Automated Inductive Causation:

Recovering Causal Structures from Observational Data

Alexander Savi

2010

University of Amsterdam

Department of Psychological Methods

Bachelor's thesis

Supervised by dr. L.J. (Lourens) Waldorp and dr. V.D. (Verena) Schmittmann

Abstract

Electronic diaries continuously generate information-rich observational data. The underlying causal network from such data could provide researchers and therapists with valuable information. The inductive (inferred) causation algorithm (IC algorithm) is a promising technique for recovering causal structures from observational data, yet has never been formally examined for its use for psychological data containing measurement error. This study examines how well, and under what circumstances, the algorithm accurately recovers causal structures from observational data. Four causal networks are created, embedded in different datasets, and then recovered using an automated IC algorithm. The results may lead to a better understanding of the characteristics of the IC algorithm, when applied to psychological data. Moreover, it could provide researchers and therapists with a tool for getting valuable causal information from observational data.

1. Introduction

In psychological research, the concept of causality has a controversial history. The debate whether true causal relationships are susceptible to recovery, is still very much alive today (Cook, Scriven, Coryn, & Evergreen, 2009). Researchers have therefore longtime neglected, and sometimes even rejected the subject, while preferring notions of correlation and dependence.

A solution to the problem of causality was presented in the form of experimental research (Fisher, 1971), a nowadays well-established concept. In experimental research, a specific set of variables is manipulated and the effects are measured in a controlled environment. The possible confounds are eliminated through randomized selection and allocation of the participants in the experiment. However, not all research is suitable for experimental design, such as observational research through electronic diaries in clinical treatment settings. Observational research is typically not conducted in a controlled environment, nor are the participants randomly selected and allocated. Observational data thus contains confounds and causation cannot be easily inferred. Yet observational data does contain valuable information on the underlying causal networks of for instance different symptoms.

Fortunately, statistical techniques for inferring the underlying causal networks from observational data have been developed in recent years. These methods include structural equation modeling (e.g. Shimizua & Kanoa, 2008) and TETRAD (e.g. Karimi & Hamilton, 2000; Spirtes, Glymour, & Scheines, 1993). This study will cover yet another statistical technique, which is a promising technique in spite of having been generally ignored in psychological research. Verma and Pearl (1991; Pearl, 2000) discovered an elegant way of recovering causal structures from observational data. The authors developed an algorithm, which only needs simple statistical dependencies to infer a causal structure, as is explained below.

The so-called inductive causation algorithm (IC algorithm) is based on the notion of statistical equivalence of causal models. Causal models are often represented as directed acyclic graphs (DAGs), graphs containing solely directed edges and no cyclic structures. Given a DAG of three nodes connected by two edges, Verma and Pearl distinguished a group of in total four different types of causal models. These four models are shown below in *Figure 1.1* to *Figure 1.4*.

Figure 1.1. Chain.

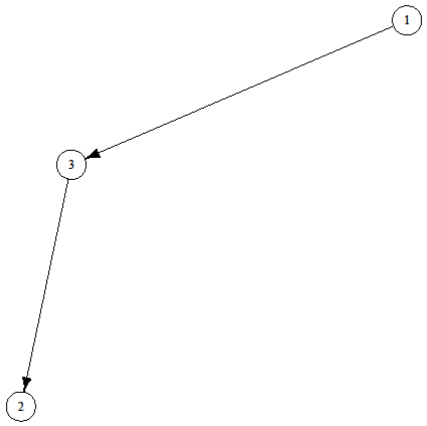


Figure 1.2. Chain.

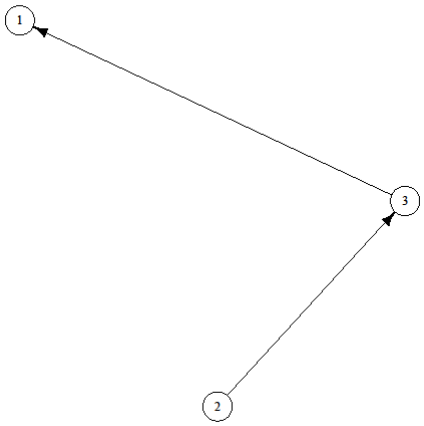


Figure 1.3. Common cause.

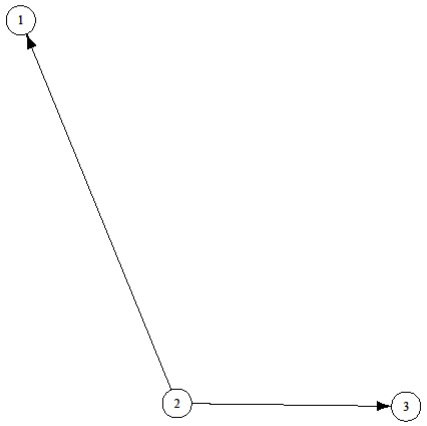
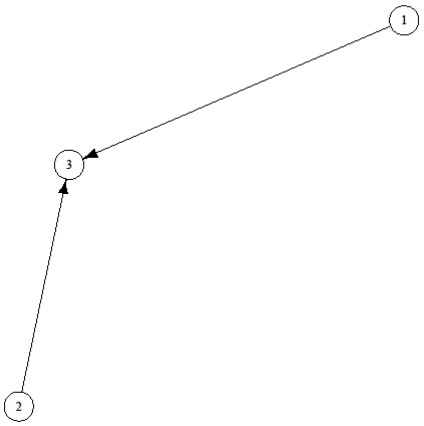


Figure 1.4. Common effect (collider).



Three of these models form a group of statistically equivalent models: the chains and the common cause model. A chain is a path with the edges in the same direction; a common cause is the middle node in a path with the edges directed away from it. These models are statistically equivalent as they share the same probability distribution, namely the outer nodes are dependent, however become statistically independent in case the middle node is conditioned on. Conditioning refers to the conditional distribution while keeping the conditioning variable fixed. Conditioning on the middle node of a path, means that the middle node is given (e.g. fixed) (Agresti & Franklin, 2007), and is thus controlled for. By controlling for a node, it can be seen whether it influences the dependency between two other nodes. For instance the distribution of node 1 and node 2 when node 3 is fixed, is called the conditional distribution of node 1 and node 2 given node 3.

The fourth model, the collider, has a different probability distribution and can thus be easily distinguished. A collider is the middle node in a path with the edges directed towards it. The outer nodes of a collider (i.e. its parents) are independent; however, they become statistically dependent in case the collider is conditioned on.

Excitingly, the statistical independencies can also be discovered in a graph, using the *d*-separation criterion. A pair of variables that is *d*-separated by (a set of) node(s), is conditional independent given the (set of) node(s). A path (such as in *Figure 1.1* to *Figure 1.4*) is *d*-separated by (a set of) node(s) Z, if one of two conditions is met (Pearl, 2000). Either the path needs to contain a chain or common cause structure and Z must contain the middle node of chain or common cause structure, or the path needs to contain a collider structure and Z must not contain the middle node of the collider structure. Discovering the *d*-separation allows for direct discovery the conditional independencies from the graph is, but only if the Markov assumption is met. The Markov assumption says that if a pair of variables is d-separated given a (set of) variable(s), the pair of variables is conditionally independent given the (set of) variable(s) (Pearl, 2000). Referring back to *Figure 1.1* to *Figure 1.4*, the *d*-

separation criterion thus prescribes when variable one and two are conditionally independent given variable three. Variable three is said to block the path, if it is the middle node in a chain connecting variables one and two, if it is the common cause of variables one and two, or if it is neither the common effect, nor a child of the common effect of variables one and two. *Figure 1.1* to *Figure 1.*3 show, that given the *d*-separation criterion, variables one and two are blocked (*d*-separated) by variable three. Thus in these figures, variables one and two are conditionally independent given variable three. *Figure 1.4* shows, that given the *d*-separation criterion, variables one and two are not blocked (*d*-separated) by variable three. Thus in this figure, variables one and two are not conditionally independent given variable three.

Verma and Pearl (1991), and later Spirtes and Glymour (Spirtes, Glymour, & Scheines, 1993), used the principle of equivalence to recover DAGs from observational data. Verma and Pearl's IC algorithm first recovers the adjacency pattern, which is a pattern containing undirected edges for all (conditional) dependencies. The adjacency pattern is created by checking the (conditional) dependencies and adding an undirected edge if all are dependent. Second, the rudimentary pattern is recovered, which is the adjacency pattern containing directed edges for the colliders. Due to their distinguishable nature, the colliders can be recovered from the adjacency pattern. Although the algorithm contains a third step, this step will not be discussed here. The prime focus of this paper is on the core of the algorithm: discovering adjacencies and colliders. The final step is a consequence of finding adjacencies and colliders.

A rudimentary pattern, partially with directed edges, can be recovered from observational data. The rudimentary pattern is a specific representation of the underlying causal structure of the data (e.g. the lower left graphs in *Figure 2.1* to *Figure 2.4*). It contains the adjacency pattern, with undirected edges for the paths that are statistically equivalent (chains and common cause structures) and directed edges for the paths that can be

distinguished statistically (colliders). Unfortunately, as the algorithm cannot distinguish between the equivalent causal structures, consequently only a limited number of causal structures can be retrieved from a DAG. Nonetheless, it is a very straightforward and promising technique for the analysis of observational data. For instance, in the analysis of electronic diaries data, which is information-rich and abundant. Typical electronic diary data consists of the scored answers to questions that measure the symptoms of a certain disorder. The (conditional) dependencies and independencies between these symptoms, and the causal network underlying the disorder, are not recovered through, for instance, regression analysis. Regression analysis returns only a single dependent variable (the other variables are independent), providing no insight in the underlying causal structure. The IC algorithm on the other hand, is a technique for recovering the underlying causal network, including the (conditional) dependencies and independencies present between the symptoms.

Although the IC algorithm may prove to be a viable option, the application of the algorithm for this type of data has been sparse so far (with the exception of e.g. Wild, Eichler, Friederich, Hartmann, Zipfel, & Herzog, 2010). Its efficaciousness for psychological research has thus yet to be studied, and statistical techniques are needed, to deal with the fair amount of measurement error which is typical of psychological data.

This paper presents an early study of the IC algorithm, when administered at psychological observational data. In a series of 35 simulated datasets, representing four different causal structures, the rudimentary pattern is recovered using the implemented IC algorithm. It is then analyzed how well and under what circumstances the algorithm is able to recover the rudimentary pattern of the underlying DAG from a certain dataset.

2. Methods

2.1. IC algorithm

In order to enable the use of the algorithm for psychological data and implement it, the IC algorithm was rephrased in statistical terms and translated to the R language (see section *2.2*). The pseudo code is provided in full in Appendix A (pp. 38-45), and in short below.

The implemented IC algorithm first creates a covariance matrix from the data (Appendix A, p. 38, *Function 1*), and subsequently all partial correlations between the variables in the data are calculated (Appendix A, p. 38, *Function 2*). Second, the adjacency pattern is retrieved, by checking all (partial) correlations of each pair of variables (Appendix A, p. 40, *Function 4*). The null-hypothesis for the (conditional) independence of a pair of variables is that their (partial) correlation in a normally distributed dataset equals zero. If the (partial) correlation is significant ($\alpha \leq .05$), the null-hypothesis is rejected and the pair of variables is dependent. If the (partial) correlation is not significant ($\alpha > .05$), the null-hypothesis is not rejected and the pair of variables is independent (given the partialled out variable(s)). An edge is present if a pair of variables is dependent under all conditions (Appendix A, p. 39, *Function 3*).

Third, the colliders are retrieved. Paths containing three variables, and of which only two pairs of variables are adjacent, are labeled as possibly unshielded colliders (Appendix A, p. 40, *Function 5*). For each possible unshielded collider, the partial correlation is checked, between two of its parents given the possible unshielded collider (Appendix A, p. 41, *Function 6*). If not significant, no statistical dependency has arisen, and thus no collider is inferred. However if significant, the correlation between the parents arises given the possible unshielded collider, and thus a collider is inferred. Finally, both patterns are combined to form the rudimentary pattern (Appendix A, p. 42, *Function 7*).

A few more functions were written to produce graphs from the patterns. The graphs of the rudimentary patterns in the results section were for instance created using these functions. In Appendix A, *Function 8* to *Function 12* (pp. 43-44), the full pseudo code is provided for these functions.

2.2. Materials

R, a free and open-source software environment for statistical computing and graphics (www.r-project.org), was used. The package containing the IC algorithm was built in R, as were the datasets and the program running the simulations.

2.3. Causal networks

To represent causal networks for the simulations, four DAGs were created, with three to six nodes. *Figure 2.1* to *Figure 2.4* show the DAGs. At the bottom left of each figure, the rudimentary pattern for the DAG is shown. The rudimentary pattern is the most accurate representation of the DAG that the current implementation of the algorithm can recover. A rudimentary pattern represents a class of equivalent models that cannot be distinguished by the algorithm (Verma & Pearl, 1991).

Each DAG contained at least one path containing a collider. For all paths containing a collider, node three is the collider node. Within the set of DAGs, three DAGs contained one path containing a collider (DAG 1, DAG 2, and DAG 3), while the fourth DAG contained three paths containing a collider (DAG 4). Finally, within the set of DAGs, some DAGs contained colliders with connected parents (node 3 of DAG 1 and DAG 4), and some DAGs contained colliders with disconnected parents (node 3 of DAG 2 and 3). Parents are the nodes with edges directed at the collider node. The DAGs were further created without any specific method.

Figure 2.1. DAG 1, containing three nodes.

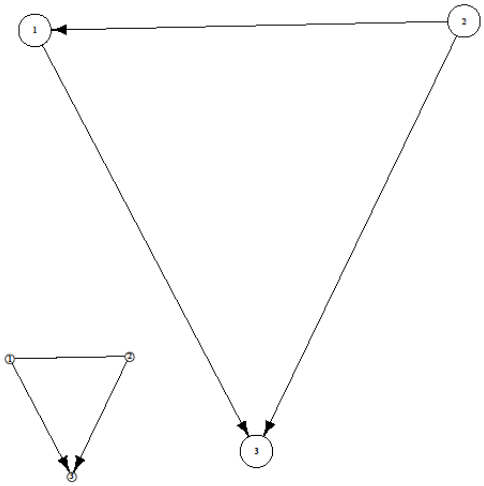


Figure 2.2. DAG 2, containing four nodes.

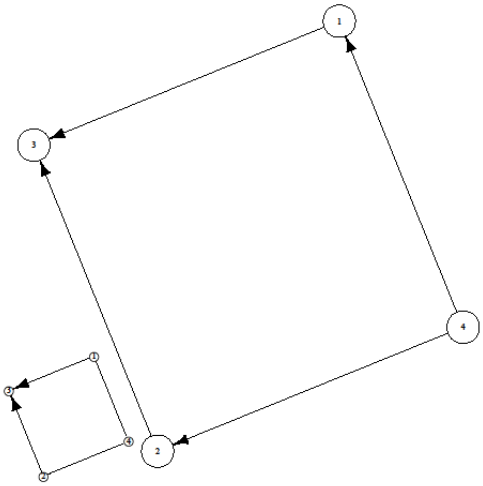


Figure 2.3. DAG 3, containing five nodes.

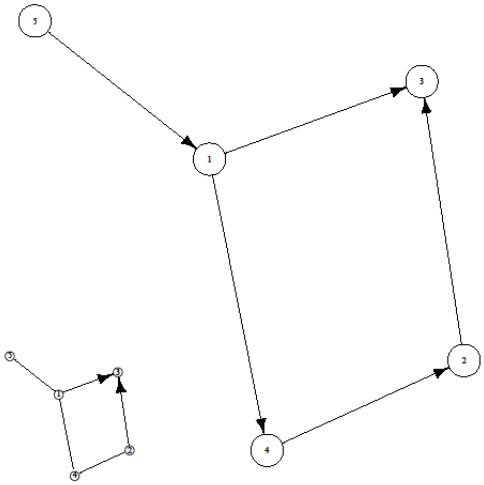
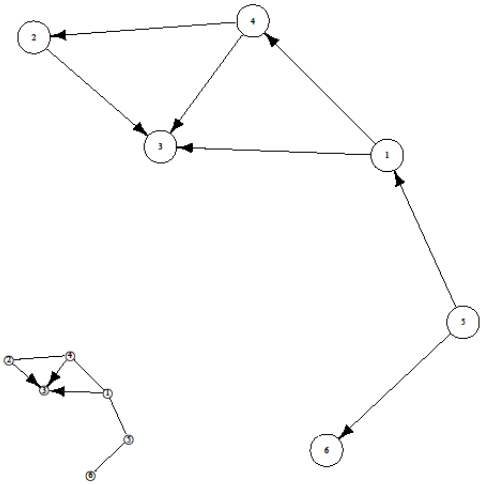


Figure 2.4. DAG 4, containing six nodes.



2.4 Data Simulation

The causal networks of the four DAGs were captured in simulated datasets. Each DAG was simulated with seven differing numbers of observations, namely 20, 30, 50, 100, 500, 2000, and 10000. The datasets were created using customary regression equation notation, e.g.: $n_i = \beta_{ij} * n_j + \varepsilon_i$, with n = node i, β = regression coefficient for edge j to i, and ε = residual for node i. The residuals were random normally distributed, with mean zero and variance one.

The regression coefficients were fixed to the value two. Besides this, data of DAG 1 was simulated with a different set of regression coefficients (henceforth: DAG 1b). The IC algorithm has a difficulty finding the parents of the collider in DAG 1, because it represents a shielded collider and not an unshielded collider, as required by the algorithm. Furthermore, in case of equal regression coefficients for all edges (henceforth: DAG 1a), the partial correlation between the parents cancels. To show this particular characteristic of the IC algorithm, data for DAG 1b was simulated with the regression coefficient for the edge between node two and one set to the value one, and the regression coefficient for one of the collider edges set to the value four. In total, there were 35 different conditions (5 DAGs x 7 sample sizes), each of which was simulated with 1000 replications.

2.5 Recovery analysis

The 35000 different datasets were each simulated and analyzed using the package containing the IC algorithm. The original adjacency, collider, and rudimentary patterns of the DAGs, were then compared to the recovered patterns. For all DAGs and each number of observations, the overall rate of correct recovered adjacency and collider patterns was stored. Furthermore, the frequencies and rudimentary patterns of all correctly and incorrectly recovered DAGs were stored. These results are presented in the next section.

3. Results

3.1 DAG 1a, equal regression coefficients

DAG 1 contained three nodes, and a collider with connected parents (*Figure 2.1*). The implemented IC algorithm reported no errors during the simulations. The adjacency pattern was correctly recovered in 100 percent of the simulations, beginning at datasets with 500 observations. The collider pattern was never correctly recovered. This is expected, as the IC algorithm requires a collider to be unshielded in order to find it, which is not the case for this

DAG. *Figure 3.1* shows a graph of the overall rate of correct recovered adjacency and collider patterns for these simulations.

The number of different recovered rudimentary patterns decreased, as the number of observations in a dataset increased. The most frequently recovered rudimentary pattern for the lower numbers of observations (20 through 30) contains two undirected edges, and lacks the edge between the parents of the collider (*Figure 3.2*). A recovered rudimentary pattern with three undirected edges is most frequent in datasets beginning at 50 observations (*Figure 3.3*). All graphs showing the results for each number of observations can be found in Appendix B, *Figure B.1* to *Figure B.8* (pp. 46-47).

3.2 DAG 1b, unequal regression coefficients

DAG 1 contained three nodes, and a collider with connected parents (*Figure 2.1*). The implemented IC algorithm reported no errors during the simulations. The adjacency pattern was correctly recovered in 100 percent of the simulations, beginning at datasets with 50 observations. In datasets with 20 and 30 observations, the correct adjacency pattern was recovered in around 90 and near 100 percent of the simulations. The collider pattern was correctly recovered in around 5 percent of the simulations, in datasets with 20 observations. In these specific cases the edge between the parents was not recovered however, and in datasets with larger numbers of observations the collider pattern was not found. *Figure 3.4* shows a graph of the overall rate of correct recovered adjacency and collider patterns for these simulations.

The number of different recovered rudimentary patterns decreased, as the number of observations in a dataset increased. The expected, but incorrect, rudimentary pattern without colliders is recovered most frequently from datasets beginning at 20 observations (*Figure 3.5*). This pattern contains three undirected edges. All graphs showing the results for

each number of observations can be found in Appendix B, *Figure B.9* to *Figure B.16* (pp. 48-49).

3.3 DAG 2

DAG 2 contained four nodes, and a collider of which the parents were *d*-separated by a common cause (*Figure 2.2*). The implemented IC algorithm reported no errors during the simulations. The adjacency pattern was correctly recovered in around 95 percent of the simulations, beginning at datasets with 30 observations. The collider pattern was correctly recovered in around 90 percent of the simulations, beginning at datasets with 2000 observations. *Figure 3.6* shows a graph of the overall rate of correct recovered adjacency and collider patterns for these simulations.

The number of different recovered rudimentary patterns decreased, as the number of observations in a dataset increased. The most frequently recovered rudimentary pattern for the lower numbers of observations (20 through 100), contains both directed collider edges, but lacks both other edges (*Figure 3.7*). The correct rudimentary pattern is recovered from datasets beginning at 500 observations (*Figure 3.8*). However for each dataset beginning at 500 observations, it is true that the rudimentary pattern is never correctly recovered for the full 100 percent of the simulations of that dataset. All graphs showing the results for each number of observations can be found in Appendix B, *Figure B.17* to *Figure B.24* (pp. 50-51).

3.4 DAG 3

DAG 3 contained five nodes, a collider of which the parents were *d*-separated by the middle node in a chain, and a node with an edge directed at this chain (*Figure 2.3*). The implemented IC algorithm reported no errors during the simulations. The adjacency pattern was correctly recovered in 100 percent of the simulations, beginning at datasets with 500

observations. It was correctly recovered in around 90 percent of the simulations, for datasets with 100 observations. The collider pattern was correctly recovered in near 100 percent of the simulations, for datasets with 10000 observations. It was correctly recovered in around 90 percent of the simulations, for datasets with 2000 observations. *Figure 3.9* shows a graph of the overall rate of correct recovered adjacency and collider patterns for these simulations.

The number of different recovered rudimentary patterns decreased, as the number of observations in a dataset increased. The most frequently recovered rudimentary patterns for the lower numbers of observations (through 100) contain either both directed collider edges, or one of both collider edges (*Figure 3.10*). Beginning at 100 observations, the most frequent recovered rudimentary pattern is near correct, however lacking the adjacency between node 1 and 4 (*Figure 3.11*). Beginning at 2000 observations, the correct rudimentary pattern is recovered most frequently (*Figure 3.12*). All graphs showing the results for each number of observations can be found in Appendix B, *Figure B.25* to *Figure B.32* (pp. 52-53).

3.5 DAG 4

DAG 4 contained six nodes, including a collider node with three edges directed towards it, and of which two pairs of parents were connected (*Figure 2.4*). The implemented IC algorithm reported no errors during the simulations. The adjacency pattern was correctly recovered in nearly 100 percent of the simulations, for datasets with 10000 observations. The collider pattern was never correctly recovered. However, two of the three collider edges were correctly recovered in nearly 100 percent of the simulations, beginning at datasets with 500 observations. *Figure 3.13* shows a graph of the overall rate of correct recovered adjacency and collider patterns for these simulations.

The number of different recovered rudimentary patterns decreased, as the number of observations in a dataset increased. The most frequent recovered rudimentary pattern for the lower numbers of observations (through 50) lacks four edges, and does not show any colliders (*Figure 3.14*). A recovered rudimentary pattern lacking one edge and lacking the head of one collider edge is most frequent in datasets beginning at 500 observations (*Figure 3.15*). A recovered rudimentary pattern lacking the head of one collider edge is most frequent in the dataset with 10000 observations (*Figure 3.16*). The correct rudimentary pattern is never recovered. All graphs showing the results for each number of observations can be found in Appendix B, *Figure B.33* to *Figure B.40* (pp. 54-55).

3.6. Figures

Figure 3.1. Percentage correct recoveries of DAG 1a (equal regression coefficients), for each pattern and each number of observations.

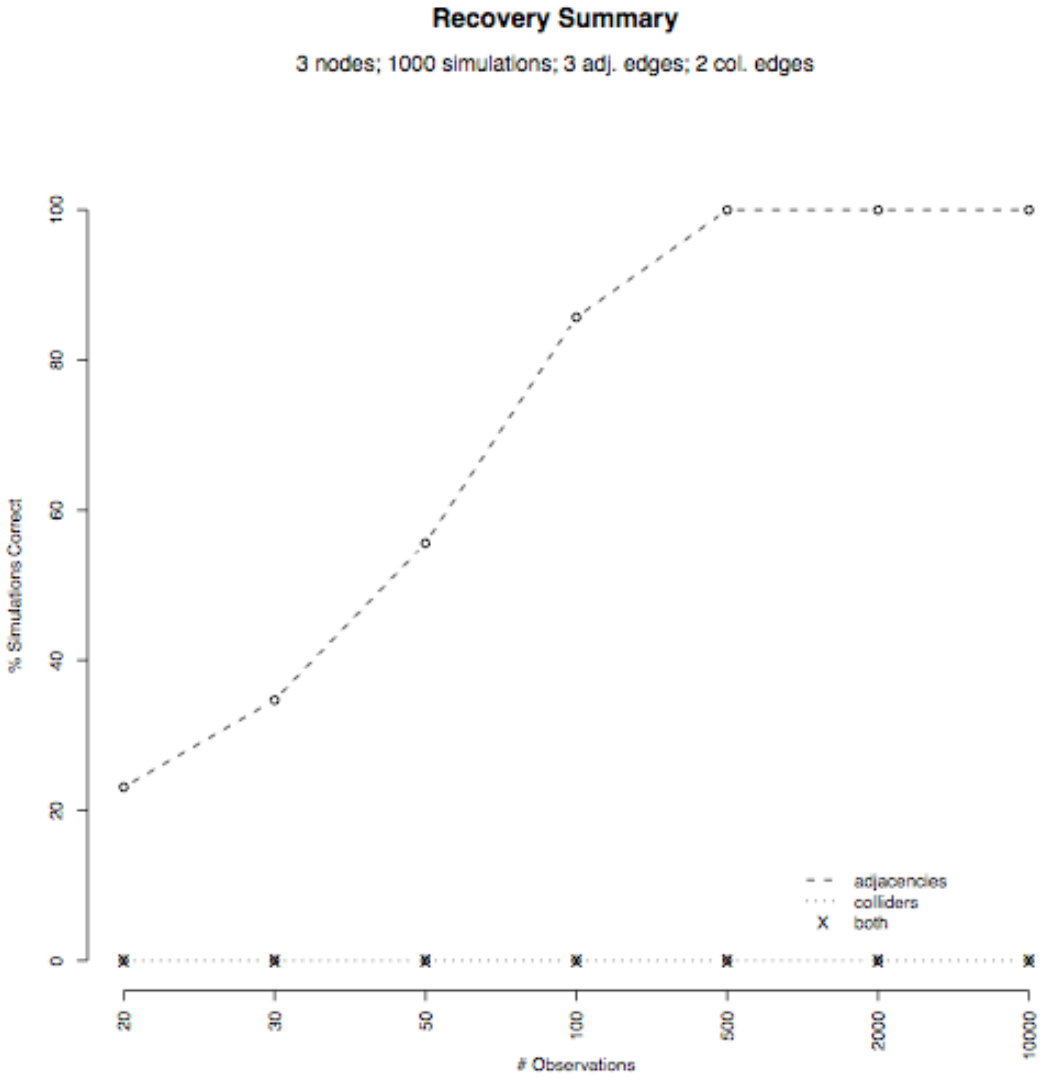
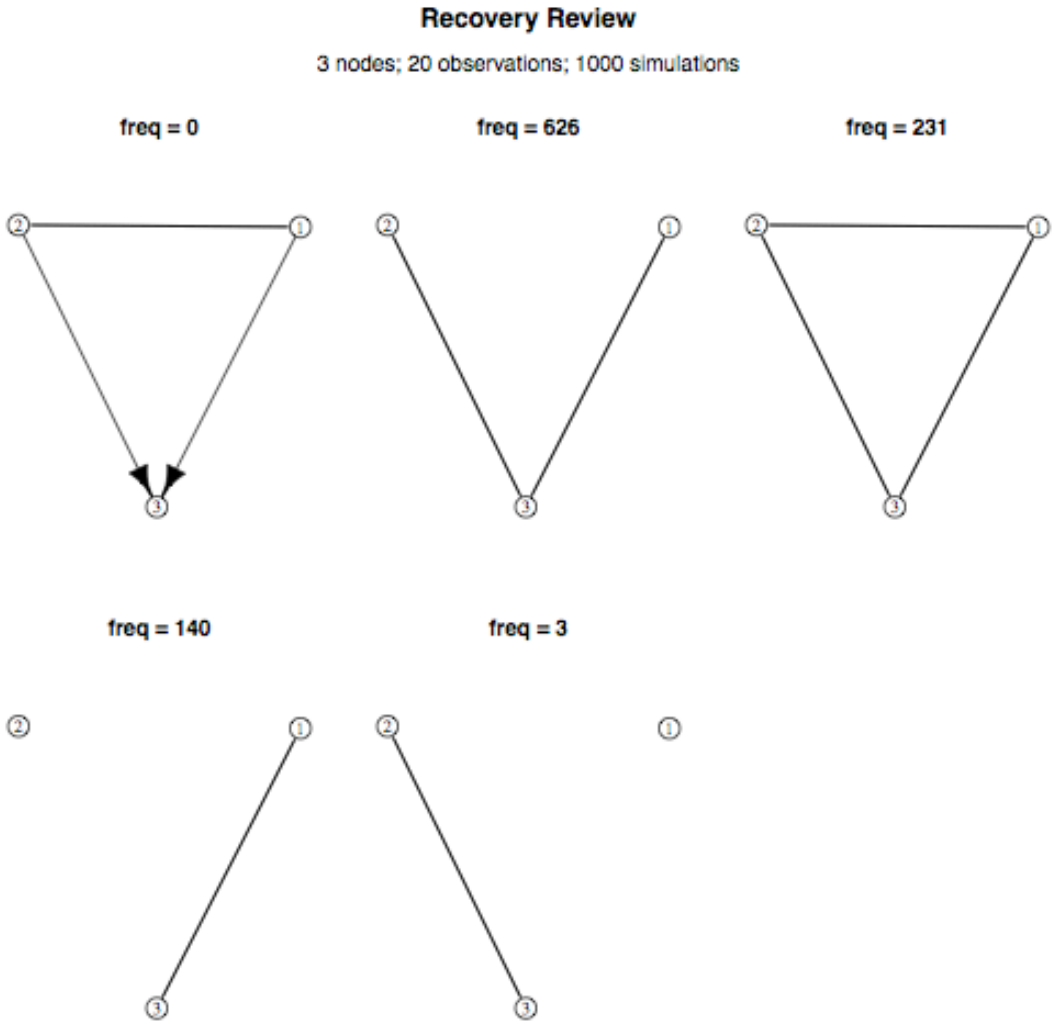
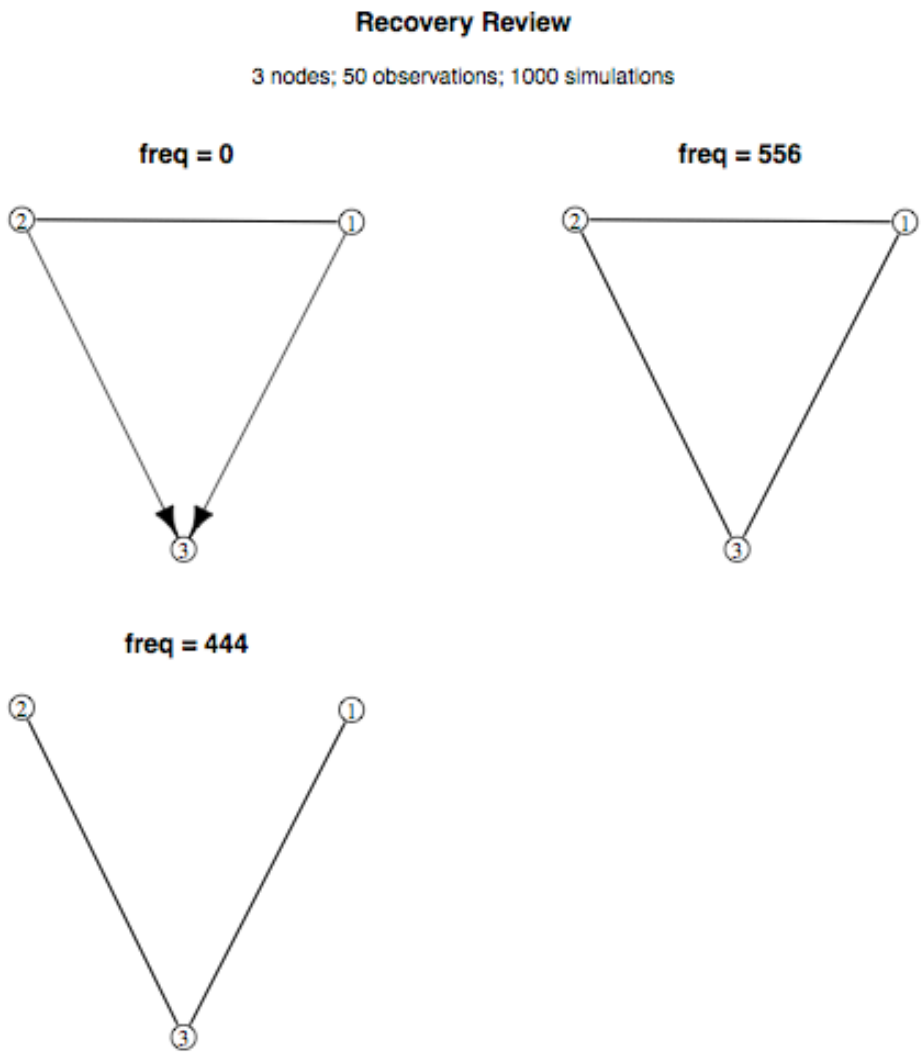


Figure 3.2. Frequencies and graphs of all recovered rudimentary patterns, for DAG 1a (equal regression coefficients) with 20 observations.



Note. Upper-left graph shows correct rudimentary pattern.

Figure 3.3. Frequencies and graphs of all recovered rudimentary patterns, for DAG 1a (equal regression coefficients) with 50 observations.



Note. Upper-left graph shows correct rudimentary pattern.

Figure 3.4. Percentage correct recoveries of DAG 1b (unequal regression coefficients), for each pattern and each number of observations.

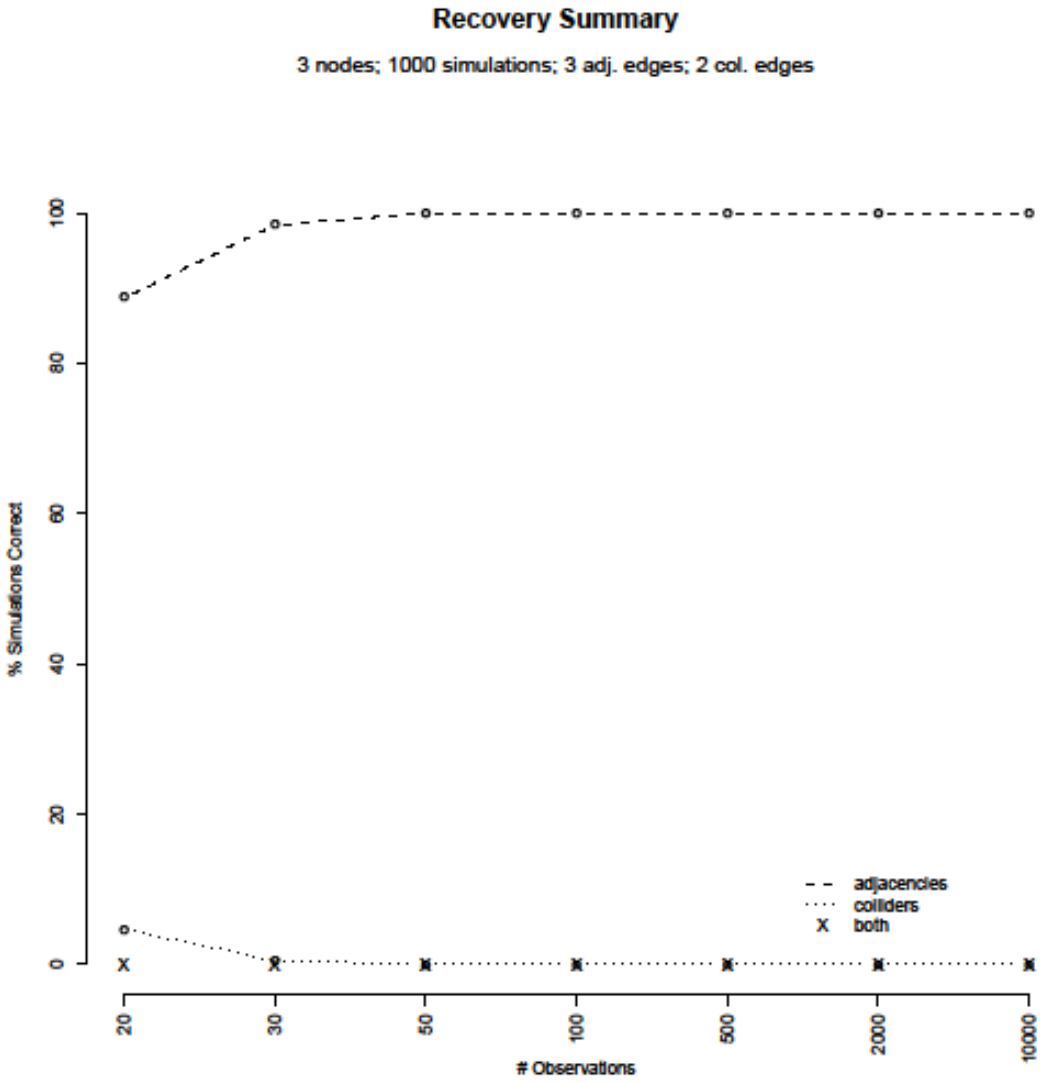
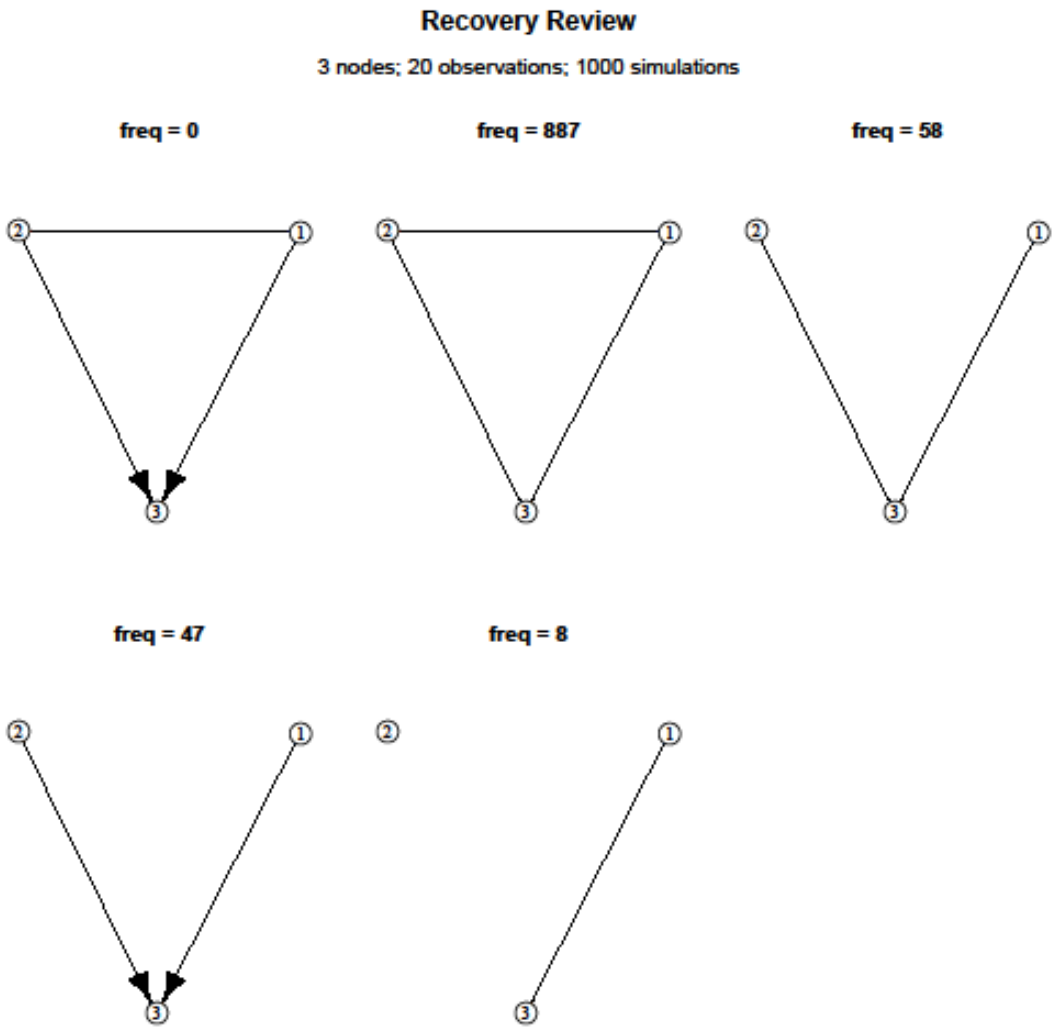


Figure 3.5. Frequencies and graphs of all recovered rudimentary patterns, for DAG 1b (unequal regression coefficients) with 20 observations.



Note. Upper-left graph shows correct rudimentary pattern.

Figure 3.6. Percentage correct recoveries of DAG 2, for each pattern and each number of observations.

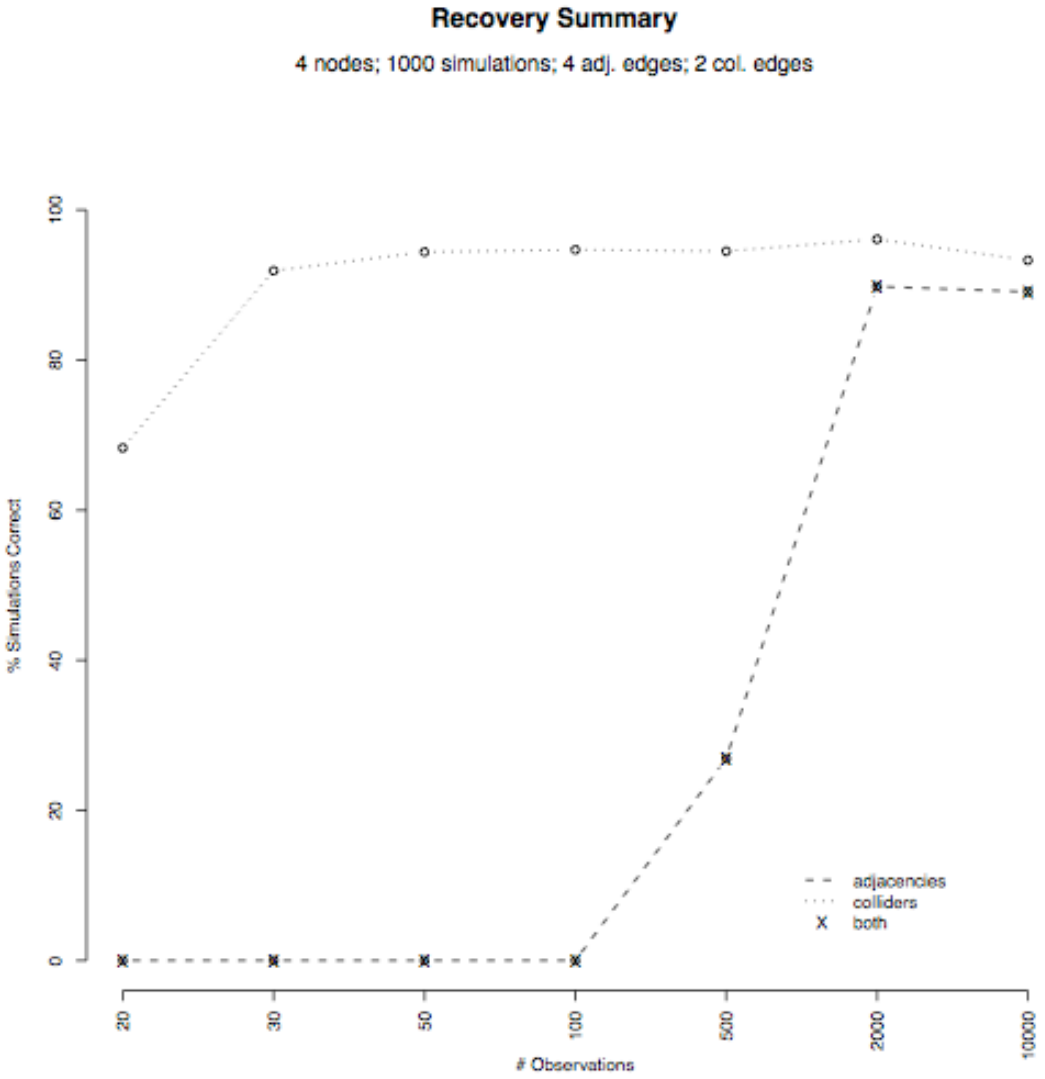
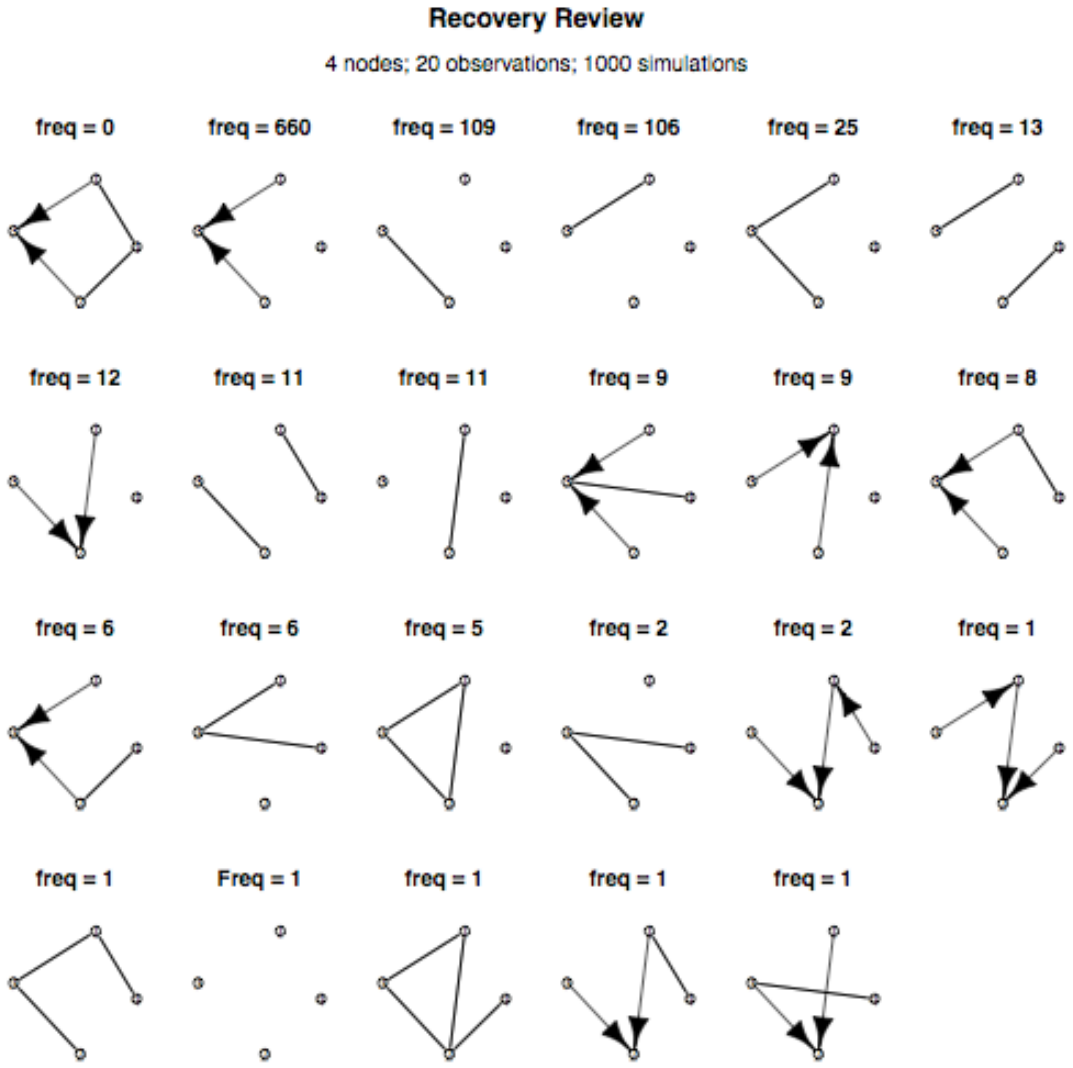
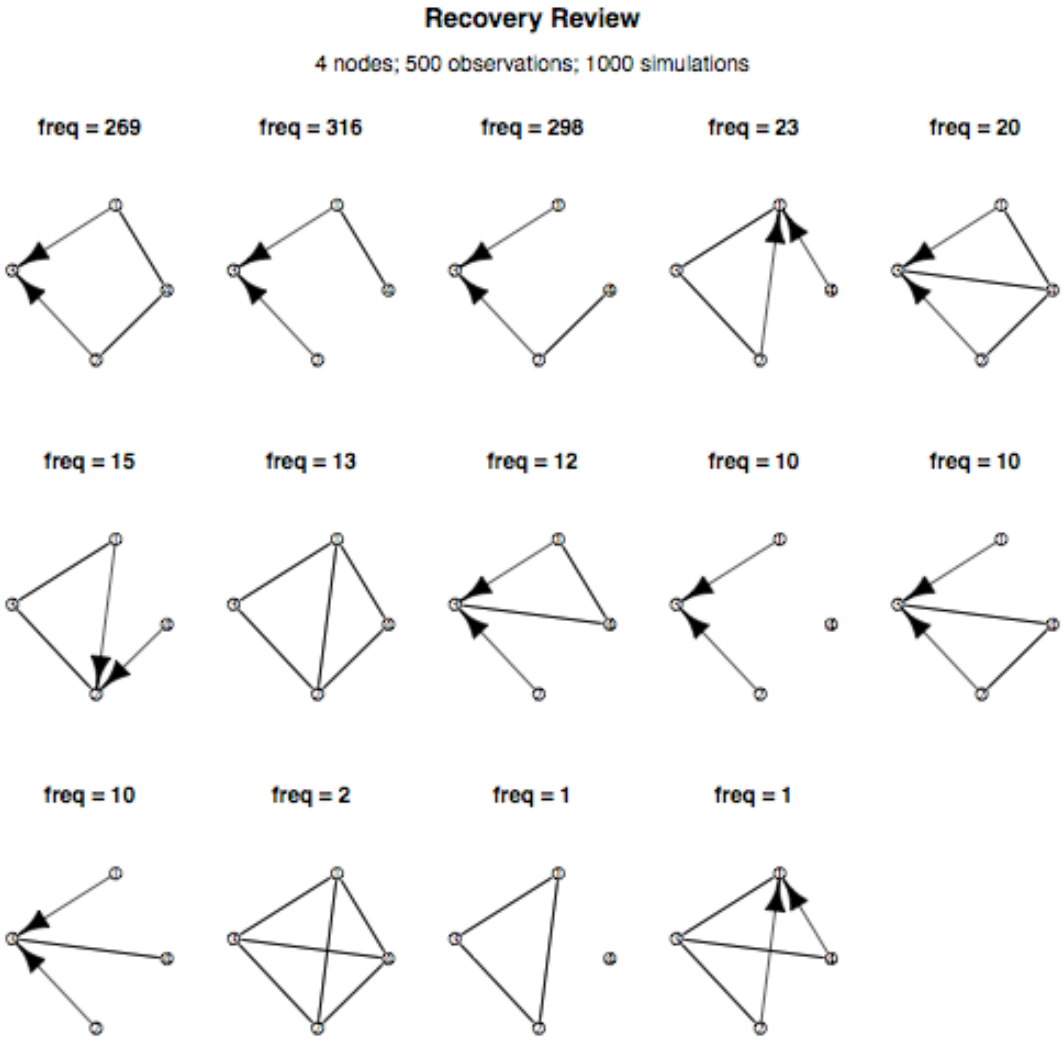


Figure 3.7. Frequencies and graphs of all recovered rudimentary patterns, for DAG 2 with 20 observations.



Note. Upper-left graph shows correct rudimentary pattern.

Figure 3.8. Frequencies and graphs of all recovered rudimentary patterns, for DAG 2 with 500 observations.



Note. Upper-left graph shows correct rudimentary pattern.

Figure 3.9. Percentage correct recoveries of DAG 3, for each pattern and each number of observations.

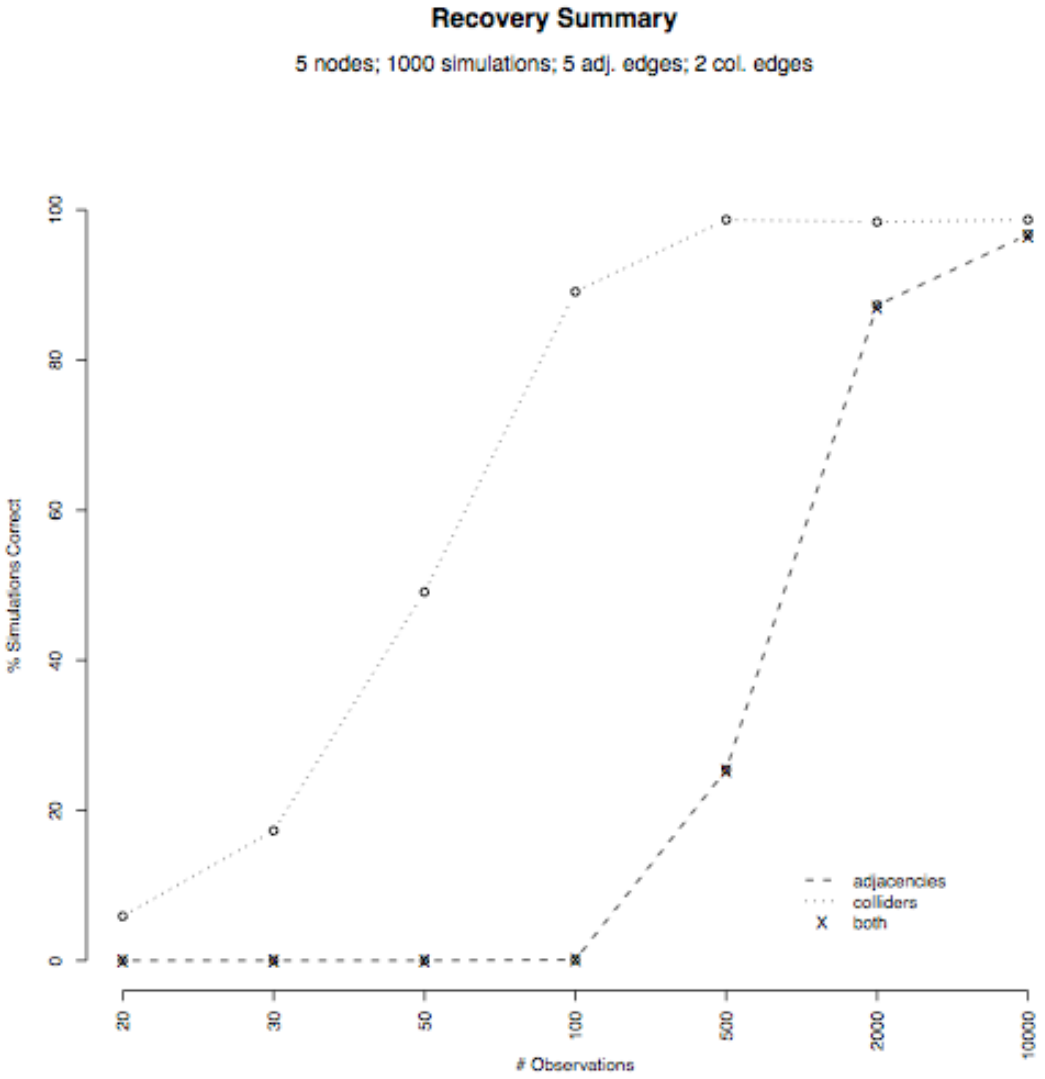
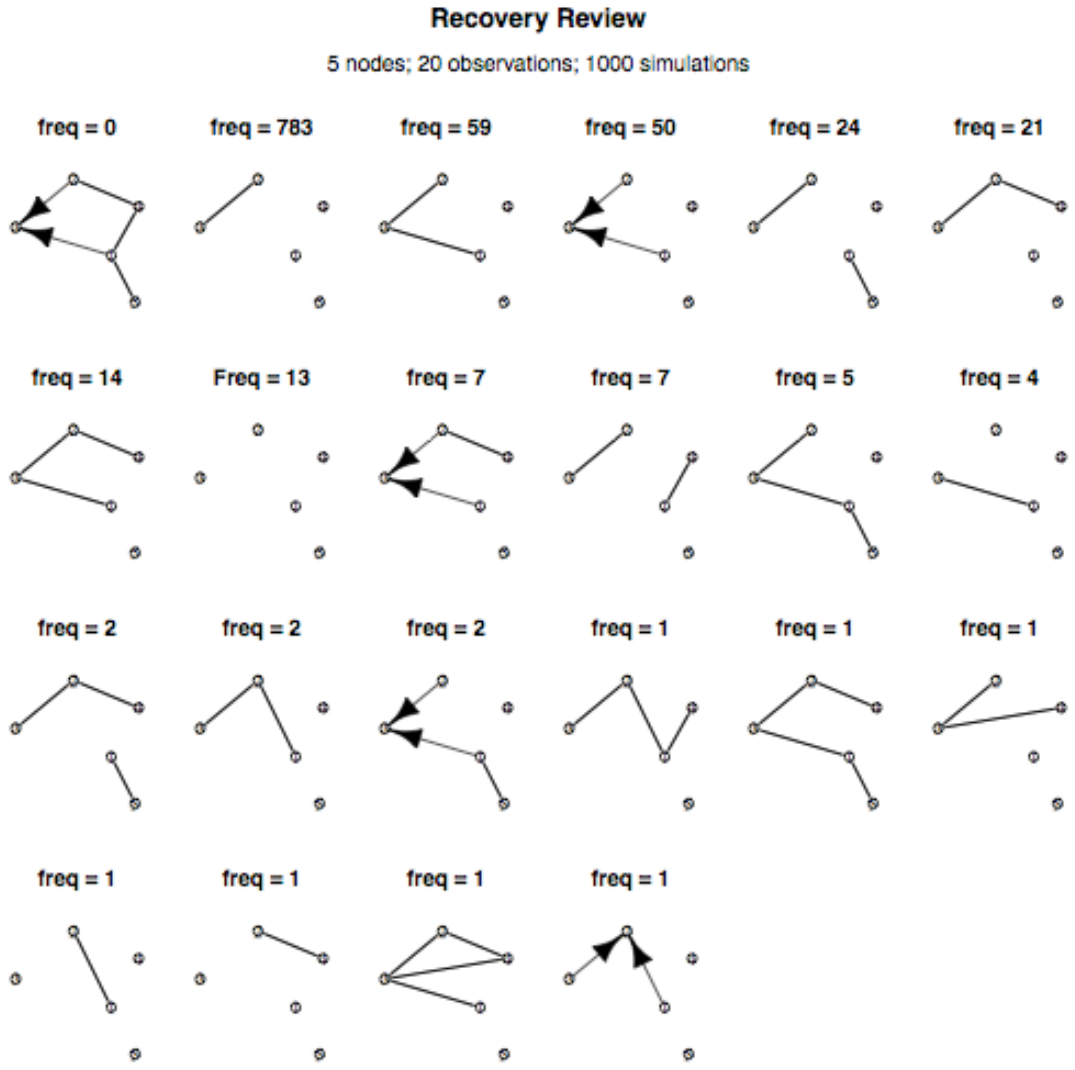
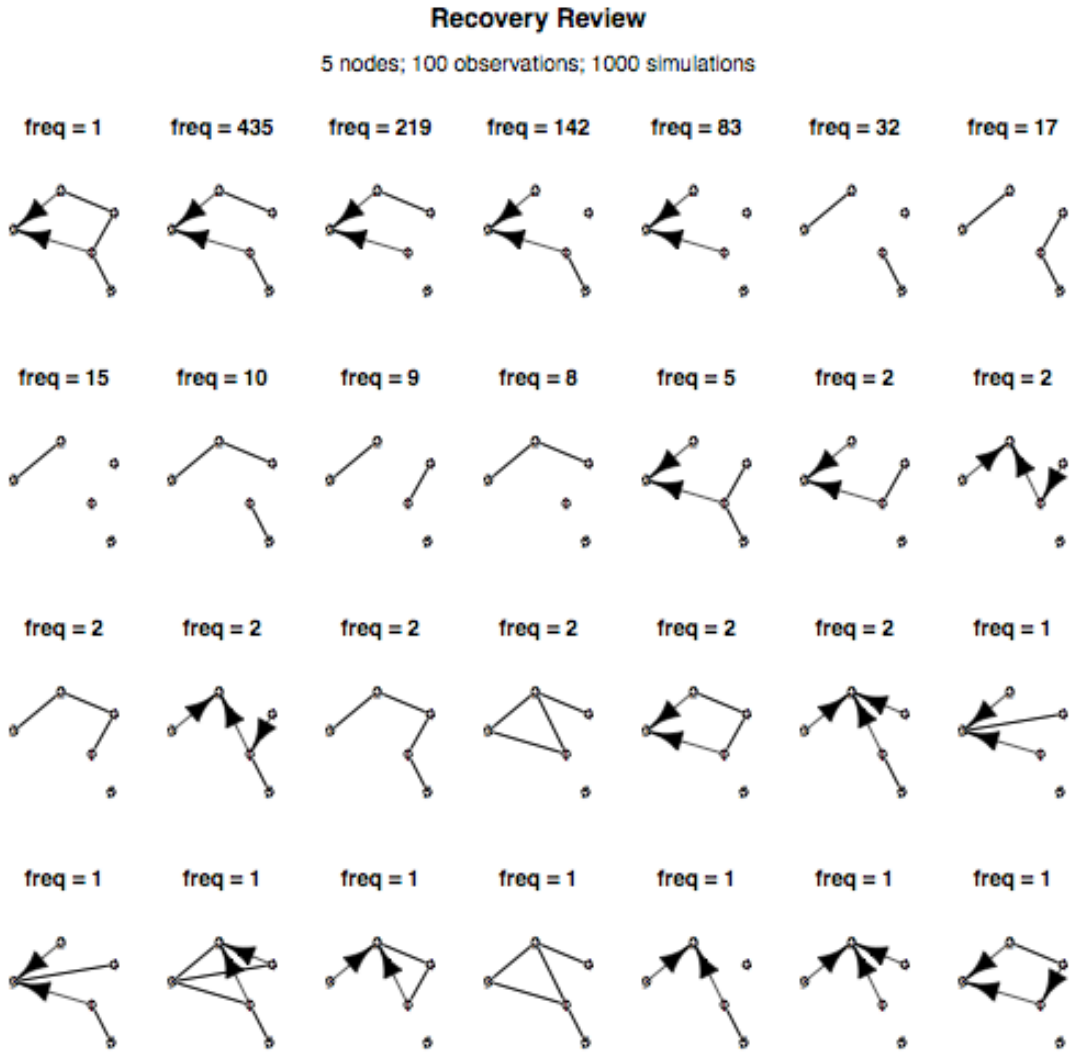


Figure 3.10. Frequencies and graphs of all recovered rudimentary patterns, for DAG 3 with 20 observations.



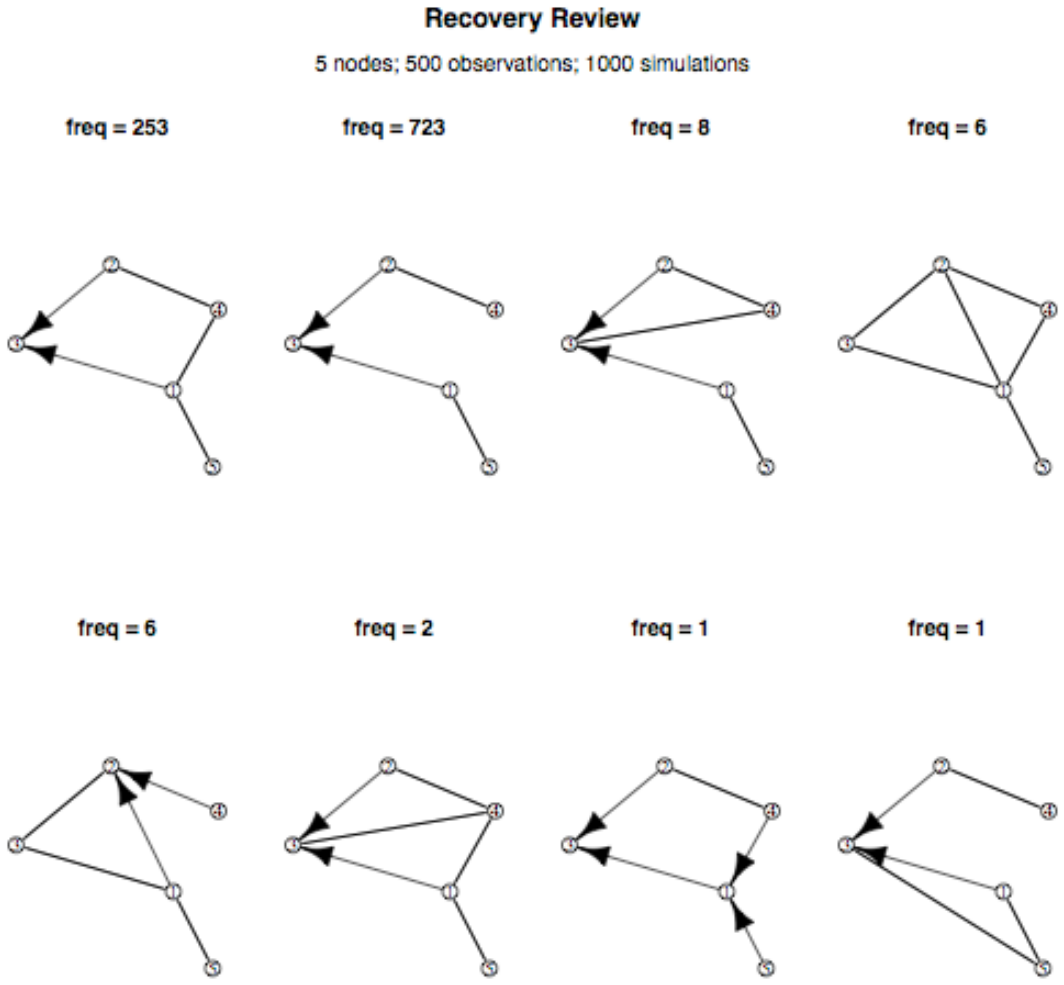
Note. Upper-left graph shows correct rudimentary pattern.

Figure 3.11. Frequencies and graphs of all recovered rudimentary patterns, for DAG 3 with 100 observations.



Note. Upper-left graph shows correct rudimentary pattern.

Figure 3.12. Frequencies and graphs of all recovered rudimentary patterns, for DAG 3 with 500 observations.



Note. Upper-left graph shows correct rudimentary pattern.

Figure 3.13. Percentage correct recoveries of DAG 4, for each pattern and each number of observations.

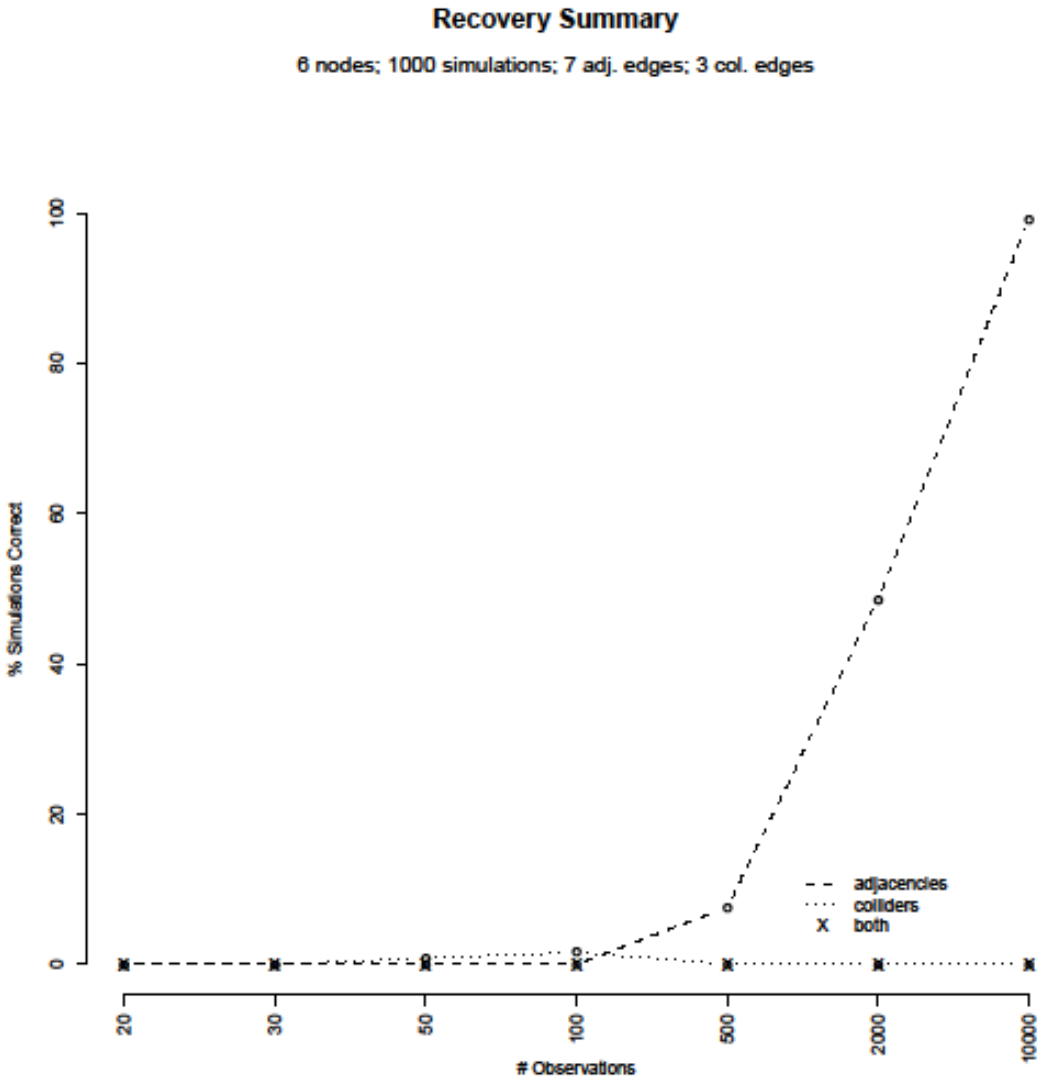
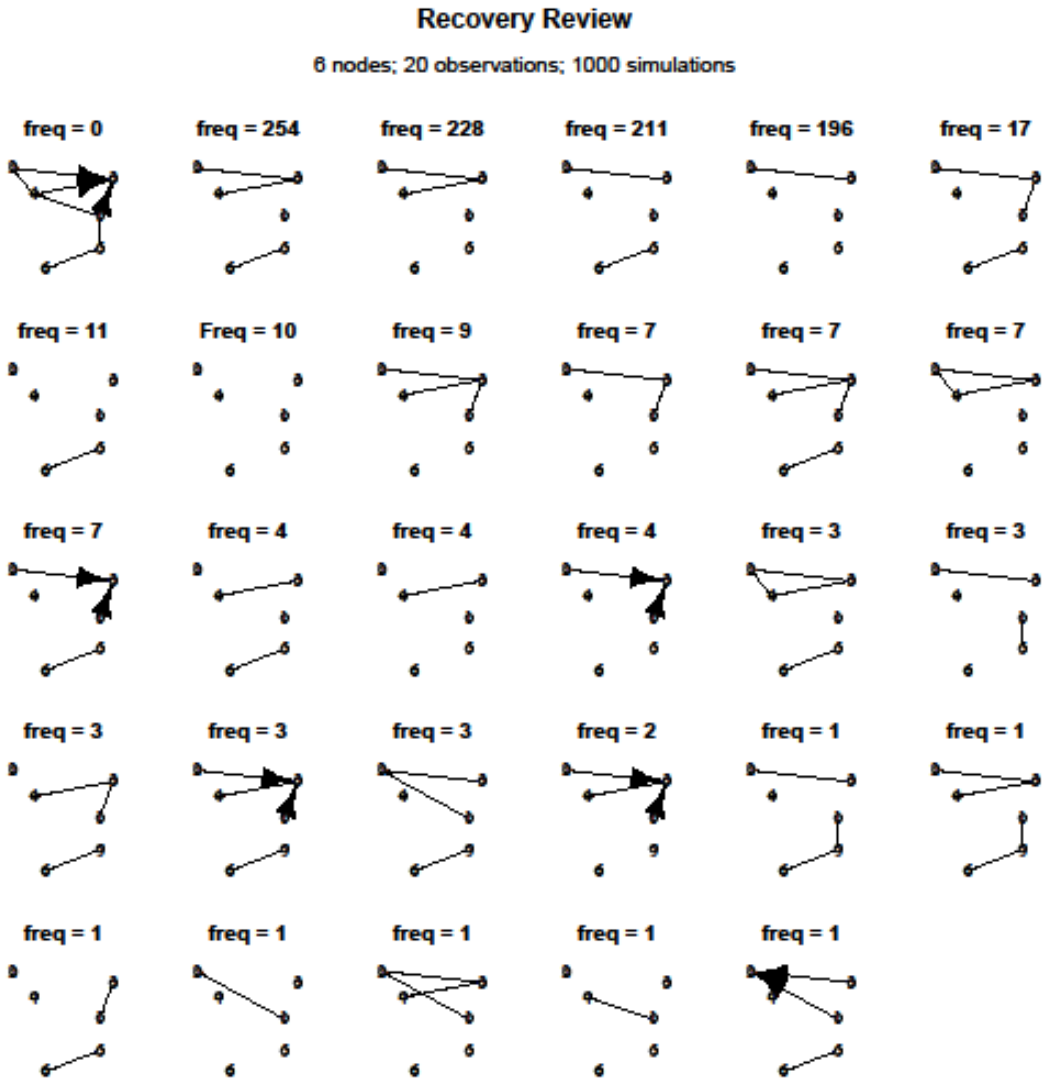
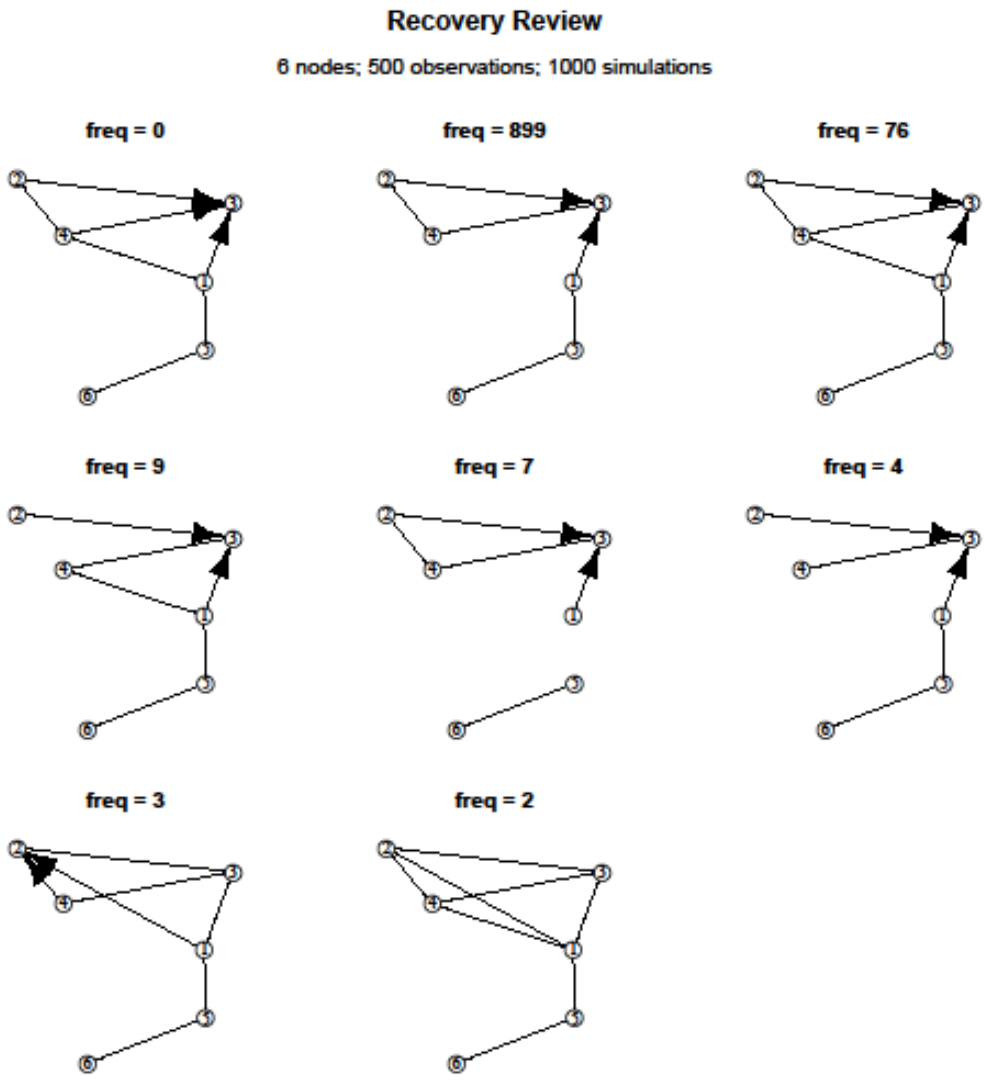


Figure 3.14. Frequencies and graphs of all recovered rudimentary patterns, for DAG 4 with 20 observations.



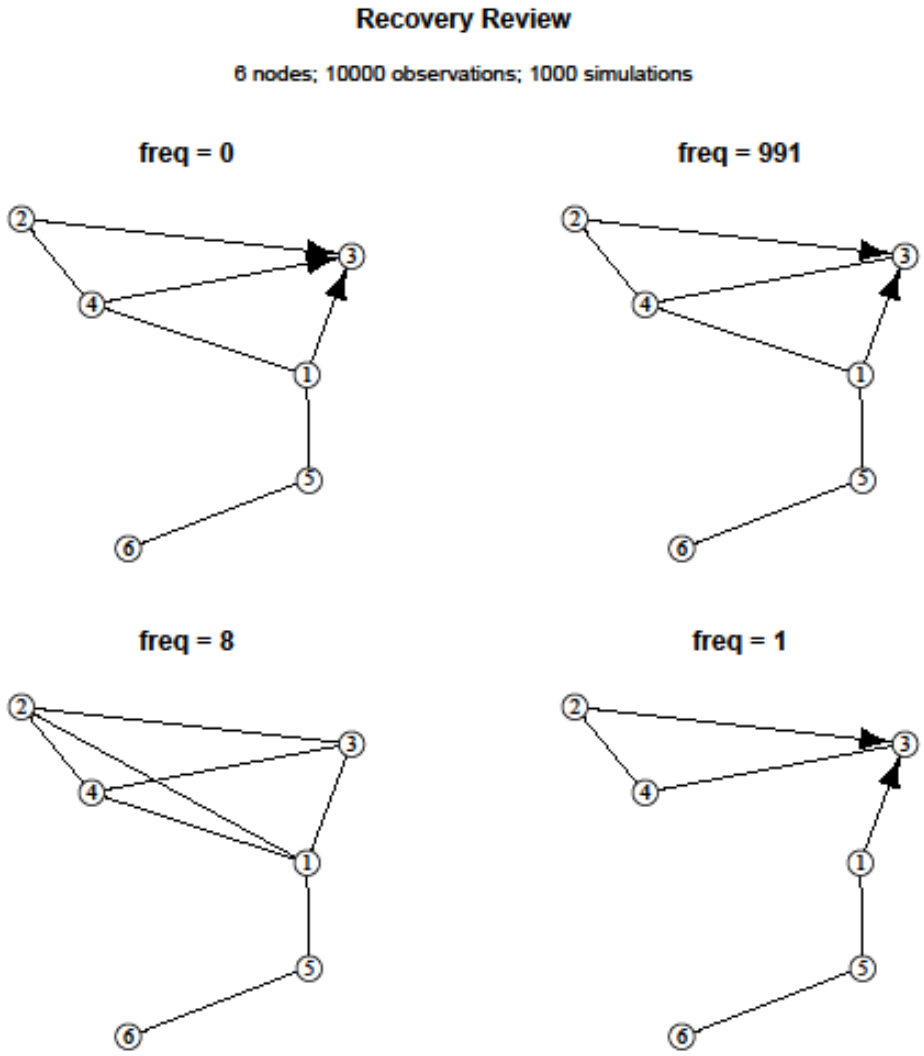
Note. Upper-left graph shows correct rudimentary pattern.

Figure 3.15. Frequencies and graphs of all recovered rudimentary patterns, for DAG 4 with 500 observations.



Note. Upper-left graph shows correct rudimentary pattern.

Figure 3.16. Frequencies and graphs of all recovered rudimentary patterns, for DAG 4 with 10000 observations.



Note. Upper-left graph shows correct rudimentary pattern.

4. Discussion

4.1. Overall Recovery Capacity

The main goal of this study was to examine how well, and under what circumstances, an automated IC algorithm is able to accurately recover the rudimentary pattern from the underlying DAG of an observational dataset. Four DAGs were created, generating different datasets, and then their rudimentary patterns were recovered using the implemented IC algorithm. The algorithm was expected to recover the rudimentary patterns accurately, given that the datasets contain a large enough number of observations. Generally, the number of observations required for correct inference is a function of the number of variables (nodes). However, an exception was expected for DAGs containing a collider with connected parents. These particular collider edges were expected to be recovered as undirected edges.

The implemented IC algorithm returned promising results. The algorithm is generally able to accurately recover the rudimentary patterns of different DAGs, given a dataset with a large number of observations (e.g. 2000 observations) and containing colliders with disconnected parents. The separately examined adjacency and collider patterns met the expectations. The algorithm accurately recovered the adjacency patterns of all DAGs around 90 to 100 percent of the simulations, beginning at datasets with 20 (DAG 1b) to 10000 (DAG 4) observations. The algorithm accurately recovered the collider patterns of DAGs containing colliders with disconnected parents around 95 percent of the simulations, beginning at datasets with 500 observations. Collider patterns of DAGs containing colliders with connected parents were not recovered accurately. The algorithm accurately recovered the rudimentary patterns of DAGs containing colliders with disconnected parents around 90 percent of the simulations, beginning at datasets with 2000 observations. The recovered rudimentary patterns of the DAGs containing colliders with connected parents on the other

hand, were not recovered accurately. As expected, colliders with connected parents were recovered as undirected edges.

4.2. Specific Findings per DAG

The implemented IC algorithm generally seems to perform as expected. However, examining the DAGs individually, a more detailed account can be given of its behavioral characteristics and light is shed at some remarkable findings.

4.2.1. Findings DAG 1

DAG 1 was specifically created to examine how the algorithm would recover a causal network containing a collider with connected parents. The DAG was embedded in two different datasets, one with equal regression coefficients and one with unequal regression coefficients. Following the rules of the IC algorithm, the recovered rudimentary patterns would be expected to incorrectly consist of three undirected edges, as the connected parents would block the algorithm from being able to find the collider. The IC algorithm indeed requires the collider to be unshielded, in order to be able to recover it (Appendix A, p. 41, *Function 6*). Moreover, the rudimentary pattern was expected to be recovered more easily from data with unequal regression coefficients, as equal regression coefficients can in some way cancel out the effects, reducing the ability to recover the dependency between the parents of the collider. As such, it finally was expected that the dependency between the parents of the collider is less easily recovered from the data with equal regression coefficients. These expectations were all confirmed in the datasets.

In the simulations of the dataset with equal regression coefficients and small numbers of observations, the pattern with only the two undirected collider edges was indeed recovered most frequently. This finding is not readily explained, but has to do with the equal regression coefficients, as was noted above. In this particular situation, the

dependency between the parents of the collider is removed by conditioning on the collider. The edge between the parents is therefore not shown, nor are the colliders recovered, as this behavior is opposite from what would be expected when following the IC algorithm. This particular situation stresses the fact that it is highly necessary to further study the influence of the regression coefficients on the recovery of the rudimentary patterns.

4.2.2. Findings DAG 2

DAG 2 was created to disconnect the parents of the collider in DAG 1 and examine the effect on the recovery of the DAG. Following the rules of the IC algorithm, the recovered rudimentary patterns would be expected to correctly consist of two directed collider edges and two undirected common cause edges. This expectation was confirmed in the datasets with large numbers of observations. In the simulations with low numbers of observations however, both edges of the common cause did not get recovered. The inability of the IC package to recover these edges is probably caused by a lack of power.

Interestingly, quite often non-existing undirected edges are present in the recovered rudimentary patterns (e.g. 11 percent in datasets with 10000 observations (Appendix B, p. 51, *Figure B.24*)). These edges are most frequently found between the outer nodes of a chain or common cause structure, and are mainly recovered from datasets with large numbers of observations. The large numbers of observations cause conditional independencies, such as between the outer nodes of a chain or common cause structure, to be found just significantly dependent. This dependency causes the algorithm to recover the incorrect adjacency.

4.2.3. Findings DAG 3

DAG 3 is an adaptation of DAG 2. However, a chain replaced the common cause structure in DAG 2, and an extra edge was introduced at the beginning of the chain. Following the rules

of the IC algorithm, the recovered rudimentary patterns would be expected to correctly consist of two directed collider edges and three undirected chain edges. Alike the results of DAG 2, this expectation was confirmed in the datasets with large numbers of observations. In the simulations with low numbers of observations however, the recovered rudimentary patterns were very much incomplete.

Interestingly, the edge between node one and four remains often unrecovered, especially in the datasets with low to moderate numbers of observations. Unfortunately, the author is currently unaware of the reason for this finding. Further research into such specific deficiencies is encouraged, as it may provide insight in the characteristics of the implemented IC algorithm.

4.2.4. Findings DAG 4

DAG 4 was created to contain three collider edges, all headed at a single node. The DAG thus contained three pairs of collider edges, of which two had connected parents. Following the rules of the IC algorithm, the recovered rudimentary patterns would be expected to consist of two of the three collider edges. This expectation was confirmed in the datasets with large numbers of observations. In the simulations with low numbers of observations however, both edges of the common cause did not get recovered.

Again, one of the undirected edges from the rudimentary pattern remains strikingly often unrecovered, especially in the datasets with low to moderate numbers of observations. Unfortunately, the author is currently unaware of the reason for this finding.

4.3. Current Limitations and Opportunities

A point of concern remains the inability of the IC algorithm to recover colliders with connected parents. However, a solution to this problem is available. It can be tested whether conditioning on the presumed collider significantly increases (or decreases) the

(partial) correlation between its parents. If this is the case, the possible collider can by those means be discovered. How this solution holds in different situations and for different data remains to be studied however.

Another current limitation is the inability to recover the complete DAG. The algorithm can only recover colliders; chains and common causes can thus not be recovered. This limitation can be partly solved by extending the implemented IC algorithm with the third step of the IC algorithm (Pearl, 2000), i.e. the complete pattern. The third step prescribes to orient as many of the undirected edges of the rudimentary pattern, according to two simple rules. First, no new collider structures may be created and second, the acyclicity of the pattern must be conserved. Although this might not always return the full DAG, it certainly is an important improvement.

Thirdly, the assumption of acyclicity is a perilous one to keep up for many psychological data. Cyclicity is explicitly involved in for instance premenstrual dysphoric disorder (Cohen, Miner, Brown, Freeman, Halbreich, Sundell, & McCray, 2002), bulimia nervosa (Cooper, Morrison, Bigman, Abramowitz, Levin, & Krener, 1988) and bipolar disorder (Calabrese & Delucchi, 1990). Fortunately this limitation can be bridged for time series data (Spirtes, Glymour, & Scheines, 1993). In case of mutually dependent variables, acyclicity can be restored by directing the edges over time points, in such a way that for instance variable one at time point one is directed at variable two at time point two, and variable two at time point one is directed at variable one at time point two.

Finally, it is noteworthy that the results of this study cannot be readily generalized. First of all, in this study simulated observational data was used, as opposed to time series data that comes available from electronic diaries. The results of this study thus apply to observational data only. This means that the number of observations from time series shall be required to be more, depending on the autocorrelation structure of the data. Moreover, in this study only a few specific DAGs were examined, with fixed distributions and regression

coefficients. When using the implemented IC algorithm for real data, it is therefore judicious to analyze a similar simulated dataset prior to analyzing the real data. This should give insight in the algorithm's behavioral characteristics for the specific dataset.

Despite the above limitations, of which most can and will be solved in the future, many opportunities lie ahead as well. The implemented IC algorithm currently has an explorative nature, but can be made suitable for the testing of hypothesized DAGs in a dataset. And many more opportunities, such as the recovery of the regression coefficients, make the IC algorithm a viable and promising technique for the recovery of causal structures from observational data.

5. References

Agresti, A, & Franklin, C. (2007). *Statistics: The art and science of learning from data.* New Jersey: Pearson Prentice Hall.

Calabrese, J. R., & Delucchi, G. A. (1990). Spectrum of efficacy of valproate in 55 patients with rapid-cycling bipolar disorder. *American Journal of Psychiatry, 147,* 431–434.

Cohen, L. S., Miner, C., Brown, E., Freeman, E. W., Halbreich, U., Sundell, K., & McCray, S. (2002). Premenstrual daily fluoxetine for premenstrual dysphoric disorder: A placebo-controlled, clinical trial using computerized diaries. Obstetrics and Gynecology, 100, 435–444.

Cook, T. D., Scriven, M., Coryn, C. L. S., & Evergreen, S. D. H. (2009). Contemporary thinking about causation in evaluation: A dialogue with Tom Cook and Michael Scriven. *American Journal of Evaluation, 00,* 1-13.

Cooper, J. L., Morrison, T. L., Bigman, O. L., Abramowitz, S. I., Levin, S., & Krener, P. (1988). Mood changes and affective disorder in the bulimic binge-purge cycle. *International Journal of Eating Disorders,* 7, 469–474.

Fisher, R.A. (1971). *The Design of Experiments* (9th ed.). New York: Macmillan Pub Co.

Karimi, K., & Hamilton, H. J. (2000). Finding temporal relations: Causal bayesian networks vs. C4.5. In Z. W. Ras & S. Ohsuga (Eds.), *Foundations of intelligent systems* (pp. 266-273). Berlin: Springer-Verlag.

Pearl, J. (2000). *Causality: models, reasoning, and inference.* New York: Cambridge University Press.

Shimizua, S., Kanoa, Y. (2008). Use of non-normality in structural equation modeling: Application to direction of causation*. Journal of Statistical Planning and Inference, 138,* 3483-3491.

Spirtes, P., and Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review, 9,* 62–72.

Spirtes, P., Glymour, C., and Scheines, R. (1993). *Causation, Prediction, and Search*. New York, N.Y.: Springer-Verlag. 2nd Edition, MIT Press (2001).

Verma, T. S., & Pearl, J. (1991). Equivalence and synthesis of causal models. *Uncertainty in Artificial Intelligence, 6,* 255–268.

Wild, B., Eichler, M., Friederich, H. C., Hartmann, M., Zipfel, S., & Herzog, W. (2010). A graphical vector autoregressive modelling approach to the analysis of electronic diary data. *BMC Medical Research Methodology, 10,* 1–16.

6. Appendix A: Pseudo Code

Function 1. Covariance matrix.

Function:	iccov(data)
Input:	A data matrix (rows=time points or participants, columns=variables).
Output:	A covariance matrix.
Purpose:	Generate a matrix showing the covariances between the variables in the data. The row and column names state the variables.

Pseudocode:

```
# Calculate the covariance matrix from the data.
# Attach informative attributes to the covariance matrix.
# Return covariance matrix.
```

Function 2. Correlation matrix.

Function:	iccor(data)
Input:	A data matrix (rows=time points or participants, columns=variables).
Output:	A (partial) correlations matrix.
Purpose:	Generate a matrix showing all (partial) correlations of the variables in the data. The row names state the possible pairs of variables. The first column shows the zero order correlations of the pairs and the subsequent columns show the partial correlations. The column names, except for the first column, state which variables are partialled out.

Pseudocode:

```
# Calculate the number of possible pairs of variables.
# Calculate the number of possible subsets (sizes zero to the number of variables minus two)
# of the set of variables.
# Generate an empty correlation matrix, with the number of rows equal to the number of
# pairs and the number of columns equal to one plus the number of subsets.
# Collect the pairs in a list.
# Collect the subsets in a list.
# Assign the former list to the row names of the correlation matrix.
# Assign the latter list to the column names of the correlation matrix.
# For each location in the correlation matrix
        # If a variable in the column matches a variable in the row
                # Assign NA to the current location in the correlation matrix.
        # Else
                # Try to calculate the (partial) correlation between the two variables in the
                # row name, given the variable(s) in the column name.
        # If an error occurs during try
                # Add one to an error log.
        # Else
```

```
                # Assign the (partial) correlation to the current location in the correlation
                # matrix.
# If the number of errors in the error log is different from zero
        # Return an error message.
# Else
        # Return the correlation matrix.
```

Function 3. Significance matrix.

Function:	icsig(data)
Input:	A data matrix (rows=time points or participants, columns=variables).
Output:	A significance matrix of the (partial) correlations.
Purpose:	Generate a matrix showing all p-values of the (partial) correlations. The row names state the possible pairs of variables. The first column shows the p-values for the zero order correlations of the pairs and the subsequent columns show the p-values for the partial correlations. The column names, except for the first column, state which variables are partialled out.

Pseudocode:

```
# Calculate the number of possible pairs of variables.
# Calculate the number of possible subsets (sizes zero to the number of variables minus two)
# of the set of variables.
# Generate an empty significance matrix, with the number of rows equal to the number of
# pairs and the number of columns equal to one plus the number of subsets.
# Collect the pairs in a list.
# Collect the subsets in a list.
# Assign the former list to the row names of the significance matrix.
# Assign the latter list to the column names of the significance matrix.
# For each location in the significance matrix
        # If a variable in the column matches a variable in the row
                # Assign NA to the current location in the significance matrix.
        # Else
                # Try to calculate the p-value of the correlation between the two variables in
                # the row name, given the variable(s) in the column name.
        # If an error occurs during try
                # Add one to an error log.
        # Else
                # Assign the p-value to the current location in the significance matrix.
# If the number of errors in the error log is different from zero
        # Return an error message.
# Else
        # Return the significance matrix.
```

Function 4. Adjacency pattern.

Function: icadj(data, alpha=.05)
Input: A data matrix (rows=time points or participants, columns=variables), and the preferred level of α. α signifies the demarcation for which a p-value in the significance matrix is considered significant or not (default is α=.05).
Output: An adjacency matrix (symmetric, undirected).
Purpose: Generate a matrix showing all adjacencies detected in the data, given the (partial) correlations. The row names and column names state the variables. A number one in the adjacency matrix indicates the presence of an adjacency between the two variables. A number zero in the adjacency matrix indicates the absence of an adjacency between the two variables.

Pseudocode:

```
# Generate an empty adjacency matrix, with the number of rows and number of columns
# equal to the number of variables.
# Assign the value zero to the diagonal.
# For each location in the upper-right half of the adjacency matrix
    # If one of the corresponding (partial) correlations in the significance matrix is not <=
    # α
        # Assign the value zero to the current location in the adjacency matrix.
        # Assign the value zero to the exact opposite of the current location in the
        # adjacency matrix.
    # Else
        # Assign the value one to the current location in the adjacency matrix.
        # Assign the value one to the exact opposite of the current location in the
        # adjacency matrix.
# Return the adjacency matrix.
```

Function 5. Unshielded collider list.

Function: icuns(data, alpha=.05)
Input: A data matrix (rows=time points or participants, columns=variables), and the preferred level of α. α signifies the demarcation for which a p-value in the significance matrix is considered significant or not (default is α=.05).
Output: A list of unshielded colliders.
Purpose: Generate a list showing all unshielded colliders detected in the data, given the (partial) correlations. Each position in the list shows an unshielded collider. The first variable in each position represents the middle node in the unshielded collider. The second and third variable are its parents.

Pseudocode:

```
# Generate from the adjacency matrix, a matrix with for every variable the number of
# adjacent variables.
# Count the number of variables with two or more adjacencies.
```

If the number of variables with two or more adjacencies equals zero
    # Return an error message (“no (unshielded) colliders”).
Else
    # Generate a matrix with, for every variable with two or more adjacencies, the
    # number of adjacencies.
    # Collect all the variables that satisfy the rule of two or more adjacencies, and their
    # adjacent variables, in a list.
    # Paste the variables that satisfy the above rule to all the combinations of two of
    # their adjacent variables.
If the list of unshielded colliders is empty
    # Return an error message (“no (unshielded) colliders”).
Else
    # Return the list of unshielded colliders and note that the first variables are the
    # middle nodes in the unshielded colliders.

Function 6. Collider pattern.

Function:	iccol(data, alpha=.05)
Input:	A data matrix (rows=time points or participants, columns=variables), and the preferred level of α. α signifies the demarcation for which a p-value in the significance matrix is considered significant or not (default is α=.05).
Output:	A collider matrix (asymmetric, directed).
Purpose:	Generate a matrix showing all colliders detected in the data, given the (partial) correlations. The row names and column names state the variables. A number one in the collider matrix indicates the presence of a directed edge from the row variable to the column variable. A number zero in the adjacency matrix indicates the absence of a directed edge from the row variable to the column variable.

Pseudocode:
Generate a collider matrix filled with zero’s, with the number of rows and number of
columns equal to the number of variables.
Count the number of unshielded colliders.
If zero unshielded colliders
    # Return collider matrix filled with zeros.
Else
    # For each unshielded collider
        # Paste the two variables that are adjacent to the unshielded collider.
        # If at each location of the significance matrix, where the unshielded collider
        # is partialled out of the correlation between the pair of variables
        # mentioned above, the p-value is not <= α
            # Do nothing.
        # Else

```
                    # Assign the value one to the collider matrix, in the column
                    # corresponding to the unshielded collider and the row
                    # corresponding to the first variable it is adjacent to.
                    # Assign the value one to the collider matrix, in the column
                    # corresponding to the unshielded collider and the row
                    # corresponding to the second variable it is adjacent to.
# Return the collider matrix.
```

Function 7. Rudimentary pattern.

Function:	icrud(data, alpha=.05)
Input:	A data matrix (rows=time points or participants, columns=variables), and the preferred level of α. α signifies the demarcation for which a p-value in the significance matrix is considered significant or not (default is α=.05).
Output:	A rudimentary matrix (asymmetric, directed and undirected).
Purpose:	Generates a matrix showing all adjacencies and colliders detected in the data, given the (partial) correlations. The row names and column names state the variables. A number one with, on the exact opposite position in the matrix, a number one as well, indicates the presence of an adjacency between the two variables. A number one with, on the exact opposite position in the matrix, a number zero, indicates the presence of a directed edge from the row variable to the column variable.

Pseudocode:

```
# Duplicate adjacency matrix and use duplicate as rudimentary matrix.
# Count the number of unshielded colliders.
# If zero unshielded colliders
        # Return the adjacency matrix.
# Else
        # For each unshielded collider
                # Paste the two variables that are adjacent to the unshielded collider.
                # If at each location of the significance matrix, where the unshielded collider
                # is partialled out of the correlation between the pair of variables
                # mentioned above, the p-value is not <= α
                        # Do nothing.
                # Else
                        # Assign the value one to the collider matrix, in the column
                        # corresponding to the unshielded collider and the row
                        # corresponding to the first variable it is adjacent to.
                        # Assign the value one to the collider matrix, in the column
                        # corresponding to the unshielded collider and the row
                        # corresponding to the second variable it is adjacent to.
                        # Assign the value zero to the collider matrix, in the column
                        # corresponding to the first variable it is adjacent to and the row
                        # corresponding to the unshielded collider.
```

Assign the value zero to the collider matrix, in the column
corresponding to the second variable it is adjacent to and the row
corresponding to the unshielded collider.
Return the rudimentary matrix.

Function 8. Variables plot.

Function:	plot.var(data)
Input:	A data matrix (rows=time points or participants, columns=variables).
Output:	Plots of the variables over time (if time series data is used).
Purpose:	Generates a matrix structure of three by three, with each cell showing a variable's behavior plotted over time.

Pseudocode:
Generate a matrix layout of three by three and fill its cells by row.
For each variable
    # Plot its behavior over time.
Return the matrix with plots.

Function 9. Adjacency graph.

Function:	plot.adj(data, alpha=.05)
Input:	A data matrix (rows=time points or participants, columns=variables), and the preferred level of α. α signifies the demarcation for which a p-value in the significance matrix is considered significant or not (default is α=.05).
Output:	An adjacency graph (undirected).
Purpose:	Generates a graph from the adjacency matrix. The graph shows the adjacencies detected in the data.

Pseudocode:
Save undirected adjacency graph (package iGraph).
Adjust labels, label color and node- and edge color.
Set seed.
Fixate layout of nodes and edges.
Plot undirected adjacency graph.
Add title to graph.

Function 10. Collider graph.

Function:	plot.col(data, alpha=.05)
Input:	A data matrix (rows=time points or participants, columns=variables), and the preferred level of α. α signifies the demarcation for which a p-value in the significance matrix is considered significant or not (default is α=.05).
Output:	A collider graph.

Purpose: Generates a graph from the collider matrix. The graph shows the colliders detected in the data.

Pseudocode:
Save directed collider graph (package iGraph).
Adjust labels, label color and node- and edge color.
Set seed.
Fixate layout of nodes and edges.
Plot directed collider graph.
Add title to graph.

Function 11. Rudimentary graph.

Function: plot.rud(data, alpha=.05)
Input: A data matrix (rows=time points or participants, columns=variables), and the preferred level of α. α signifies the demarcation for which a p-value in the significance matrix is considered significant or not (default is α=.05).
Output: A rudimentary graph
Purpose: Generates a graph from the rudimentary matrix. The graph shows both the adjacencies and colliders detected in the data.

Pseudocode:
Save directed adjacency graph (package iGraph).
Adjust labels, label color and node- and edge color.
Set seed.
Fixate layout of nodes and edges.
Assign all adjacencies directed edges.
Collect all adjacent variables in a list.
For each cell in the rudimentary matrix
    # If cell contains value one and the exact opposite cell in the matrix contains value
    # one as well
        # Grab index of "place" in adjacencies list, where the row variable is present.
        # Grab index of "place" in adjacencies list, where the column variable is
        # present.
        # "Save" index of "place" where both row and column variables are present.
        # Remove the direction from the list with adjacent variables with this index.
    # Else
        # Do nothing.
Plot rudimentary graph.
Add title to graph.

Function 12. Overview of all graphs.

Function: plot.all(data, alpha=.05)
Input: A data matrix (rows=time points or participants, columns=variables), and the

preferred level of α. α signifies the demarcation for which a p-value in the significance matrix is considered significant or not (default is α=.05).

Output: An adjacency graph, collider graph, rudimentary graph, and complete graph.

Purpose: Give an overview of all graphs.

Pseudocode:

Generate a matrix layout of two by two and fill its cells by row.
Plot adjacency graph.
Plot collider graph.
Plot rudimentary graph.
Plot text “complete graph not yet available”.

7. Appendix B: Simulation results.

Figure B.1. – B.8. Results for DAG 1a, equal regression coefficients.

Figure B.1.

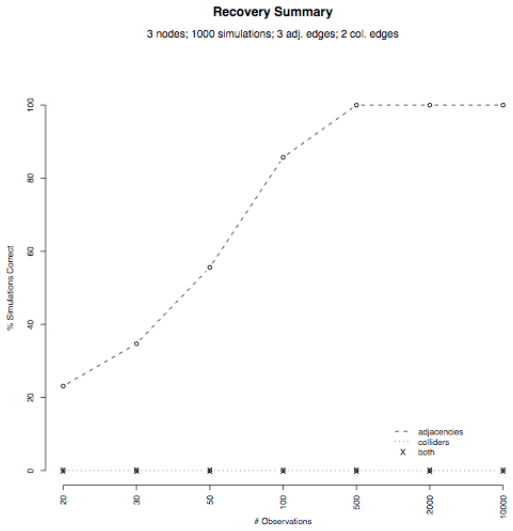


Figure B.2.

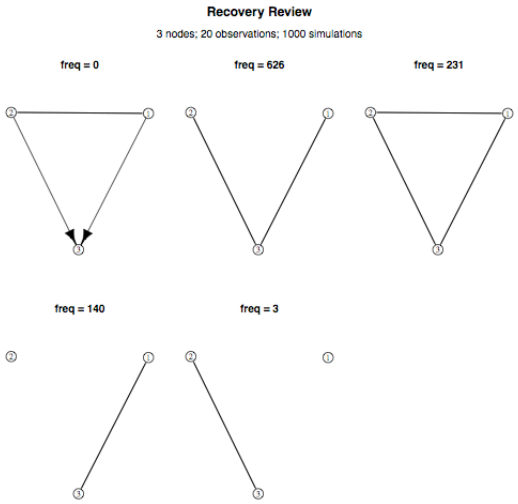


Figure B.3.

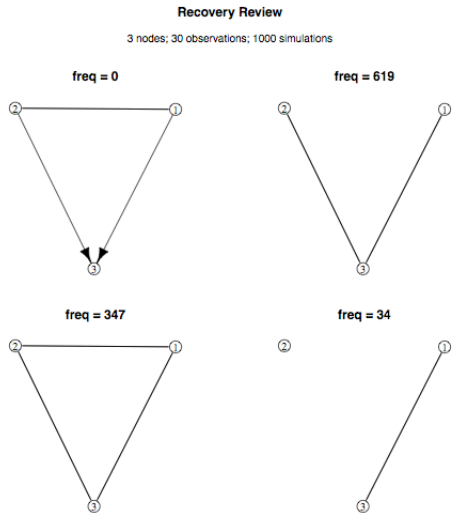


Figure B.4.

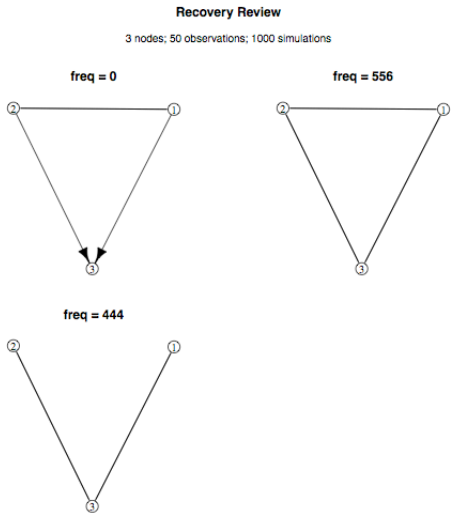


Figure B.5.

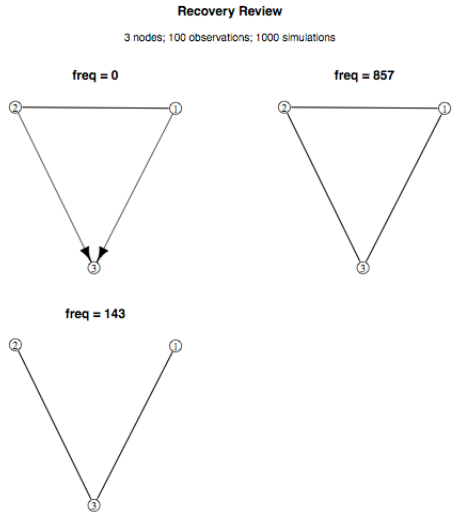


Figure B.6.

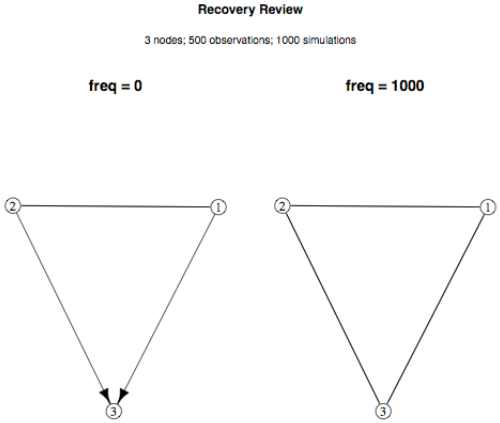


Figure B.7.

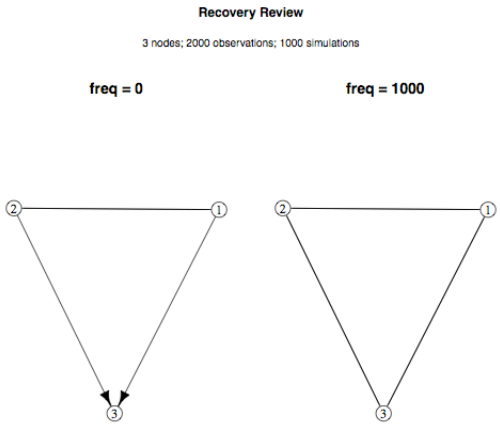


Figure B.8.

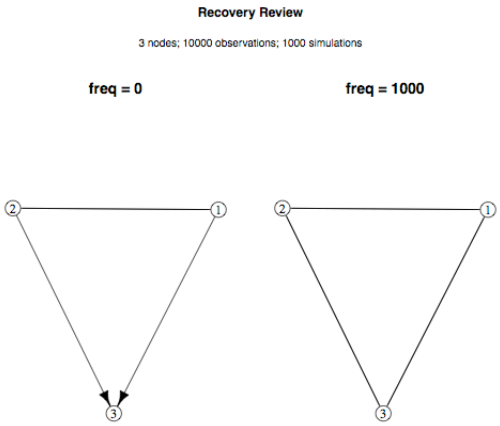


Figure B.9. – B.16. Results for DAG 1b, unequal regression coefficients.

Figure B.9.

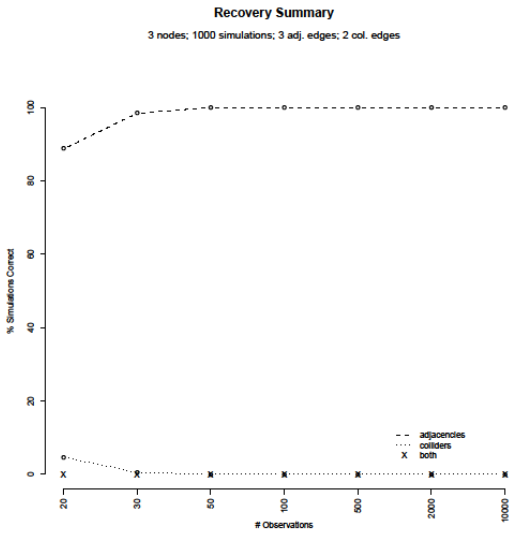


Figure B.10.

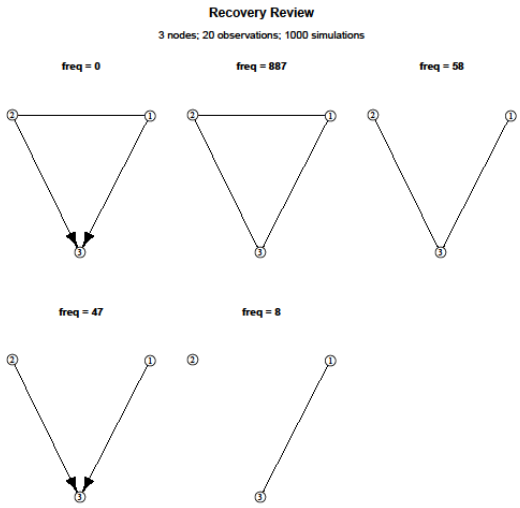


Figure B.11.

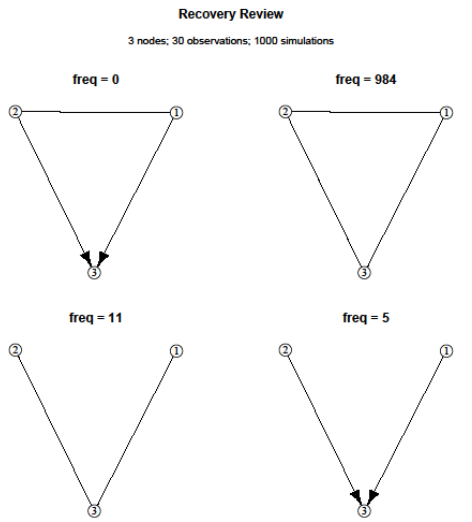


Figure B.12.

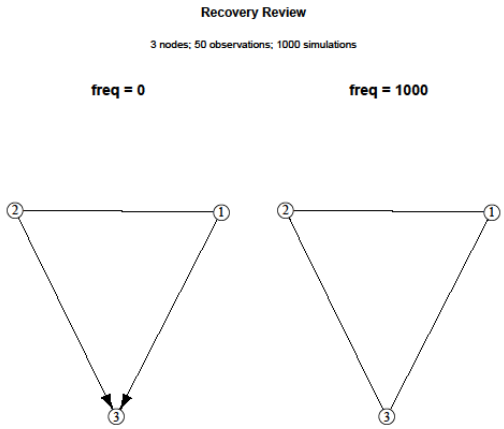


Figure B.13.

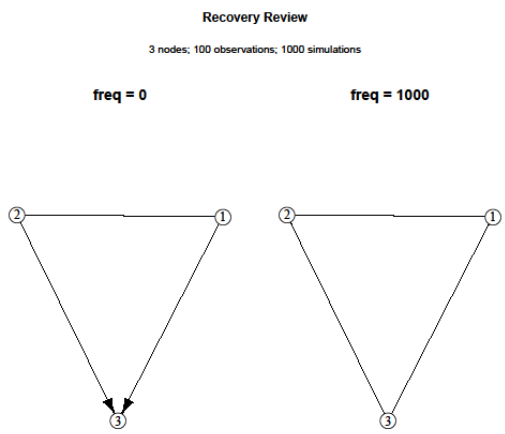


Figure B.14.

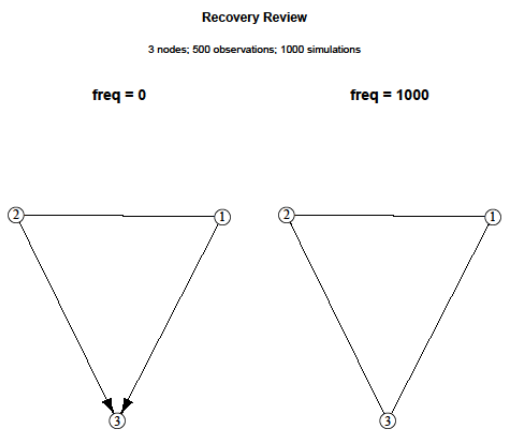


Figure B.15.

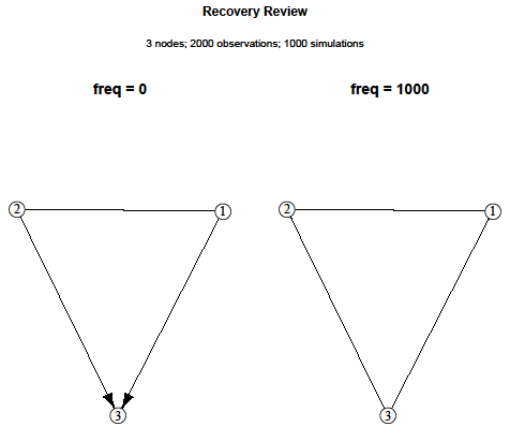


Figure B.16.

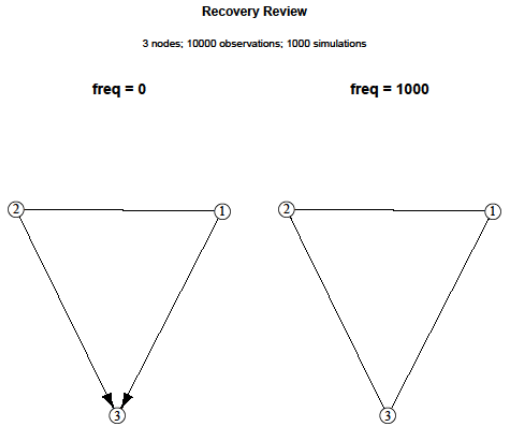


Figure B.17. – B.24. Results for DAG 2.

Figure B.17.

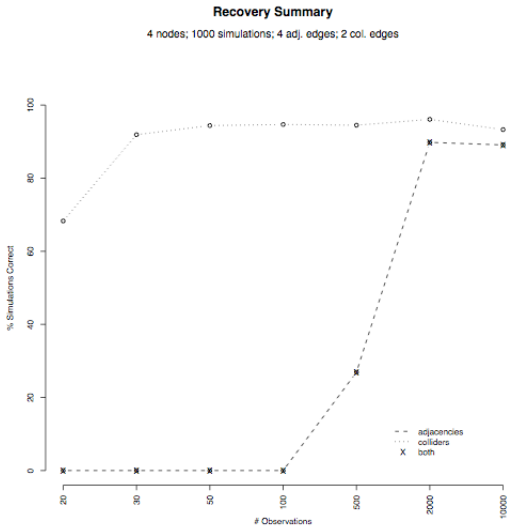


Figure B.18.

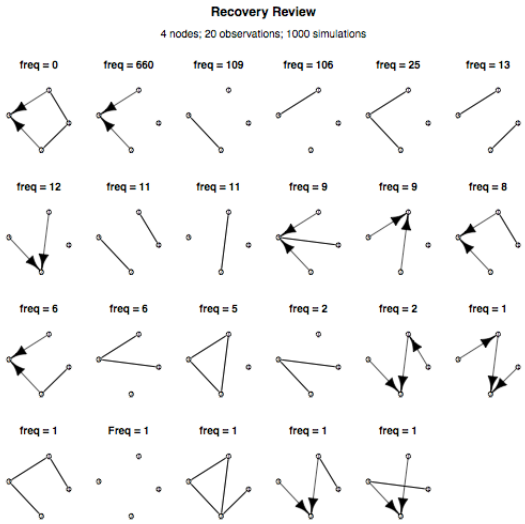


Figure B.19.

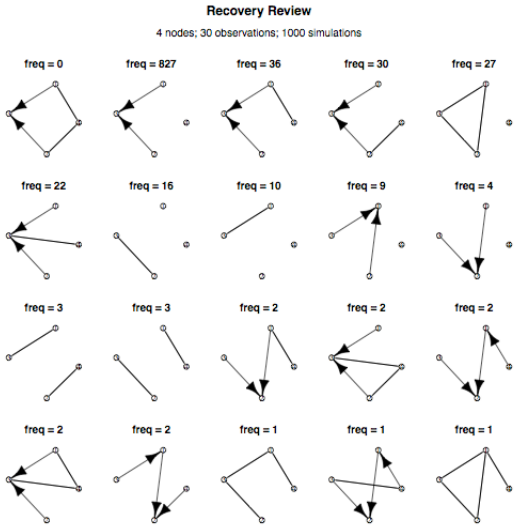


Figure B.20.

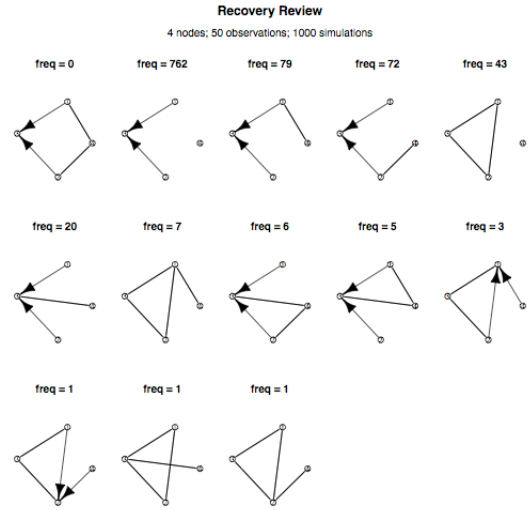


Figure B.21.

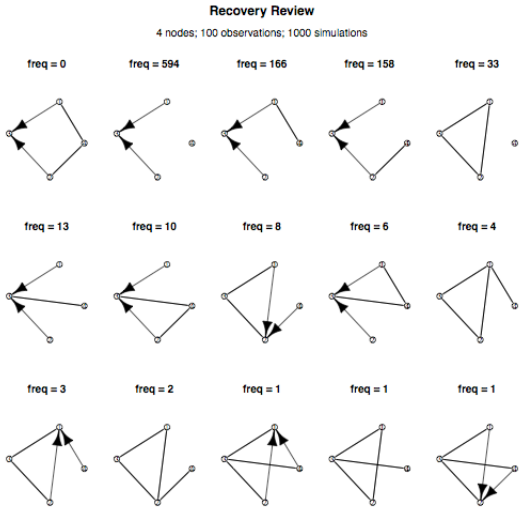


Figure B.22.

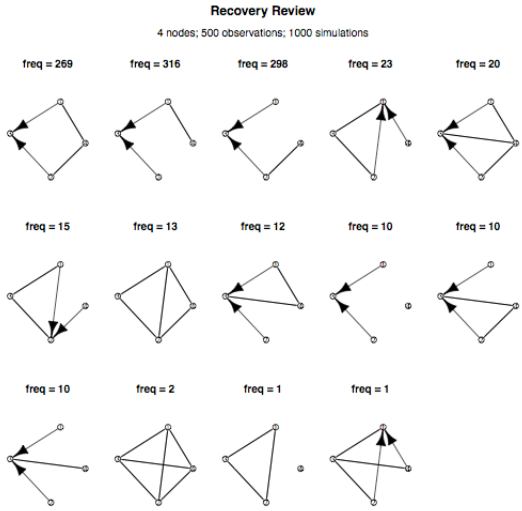


Figure B.23.

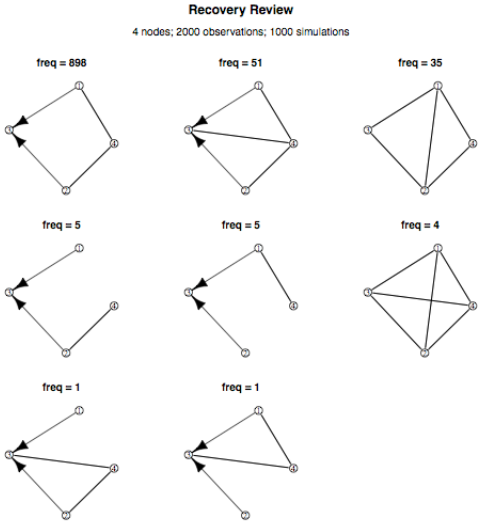


Figure B.24.

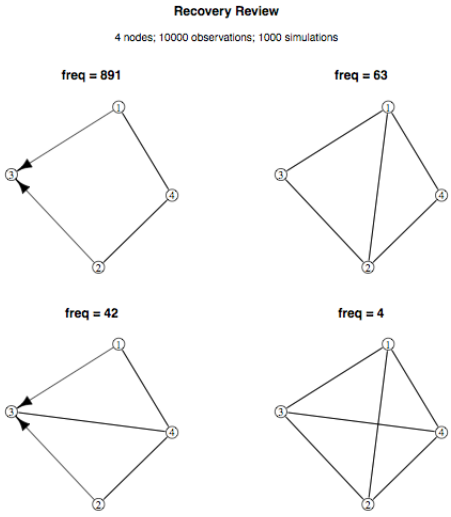


Figure B.25. – B.32. Results for DAG 3.

Figure B.25.

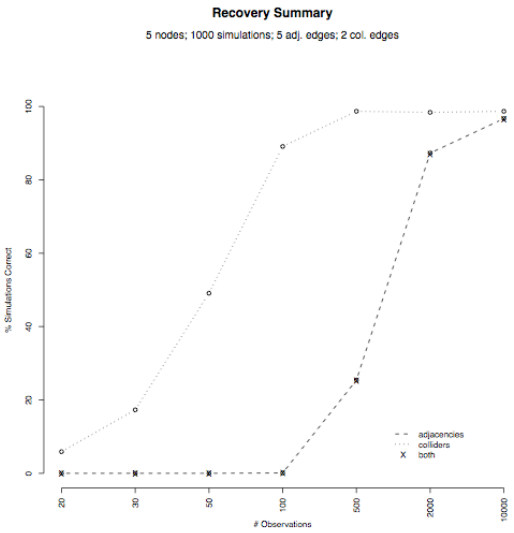


Figure B.26.

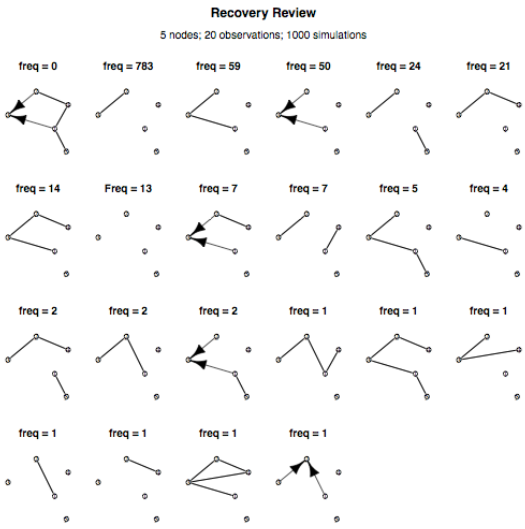


Figure B.27.

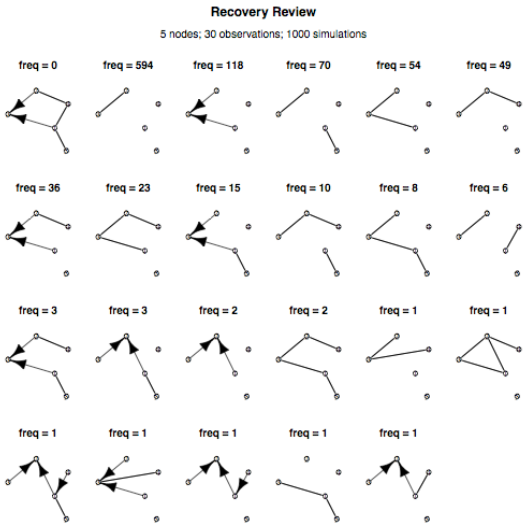


Figure B.28.

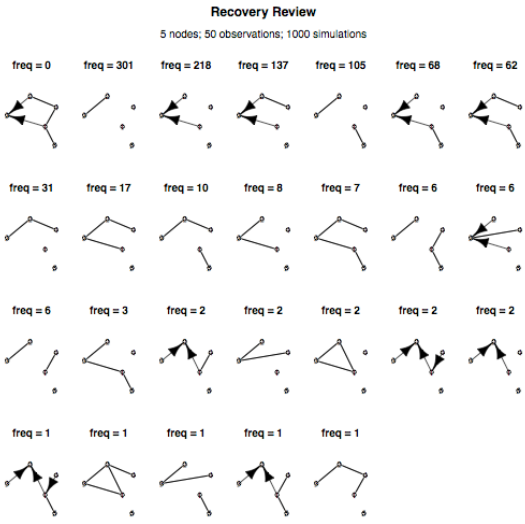


Figure B.29.

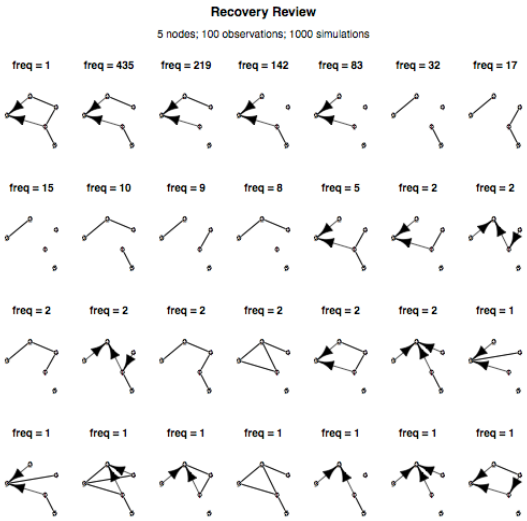


Figure B.30.

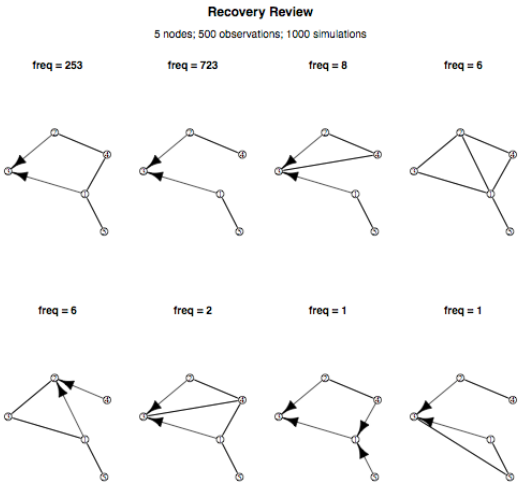


Figure B.31.

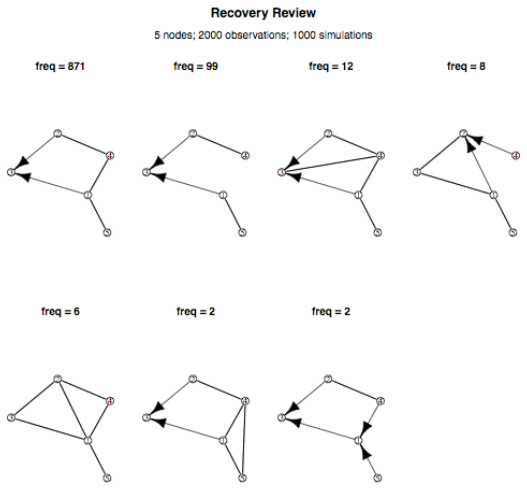


Figure B.32.

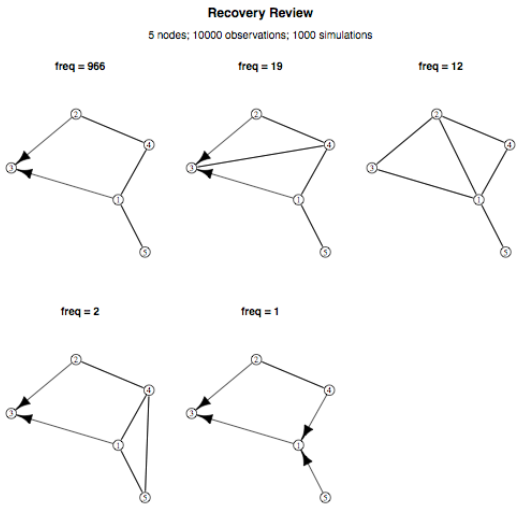


Figure B.33. – B.40. Results for DAG 4.

Figure B.33.

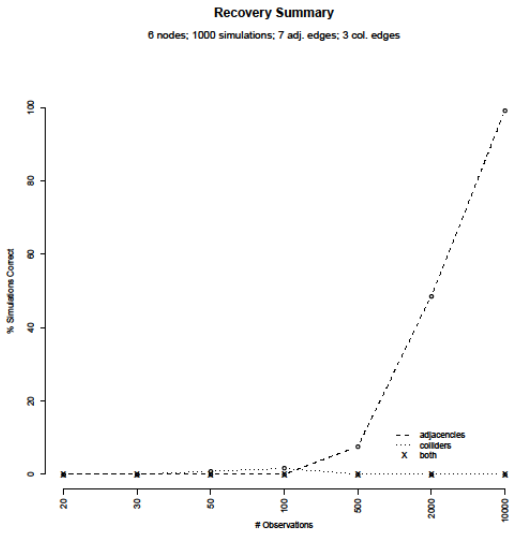


Figure B.34.

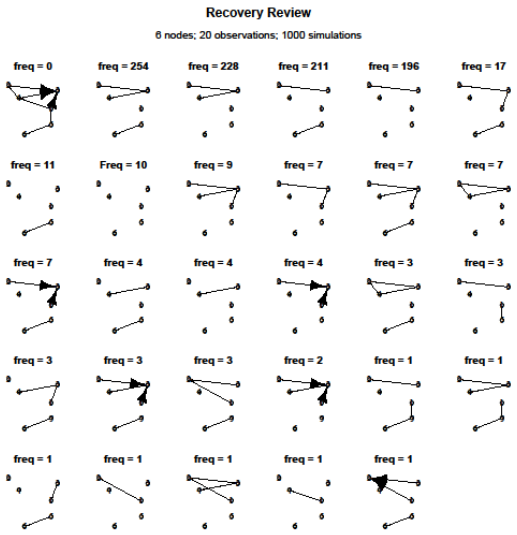


Figure B.35.

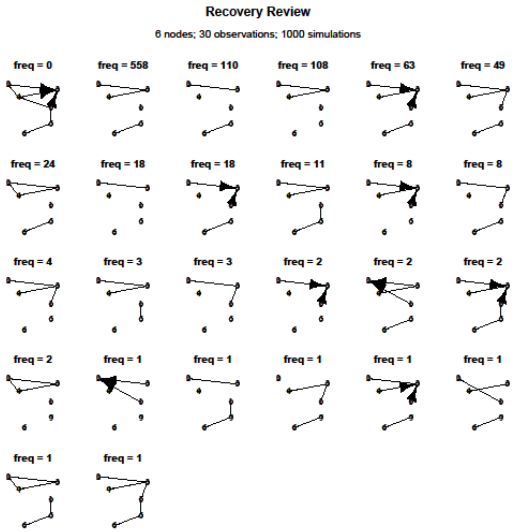


Figure B.36.

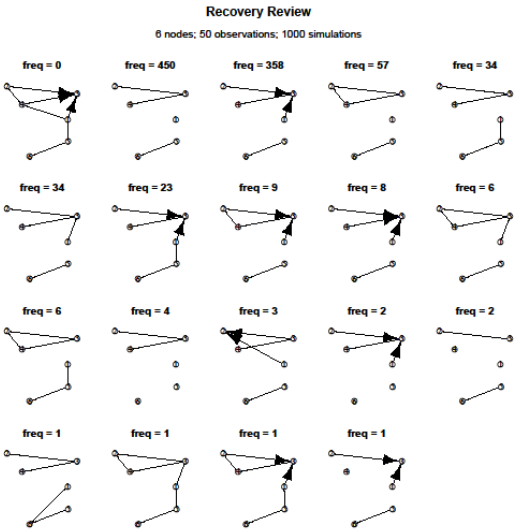


Figure B.37.

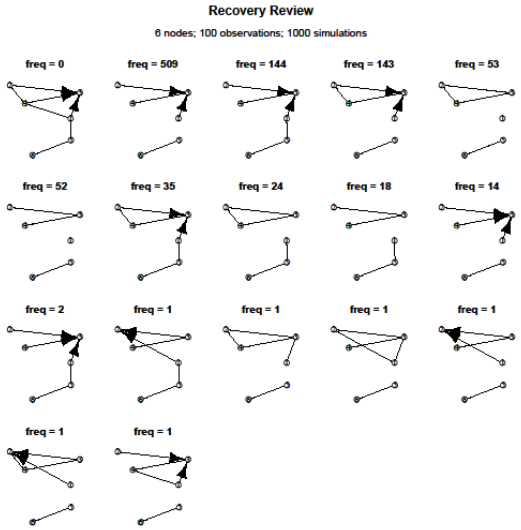


Figure B.38.

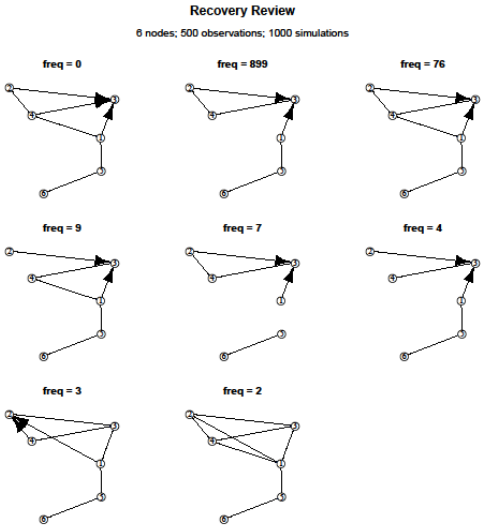


Figure B.39.

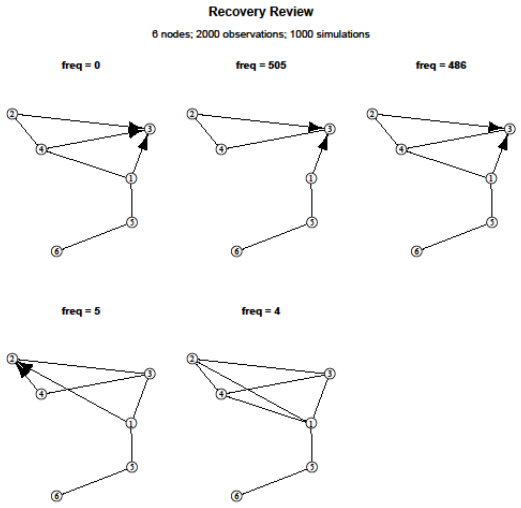


Figure B.40.