2008). Robustness of community structure in networks. *Physical Review E*, *77*(4), 046119.

Kehagias, A., & Pitsoulis, L. (2013). Bad communities with high modularity. *The European Physical Journal B*, *86*(7), 330.

Killworth, P., & Bernard, H. (1976). Informant accuracy in social network data. *Human Organization*, *35*(3), 269–286.

King, J. E. (1997). Methods of assessing replicability in canonical correlation analysis (cca).

Lane, S., Gates, K. M., Pike, H., Beltz, A., & Wright, A. G. (in press). Uncovering general, shared, and unique temporal patterns in ambulatory assessment data. *Psychological Methods*.

Meilă, M. (2007). Comparing clusterings —an information based distance. *Journal of multivariate analysis*, *98*(5), 873–895.

Meunier, D., Achard, S., Morcom, A., & Bullmore, E. (2009). Age-related changes in modular organization of human brain functional networks. *Neuroimage*, *44*(3), 715–723.

Mezer, A., Yovel, Y., Pasternak, O., Gorfine, T., & Assaf, Y. (2009). Cluster analysis of resting-state fmri time series. *Neuroimage*, *45*(4), 1117–1125.

Milligan, G. W. (1981). A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, *46*(2), 187–199.

Morey, L. C., Blashfield, R. K., & Skinner, H. A. (1983). A comparison of cluster analysis techniques withing a sequential validation framework. *Multivariate Behavioral Research*, *18*(3), 309–329.

Mulligan, C. N., & Cooper, D. G. (1985). Pressate from peat dewatering as a substrate for bacterial growth. *Applied and environmental microbiology*, *50*(1), 160–162.

Newby, P., & Tucker, K. L. (2004). Empirically derived eating patterns using factor or cluster analysis: a review. *Nutrition reviews*, *62*(5), 177–203.

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, *103*(23), 8577–8582. doi: 10.1073/pnas.0601602103

Newman, M. E., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review E*, *69*(2), 026113.

Orman, G. K., & Labatut, V. (2009). A Comparison of Community Detection Algorithms on Artificial Networks. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 5808 LNAI, pp. 242–256). doi: 10.1007/978-3-642-04747-3_20

Papadopoulos, S., Kompatsiaris, Y., Vakali, A., & Spyridonos, P. (2012). Community detection in social media. *Data Mining and Knowledge Discovery*, *24*(3), 515–554.

Pons, P., & Latapy, M. (2006). Computing communities in large networks using random walks. *J. Graph Algorithms Appl.*, *10*(2), 191. Retrieved from `http://www.emis.ams.org/journals/JGAA/accepted/2006/PonsLatapy2006.10.2.pdf`

Porter, M. A., Onnela, J.-P., & Mucha, P. J. (2009). Communities in Networks. *Notices of the AMS*, 1082–1097. Retrieved from `http://arxiv.org/abs/0902.3788`

Power, J. D., Fair, D. A., Schlaggar, B. L., & Petersen, S. E. (2010). The development of human functional brain networks. *Neuron*, *67*(5), 735–748.

Price, R. B., Gates, K. M., Kraynak, T. E., Thase, M. E., & Siegle, G. J. (2017). Data-driven subgroups in depression derived from directed functional connectivity paths at rest. *Neuropsychopharmacology*, *42*(13), 2623.

R Core Team. (2017). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., & Parisi, D. (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, *101*(9), 2658–2663. doi: 10.1073/pnas.0400054101

Reichardt, J., & Bornholdt, S. (2006). Statistical mechanics of community detection. *Physical Review E*, *74*(1), 016110.

Rubinov, M., & Sporns, O. (2011). Weight-conserving characterization of complex functional brain networks. *NeuroImage*, *56*(4), 2068–2079. doi: 10.1016/j.neuroimage.2011.03.069

Shai, S., Stanley, N., Granell, C., Taylor, D., & Mucha, P. J. (2017). Case studies in network community detection. *arXiv preprint arXiv:1705.02305*.

Snijders, T. A., & Borgatti, S. P. (1999). Non-parametric standard errors and tests for network statistics. *Connections*, *22*(2), 161–170.

Steinley, D. (2004). Properties of the Hubert-Arabie adjusted Rand index. *Psychological methods*, *9*(3), 386–396. doi: 10.1037/1082-989X.9.3.386

Stevens, A. A., Tappon, S. C., Garg, A., & Fair, D. A. (2012). Functional brain network modularity captures inter-and intra-individual variation in working memory capacity. *PloS one*, *7*(1), e30468.

Thompson, B. (1994). The pivotal role of replication in psychological research: Empirically evaluating the replicability of sample results. *Journal of Personality*, *62*(2), 157–176.

Tonidandel, S., & Overall, J. E. (2004). Determining the number of clusters by sampling with replacement. *Psychological Methods*, *9*(2), 238.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, *58*(301), 236–244. doi: 10.1080/01621459.1963.10500845

Weir, W. H., Emmons, S., Gibson, R., Taylor, D., & Mucha, P. J. (2017). Post-processing partitions to identify domains of modularity optimization. *Algorithms*, *10*(3), 93.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of anthropological research*, *33*(4), 452–473.

Tables and Figures

***Table 1. Output values provided by* `perturbR`** *Note: N indicates number of nodes; reps indicates number of repetitions (default = 100); n_α indicates the number of values at which rewiring occurs in α increments (default of 0.01); VI = Variation of Information; ARI = Adjusted Rand Index.*

Name	Dimension	Contents
comm.assign	1 x N	Cluster assignments for each node.
VI	*reps* x n_α	VI values for each repetition across αs for original matrix
ARI	*reps* x n_α	ARI values for each repetition across αs for original matrix
VI.rando	*reps* x n_α	VI values for each repetition across αs for random matrix
ARI.rando	*reps* x n_α	ARI values for each repetition across αs for the random matrix
modularity	*reps* x n_α	Modularity values for each repetition across αs for original matrix
modularity.rando	*reps* x n_α	Modularity values for each repetition across αs for random matrix
percent	1 x n_α	Proportion of edges perturbed at each α
ari10mark	1	ARI value: value for bottom line in figure (see text)
vi10mark	1	VI value: value for top line in figure (see text)
ari20mark	1	ARI: value for top line in figure (see text)
vi20mark	1	VI: value for bottom line in figure (see text)
cutoff	1	Modularity value where 95% of random matrices fall below

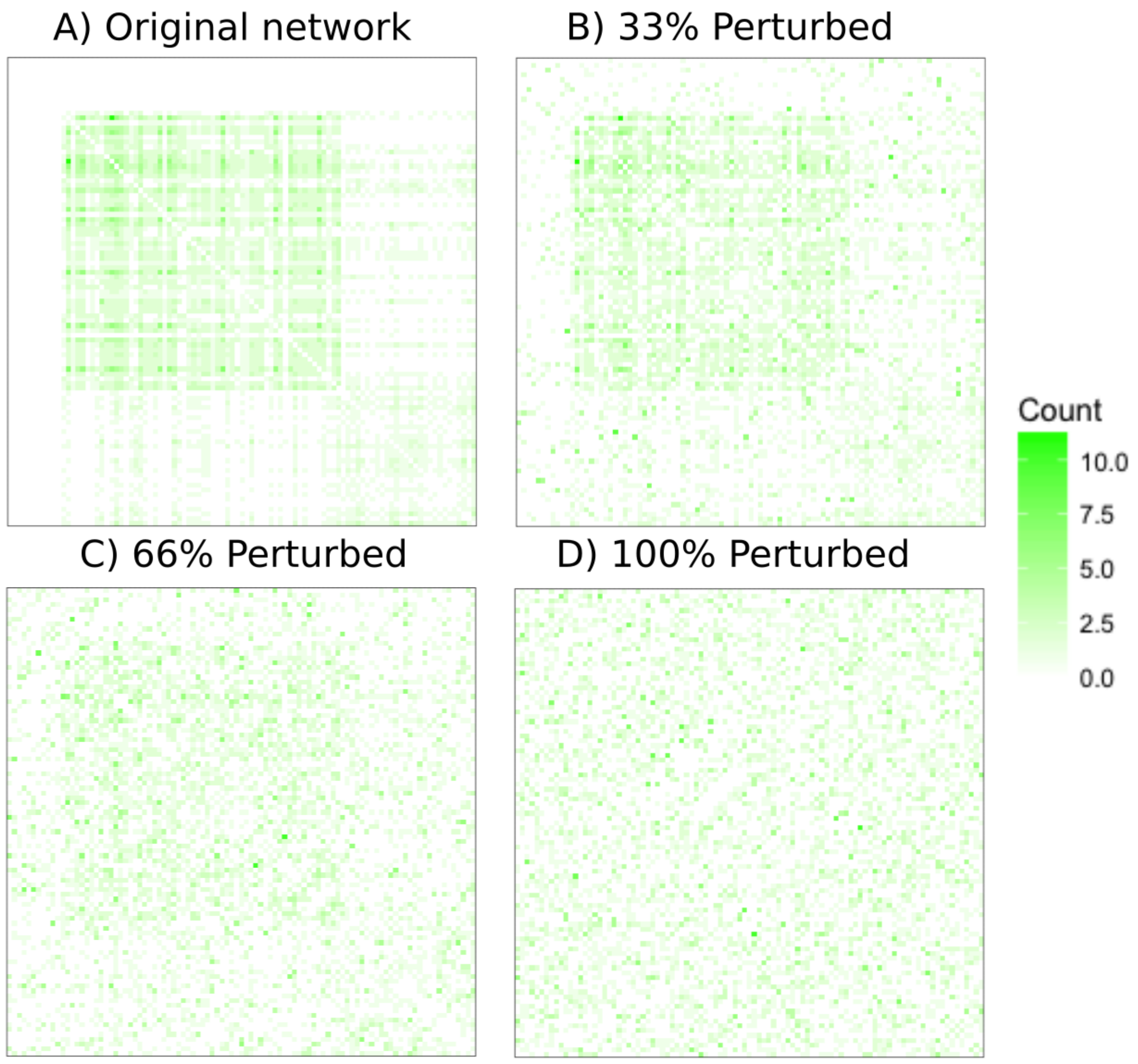

Figure 1. **Example of increasing degrees of perturbation for count matrices.**Panel A depicts an original matrix. A clear cluster can be seen from the dark square, which represents individuals who have high edge weights. The blank edges represent zeros. Panel B depicts what the matrix looks like when 33% when the edges of the original matrix are randomly perturbed. Panel C depicts the matrix with 66% perturbed; the clear cluster structure is beginning to disappear. Finally, in Panel D every edge in the matrix has been randomly perturbed. The matrix is completely random.

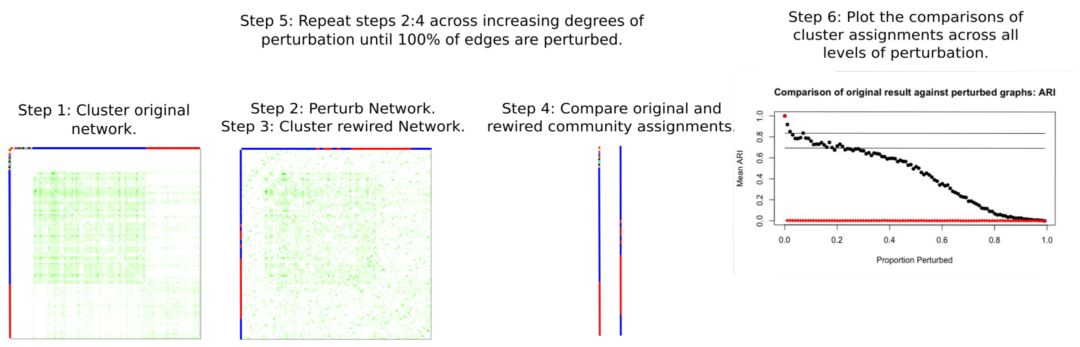
Step 5: Repeat steps 2:4 across increasing degrees of perturbation until 100% of edges are perturbed.
Step 6: Plot the comparisons of cluster assignments across all levels of perturbation.
Step 1: Cluster original network.
Step 2: Perturb Network.
Step 3: Cluster rewired Network.
Step 4: Compare original and rewired community assignments.
Comparison of original result against perturbed graphs: ARI
Mean ARI
Proportion Perturbed
0.0
0.2
0.4
0.6
0.8
1.0

Figure 2. **Steps for comparing community assignments between original and rewired matrices.** The color codes in steps 1 through 4 indicate cluster assignments.

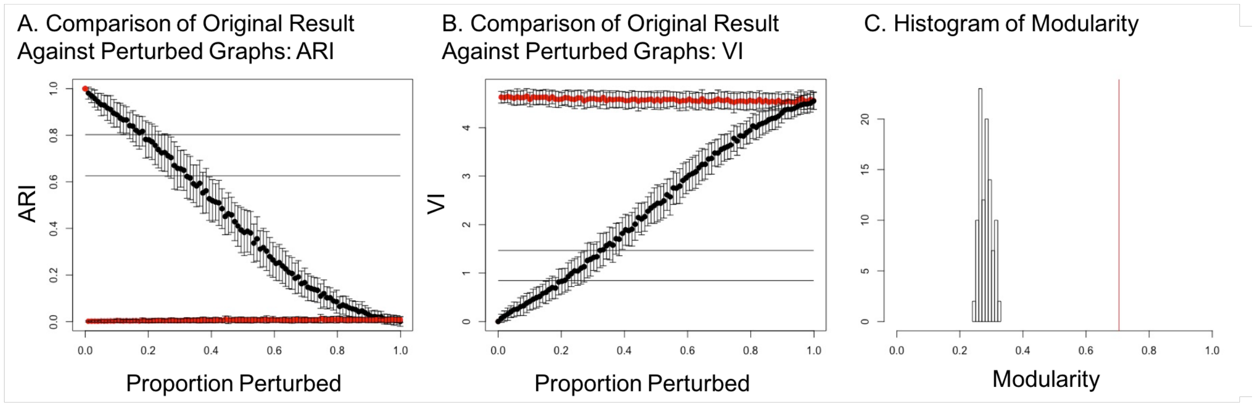

Figure 3. **Example: Results from simulated data.** In Panels A and B, Red triangles represent original random matrices for scaling; black circles represent mean differences between the perturbed and original matrix solutions; the horizontal lines indicate mean difference between solutions where 10% (upper line for VI, lower for ARI) and 20% of the nodes are randomly switched to different clusters. Error bars indicate plus and minus 2 standard deviations from the mean. Panel C depicts the histogram from modularity results obtained from a random matrix with similar strength distribution as the original matrix. Here, the red line indicates the modularity of the results from the original matrix.

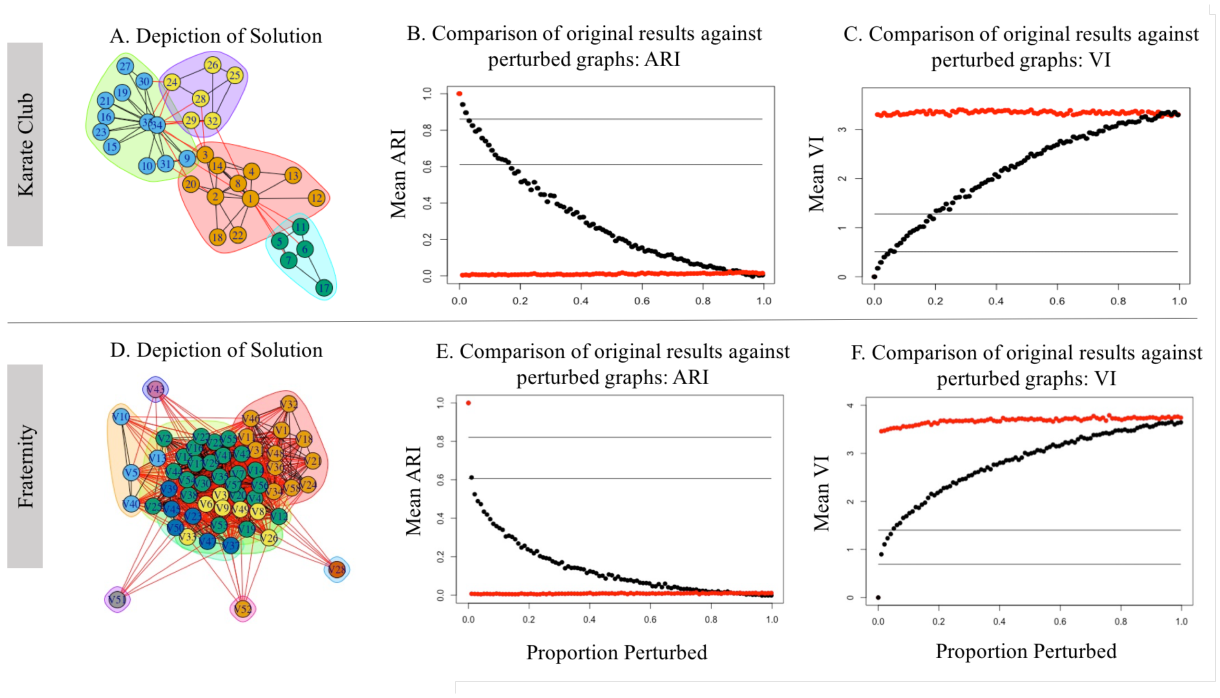

Figure 4. **Social Network Results**Panels A and D depict the cluster solutions results for the Karate Club and Fraternity social networks, respectively. Panels B, C, E, and F depict the average differences between the original solution and iteratively perturbed matrices. Red triangles represent original random matrices for scaling; black circles represent the original matrices; the horizontal lines indicate difference between solutions 10% and 20% of the nodes are randomly switched to different clusters. According to both the VI and ARI about 20% of the edges need to be perturbed before results differ as much as would be expected if 20% of the nodes were assigned different communities for the Karate Club network. The Fraternity network does not appear robust according to either the VI or ARI figures.

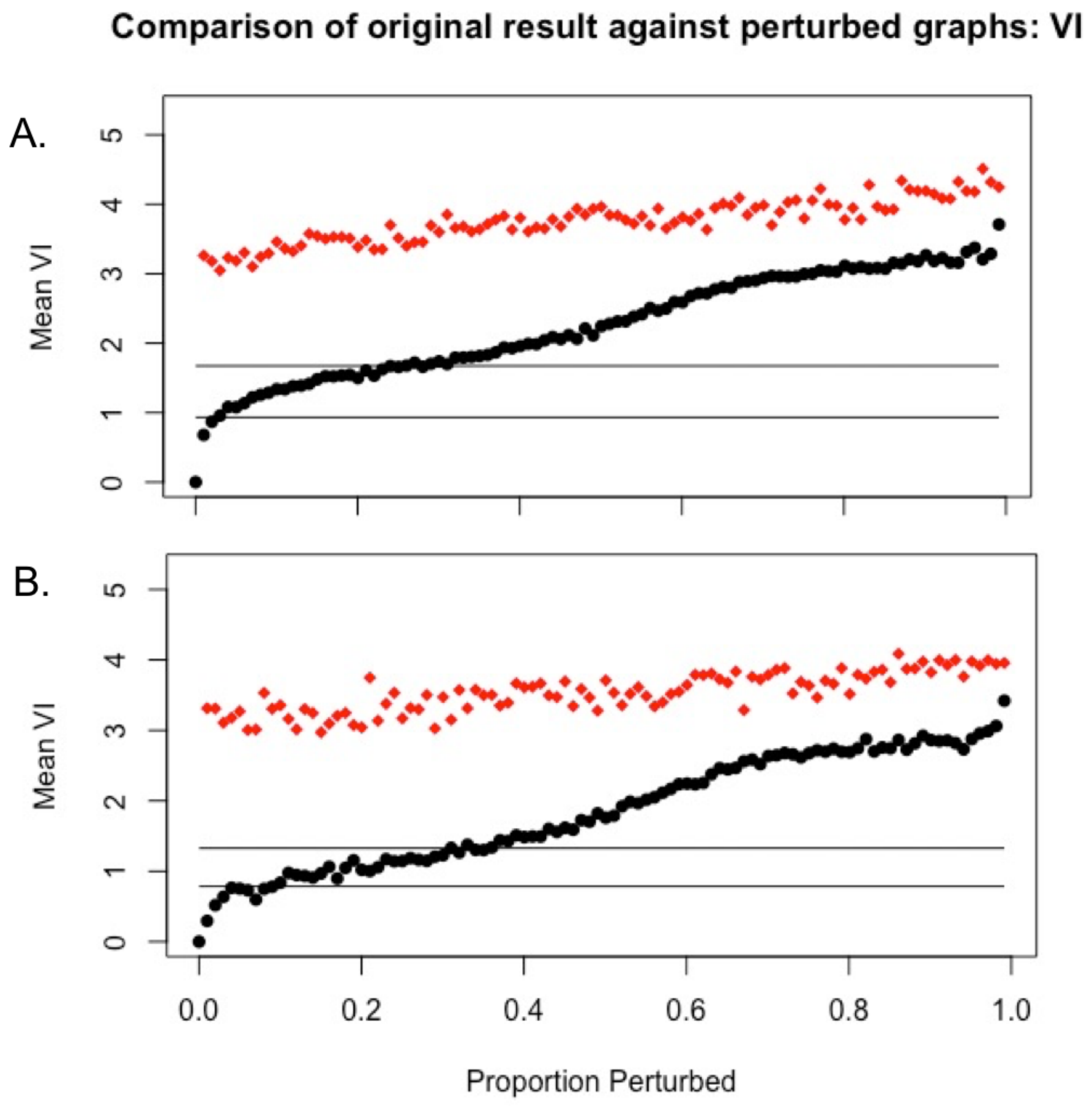

Figure 5. **DTI Structural Brain Data Example Results.** Average differences between the original solution and solutions from iteratively perturbed matrices presented here in terms of Variation of Information for two individuals separately in panels A and B. Red triangles represent original random matrices for scaling; black circles represent the original matrices; the horizontal lines indicate difference between solutions 10% (top line) and 20% (bottom) of the nodes are randomly switched to different clusters.