A General Monte Carlo Method for Sample Size Analysis in the Context of Network Models

Mihai A. Constantin[1], Noémi K. Schuurman[2], and Jeroen K. Vermunt[1]
[1]Tilburg University
[2]Utrecht University

This is a ***revised*** preprint of a manuscript that is in press at Psychological Methods.
November 30, 2022

We introduce a general method for sample size computations in the context of cross-sectional network models. The method takes the form of an automated Monte Carlo algorithm, designed to find an optimal sample size while iteratively concentrating the computations on the sample sizes that seem most relevant. The method requires three inputs: 1) a hypothesized network structure or desired characteristics of that structure, 2) an estimation *performance measure* and its corresponding target value (e.g., a sensitivity of 0.6), and 3) a statistic and its corresponding target value that determines how the target value for the performance measure be reached (e.g., reaching a sensitivity of 0.6 with a probability of 0.8). The method consists of a Monte Carlo simulation step for computing the performance measure and the statistic for several sample sizes selected from an initial candidate sample size range, a curve-fitting step for interpolating the statistic across the entire candidate range, and a stratified bootstrapping step to quantify the uncertainty around the recommendation provided. We evaluated the performance of the method for the Gaussian Graphical Model, but it can easily extend to other models. The method displayed good performance, providing sample size recommendations that were, on average, within three observations of a benchmark sample size, with the highest standard deviation of 25.87 observations. The method discussed is implemented in the form of an **R** package called **powerly**, available on GitHub and CRAN.

Keywords: network models, Gaussian Graphical Model, Monte Carlo simulation, monotone splines, stratified bootstrapping, sample size analysis

The network approach to psychology is a popular framework for conceptualizing a wide range of psychological phenomena. Within this framework, one can investigate the extent to which a set of variables, called *nodes* in network terminology, are connected in a network structure via *edges*, which reflect estimated associations between pairs of variables. It was initially proposed, and further developed, as a framework for explaining the onset and maintenance of mental disorders (Borsboom & Cramer, 2013; Borsboom et al., 2011; Cramer et al., 2010, 2016), where the nodes are symptoms, and the edges reflect the causal influences among the symptoms, forming a complex system that leads to the emergence of a mental disorder (Borsboom, 2017). The network approach is, however, rapidly spreading to other fields of psychology, such as personality research, or health sciences (e.g., Costantini et al., 2015; Kossakowski et al., 2016). This surge in popularity and its promise for enabling new insights have drawn significant attention to the method, paving the way for integrating graphical models into the empirical psychological literature.

Unlike social networks, which are constructed (i.e., an edge represents an observable quantity), psychological networks need to be estimated from data. Hence, in order to numerically encode the relationship between two nodes, probabilistic models are used to estimate this relationship.

Mihai A. Constantin, Noémi K. Schuurman, Jeroen K. Vermunt.

The code for the **R** package **powerly** implementing the method discussed in this paper is publicly available on Github at github.com/mihaiconstantin/powerly. The simulation scripts and other supplementary materials are publicly available on OSF at osf.io/qksd7. Ideas in this paper were presented at several conferences and on the powerly.dev website.

Correspondence concerning this article should be addressed to Mihai A. Constantin, Department of Methodology and Statistics, Tilburg University, PO Box 90153 5000 LE, Tilburg. E-mail: contact@mihaiconstantin.com.

The result of this estimation is called a network structure, which is encoded as a matrix of *edge weights* (i.e., the estimated associations), and can be visualized to grasp the multivariate dependencies in the data. The network structure can be further analyzed by calculating descriptive measures (e.g., density, small worldliness, etc.; O'Malley & Marsden, 2008; Sharma & Srivastava, 2015) or by looking at node centrality indices (e.g., strength, betweenness, closeness, etc.; Bonacich, 2007; Jones et al., 2019; Opsahl et al., 2010), a process known as network inference. Regardless of what further analyses are employed, the quality of the estimated network structure directly impacts the extent to which any kind of inference performed is valid for interpretation.

Accurately estimating network structures from psychological data (i.e., psychological networks) is a subject of debate and catalyst of methodological advancements (e.g., Williams & Rast, 2020). One crucial factor to consider in order to accurately estimate network parameters is the sample size (Epskamp, Borsboom, & Fried, 2018; Fried et al., 2018). The problem of knowing which sample sizes to select is particularly relevant to psychological research where data collection is often limited to small numbers of participants and convenience sampling schemes (e.g., Shen et al., 2011), all in the midst of a replication crisis (Open Science Collaboration, 2015). For psychological networks, the necessity for making informed decisions about the sample size is further emphasized by the fact that the number of parameters that needs to be estimated increases rapidly with the number of variables included in the network. The quest for accuracy and stability has advanced the psychological network modeling field by generating new methodologies intended for safeguarding against erroneous conclusions (Epskamp, Borsboom, & Fried, 2018) and launched debates and initiatives about how well these models replicate in different samples (Borsboom et al., 2017; Forbes et al., 2017). Sample size lies at the foundation of these efforts, yet, despite previous work (e.g., Epskamp, Borsboom, & Fried, 2018), the practical question of what sample size is needed to reliably retrieve a hypothesized network structure remains largely unanswered.

The goal of this paper is to provide a general method for obtaining sample size recommendations to accurately estimate cross-sectional network models. Obtaining sample size recommendations for network models is far from trivial. When conducting a power analysis (Aberson, 2019; Cohen, 1988), one needs to specify an effect size and fix a significance level for testing. In the context of a power analysis, the effect size is often a single numerical quantity. However, specifying an effect size for network models is far more complex. To determine the sample size required, one first needs to know what is important to recover for the network, and this will likely differ from one research project to another. Some may be interested in network centrality, while others may be interested in correctly recovering all true edges. Sample size recommendations will and should differ depending on these different demands. Furthermore, once the demand is identified, one needs to decide how to find a suitable sample size to satisfy this demand; and what sample size is enough depends on what aspects of the network structure impact this demand. There may be many different aspects of the network structure relevant to a particular demand, such as the network type, the density of the network, or the value of all or some of the edge weights. In other words, specifying the effect size for such a "power analysis" is not straightforward and may very well require specifying details about the entire network structure. As a result, providing general sample size recommendations is not trivial and will most likely require elaborate simulation studies (e.g., Arend & Schäfer, 2019; Hancock & French, 2013), which every researcher must perform to suit the particular demands of their research projects.

Moreover, another issue that arises when conducting power analysis for network models is the computation time. Regular Monte Carlo (MC) simulations have already been suggested to tackle sample size calculations for network models (Epskamp, Borsboom, & Fried, 2018). However, while MC simulations are a versatile tool for solving intractable problems, common approaches often involve simulating many datasets for a broad range of sample sizes and checking whether the power requirements for each sample size are met (e.g., Muthén & Muthén, 2002; Schoemann et al., 2014; Zhang, 2014). Since each MC simulation requires generating data and estimating a model, such brute-force approaches run the risk of using computational resources for irrelevant sample sizes. We present a more efficient approach designed to concentrate the MC simulations as much as possible on sample sizes that seem most relevant. We use an iterative strategy to narrow down the relevant sample sizes and obtain a more precise estimate of the sample size once we are close to the required sample size.

A final challenge in providing sample size recommendations for network models is put forward by the rapid methodological advancements in the field. The field of psychological networks is relatively young and what is considered best practice is an ongoing debate (e.g., Epskamp et al., 2017; Williams & Rast, 2020). To this end, we design our method with a high level of generality in mind. Our method remains agnostic to the estimation algorithm, and it performs well with popular approaches for estimating psychological networks (e.g., regularized estimation; Epskamp & Fried, 2018) while also allowing for other approaches (e.g., non-regularized and Bayesian estimation). Similarly, we enable researchers to investigate various estimation performance measures of the network structure while also enabling them to provide their own definitions for such measures. Our method requires only minimal input from researchers who wish to apply it but remains open for extension.

The method is based on three steps once the input is provided. The method starts with a Monte Carlo simulation step for computing the performance measure and the statistic for several sample sizes selected from a candidate sample size range. It continues with a curve-fitting step for interpolating the statistic across the entire candidate range. The final step employs a stratified bootstrapping scheme to quantify the uncertainty around the recommendation provided. The method iteratively repeats these three steps until the candidate range of sample sizes shrinks to a tolerable threshold. Together with existing methodologies (Epskamp, Borsboom, & Fried, 2018), our approach aims to close the gap by providing empirical researchers with the tools necessary to plan data collection when network modeling is the strategy of choice.

The remainder of the paper is organized as follows. We start with background information on psychological network models and narrow our focus on the more specific Gaussian Graphical Model (GGM), which we use as an example model to demonstrate our method. In this context, we discuss multiple characteristics of the estimation of a network structure that researchers may want to optimize for. Then, we discuss various network characteristics that are likely to impact the sample size required to observe a specific value on these estimation performance measures of interest. In the subsequent section, we describe the proposed method and a strategy to validate the sample size provided as a recommendation. The next section is dedicated to a simulation study to evaluate the performance of the proposed sample size computation method. We conclude by discussing extensions of this method to other network models and improvement points.

Background on Network Models

Psychological network models are used to estimate network structures encoding pairwise relationships between a group of variables. A core concept in the estimation of psychological network structures is the property of *conditional independence* (Dawid, 1979). An edge that connects two nodes indicates that the two variables are dependent, while conditioning on all other variables in the network. Conversely, when an edge is absent between two nodes, this implies that the two variables are conditionally independent, given all other variables in the network. We say that an edge $\{i, j\}$ connecting nodes i and j is present in the edge set E of a graph G if i and j are not conditionally independent given all other nodes in the network, usually denoted as $\{i, j\} \notin E(G) \iff Y_i \perp\!\!\!\perp Y_j \mid \boldsymbol{y}^{-(i,j)}$, where $\notin$ indicates that edge $\{i, j\}$ is not present in the edge set $E(G)$, $\boldsymbol{y}$ represents a vector of random variables, $\perp\!\!\!\perp$ indicates conditional independence (Dawid, 1979), and $\boldsymbol{y}^{-(i,j)}$ indicates all variables in $\boldsymbol{y}$, excluding i and j.

There are many techniques for estimating psychological network models, and all fall under the category of graphical modeling (Lauritzen, 1996). Most commonly, researchers focus on a specific class of models known as Pairwise Markov Random Fields (PMRF; Epskamp & Fried, 2018). Figure 1 displays a PMRF, where the absence of an edge between nodes F and B indicates that they are conditionally independent given all other nodes in the network. The choice of PMRF depends on the type of data and distributional assumptions. For instance, when the data is continuous and multivariate normally distributed, the common choice is the GGM (Epskamp, Waldorp, et al., 2018; Uhler, 2017). In contrast, when the data is binary, the model of choice is the Ising model (Cipra, 1987; van Borkulo et al., 2014). In essence, these models aim to estimate a matrix of model parameters $\boldsymbol{\Theta}$ that contains the estimated conditional associations. These estimated conditional associations encode the weights—often reflected graphically by the width of the edges—associated with each edge in the network. As a result, the $\boldsymbol{\Theta}$ matrix provides two pieces of information. On the one hand, about the network structure (i.e., the presence or absence of an edge) and, on the other hand, the strength and the direction of the conditional association between any pair of variables in the network (e.g., the value and sign of the coefficients; see Figure 1).

The Gaussian Graphical Model

When the variables (e.g., items of a questionnaire) in the vector $\boldsymbol{y}$ are independent random samples from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with the mean vector $\boldsymbol{\mu}$ of length P and the $P \times P$ covariance matrix $\boldsymbol{\Sigma}$, the specific PMRF type of interest is the GGM. The GGM is a network of partial correlations usually constructed from the precision matrix. The precision matrix $\boldsymbol{\Omega}$ is obtained from the inverse of the covariance matrix:

$$\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1} . \tag{1}$$

The ω_{ij} entries in the precision matrix $\boldsymbol{\Omega}$ can readily be used as edge weights and visualized in a network structure. However, it is often preferred to compute and use partial correlations as edge weights instead of the values in the precision matrix $\boldsymbol{\Omega}$ (Epskamp & Fried, 2018). The $\boldsymbol{\Omega}$ matrix encodes all the information required to obtain the partial correlation coefficient between any two variables θ_{ij} as:

$$\mathrm{Cor}\Big(Y_i, Y_j \mid \boldsymbol{y}^{-(i,j)}\Big) = -\frac{\omega_{ij}}{\sqrt{\omega_{ii}}\sqrt{\omega_{jj}}} , \tag{2}$$

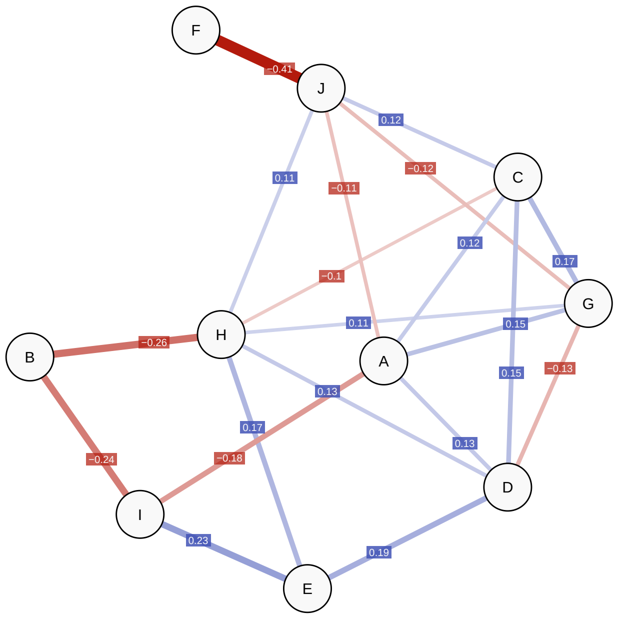
where, as discussed elsewhere (see Lauritzen, 1996), for a GGM, an entry $\theta_{ij} = 0$ if and only if $Y_i \perp\!\!\!\perp Y_j \mid \boldsymbol{y}^{-(i,j)}$.

A popular approach is to estimate the precision matrix $\boldsymbol{\Omega}$ with regularization paired with the Extended

Bayesian Information Criterion (EBIC) for model selection (see Foygel & Drton, 2010, for a detailed discussion). The type of regularization used is the Least Absolute Shrinkage and Selection Operator (LASSO; Hastie et al., 2009), implemented as a fast algorithm optimized for precision matrix estimation termed graphical LASSO (*glasso*; Friedman et al., 2008). For a more detailed overview of this methodology termed *EBICglasso* and how it is applied to psychological networks, we refer the readers to work by Epskamp and Fried (2018). Given the popularity of this approach in the literature, we choose the *EBICglasso* estimation procedure to illustrate our sample size method. However, our method easily generalizes to other estimation procedures (e.g., non-regularized solutions for selecting edges based on significance tests, or Bayesian estimation), a point to which we return in the discussion section.

Figure 1

Example of a Gaussian Graphical Model represented as an undirected network with 10 *nodes and* 20 *edges.*



Note. The model was generated using the function **generate_model** from the **R** package **powerly** with 10 nodes, a network density of 0.45, and 50% positive edges. The plot was created using the **R** package **qgraph** (Epskamp et al., 2012).

Sample Size Recommendations for Network Models

Sample size studies typically revolve around attaining a certain power level. Using the setup of a typical power analysis is a helpful analogy for explaining what elements are relevant to provide adequate sample size recommendations to reach some desirable properties of the estimated networks. In a typical power analysis, the goal is to establish the sample size needed to find a significant effect. The power of a test is defined as the probability of detecting an effect size when there truly is an effect of a specific size in the population (Cohen, 1988). In other words, the power is the probability of rejecting the null hypothesis (i.e., no effect size) when the true effect size is some value other than zero. A typical power analysis hence depends on three elements. First, it depends on the effect size the researcher wants to detect, that is, the assumed true value of some aspect of the population (e.g., a correlation coefficient equal to a certain hypothesized value). Next to that, it depends on the selected significance level used for deciding whether or not to reject the null hypothesis. One is usually interested in detecting a true effect size when conducting a significance test with a certain Type I error rate. Finally, it depends on the desired value for the power. Assuming a true effect size, the value specified for power quantifies the probability of attaining a significant effect size. That is, the value for the power represents in what proportion of studies we want to be able to reject the null hypothesis of no effect.

Sample size recommendations for network models will typically be much more complicated than a basic power analysis for a single hypothesis test. First, compared to a basic power analysis where we can quantify the effect size as a scalar, the effect size for networks is regarded as a "high-dimensional interplay of the network structure [...] and the strength of the edges" (Epskamp, Borsboom, & Fried, 2018, p. 196). That is, the network consists of a matrix of edge weights (i.e., partial correlations for a GGM), and the sample size recommendation typically needs to consider the network as a whole to evaluate the power. This implies that instead of specifying a scalar, the researcher needs to specify an entire matrix of effect sizes before continuing their sample size analysis. Specifying a matrix of partial correlation coefficients is difficult, given that a partial correlation coefficient implies controlling for all other variables in the network (Schober et al., 2018). The task is further complicated by the fact that when specifying the edge weights, we may also need to consider other characteristics of the network structure (e.g., how nodes are expected to cluster or what nodes are expected to be central).

Second, the analog of "detecting a significant effect" for network models most likely implies reaching a certain target value for some performance measure of interest (e.g., obtaining a sensitivity of 0.6) for the network as a whole, rather than for specific edge weights in the network. As we discuss in the following subsection, the selected performance measure and its target value will typically depend on the research question of interest. Finally, we require a working definition of power in the context of network models. Earlier, we indicated that for a typical power analysis, this constitutes selecting the probability with which we want to detect a significant effect size. For network models, we are interested in defining a statistic that quantifies how well we want to be able to reach the target value on the performance measure. This statistic may take the form of typical power, that is, reaching a target value on the performance measure with a certain probability (e.g., in 80% of the cases). However, other researchers may find other definitions more useful. Defining how well the performance measure should be reached depends, again, on the particular research question.

Sample Size and Reliable Parameter Estimation

The extent to which we reliably estimate the precision matrix $\mathbf{\Omega}$ is essential for further analyses of the network structure. Further analyses usually involve computing descriptive graph indices such as centrality indices, clustering coefficients, or connectivity measures. Since these measures are computed on either the weights matrix $\mathbf{\Theta}$ or the adjacency matrix[1], any further information extracted from the network structure is contingent on how accurate the precision matrix $\mathbf{\Omega}$ was estimated in the first place. Therefore, the sample size must be "large enough" to guarantee that the precision matrix is reliably estimated, relative to a specific performance measure of interest, which quantifies the quality of the estimation. That is, what sample size is "large enough", depends to a large extent on the research question and goals of the researcher. Next to this, it depends on the characteristics of the underlying network structure that is estimated: the number of variables in the network and the network architecture.

Performance Measures

Different qualities of the estimation are important to different researchers. We may refer to these qualities as abilities to find true effects according to a certain target (e.g., obtaining a sensitivity of 0.6), and we operationalize them using estimation performance measures. Therefore, what constitutes a "large enough" sample size depends, on the one hand, on what performance measure the researcher is interested in, that is, what aspects of the network estimation are deemed important for the research question.

When the purpose lies with retrieving the network structure, most studies (e.g., van Borkulo et al., 2014; Williams & Rast, 2020) rely on performance measures like *sensitivity*, *specificity*, variants of *F scores* (Goutte & Gaussier, 2005), or the *Matthews correlation coefficient* (MCC; Matthews, 1975). These performance measures consider different aspects of binary classification (i.e., the number of true positives [TP] and negatives [TN], as well as the number of false positives [FP] and negatives [FN]) to quantify the extent to which the edges in a network are correctly estimated to be present or not. For example, sensitivity is the true positive rate, that is the proportion of edges in the true network structure that were correctly estimated to be non-zero:

$$\text{SEN} = \frac{\text{TP}}{\text{TP} + \text{FN}} \,. \tag{3}$$

Specificity refers to the true negative rate and provides information about the proportion of edges in the true network structure that were correctly estimated to be zero (i.e., absent):

$$\text{SPE} = \frac{\text{TN}}{\text{TN} + \text{FP}} \,. \tag{4}$$

Finally, the MCC, equivalent to the ϕ coefficient for 2×2 confusion matrices, takes into consideration all aspects of binary classification and is defined as:

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \,. \tag{5}$$

The MCC ranges between -1 and 1 (Powers, 2020), where a value of 1 indicates perfect correspondence between the edges in the true network structure and the edges in the estimated network structure.

In addition to performance measures emphasizing the retrieval of the network structure, the *edge-weights correlation coefficient* (ECC) can also be used to quantify the similarity between the true and the estimated model (Epskamp & Fried, 2018). The ECC is computed as the Pearson correlation between the upper triangles of the true and estimated edge weights matrices.

Whereas in this paper we focus on *sensitivity*, *MCC*, and *ECC*, our sample size computation approach is also applicable to other measures. For example, researchers may instead be more interested in other correlation coefficients (Chicco & Jurman, 2020; Schober et al., 2018), or measures of entropy and divergence (Rényi, 1961; van Erven & Harremos, 2014). In practice, the research question one aims to answer should largely guide the choice of the performance measure. Given that this choice will likely differ from study to study, we focus on developing a general method that extends easily to different performance measures.

Network Characteristics

The sample size also depends on various characteristics of the network. For instance, a larger number of variables in the network will likely require a larger sample size. However, there are other network characteristics for which the impact on the sample size required is less obvious, such as the strength and sign of the edge weights, the network architecture (e.g., small-world, random, scale-free; Barabási, 2009; Barabási & Albert, 1999; Watts & Strogatz,

[1]The adjacency matrix, denoted as $\boldsymbol{A}$, is an unweighted version of the $\mathbf{\Theta}$ matrix, where each entry $\theta_{ij} > 0$ is set to 1. It is usually considered the backbone of the network structure.

1998), network connectivity (e.g., how well the network is connected; Sharma & Srivastava, 2015), and the clustering of nodes (e.g., how nodes group together in the network; Opsahl & Panzarasa, 2009). Furthermore, the required sample size depends on each combination of network characteristics and performance measures, hence making it difficult to provide an analytical solution. This gap is precisely the problem our method aims to solve. We provide a solution to this problem in the form of a general MC method to conduct sample size analysis depending on the needs of researchers.

Methods

We propose a general MC method for providing sample size recommendations for cross-sectional network models. The method takes the form of an algorithm that iteratively searches for the optimal sample size, given a model specification (i.e., a combination of network characteristics), a performance measure of interest and its corresponding value (e.g., a sensitivity of 0.6), and a definition of power and its corresponding value (e.g., reaching a sensitivity of 0.6 in 80% of the cases). The search for the optimal sample size is conducted in three steps, which is followed by a validation of the sample size provided as a recommendation. We provide an open-source implementation for the method discussed here in the form of an `R` package called `powerly` (see Figure 2), available on GitHub[2] and CRAN[3].

In the first step, we use MC simulations to get a rough understanding of how the performance measure changes as a function of sample size. We perform several MC replications for several sample sizes selected from a candidate sample size range. For each MC replication, we simulate data from a hypothesized network structure (i.e., the true model), estimate the model parameters, and compute the performance measure. At the end of this step, for each sample size, we compute a statistic of interest (e.g., the probability of reaching the target value on the performance measure) on the performance measure values and have preliminary information on the value of the statistic observed at specific points within the initial candidate sample size range (see Figure 3).

Whereas in the first step we strategically select several sample sizes within a candidate range for which we compute the corresponding statistic of interest, in the second step, we interpolate the statistic across all sample sizes comprised in that range. To do so, we fit a spline to the values of the statistic determined in the previous step. The estimated spline coefficients enable us to obtain a smooth function by interpolating the value of the statistic for all sample sizes in the initial range (see Figure 3b).

However, since the statistic we compute is based on a restricted number[4] of MC replications, the interpolated spline (i.e., the power function) is subject to MC errors. In the third step, we account for these errors and provide a measure of uncertainty around the interpolated spline. To achieve this, we use bootstrapping to quantify the uncertainty around the statistic of interest (see Figure 4). The uncertainty around the spline allows us to narrow down the candidate sample size range. We repeat the three steps with the new range and efficiently concentrate the MC simulations around sample sizes that seem most relevant. The algorithm stops when the candidate range shrinks to a tolerable threshold. Ultimately, we provide our recommendation in terms of the median of the shrunken sample sizes range, as well as the distribution of statistic values associated with the recommended sample size (see Figure 4).

In the following subsections, we explain the procedure in more detail. We first discuss the required user input for the method to work, analogous to the input one usually provides in the context of a typical power analysis. Then, we provide a detailed description of each step of the algorithm. We provide information on the convergence of the algorithm, and we end by discussing how to validate the resulting sample size recommendation provided.

Required Input

The following input needs to be provided by the researcher to kick-start the algorithm. First, we need an effect size, specified as parameter values for the true network that we use to generate data from. The true parameter values are contained in the $P \times P$ true model matrix $\mathbf{\Theta}$, where p is the number of variables in the network. The θ_{ij} entries in $\mathbf{\Theta}$ represent the true edge weights that connect variables i and j, respectively. In the case of the GGM, the θ_{ij} entries encode the partial correlations among the variables in the network. The researcher may either specify $\mathbf{\Theta}$ directly, however, since this is non-trivial, it may also be generated based on specific characteristics of the network, such as the number of nodes and network density (see Yin & Li, 2011, for an example algorithm on how to generate true parameters for the GGM).

Second, we require the desired performance measure expressed in the form of a function $f\left(\mathbf{\Theta}, \widehat{\mathbf{\Theta}}\right)$ that takes arguments an ordered pair of model matrices, where $\mathbf{\Theta}$ is the matrix of true model parameters and $\widehat{\mathbf{\Theta}}$ the matrix of estimated model parameters with $P \times P$ dimensions. The performance measure is a scalar value that can be regarded as a measure of discrepancy between two sets of model parameters. The function $f\left(\mathbf{\Theta}, \widehat{\mathbf{\Theta}}\right)$ computes the performance measure by comparing the true model parameters in $\mathbf{\Theta}$ with the estimated model parameters in $\widehat{\mathbf{\Theta}}$ recovered from the data. Together with the performance measure, the researcher

[2]GitHub URL: github.com/mihaiconstantin/powerly.

[3]CRAN URL: cran.r-project.org/package=powerly.

[4]The number of MC simulations may be restricted based on model complexity to gain computation speed while allowing for more error when computing the statistic of interest.

Figure 2

Code example for running the method and visualizing the output using the **R** *package* **powerly***, for a randomly generated GGM with* 10 *nodes and a density of* 0.4.

Analysis

```r
# Load library.
library(powerly)

# Run the method.
results <- powerly(
    range_lower = 150,
    range_upper = 800,
    samples = 30,
    replications = 50,
    measure = "sen",
    measure_value = 0.6,
    statistic = "power",
    statistic_value = 0.8,
    model = "ggm",
    nodes = 10,
    density = 0.4
)
```

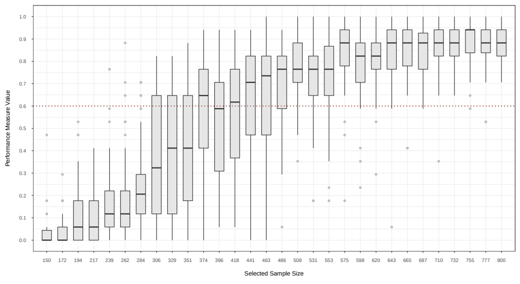
Output

```r
# Show summary.
summary(results)
```

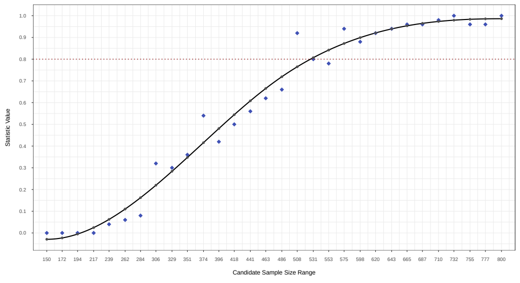
```
Method run (7.17 sec):
- converged: yes
- iterations: 1
- recommendation: 528
```

Plots

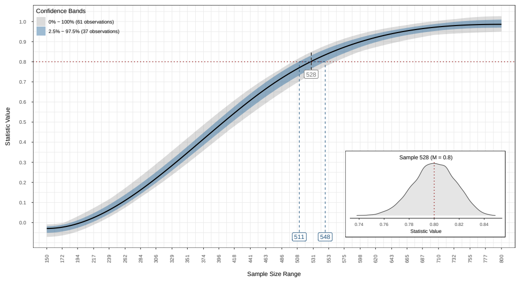
```r
plot(results, step = 1)
```



```r
plot(results, step = 2)
```



```r
plot(results, step = 3)
```



Note. More information about the sample size computation can be extracted from the **`results`** object. The documentation for the **`powerly`** function can be consulted by running **`?powerly`**. The output can be reproduced by running **`set.seed(20031993)`** before the **`powerly`** function call, using package version **`1.8.0`**.

must also specify a target value δ, indicating the desired value to be reached for the performance measure. For example, the researcher may be interested in sensitivity as a performance measure, with a target value of $\delta = 0.6$.

The third input requires the researcher to specify a statistic—and its corresponding target value—that captures *how* the target value δ for the performance measure should be reached. For instance, the researcher may be interested in a sample size recommendation that can be used to reach the target sensitivity $\delta = 0.6$ in 80% of the cases. This statistic can generally be expressed as a function $g(\boldsymbol{\xi})$ that takes input a vector $\boldsymbol{\xi}$ of size R with index $r(r = 1, \dots, R)$, which consists of replicated performance measures associated with a particular sample size. Each element of $\boldsymbol{\xi}$ is, therefore, a replicated performance measure computed using the function $f\left(\boldsymbol{\Theta}, \widehat{\boldsymbol{\Theta}}\right)$ evaluated for a particular sample size.

While the statistic can take any form (e.g., mean or median), in this paper, we focus on defining the function $g(\boldsymbol{\xi})$ in terms of the probability of observing a desired target value for the performance measure. In this case, the statistic definition is analogous to typical power computation:

$$g(\boldsymbol{\xi}) = \frac{1}{R} \sum_{r=1}^{R} [\xi_r \geq \delta], \tag{6}$$

where the notation $[\cdot]$ represents the Iverson brackets (Knuth, 1992), with $[\xi_r \geq \delta]$ defined to be 1 if the statement $\xi_r \geq \delta$ is true and 0 otherwise[5]. The recommended sample size is chosen to satisfy $g(\boldsymbol{\xi}) \geq \tau$, where τ is an arbitrary threshold indicating a certain probability of interest, that is, the desired value for the statistic (e.g., 0.8 in the example above).

Optionally, the researcher may also specify a candidate sample size range $\mathbb{N}_s$ to restrict the algorithm search for an optimal sample size within that range. When this optional range $\mathbb{N}_s$ is not provided, it is constructed considering the complexity associated with the estimation of the model matrix $\widehat{\boldsymbol{\Theta}}$. This is done by setting the lower bound of the range $\mathbb{N}_s$ to a sufficiently small sample size such that for that sample size, the performance measure is smaller than the target value. Conversely, the upper bound of $\mathbb{N}_s$ is set to a sufficiently large sample size for the performance measure to yield a value greater than the target.

Step 1: Performing Monte Carlo Simulations

The goal of performing MC simulations is to get a rough understanding of the behavior of the performance measure as a function of sample size. First, we compute the performance measure (e.g., sensitivity) for several equidistant sample sizes within the initial sample size range $\mathbb{N}_s$. Then, for each of those sample sizes, we compute the statistic of interest using the function $g(\boldsymbol{\xi})$ in Equation 6 based on the values of the replicated performance measure.

More specifically, let T be the total number of selected sample sizes, indexed by $t(t = 1, \dots, T)$. We start by selecting T equidistant sample sizes from the initial candidate sample size range as $S = \{s_1, \dots, s_T\} \subseteq \mathbb{N}_s$. For each selected sample size $s_t \in S$ we perform a total of R MC simulation replications where r indicates the current replication.

Each MC replication consists of the following procedure:

1. Simulate data for each selected sample size s_t by generating a $s_t \times p$ data matrix $\boldsymbol{Y}^{(rt)}$ using the true model matrix $\boldsymbol{\Theta}$, where the rows represent observations and the columns variables in the network.

2. Estimate the matrix of model parameters $\widehat{\boldsymbol{\Theta}}^{(rt)}$ for each simulated data set $\boldsymbol{Y}^{(rt)}$. For the GGM, we use the popular *EBICglasso* methodology (see section *The Gaussian Graphical Model*).

3. Evaluate the function $f\left(\boldsymbol{\Theta}, \widehat{\boldsymbol{\Theta}}^{(rt)}\right)$ to compute the discrepancy between the true and the estimated model parameters and obtain a value for performance measure.

Repeating the procedure above R times for all sample sizes in S we obtain a $R \times T$ matrix $\boldsymbol{\Xi}$, where each ξ_{rt} entry represents a performance measure computed for the t-th sample size, during the r-th MC replication. Based on these performance measure values, we compute the statistic of interest across the different replications for each selected sample size. Specifically, we do this by applying the function $g(\boldsymbol{\xi}_t)$ in Equation (6) to all ξ_{rt} entries in $\boldsymbol{\Xi}$ associated with a particular sample size s_t, which we denote as a vector $\boldsymbol{\xi}_t$. The result of this operation is stored in a $T \times 1$ column vector $\boldsymbol{g}^{\top} = \left[g(\boldsymbol{\xi}_1), \dots, g(\boldsymbol{\xi}_T)\right]$, where each g_t element refers to a value of the statistic obtained for a sample size $s_t \in S$ based on R MC replications. Figure 3a displays the columns of matrix $\boldsymbol{\Xi}$ as box plots, providing a representation of how much the performance measure varies for each sample size based on the number of replications performed. Figure 3b shows how the statistic behaves as a function of sample size, which we model using a spline methodology in the next step.

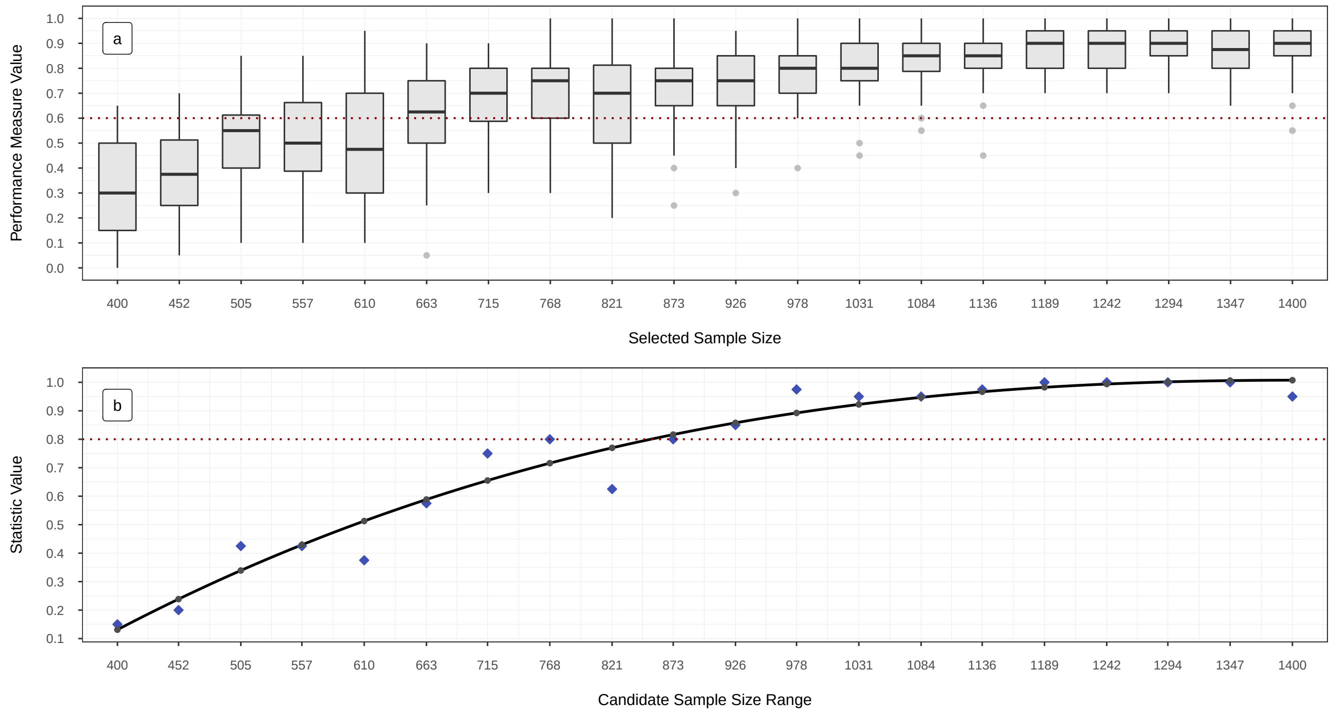
Step 2: Fitting a Smooth Curve

The goal of this step is to obtain a smooth function that enables us to have an estimate of the statistic of interest for all sample sizes in the range $\mathbb{N}_s$. To achieve this, we fit a spline to the results of the previous MC step and use the resulting spline coefficients to interpolate the statistic across the entire range $\mathbb{N}_s$. This step hinges on the assumption that the statistic behaves consistently as a function of sample size by

[5]Simialrly, the defition of the statistic in Equation 6 can be changed to $[\xi_r \leq \delta]$ for performance measures that are assumed to follow a monotone non-increasing pattern.

Figure 3

Visual representation of steps one and two. Panel (a) shows the performance measure values obtained from the Monte Carlo replications. Panel (b) shows the values for the statistic computed on the performance measure values and the monotone spline used to interpolate the statistic across the entire candidate sample size range $\mathbb{N}_s$.



Note. The dotted horizontal lines indicate the desired target for the performance measure (i.e., 0.6) and the statistic (i.e., 0.8). The box plots in panel (*a*) represent sensitivity values computed for the model in Figure 1 based on $R = 30$ replications and $T = 20$ sample sizes. The statistic values in panel (*b*) are computed according to Equation (6).

displaying a monotone trend. For example, a monotone non-decreasing spline (Ramsay, 1988) implies the assumption that the statistic increases as a function of sample size. To obtain the spline coefficients for this scenario, we use the **R** package **splines2** (Wang & Yan, 2021) package to create a basis[6] of cubic *I-Splines* (de Leeuw, 2017; Ramsay, 1988) from the selected sample sizes in S. This gives us the design matrix $\boldsymbol{H}$ containing the basis functions.

Assuming a monotone non-decreasing trend for the statistic, we obtain our spline (see Figure 3b) as the linear combination of matrix $\boldsymbol{H}$ with a non-negative[7] vector of spline coefficients $\boldsymbol{a}$ (Ramsay, 1988). At this point, computing the spline coefficients boils down to solving a least-squares optimization problem with non-negativity constraints of the form:

$$\begin{aligned} \min_{\boldsymbol{a}} \quad & \left\| \boldsymbol{H}\boldsymbol{a} - \boldsymbol{g} \right\|_2 \\ \text{s.t.} \quad & \boldsymbol{a} \geq 0 \,, \end{aligned} \tag{7}$$

where $\left\| \cdot \right\|_2$ denotes the Euclidean norm, all elements of vector $\boldsymbol{a}$ are constrained[8] to be non-negative, and $\boldsymbol{g}$ is the vector of statistics computed in the previous step. To solve this problem, we use the quadratic programming solver implemented in the **R** package **quadprog** (Weingessel, 2019).

Finally, we use the estimated spline coefficients in the vector $\boldsymbol{a}$ to interpolate values on the statistic of interest across all sample sizes in the candidate range $\mathbb{N}_s$. To accomplish this, we create a new *I-Spline* basis $\boldsymbol{H}^*$ from all sample sizes in the range $\mathbb{N}_s$. The interpolated statistics are then obtained from the linear combination of the basis functions in matrix $\boldsymbol{H}^*$ and the estimated spline coefficients in $\boldsymbol{a}$. Figure 3 shows an example of the resulting smooth function.

[6]The number of inner knots used in the creation of the basis matrix is determined based on Leave-One-Out-Cross-Validation (LOOCV; Hastie et al., 2009), and the inner knots are placed at the quantiles of S.

[7]A monotone non-increasing trend can also be assumed for statistics that decrease as a function of sample size, in which case, the spline coefficients $\boldsymbol{a}$ are constrained to be non-positive.

[8]In our application, the first basis function in $\boldsymbol{H}$ was a constant function (i.e., 1), and the first element of $\boldsymbol{a}$ was freely estimated.

Step 3: Performing Stratified Bootstrapping

In the third step of the algorithm, we deploy non-parametric bootstrapping to quantify the uncertainty in the estimated spline. Since the number of MC replications performed during the first step is limited by the computation time associated with each MC replication, we use bootstrapping to represent the variability in the replicated performance measures associated with each sample size $s_t \in S$. In this case, we require relatively little computational time since we bootstrap the performance measures and, thus, re-estimating the model and generating data is not necessary. Non-parametric bootstrapping (Hastie et al., 2009) seems like a logical way to construct a measure of uncertainty around the estimated spline and account for the MC error. Furthermore, since we partition the sample size range $\mathbb{N}_s$ into several equidistant sample sizes $S \subseteq \mathbb{N}_s$, the bootstrapping scheme takes the form of stratified sampling (Parsons, 2017), where each $s_t \in S$ represents a stratum, associated with a countably infinite set of MC replications, i.e., the population of interest.

Specifically, during this step, we perform B bootstrap runs as follows:

1. Emulate the MC step of the algorithm in a computationally efficient manner by sampling with replacement R values from each vector $\boldsymbol{\xi}_t$ containing the replicated performance measures for the sample size $s_t \in S$. Then, compute a bootstrapped version of the vector of statistics $\boldsymbol{g}^{(b)} = \left[g\left(\boldsymbol{\xi}_1^{(b)}\right), \ldots, g\left(\boldsymbol{\xi}_T^{(b)}\right)\right]$, as discussed in *Step 1*, where $b(b = 1, \ldots, B)$ indicates the current bootstrap run.

2. Estimate bootstrapped spline coefficients $\boldsymbol{a}^{(b)}$ by repeating *Step 2* of the method, using the bootstrapped vector of statistics $\boldsymbol{g}^{(b)}$ instead of $\boldsymbol{g}$. Then, obtain bootstrapped statistic values for all sample sizes in range $\mathbb{N}_s$ by using the bootstrapped spline coefficients $\boldsymbol{a}^{(b)}$ for interpolating.

3. Since each sample size in the range $\mathbb{N}_s$ has a corresponding bootstrap distribution of statistic values (see subplot in Figure 4), use these distributions to compute 95% confidence intervals (CI) for each sample size in the range.

At the end of this step, we locate the first sample size for which the upper bound of its 95% CI reached the value τ for the statistic, and denote it as N_l. Similarly, we also locate the sample size for which the lower bound of its 95% CI reached the value τ and denote it as N_u. Together, the N_l and N_u sample sizes constitute the new lower and upper bounds for a shrunken version of the initial candidate range $\mathbb{N}_s$, where the optimal sample size is most likely to be found (see Figure 4). The distance between N_l and N_u is ultimately used to decide whether subsequent algorithm iterations are needed, which we discuss next.

Convergence Rule

To decide whether subsequent iterations of the algorithm are needed, we compute the distance between N_l and N_u and compare it to a scalar ε indicating the tolerance for the uncertainty around the sample size recommendation. If $N_u - N_l > \varepsilon$ then the initial range $\mathbb{N}_s$ is updated by setting its lower bound to N_l and its upper bound to N_u. Using the updated sample size range $\mathbb{N}_s$ the three steps of the algorithm steps are performed again, concentrating the new set of MC replications on the narrowed-down range of sample sizes. The algorithm continues to iterate until either $N_u - N_l \leq \varepsilon$, or a certain number of iterations has been elapsed. If the candidate range shrinks to ε within the allocated number of iterations, we consider that the algorithm has converged. Upon convergence, the median sample size of the shrunken range $\mathbb{N}_s$ and its corresponding distribution of statistic values are provided as recommendations to the researcher (see Figure 2 for an example of the output provided).

Validating the Recommendation

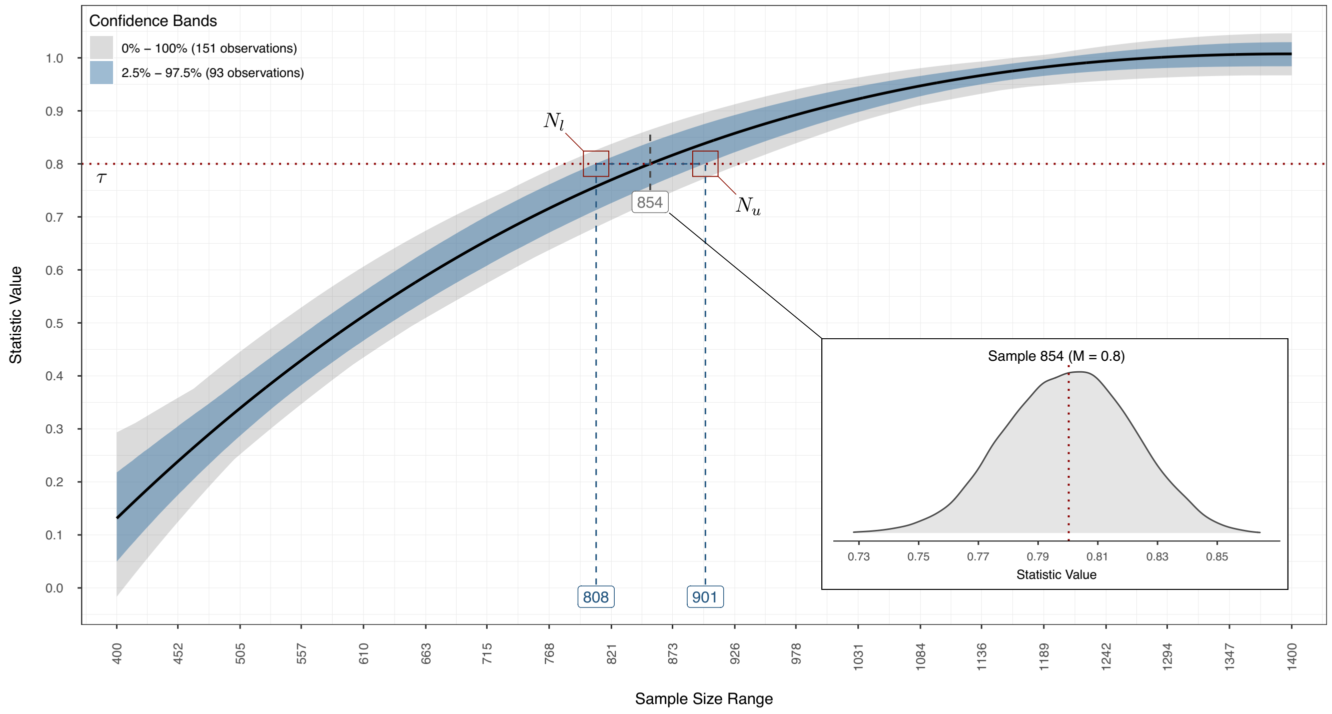
In addition to the three method steps discussed above, our method also performs an optional validation check for the recommended sample size. The goal of this validation is to determine whether the recommendation provided consistently recovers the desired value δ for the performance measure according to the statistic of interest. To do so, we perform an additional number of MC replications using the recommended sample size, resulting in a vector of replicated performance measures. Finally, we assess the quality of the recommendation by computing the statistic of interest. The value of the statistic obtained during the validation procedure should come close to the desired value specified by the researcher. In this case, this can be considered as strong evidence that the recommended sample size can reliably capture the target value on the performance measure according to the statistic of interest.

Simulation Study

In this section, we evaluate the performance of the proposed sample size computation method using a simulation study. The goal of this simulation is to test whether the method produces a good estimate for the sample size recommendation, for both small and large networks alike. The simulation consists of running the method for several true models of varying "complexity" and comparing how close the recommendations are to the *optimal sample size* corresponding to each true model. To check the robustness of the method, we investigate three performance measures, i.e., sensitivity (SEN), MCC, and ECC, and also

Figure 4

Visual representation of step three. The main plot displays the Monte Carlo error around the observed spline (i.e., the dark line) in terms of confidence intervals (CI), after one method iteration. The subplot in the bottom right shows the distribution of statistics for a given sample size (e.g., the median of all sample sizes that reached the desired target for the statistic).



Note. The horizontal red dotted line indicates the desired target for the statistic (i.e., 0.8), and the vertical red dotted line indicates the mean (i.e., 0.8) of the bootstrapped statistics for the median sample size 854. The vertical blue dashed lines indicate the lower N_l and upper N_u sample size bounds based on the 95% CI. The CI was constructed using the percentile method (Diciccio & Romano, 1988) based on 10000 bootstraps.

vary the number of selected sample sizes T and the number of MC replications R. For the remainder of this section, we start by presenting the simulation design and procedure in more detail and end by discussing the results of the simulation design.

True Models

While other network characteristics may also be relevant for the sample size calculation, we focus on the number of nodes and network density (i.e., the proportion of present edges) because they seem immediately obvious to how they may impact the required sample size. For instance, for a random network structure, a higher number of nodes implies more pairwise connections, and a higher density value implies that more of these pairwise connections will be present in the network. Therefore, we expect that higher values for either of these two characteristics will result in increasingly complex true models that likely require a larger sample size to accurately estimate the network structure. For this reason, we vary the number of nodes to generate several network models of varying estimation “complexity”. We consider three network models with a random architecture (Barabási & Albert, 1999), 90% positive edges, and a fixed density of 0.4. We select three values for the number of nodes, namely $P = \{10, 15, 20\}$. These values result in three network structures (see Figure 5), which serve as our first three true models, and reasonably cover typical networks encountered in the psychological literature (e.g., Isvoranu et al., 2020). Each network resulted in 17, 45, and 74, respectively, edge weights parameters.

We also consider three additional true models that may be less realistic but can help further investigate the robustness of the method. The additional true models use the same edge set as the first model but differ in terms of the strength and sign of the edge weights[9]. More specifically, the edge weights for true models four and five are 25% weaker and stronger, respectively, relative to the edge weights in the first true model. Finally, true model six contains the same edge set

[9]The network plots for the additional true models can be consulted in the supplementary materials.

and weights as true model one, but the proportion of positive edges is set to 50%. All models were generated using the function **generate_model** in the **R** package **powerly** version **1.8.0**.

Using the true models depicted in Figure 5 we identified the corresponding optimal sample size for each choice of performance measure. For the first true true model with 10 nodes (i.e., P_{10}) we observed: 526 for SEN, 514 for MCC, and 339 for ECC. For the second true model with 15 nodes (i.e., P_{15}): 1424 for SEN, 1415 for MCC, and 1413 for ECC. For the third true model with 20 nodes (i.e., P_{20}): 2007 for SEN and 2004 for both MCC and ECC. For the additional true models we only investigated SEN and observed the following optimal sample size values: 1127 for the fourth true model with 10 nodes and weaker edge weights (i.e., $P_{10}^{(-)}$), 249 for the fifth true model with 10 nodes and stronger edge weights (i.e., $P_{10}^{(+)}$), and 563 for the last true model with 10 nodes and 50% negative edges (i.e., $P_{10}^{(*)}$). See the *Simulation Procedure* section for details on how the optimal sample size values were determined. All optimal sample sizes yielded the desired target for the statistic with a three-decimal precision. Based on the observed optimal sample sizes, we fixed the initial lower and upper bounds of the candidate sample size range $\mathbb{N}_s$, ensuring each range spanned at least 1000 observations. As a result, the following ranges were used during the simulation: [150..1150] for model P_{10}, [500..2000] for model P_{15} and $P_{10}^{(-)}$, [1000..2500] for model P_{20}, [100..1100] for model $P_{10}^{(+)}$, and [150..1150] for model $P_{10}^{(*)}$.

Method Parameters

Preliminary simulations not presented here indicate that the robustness of the method depends on the amount of information the method works with, given by the total number of MC replications (i.e., $T \times R$). To check the robustness of the method, we vary the number of equidistant sample sizes T selected from the candidate sample size range $\mathbb{N}_s$, with values set to $T = \{20, 40, 60\}$. For the second method parameter, we vary the number of MC replications R performed for each selected sample size T, with values set to $R = \{10, 30, 50\}$. Combining these two method parameters, we obtain 9 method configurations (i.e., 3 quantities of selected sample sizes × 3 quantities of MC replications) that we apply to each true model. Throughout this simulation, we fix the sample size tolerance $\varepsilon = 50$, and we set the number of method iterations to 10.

Simulation Outcomes

To evaluate the performance of the method, we run the method many times and monitor how close the recommendations provided are to the optimal sample sizes. To this end, our simulation outcomes consist primarily of the means and standard deviations of the recommendations provided by the method. Additionally, we also compute the Mean Absolute Error (MAE) as an indication of how well each method configuration performed in recovering the optimal sample size. Furthermore, we also count how often the shrunken candidate range $\mathbb{N}_s$ (i.e., as observed in the last iteration of the method) contained the optimal sample size. For this simulation, we focus on SEN (i.e., for all six models), and MCC and ECC (i.e., for the first three models) as performance measures, and the statistic defined in Equation 6 (i.e., power). We select an *arbitrary* value of 0.6 as the target for all performance measures, which we want to attain in 80% of the cases (i.e., a power value of $\tau = 0.8$).

Simulation Procedure

The simulation procedure consists of two phases, a ground truth phase for establishing the optimal sample size associated with each true model, and an evaluation phase for assessing how well the method performs in recovering each optimal sample size.

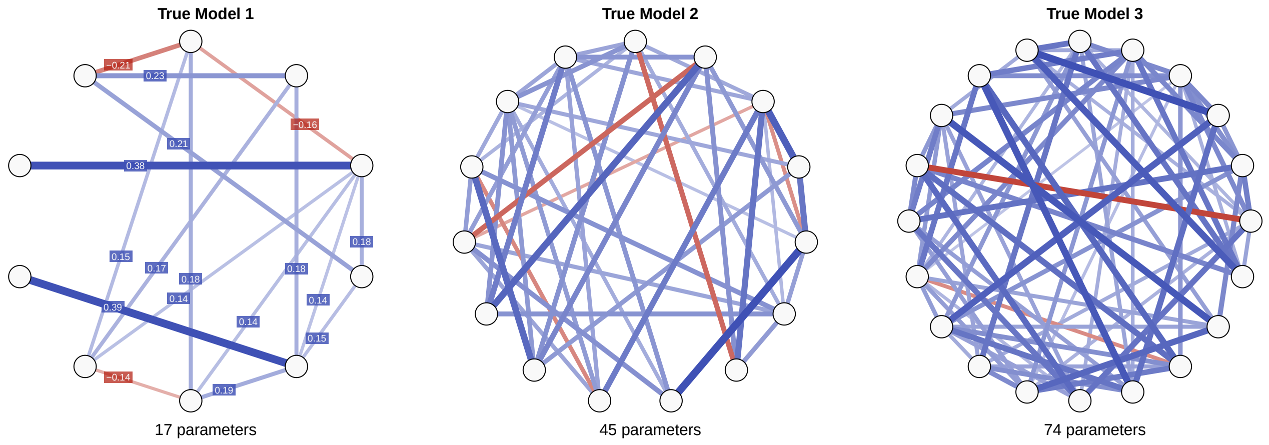
Since our simulation hinges on comparing the method results to the optimal sample size corresponding to each true model and performance measure, we need a way to find the optimal sample size for these models. To achieve this, we deploy a trial-and-error and computationally intensive MC approach. For each model, we start by guessing a range where the optimal sample size might be located, and we run 300000 MC replication for each sample size in that range (i.e., following the same strategy discussed during *Step 1* of the method). For each sample size, we then compute the statistic of interest and select the first sample size that reaches the value of $\tau = 0.8$. If such a sample is not observed, then we readjust the range and repeat the MC simulations and the computation of the statistics. By checking every sample size in the range and running a large number of MC simulations, we can expect to find the optimal sample size for each true model, or at least be reasonably close to the actual true value. These sample sizes serve as ground truth and constitute the benchmark sample sizes against which we compare the performance of our method.

Once the optimal sample sizes are established, we proceed by evaluating how well each method configuration performs in recovering these optimal sample sizes. To do this, for each true model and performance measure, we apply all nine method configurations and repeat the process 6000 times. Then, we compute the simulation outcomes indicated earlier. This enables us to find out, on average, how well the method does in recovering the optimal sample size and how precise it is in its recovery.

All computations were carried out on the Lisa Dutch national compute cluster and amounted to approximately 16500 CPU hours. We used different nodes on the cluster with Intel® Xeon® Gold 6130 processors, 2.10 GHz base

Figure 5

The first three true models (i.e., P_{10}, P_{15}, and P_{20}) used during the simulation study to assess the performance of the method.



Note. All models were generated with 0.4 density, 90% positive edges, and varying number of nodes (i.e., 10, 15, and 20). The edge weights for the first model have a mean absolute strength of 0.074 and are comprised the interval $[-0.213, 0.387]$. A mean absolute value of 0.048 in the interval $[-0.122, 0.184]$ for the second model, and 0.035 with $[-0.113, 0.139]$ for the third model. Models four to six represent variations of the first model in terms of strength and proportion of positive edge weights (i.e., see supplementary materials).

frequency, 16 cores, and 96 GB RAM. All simulations were performed in a Debian Linux environment using **`R`** version **`4.1.0`** linked against the **`OpenBLAS`** library and **`powerly`** version **`1.8.0`**. Multithreading was disabled and, instead, parallelization for a given method run was achieved using the built-in functionality in **`powerly`**, with the number of cores set to 15. The script for performing the simulation described above is available, with instructions, in the supplementary materials on OSF[10].

Simulation Results

Tables 1 and 2 show the means, standard deviations, and mean absolute errors of the sample size recommendations provided by the method. On average, the method showed good performance in recovering the optimal sample sizes. The mean of the recommendations provided by the method was, at most, three observations away from the truth for all true models and performance measures. The largest discrepancies were observed for models P_{10} (i.e., $M = 516.12$ for MCC and $M = 341.59$ for ECC) and $P_{10}^{(+)}$ (i.e., $M = 252$ for SEN). For all other models and performance measures, discrepancies were less than one observation. The variability of the recommendations is captured in Figure 6 for SEN, Figure 7 for MCC, and Figure 8 for ECC, where density and box plots are provided for all method configurations. For all true models, the highest variability was observed for the lowest number of selected sample sizes (i.e., $T = 20$) and the lowest number of MC replications (i.e., $R = 10$), with the highest standard deviations of 25.87 observations for $P_{10}^{(-)}$. As a general trend, increasing T or R resulted in lower variability, with the highest decrease resulting from simultaneously increasing T and R (e.g., from $SD = 25.87$ to $SD = 12.03$ for model $P_{10}^{(-)}$ when increasing from T from 20 to 60 and R from 10 to 60). A similar trend was also observed for the mean absolute errors of the recommendations provided (see Table 2).

Table 3 shows the inclusion rates for how often the shrunken candidate sample size range $\mathbb{N}_s$ contained the optimal sample size. As presented in Table 3, the number of MC replications had a large impact on the inclusion rates. Overall, the lowest number of MC replications (i.e., $R = 10$) resulted in the lowest inclusion rates, regardless of the number of selected sample sizes T. The biggest increase was observed when going from $R = 10$ to $R = 30$. To obtain roughly 80% inclusion rates, at least $R = 50$ were necessary when $T = 20$, and at least $R = 30$ when $T = 40$ or $T = 60$. Exception from this rule were MCC and SEN for P_{10}, where an inclusion rate of 80% was not consistently observed.

Method Characteristics

Table 4 displays the average number of iterations the method needed to converge. As expected, the method required more iterations to convergence when lower values of $T \times R$ were used (i.e., $M_{20\times10} = 3.56$, $M_{40\times30} = 1.86$, and $M_{60\times50} = 1.28$). Furthermore, the number of iterations increased as the true model became more complex (i.e., from $M = 1.80$ for model P_{10}, to $M = 2.41$ for model P_{20}). Most notably, model $P_{10}^{(-)}$ (i.e., with the scaled-down edge weights)

[10]OSF repository URL: osf.io/qksd7.

Table 1

Means and standard deviations (i.e., in parentheses) for the sample size recommendations for six true models and different parameter configurations: number of selected sample sizes (i.e., T) and number of MC replications performed for each sample size (i.e., R).

		$T = 20$			$T = 40$			$T = 60$		
		$R = 10$	$R = 30$	$R = 50$	$R = 10$	$R = 30$	$R = 50$	$R = 10$	$R = 30$	$R = 50$
True Model 1										
526	*SEN*	524.83 (17.21)	525.54 (11.99)	526.44 (11.20)	524.98 (14.00)	526.76 (12.37)	526.21 (11.86)	525.02 (13.48)	526.43 (12.13)	526.14 (11.07)
514	*MCC*	516.67 (19.56)	515.36 (13.20)	515.50 (12.03)	515.41 (15.34)	516.15 (13.03)	516.88 (13.74)	515.86 (14.12)	517.46 (13.68)	515.79 (12.40)
339	*ECC*	339.62 (14.90)	342.47 (11.00)	342.31 (11.44)	341.19 (12.65)	342.15 (11.83)	341.51 (10.34)	342.81 (13.29)	341.63 (10.78)	340.62 (9.33)
True Model 2										
1424	*SEN*	1422.97 (21.37)	1423.84 (14.36)	1423.80 (12.56)	1423.97 (17.22)	1423.78 (13.12)	1423.39 (12.73)	1423.87 (15.25)	1423.48 (13.01)	1424.06 (12.20)
1415	*MCC*	1416.28 (20.83)	1415.30 (14.70)	1415.68 (13.11)	1415.70 (17.50)	1415.39 (13.77)	1414.27 (12.68)	1415.43 (15.80)	1414.46 (13.37)	1415.01 (12.38)
1413	*ECC*	1414.00 (21.37)	1413.16 (14.89)	1413.15 (13.39)	1414.16 (17.39)	1412.96 (13.97)	1412.50 (12.74)	1413.04 (16.20)	1412.10 (13.30)	1413.23 (12.34)
True Model 3										
2007	*SEN*	2007.80 (22.95)	2007.84 (16.97)	2007.02 (13.57)	2008.19 (19.22)	2006.44 (13.44)	2005.78 (13.17)	2007.76 (17.66)	2005.53 (12.82)	2006.24 (12.56)
2004	*MCC*	2005.05 (22.66)	2004.80 (17.03)	2004.14 (13.57)	2005.22 (18.84)	2003.11 (13.32)	2002.17 (12.43)	2004.79 (17.42)	2002.53 (12.60)	2003.58 (12.32)
2004	*ECC*	2004.96 (22.06)	2004.93 (16.76)	2003.73 (13.32)	2005.12 (19.07)	2003.33 (13.08)	2002.18 (12.60)	2004.66 (17.30)	2002.05 (12.49)	2003.25 (12.50)
True Model 4										
1127	*SEN*	1125.44 (25.87)	1126.96 (18.81)	1127.17 (15.95)	1127.08 (21.23)	1127.12 (15.00)	1127.18 (12.54)	1127.42 (19.52)	1127.06 (13.15)	1127.96 (12.03)
True Model 5										
249	*SEN*	256.11 (22.67)	253.90 (18.02)	254.59 (16.84)	251.14 (15.57)	250.41 (10.09)	250.43 (8.57)	250.71 (13.07)	250.45 (8.96)	250.28 (7.51)
True Model 6										
563	*SEN*	562.31 (17.06)	563.20 (14.47)	562.76 (12.59)	562.78 (15.87)	562.83 (12.18)	563.13 (11.33)	563.78 (15.85)	563.20 (11.49)	563.70 (9.81)

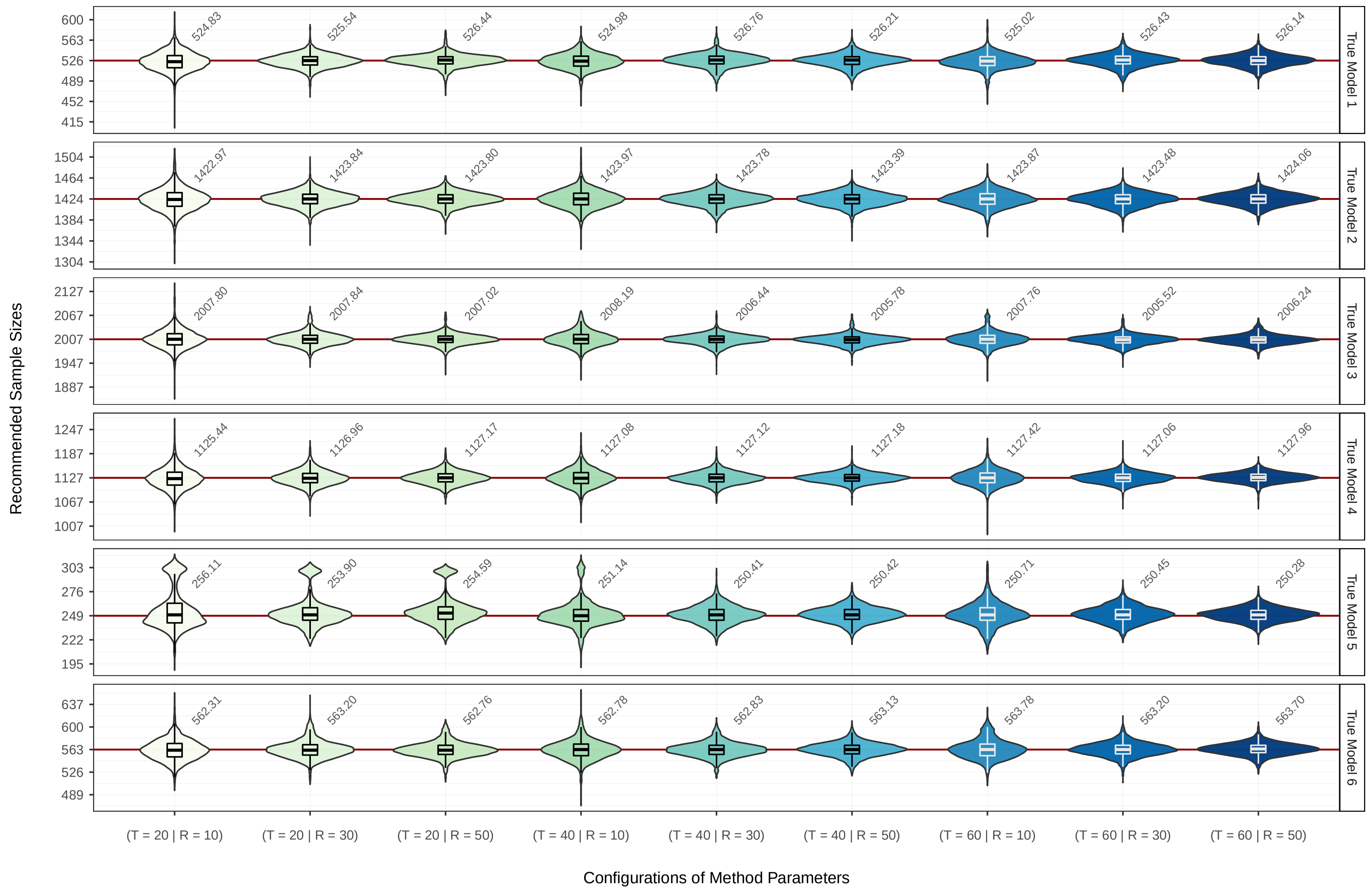
Note. All values are calculated based on 6000 method repetitions. The numerical boldface values represent the *optimal* sample sizes.

resulted in the highest number of iterations, with a mean of $M = 3.38$. In general, configurations employing a higher number of selected sample sizes T and MC replications R resulted in longer computation time for all six true models (i.e., see supplementary materials). Moreover, as the true model increased in "complexity", the method required more seconds to converge (i.e., $M = 2.92$ for $P_{10}^{(+)}$, $M = 3.74$ for P_{10}, $M = 3.76$ for $P_{10}^{(*)}$, $M = 6.24$ for P_{15}, $M = 7.21$ for $P_{10}^{(-)}$, and $M = 8.54$ for P_{20}).

The convergence of the method was evaluated based on whether the method succeeded in shrinking the candidate sample size range $\mathbb{N}_s$ to 50 sample sizes within a maximum of 10 iterations. As the true model increased in "complexity", lower values of T and R resulted in lower

Figure 6

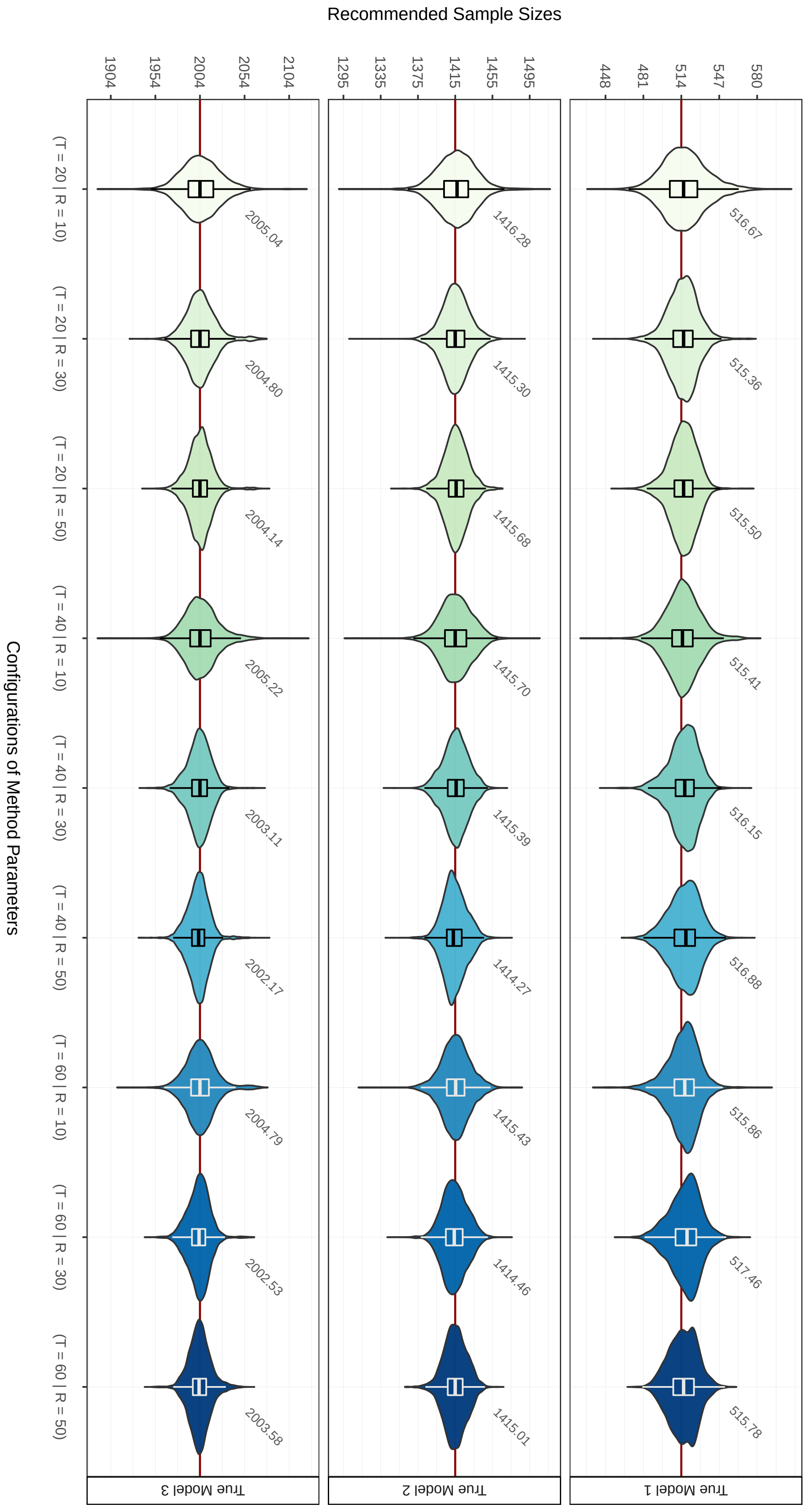
The performance of the method in recovering the optimal sample size for sensitivity under different configurations of method parameters: number of selected sample sizes (i.e., T), and number of MC replications performed for each sample size (i.e., R).



Note. The red lines indicate the *optimal* sample sizes. The gray values indicate the means for each configuration of method parameters.

Figure 7

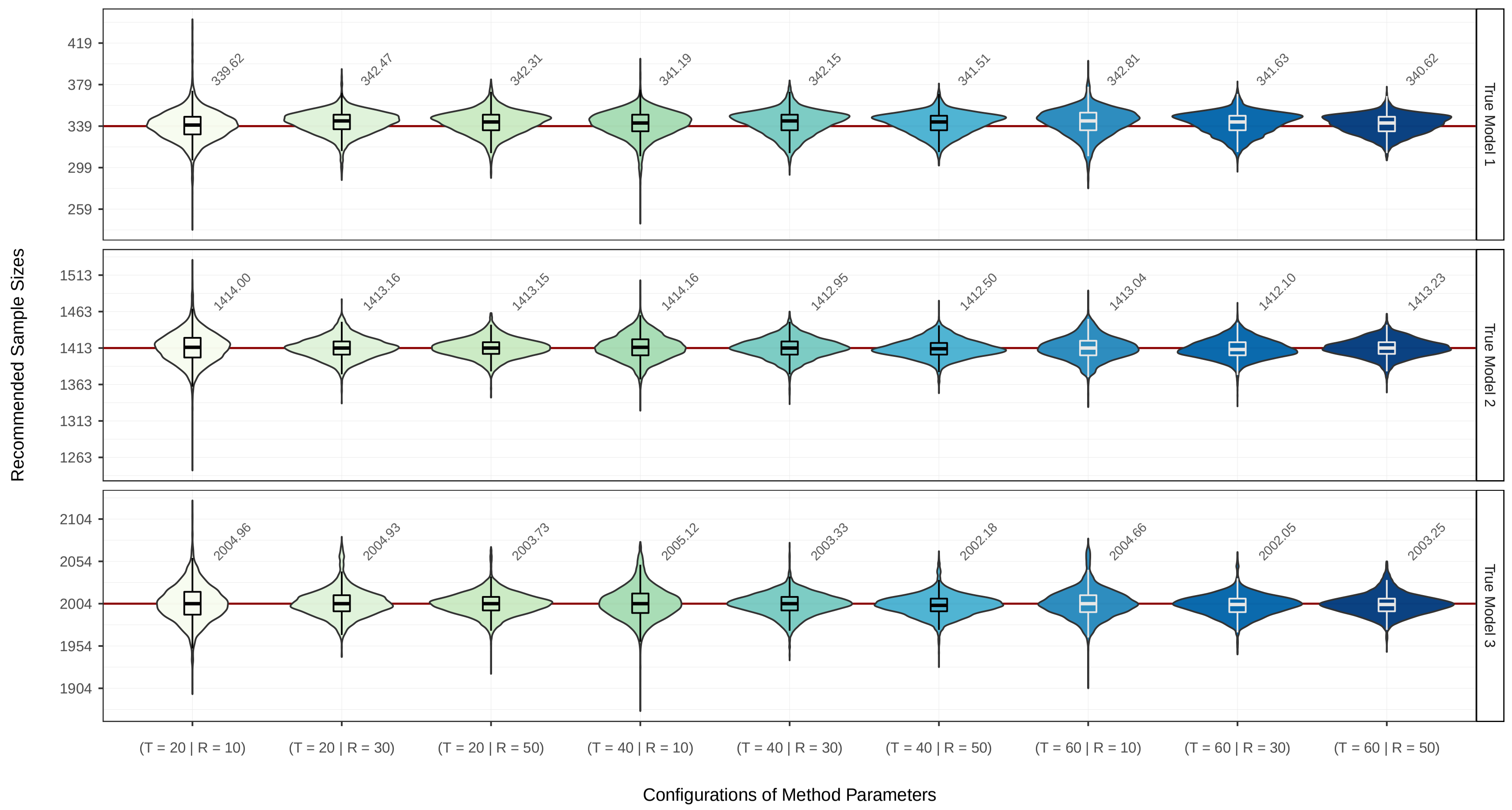
The performance of the method in recovering the optimal sample size for MCC *under different configurations of method parameters: number of selected sample sizes (i.e., T), and number of MC replications performed for each sample size (i.e., R).*



Note. The red lines indicate the *optimal* sample sizes. The gray values indicate the means for each configuration of method parameters.

Figure 8

The performance of the method in recovering the optimal sample size for ECC *under different configurations of method parameters: number of selected sample sizes (i.e., T), and number of MC replications performed for each sample size (i.e., R).*



Note. The red lines indicate the *optimal* sample sizes. The gray values indicate the means for each configuration of method parameters.

Table 2

Mean absolute errors for the sample size recommendations for six true models and different parameter configurations: number of selected sample sizes (i.e., T) from the candidate sample size range $\mathbb{N}_s$, and number of MC replications performed for each sample size (i.e., R).

	$T = 20$			$T = 40$			$T = 60$		
	$R = 10$	$R = 30$	$R = 50$	$R = 10$	$R = 30$	$R = 50$	$R = 10$	$R = 30$	$R = 50$
True Model 1									
SEN	13.42	9.29	8.29	10.96	9.32	9.01	10.13	9.24	8.60
MCC	15.23	10.28	9.49	11.67	10.46	11.31	10.90	11.35	10.18
ECC	11.03	9.20	9.50	9.96	9.97	8.73	11.03	9.17	7.78
True Model 2									
SEN	16.21	11.24	9.77	13.17	10.17	10.03	11.86	10.23	9.55
MCC	16.17	11.42	10.13	13.70	10.80	9.98	12.16	10.64	9.76
ECC	16.51	11.52	10.38	13.63	10.85	9.97	12.49	10.57	9.67
True Model 3									
SEN	17.31	12.66	10.14	14.60	10.10	9.73	13.12	9.78	9.50
MCC	17.38	12.64	10.14	14.38	10.15	9.39	12.90	9.60	9.30
ECC	16.82	12.41	10.07	14.54	9.98	9.59	12.86	9.58	9.52
True Model 4									
SEN	19.78	14.53	12.41	16.40	11.58	9.67	15.06	10.33	9.31
True Model 5									
SEN	16.63	12.10	11.80	10.36	7.86	6.76	9.71	7.14	6.01
True Model 6									
SEN	13.23	10.87	9.39	11.94	9.23	8.69	12.15	8.76	7.59

Note. All values are calculated based on 6000 method repetitions.

convergence rates, with the lowest rate of 92.80% observed for model $P_{10}^{(-)}$ at $T = 20$ and $R = 10$. Increasing the number of selected sample sizes and the number of replications to at least $T = 40$ and $R = 30$ resulted in 100% rates across all true models and performance measures.

Discussion

In this paper, we introduced a new method for performing sample size analysis in the context of psychological network models. We designed our method keeping in mind the challenges associated with providing sample size recommendations for psychological networks. Our method requires three researcher inputs, analogous to a typical power analysis, namely: 1) a hypothesized network structure, 2) a performance measure of interest (e.g., a sensitivity of 0.6), and 3) a statistic quantifying how the performance measure should be obtained (e.g., obtaining a sensitivity of 0.6 with a probability of 0.8). The method takes the form of a three-step recursive algorithm that narrows down a sample size range until the optimal sample size is identified. It starts with an MC simulations step, followed by a monotone spline fitting step, and ends with a bootstrapping step for quantifying the MC error around the spline. To help researchers assess the quality of the recommendation provided by the method, we also provide a validation procedure. Both the method and the validation procedure are implemented in a freely available **R** package **powerly** (see Figure 2).

Simulation Results

We evaluated the performance of the method by means of a simulation study for six Gaussian Graphical Models of varying estimation "complexity", three performance measures, and several configurations of method parameters. First, our results show that the method is reliable in recovering the optimal sample size, with the means of the recommendations provided being at most three observations away from the optimal values, regardless of model "complexity". It is worth mentioning that the method showed some difficulty in recovering the optimal sample size for the least complex true model (i.e., $P_{10}^{(+)}$ consisting of 10

Table 3

Percentages for the number of times the shrunken candidate sample size range $\mathbb{N}_s$ contained the optimal sample size for six true models and different parameter configurations: number of selected sample sizes (i.e., T) from the candidate sample size range $\mathbb{N}_s$, and number of MC replications performed for each sample size (i.e., R).

	$T = 20$			$T = 40$			$T = 60$		
	$R = 10$	$R = 30$	$R = 50$	$R = 10$	$R = 30$	$R = 50$	$R = 10$	$R = 30$	$R = 50$
True Model 1									
SEN	74.50%	86.50%	87.90%	82.30%	88.50%	89.80%	84.60%	89.40%	91.00%
MCC	67.10%	79.90%	82.40%	75.50%	78.60%	76.40%	77.80%	75.30%	75.30%
ECC	77.50%	81.50%	79.60%	78.50%	76.80%	74.90%	76.40%	74.30%	75.60%
True Model 2									
SEN	64.10%	82.80%	85.70%	73.10%	84.20%	84.90%	78.60%	84.30%	86.80%
MCC	62.80%	80.70%	84.70%	72.60%	83.10%	84.90%	78.00%	84.10%	85.30%
ECC	62.90%	80.70%	84.00%	71.90%	84.20%	85.80%	78.20%	84.60%	85.20%
True Model 3									
SEN	62.90%	79.60%	84.50%	71.80%	84.50%	86.00%	77.00%	85.40%	89.40%
MCC	63.60%	79.40%	85.00%	72.80%	83.90%	86.70%	77.10%	86.00%	89.70%
ECC	64.50%	80.20%	84.60%	72.50%	84.30%	86.10%	76.50%	85.20%	89.10%
True Model 4									
SEN	59.80%	75.40%	82.20%	67.10%	82.20%	86.80%	71.00%	84.40%	86.70%
True Model 5									
SEN	71.90%	83.10%	84.70%	85.70%	91.40%	92.00%	86.50%	91.20%	92.00%
True Model 6									
SEN	75.90%	84.20%	87.50%	78.80%	89.20%	91.30%	81.20%	91.30%	92.90%

Note. All values in the table are calculated based on 6000 method repetitions.

nodes and the strongest edge weights) when a low number of selected sample sizes T (i.e., 20) and MC replications R (i.e., 10) was used. Probably *EBICglasso* is not the best choice of estimation method for this particular scenario (e.g., Williams & Rast, 2020), but despite this, the method was still able to find the *optimal* sample size and the difficulty was rapidly mitigated by increasing T and R.

Second, the variability of the method recommendations was rather small and decreased consistently as the number of MC replications R and selected sample sizes T increased. Even at low settings (i.e., $T = 20$, with $R = 10$), the standard deviations remained below 30 sample sizes, and below 20 if R was increased to 30. Finally, the number of MC replications played an important role in ensuring that the narrowed-down sample size range in the last iteration of the method contained the optimal sample size. Our results suggest that whenever possible, a larger number of MC simulations should be preferred (i.e., ≥ 30), to obtain high precision.

Using the Method

Overall, our results suggest that with a reasonable amount of selected sample sizes and MC replications, the method will yield the optimal sample size with high precision. The method will, however, fail to provide the optimal sample size if the initial candidate range does not contain the optimal sample size in the first place. In such cases, if the lower bound of the candidate range was higher than the optimal sample, the method will provide the lower bound as the recommendation and a warning in the graphical output for *Step 3*. Conversely, if the upper bound of the range is lower than the optimal sample, the method will provide the upper bound as the recommendation. For best performance, researchers are advised to set the initial sample size range reasonably large (e.g., ≥ 1500 sample sizes).

Since the method relies on MC simulations, there is a computational cost associated. However, the method tries to mitigate this as much as possible by concentrating the computations on the sample sizes that seem most relevant. The highest computation penalty is given by the estimation

Table 4

The average number of iterations the method needed to converge for six true models and under different parameter configurations: number of selected sample sizes (i.e., T) from the candidate sample size range $\mathbb{N}_s$, and number of MC replications performed for each sample size (i.e., R).

	$T = 20$			$T = 40$			$T = 60$		
	$R = 10$	$R = 30$	$R = 50$	$R = 10$	$R = 30$	$R = 50$	$R = 10$	$R = 30$	$R = 50$
True Model 1									
SEN	3.32	2.23	1.86	2.45	1.57	1.33	2.00	1.33	1.16
MCC	3.52	2.42	1.98	2.63	1.72	1.40	2.16	1.42	1.26
ECC	2.40	1.60	1.38	1.73	1.20	1.10	1.42	1.09	1.03
True Model 2									
SEN	3.80	2.69	2.21	2.94	2.00	1.68	2.52	1.75	1.25
MCC	3.71	2.62	2.19	2.90	1.96	1.58	2.48	1.63	1.25
ECC	3.70	2.62	2.17	2.89	1.92	1.55	2.47	1.59	1.25
True Model 3									
SEN	3.95	2.75	2.31	3.04	2.12	1.78	2.58	1.85	1.36
MCC	3.93	2.76	2.28	3.05	2.10	1.76	2.59	1.84	1.35
ECC	3.94	2.74	2.30	3.06	2.12	1.76	2.57	1.83	1.36
True Model 4									
SEN	5.39	3.96	3.26	4.28	2.94	2.39	3.68	2.48	2.02
True Model 5									
SEN	1.90	1.37	1.12	1.55	1.14	1.04	1.31	1.05	1.01
True Model 6									
SEN	3.17	2.12	1.76	2.32	1.53	1.22	1.95	1.24	1.11

Note. A method iteration consisted of running all three steps and updating the candidate sample size range $\mathbb{N}_s$. All values are calculated based on 6000 method repetitions.

procedure of the model and the data generation mechanism. If one is faced with a situation in which the estimation procedure is extremely demanding, the adequate choice of initial sample size range can help speed up the convergence of the algorithm. In this situation, researchers are advised to run the method once with a large initial range and many selected sample sizes, but fewer MC replications (e.g., 3). This will help researchers quickly locate which sample sizes in the initial range seem most relevant. Then, the researcher can run the method a second time, with a reduced initial range, and, this time, with fewer selected sample sizes, but more MC replications, to ensure the quality of the recommendation.

While it is not to goal of this paper to provide sample size recommendations, our simulation study shows that, aside from the number of nodes and density, the strength of the edge weights plays an important role in the sample size determination, leading to drastically different optimal values. For example, our results show an increase from 526 to 1127 observations when the edge weights were scaled down by 25% (i.e., model $P_{10}^{(-)}$). Similarly, we observed a decrease from 526 to 249 observations for the model $P_{10}^{(+)}$ with weights 25% stronger. Furthermore, our results indicate that the sign of the edge weights also plays a role, albeit smaller, with the optimal sample increasing from 526 to 563 when adjusting the proportion of negative edges from 10% to 50% (i.e., from model P_{10} to $P_{10}^{(*)}$). These ad-hoc findings further emphasize the difficulty of obtaining sample size recommendations for psychological networks. Even though our method can help recover the optimal sample size in these and other scenarios, the burden still falls heavily on the researchers that need to specify their hypothesized network.

One of the method inputs requires researchers to specify their hypothesized network either as a model matrix or proceed by generating such a model matrix based on various hyperparameters (e.g., number of nodes and network density). This places the responsibility for carefully selecting true models in line with theoretical assumptions on the researchers. There is, however, the unfortunate risk that the method may yield an answer too precise for a relatively vague research question, as a result of a lack of theoretical justifications in selecting a true model. While the method cannot account for such theoretical aspects, we suggest two options for the researchers unsure about the theoretical correctness of their true models. First, if one is presented with a choice between a small number of competing true models, one can run the method for all these models in turn.

Second, if one has no prior information about the hypothesized network, we advise varying the true model either within or between method runs. Varying the true model within a method run can be achieved by randomly generating true models during the MC phase in *Step 1*. This ensures that during each MC replication, the performance measure is computed for a new true model generated according to some hyperparameters, which will likely require more iterations for the method to converge. A more sensible approach would be to vary the true model between different method runs by repeatedly running the method for randomly generated true models, which is still feasible given the short computation time. In this scenario, the meaning of the sample size recommendation changes from a specific network structure to a family of network structures generated according to some characteristics. It is important to note that a sufficient number of randomly generated true models should be used to ensure an adequate recommendation is obtained. This second scenario could, for instance, be helpful in the context of meta-research simulations for assessing whether certain studies are underpowered.

Extending the Method

One particularly strong characteristic of our method is its *ability to generalize* to other types of models, performance measures, and statistics. While our method already supports other network methodologies through the **R** package **bootnet** (Epskamp, Borsboom, & Fried, 2018), this characteristic allows us to go beyond network models and turn our method into a *general framework for conducting sample size analysis*. There are three requirements when considering extending the method to other models, namely: 1) the ability to generate true model parameters and represent these parameters in matrix form, 2) the ability to use the true model parameters as a data-generating mechanism, and 3) the ability to estimate the model from data. If these three requirements are satisfied, the model in question can readily be integrated into our framework. In terms of computational scalability, as discussed earlier, the biggest factors to consider are model estimation and data generation algorithms. Given that these two algorithms are executed for each MC replication, the number of $T \times R$ will roughly dictate the duration of one method iteration. Therefore, upscaling the sample size computation to more complex models, which may require hours to obtain a sample size, is still feasible with the proposed approach and would be impractical otherwise using a trial-and-error approach. Furthermore, while here we focused on sensitivity, MCC, and edge weights correlation, the performance measure and the statistic can also be modified.

Limitations

One potential weak spot for the method is given by the assumption of monotonicity enforced to the spline in *Step 2*. The rationale behind this assumption is that the statistic must change consistently as a function of sample size (i.e., either by increasing or decreasing) to identify a unique sample size that reaches the target value for the statistic. However, the MC error associated with a low number of replications can mask the monotone trend of the statistic if the candidate range is too narrow, displaying a seemingly random pattern. In such cases, the monotonicity assumption is violated, and the resulting spline has a slope close to 0, which, in turn, causes the method to fail to reduce the candidate range. Unfortunately, there is no clear way around this since relaxing the monotonicity assumption results in a spline that reaches the target value for the statistic at multiple sample sizes throughout the range. The most reasonable way around this limitation is to ensure that a sufficiently large number of MC replications are used.

Another point to consider is that, despite its generalizability, the method we introduce is still fairly complex, including many components (e.g., generating and estimating models, generating data, defining performance measures and statistics, creating spline bases, constrained optimization, cross-validation, stratified bootstrapping, and others). However, we attempted to take many of these steps out of the equation for potential users of our method by providing freely available software for using and extending the method.

Conclusion

This paper contributes a method for sample size analysis for complex models and complex requirements, abstracting away the necessity for researchers to implement custom trial-and-error and brute-force simulation designs. Furthermore, in addition to the method, we also introduced an **R** package **powerly** for which we are developing a well-defined API to allow researchers to perform simulations and generalize our method to new models (e.g., SEM models via the **lavaan** package; Rosseel, 2012) Taken together, this paper and **powerly** act as a general framework for conducting all sorts of sample size analyses. Sample size analysis for psychological networks has been a missing piece in the psychological network toolbox. With this paper, we aimed to contribute at completing the toolbox, providing researchers with the necessary tools to carry out their analyses.

References

Aberson, C. L. (2019, February 6). *Applied Power Analysis for the Behavioral Sciences* (2nd ed.). Routledge. https://doi.org/10.4324/9781315171500

Arend, M. G., & Schäfer, T. (2019). Statistical power in two-level models: A tutorial based on Monte Carlo simulation. *Psychological Methods*, *24*(1), 1–19. https://doi.org/10.1037/met0000195

Barabási, A.-L. (2009). Scale-Free Networks: A Decade and Beyond. *Science*, *325*(5939), 412–413. https://doi.org/10.1126/science.1173299

Barabási, A.-L., & Albert, R. (1999). Emergence of Scaling in Random Networks. *Science*, *286*(5439), 509–512. https://doi.org/10.1126/science.286.5439.509

Bonacich, P. (2007). Some unique properties of eigenvector centrality. *Social Networks*, *29*(4), 555–564. https://doi.org/10.1016/j.socnet.2007.04.002

Borsboom, D. (2017). A network theory of mental disorders. *World Psychiatry*, *16*(1), 5–13. https://doi.org/10.1002/wps.20375

Borsboom, D., Cramer, A. O. J., Schmittmann, V. D., Epskamp, S., & Waldorp, L. J. (2011). The Small World of Psychopathology. *PLoS ONE*, *6*(11), e27407. https://doi.org/10.1371/journal.pone.0027407

Borsboom, D., & Cramer, A. O. (2013). Network Analysis: An Integrative Approach to the Structure of Psychopathology. *Annual Review of Clinical Psychology*, *9*(1), 91–121. https://doi.org/10.1146/annurev-clinpsy-050212-185608

Borsboom, D., Fried, E. I., Epskamp, S., Waldorp, L. J., van Borkulo, C. D., van der Maas, H. L. J., & Cramer, A. O. J. (2017). False alarm? A comprehensive reanalysis of "Evidence that psychopathology symptom networks have limited replicability" by Forbes, Wright, Markon, and Krueger (2017). *Journal of Abnormal Psychology*, *126*(7), 989–999. https://doi.org/10.1037/abn0000306

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1), 6. https://doi.org/10.1186/s12864-019-6413-7

Cipra, B. A. (1987). An Introduction to the Ising Model. *The American Mathematical Monthly*, *94*(10), 937–959. https://doi.org/10.1080/00029890.1987.12000742

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed). L. Erlbaum Associates.

Constantin, M. A., Schuurman, N. K., & Vermunt, J. K. (2022). Supplementary Materials: A General Monte Carlo Method for Sample Size Analysis in the Context of Network Models. https://doi.org/10.17605/OSF.IO/QKSD7

Costantini, G., Epskamp, S., Borsboom, D., Perugini, M., Mõttus, R., Waldorp, L. J., & Cramer, A. O. (2015). State of the aRt personality research: A tutorial on network analysis of personality data in R. *Journal of Research in Personality*, *54*, 13–29. https://doi.org/10.1016/j.jrp.2014.07.003

Cramer, A. O. J., van Borkulo, C. D., Giltay, E. J., van der Maas, H. L. J., Kendler, K. S., Scheffer, M., & Borsboom, D. (2016). Major Depression as a Complex Dynamic System (I. Branchi, Ed.). *PLoS ONE*, *11*(12), e0167490. https://doi.org/10.1371/journal.pone.0167490

Cramer, A. O. J., Waldorp, L. J., van der Maas, H. L. J., & Borsboom, D. (2010). Complex realities require complex theories: Refining and extending the network approach to mental disorders. *Behavioral and Brain Sciences*, *33*(2-3), 178–193. https://doi.org/10.1017/S0140525X10000920

Dawid, A. P. (1979). Conditional independence in statistical theory. *Journal of the Royal Statistical Society: Series B (Methodological)*, *41*(1), 1–15.

de Leeuw, J. (2017). Computing and Fitting Monotone Splines. https://doi.org/10.13140/RG.2.2.36758.96327

Diciccio, T. J., & Romano, J. P. (1988). A Review of Bootstrap Confidence Intervals. *Journal of the Royal Statistical Society: Series B (Methodological)*, *50*(3), 338–354. https://doi.org/10.1111/j.2517-6161.1988.tb01732.x

Epskamp, S., Borsboom, D., & Fried, E. I. (2018). Estimating psychological networks and their accuracy: A tutorial paper. *Behavior Research Methods*, *50*(1), 195–212. https://doi.org/10.3758/s13428-017-0862-1

Epskamp, S., Cramer, A. O. J., Waldorp, L. J., Schmittmann, V. D., & Borsboom, D. (2012). Qgraph: Network Visualizations of Relationships in Psychometric Data. *Journal of Statistical Software*, *48*(4). https://doi.org/10.18637/jss.v048.i04

Epskamp, S., & Fried, E. I. (2018). A Tutorial on Regularized Partial Correlation Networks. *Psychological Methods*, *23*(4), 617–634. https://doi.org/10.1037/met0000167

Epskamp, S., Kruis, J., & Marsman, M. (2017). Estimating psychopathological networks: Be careful what you wish for. *PLoS ONE*, *12*(6), e0179891. https://doi.org/10.1371/journal.pone.0179891

Epskamp, S., Waldorp, L. J., Mõttus, R., & Borsboom, D. (2018). The Gaussian Graphical Model in Cross-Sectional and Time-Series Data. *Multivariate Behavioral Research*, *53*(4), 453–480. https://doi.org/10.1080/00273171.2018.1454823

Forbes, M. K., Wright, A. G. C., Markon, K. E., & Krueger, R. F. (2017). Evidence that psychopathology symptom networks have limited replicability. *Journal of Abnormal Psychology*, *126*(7), 969–988. https://doi.org/10.1037/abn0000276

Foygel, R., & Drton, M. (2010). Extended Bayesian information criteria for Gaussian graphical models. *Advances in neural information processing systems*, *23*, 604–612.

Fried, E. I., Eidhof, M. B., Palic, S., Costantini, G., Huisman-van Dijk, H. M., Bockting, C. L. H., Engelhard, I., Armour, C., Nielsen, A. B. S., & Karstoft, K.-I. (2018). Replicability and Generalizability of Posttraumatic Stress Disorder (PTSD) Networks: A Cross-Cultural Multisite Study of PTSD Symptoms in Four Trauma Patient Samples. *Clinical Psychological Science*, *6*(3), 335–351. https://doi.org/10.1177/2167702617745092

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso.

Biostatistics, *9*(3), 432–441. https://doi.org/10.1093/biostatistics/kxm045

Goutte, C., & Gaussier, E. (2005). A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In D. E. Losada & J. M. Fernández-Luna (Eds.), *Advances in information retrieval* (pp. 345–359). Springer Berlin Heidelberg.

Hancock, G. R., & French, B. F. (2013). Power analysis in structural equation modeling. In *Structural equation modeling: A second course, 2nd ed.* (pp. 117–159). IAP Information Age Publishing.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer New York. https://doi.org/10.1007/978-0-387-84858-7

Isvoranu, A.-M., Epskamp, S., & Cheung, M. W.-L. (2020). Network Models of Post-traumatic Stress Disorder: A Meta-analysis. https://doi.org/10.31234/osf.io/8k4u6

Jones, P. J., Ma, R., & McNally, R. J. (2019). Bridge Centrality: A Network Approach to Understanding Comorbidity. *Multivariate Behavioral Research*, 1–15. https://doi.org/10.1080/00273171.2019.1614898

Knuth, D. E. (1992). Two Notes on Notation. *The American Mathematical Monthly*, *99*(5), 403–422. https://doi.org/10.1080/00029890.1992.11995869

Kossakowski, J. J., Epskamp, S., Kieffer, J. M., van Borkulo, C. D., Rhemtulla, M., & Borsboom, D. (2016). The application of a network approach to Health-Related Quality of Life (HRQoL): Introducing a new method for assessing HRQoL in healthy adults and cancer patients. *Quality of Life Research*, *25*(4), 781–792. https://doi.org/10.1007/s11136-015-1127-z

Lauritzen, S. L. (1996). *Graphical models* (Vol. 17). Clarendon Press.

Matthews, B. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, *405*(2), 442–451. https://doi.org/10.1016/0005-2795(75)90109-9

Muthén, L. K., & Muthén, B. O. (2002). How to Use a Monte Carlo Study to Decide on Sample Size and Determine Power. *Structural Equation Modeling: A Multidisciplinary Journal*, *9*(4), 599–620. https://doi.org/10.1207/S15328007SEM0904_8

O'Malley, A. J., & Marsden, P. V. (2008). The analysis of social networks. *Health Services and Outcomes Research Methodology*, *8*(4), 222–269. https://doi.org/10.1007/s10742-008-0041-z

Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. *Science*, *349*(6251), aac4716–aac4716. https://doi.org/10.1126/science.aac4716

Opsahl, T., Agneessens, F., & Skvoretz, J. (2010). Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, *32*(3), 245–251. https://doi.org/10.1016/j.socnet.2010.03.006

Opsahl, T., & Panzarasa, P. (2009). Clustering in weighted networks. *Social Networks*, *31*(2), 155–163. https://doi.org/10.1016/j.socnet.2009.02.002

Parsons, V. L. (2017, February 15). Stratified Sampling. In N. Balakrishnan, T. Colton, B. Everitt, W. Piegorsch, F. Ruggeri, & J. L. Teugels (Eds.), *Wiley StatsRef: Statistics Reference Online* (1st ed., pp. 1–11). Wiley. https://doi.org/10.1002/9781118445112.stat05999.pub2

Powers, D. M. W. (2020). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. https://doi.org/10.48550/ARXIV.2010.16061

Ramsay, J. O. (1988). Monotone Regression Splines in Action. *Statistical Science*, *3*(4), 425–441. https://doi.org/10.1214/ss/1177012761

Rényi, A. (1961). On measures of entropy and information, 547–561.

Rosseel, Y. (2012). Lavaan: An R Package for Structural Equation Modeling. *Journal of Statistical Software*, *48*(2). https://doi.org/10.18637/jss.v048.i02

Schober, P., Boer, C., & Schwarte, L. A. (2018). Correlation Coefficients: Appropriate Use and Interpretation. *Anesthesia & Analgesia*, *126*(5), 1763–1768. https://doi.org/10.1213/ANE.0000000000002864

Schoemann, A. M., Miller, P., Pornprasertmanit, S., & Wu, W. (2014). Using Monte Carlo simulations to determine power and sample size for planned missing designs. *International Journal of Behavioral Development*, *38*(5), 471–479. https://doi.org/10.1177/0165025413515169

Sharma, H., & Srivastava, V. (2015). Descriptive Analysis of Social Network Measures. *3*(7), 13.

Shen, W., Kiger, T. B., Davies, S. E., Rasch, R. L., Simon, K. M., & Ones, D. S. (2011). Samples in applied psychology: Over a decade of research in review. *Journal of Applied Psychology*, *96*(5), 1055–1064. https://doi.org/10.1037/a0023322

Uhler, C. (2017, July 13). *Gaussian Graphical Models: An Algebraic and Geometric Perspective*. Retrieved January 30, 2020, from http://arxiv.org/abs/1707.04345

van Borkulo, C. D., Borsboom, D., Epskamp, S., Blanken, T. F., Boschloo, L., Schoevers, R. A., & Waldorp, L. J. (2014). A new method for constructing networks from binary data. *Scientific Reports*, *4*(1), 5918. https://doi.org/10.1038/srep05918

van Erven, T., & Harremos, P. (2014). Rényi Divergence and Kullback-Leibler Divergence. *IEEE Transactions on Information Theory*, *60*(7), 3797–3820. https://doi.org/10.1109/TIT.2014.2320500

Wang, W., & Yan, J. (2021). *Splines2: Regression spline functions and classes*. https://CRAN.R-project.org/package=splines2

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*(6684), 440–442. https://doi.org/10.1038/30918

Weingessel, A. (2019). *Quadprog: Functions to solve quadratic programming problems*. https://CRAN.R-project.org/package=quadprog

Williams, D. R., & Rast, P. (2020). Back to the basics: Rethinking partial correlation network methodology. *British Journal of Mathematical and Statistical Psychology*, *73*(2), 187–212. https://doi.org/10.1111/bmsp.12173

Yin, J., & Li, H. (2011). A sparse conditional Gaussian graphical model for analysis of genetical genomics data. *The*

Annals of Applied Statistics, *5*(4), 2630–2650. https://doi.org/10.1214/11-AOAS494

Zhang, Z. (2014). Monte Carlo based statistical power analysis for mediation models: Methods and software. *Behavior Research Methods*, *46*(4), 1184–1198. https://doi.org/10.3758/s13428-013-0424-0