Supplementary Material: The Distance Precision Matrix: computing networks from non-linear relationships

Mahsa Ghanbari[a,b], Julia Lasserre[c], and Martin Vingron[a]

[a]Department of Computational Molecular Biology, Max Planck Institute for Molecular Genetics, D-14195 Berlin, Germany.
[b]Current address: The Berlin Institute for Medical Systems Biology, Max Delbrück Center for Molecular Medicine, Robert Rössle Str. 10, D-13125 Berlin, Germany
[c]Zalando Research, Charlottenstr. 4, D-10969 Berlin, Germany

June 12, 2018

1 Examples of non-linear relationships

Figure 1 shows the scatter plot of two variables with a simple non-linear (here quadratic) relationship for varying numbers of samples.

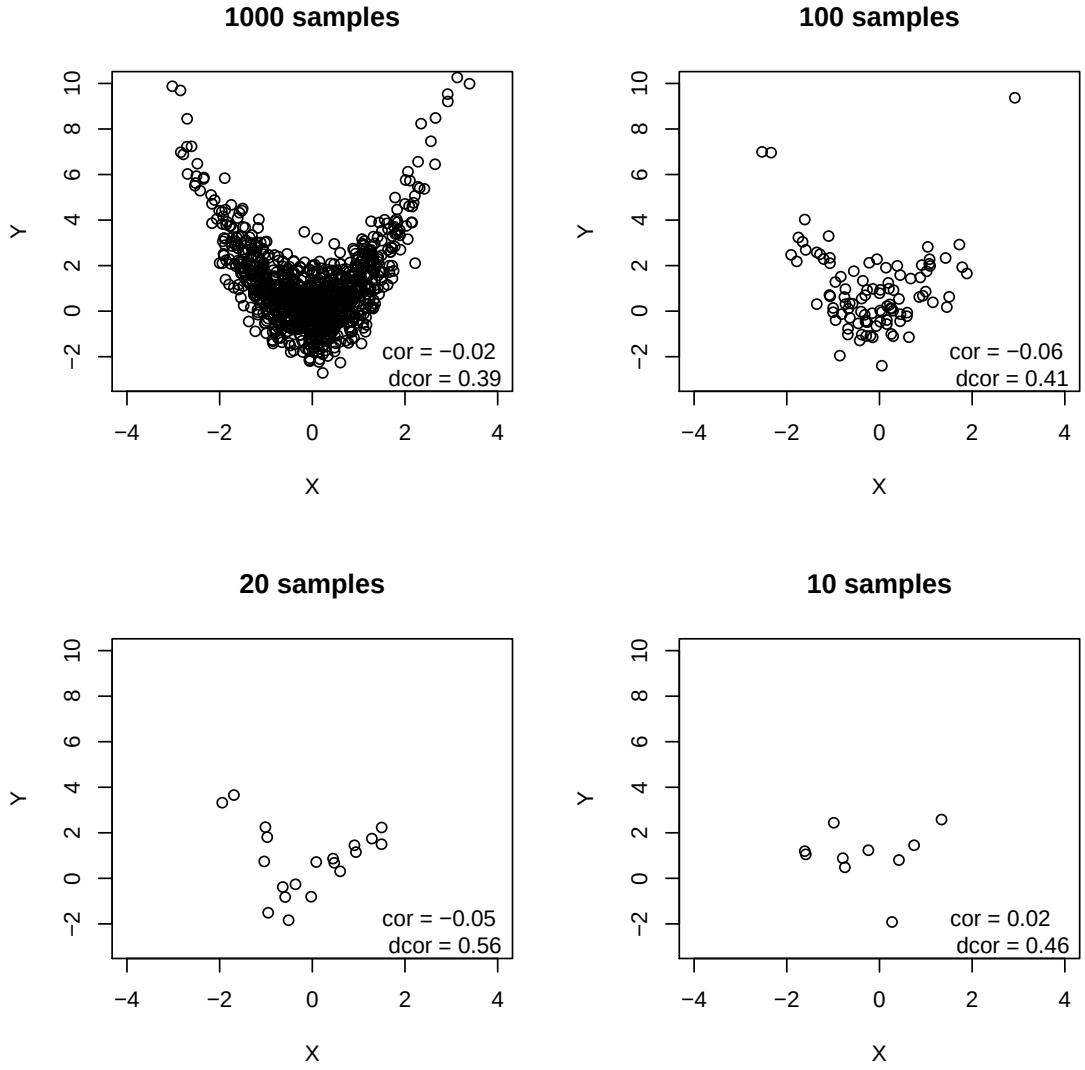


Figure 1: **Scatter plots of a non-linear relationship with different number of samples.**

In the case of high sample size, we can recognize the relationship visually and build a specific model. In the case of low sample size however, and in the presence of noise, visual signs can be misleading, and one might even think that there is no relationship between the variables as the correction coefficient (cor) oscillates around zero. Distance correlation (dcor) on the other hand captures the relationship at all sample sizes in the plot. The coefficient is sensitive to sample size and varies more but it does not go to 0.

Figure 2 shows the scatter plot of two genes in DREAM4 (10 nodes).

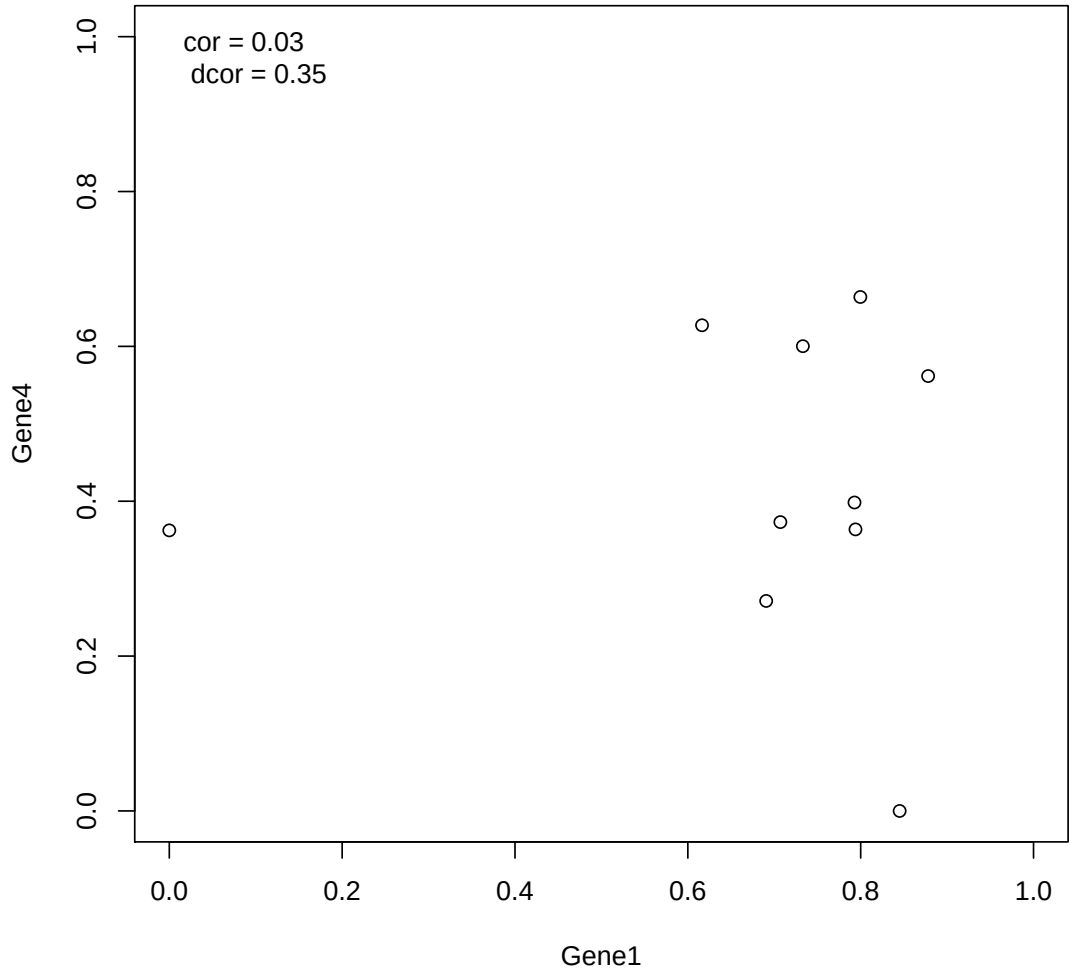


Figure 2: **Scatter plot of a (non-linear) relationship between two genes in DREAM4 data.**

This plot shows non-linear data with small sample size and high noise, quite a typical case in biology. The relationship here would be really hard to model visually and it is preferable to use a generic method than can handle all forms of dependencies.

2 Functions used to produce non-linear relationships

Each variable corresponds to a node, and there are 11 variables ($A...K$). The following functions were used to simulate the non-linear data:

$$A = U(-2, 2)$$

$$B = U(-2, 2)$$

$$C = 5 * A^2 - 3 + \epsilon$$

$$D = 3 * A^3 - 5 * A + 5 + \epsilon$$

$$E = (2 * B + 0.5)^2 - 4 + \epsilon$$

$$F = 2 * log(|C - 6|) + 1 + \epsilon$$

$$G = 0.2 * (C - 6)^2 + 0.4 * |F|^{(1/2)} - 14 + \epsilon$$

$$H = 0.4 * (D/2)^2 - 0.3 * (E/5)^3 - 1.5 * E - 10 + \epsilon$$

$$I = 0.4 * (E - 5)^2 - 0.5 * (E - 5) - 20 + \epsilon$$

$$J = 0.2 * (H/10)^4 + 1.3 * (H/10)^3 + 0.4 * (H/10)^2 - 5 + \epsilon$$

$$K = 0.6 * (I/5)^2 - 0.8 * B^3 - 3 + \epsilon$$

Figure 3 shows typical non-linear data used in the experiments.

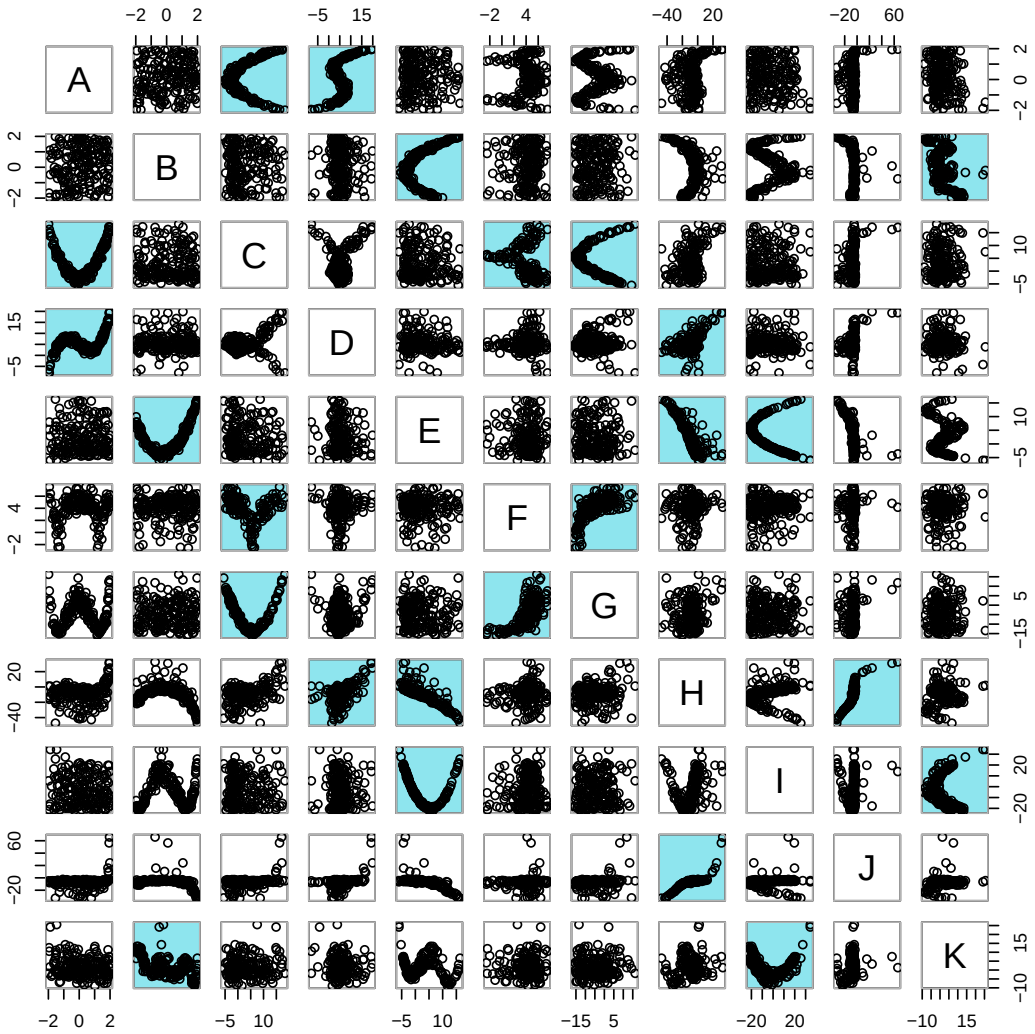


Figure 3: **Scatter plots of one realization of data simulation from the network H in Main Document Figure 1** (highlighted cells correspond to real edges).

3 Simulation of Gaussian data

To simulate Gaussian data, a random directed acyclic graph G is generated using the function randomDAG from the R package pcalg [1, 2], with a connection probability of 0.2. Random edges are added when necessary to make sure every node is connected to at least one other node. An undirected graph is then created by making G symmetric using $G = G + G^T$. A covariance matrix is generated from the skeleton of G using the function qpG2Sigma from the R package qpgraph [3, 4, 5], with rho=0.5. This covariance matrix is then used to simulate multivariate Gaussian data with arbitrary means using the function rmvnorm from the R package mvtnorm [6, 7].

4 Evaluation methodology

In network reconstruction, only a small fraction of the possible edges among nodes is expected to be real. In such unbalanced situations, it has been shown that the precision-recall (PR) curve is very informative in order to compare method performances [8].

Precision-recall (PR) curves are very informative when it comes to network reconstruction tasks, because they show the sorting power of the algorithm. Indeed, by plotting the precision as a function of the recall, we can see how often the edges the algorithm is most confident about are true edges. In contrast, the well-known receiver-operating-characteristic (ROC) curve shows the separation power of an algorithm. Indeed, by plotting the recall as a function of the false positive rate, we can see how many edges we can retrieve in general.

ROC curves typically display a task-independent behavior for a random binary classifier. The expected ROC curve is a straight line from $[0, 0]$ to $[1, 1]$, and its area under the curve (AUC) is 0.5.

But, unlike ROC curves, expected PR curves are task-dependent. The expected PR curve is a horizontal line from $[0, p]$ to $[1, p]$ and the AUC is p, where p is the proportion of positives (i.e. edges) in the network. If a network has 10 nodes, and we are interested in undirected edges, there are $\frac{10 \times 9}{2} = 45$ candidate edges. If this network has 9 actual edges, then $p = \frac{9}{45} = 0.2$. Equally, if the expected AUPRC is 0.4 and the network has 20 nodes, the number of actual edges in the network is $0.4 \times \frac{20 \times 19}{2} = 38$.

In this study, we only consider undirected edges, i.e. $A \rightarrow B$ and $B \rightarrow A$ are treated equally. All methods return a score for each (undirected) candidate edge. All candidate edges are sorted in ascending order of the absolute value of their scores given by the method of interest. PR and ROC curves are then built by growing the network from the top scoring edge to the bottom one and computing the false positive rate, the true positive rate (or recall) and the precision at each stage. When replicates are available, these curves are averaged by averaging the false positive rates, recalls and precisions of every stage.

Note that in this study, all methods are assessed equivalently. No parameters are tuned, and if parameters are required (for example in ARACNE [9], they are set to their default value. The PR and ROC curves are built using a moving threshold on the obtained scores, not by changing the parameters of the methods and obtaining new scores.

5 Competitor methods

In this study, we compare the performance of DPM and reg-DPM with nine other methods:

- Pearson correlation (cor).
- partial correlation (pcor), computed using the R package corpcor [10].
- regularized partial correlation following Schäfer *et al.* [10] (reg-pcor), computed using the R package corpcor [10].
- network deconvolution [11] with the correlation matrix as input (nd), using our own R implementation copied from the available python code.
- mutual information (mi), using the R package minet [12].All runs use equal frequency discretization and the Miller-Madow estimator. pmi uses the available Matlab code.
- part mutual information [13] (pmi), using the available Matlab code.
- the well-established network reconstruction method ARACNE [9] (arac), using the R package minet [12]. All runs use equal frequency discretization and the Miller-Madow estimator.
- distance correlation (dcor), using the provided R package energy [14].
- Székely *et al.*'s version of partial distance correlation [15] (pdcor), using source code provided by the authors.

6 Performance comparison on Gaussian data

Figure 4 shows the PR curves corresponding to Main Document Figure 2a. The expected PR curve on this task is a horizontal line at precision=0.27.

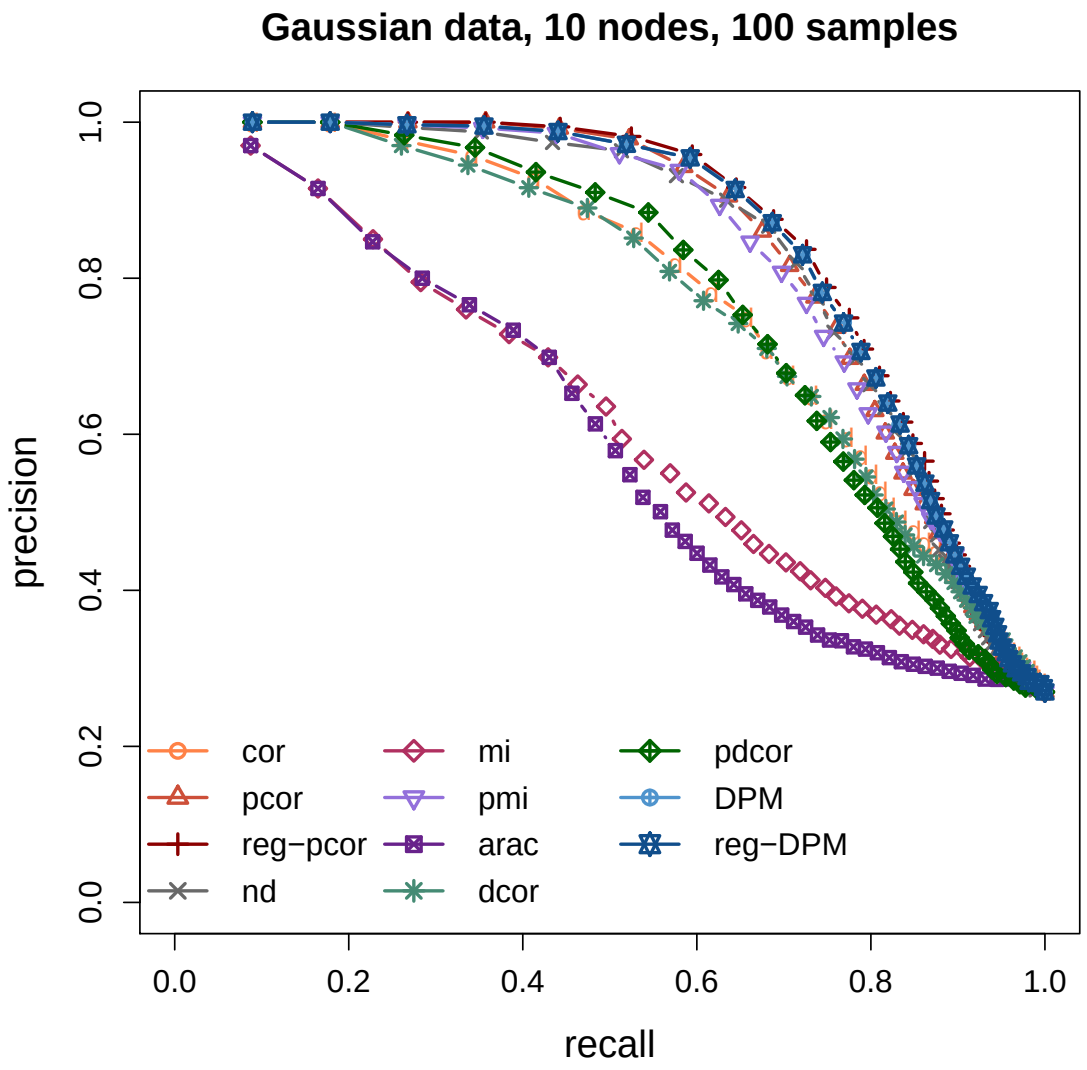
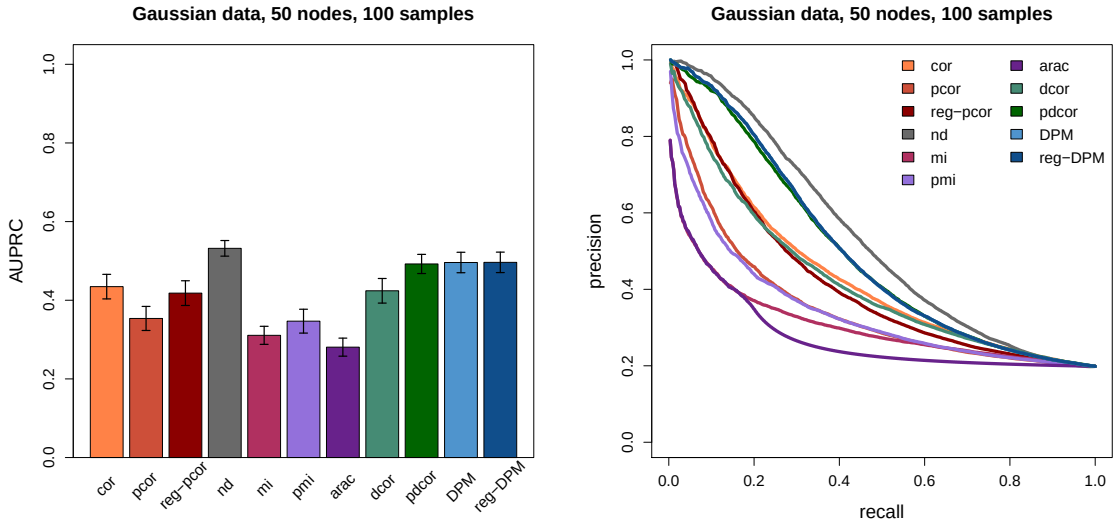


Figure 4: **Performance on simulated Gaussian data.** The plots show the PR curves (averaged over 100 replicates with 100 samples per replicate) of all selected methods for the reconstruction of Gaussian networks. The expected PR curve on this task is a horizontal line at precision=0.27.

Figure 5 shows the 50 nodes counterpart of Main Document Figure 2a and Figure 4 above. Error bars show one standard deviation. The expected AUPRC on this task (averaged over the 100 simulated networks) is 0.27 for 10 nodes, 0.2 for 50 nodes.



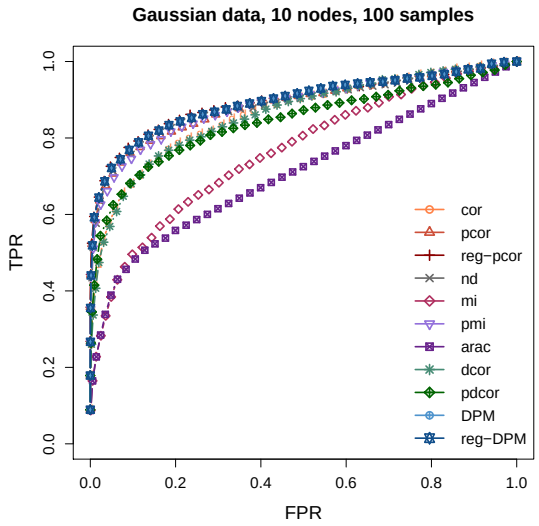
(a) Average PR curves. The expected PR curve on this task is a horizontal line at precision=0.2.

(b) Average AUPRCs. The expected AUPRC on this task is 0.2.

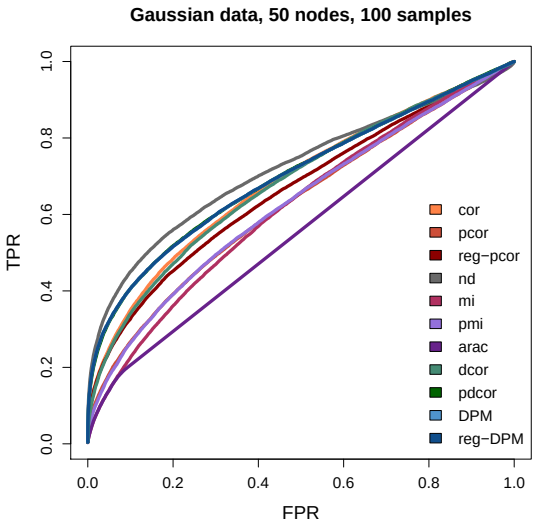
Figure 5: **Performance on simulated Gaussian data with 50 nodes networks.** The plots show the AUPRCs (averaged over 100 replicates with 100 samples per replicate) of all selected methods for the reconstruction of Gaussian networks. Error bars show one standard deviation. Overall, nd, DPM, and reg-DPM perform best.

Mutual information based methods (mi and arac) perform worst on this task, probably because of the discretization step. pmi is solved analytically and does not suffer. nd, DPM, and reg-DPM (overlapping) perform best for both network sizes, together with pcor, reg-pcor and pmi for the 10 nodes networks, and with pdcor for 50 nodes networks. Note that nd performs slightly better then DPM on the 50 nodes networks, however this difference decreases as the number of samples increases (see Figure 7 for the same figure with 200 samples). pcor and pmi do surprisingly badly on the 50 nodes networks, probably due to the relatively small amount of samples. Indeed, when using 200 samples, pcor, reg-pcor and pmi get closer. In line with this assumption, Figure 5 shows that regularization improves pcor for the 50 nodes networks, though not enough to match DPM. For DPM both the standard and regularized versions work well, which suggests that DPM requires fewer samples than pcor, probably due to the quadratic inflation of the number of samples in the D-centered vectors. As could be expected, cor and dcor go hand-in-hand for both network sizes. Regarding ROC curves and AUROCCs (see Figure 6), all methods are roughly comparable except those based on mutual information which never perform as well.

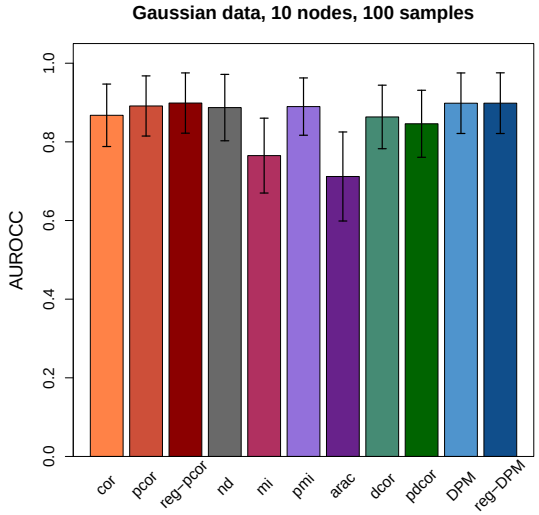
Figure 6 shows the ROC counterpart of Main Document Figure 2a and Figure 5 above.



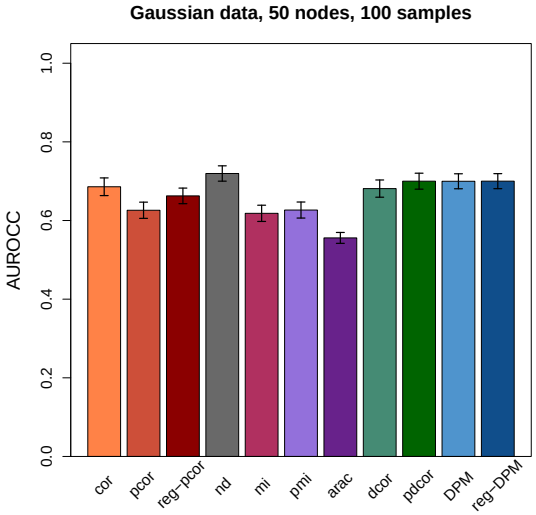
(a) Average ROC curves, 10 nodes networks.



(b) Average ROC curves, 50 nodes networks.



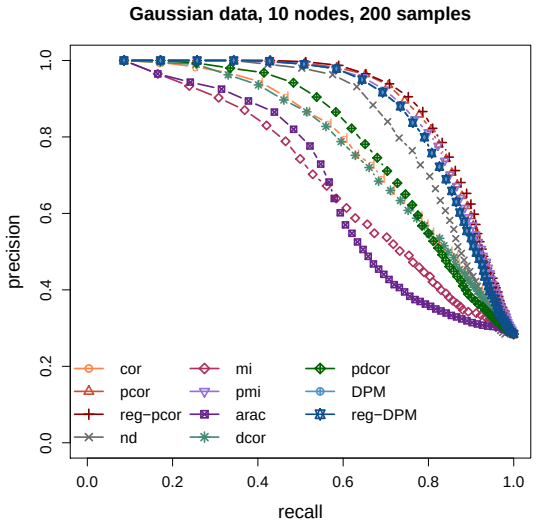
(c) Average AUROCCs, 10 nodes networks. Error bars show one standard deviation.



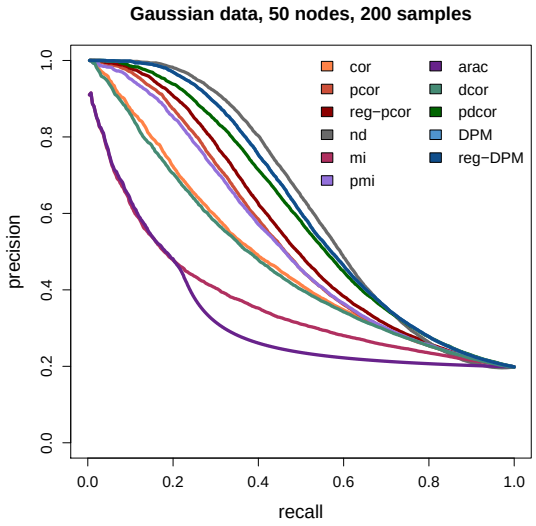
(d) Average AUROCCs, 50 nodes networks. Error bars show one standard deviation.

Figure 6: **ROC-based performance on simulated Gaussian data.** The plots show the performance (averaged over 100 replicates with 100 samples per replicate) of all selected methods for the reconstruction of Gaussian networks.

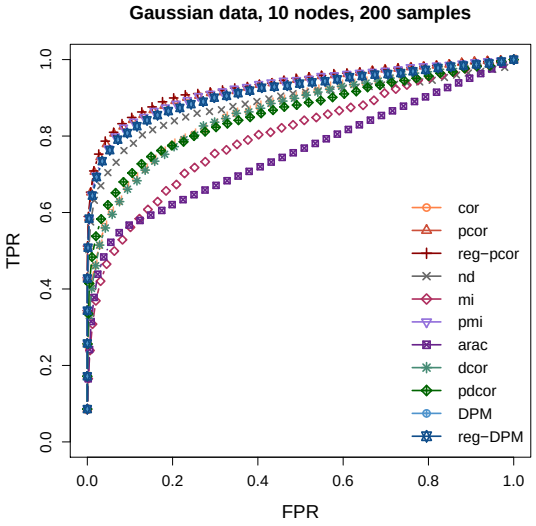
We run the same set of experiments as in Main Document Figure 2a but with 200 samples. Results are shown in Figure 7 and provide a similar message. The expected PR curve on this task (averaged over the 100 simulated networks) is a horizontal line at precision=0.28 for 10 nodes, at precision=0.2 for 50 nodes.



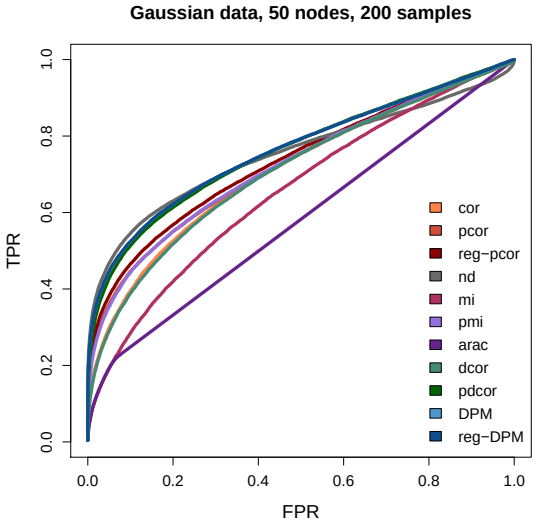
(a) Average PR curves, 10 nodes networks. The expected PR curve on this task is a horizontal line at precision=0.28.



(b) Average PR curves, 50 nodes networks. The expected PR curve on this task is a horizontal line at precision=0.2.



(c) Average ROC curves, 10 nodes networks.



(d) Average ROC curves, 50 nodes networks.

Figure 7: **Performance on simulated Gaussian data with 200 samples.** The plots show the performance (averaged over 100 replicates with 200 samples per replicate) of all selected methods for the reconstruction of Gaussian networks.

7 Performance comparison on non-linear data

Figure 8 shows the PR curves corresponding to Main Document Figure 2b. The expected PR curve on this task is a horizontal line at precision=0.2.

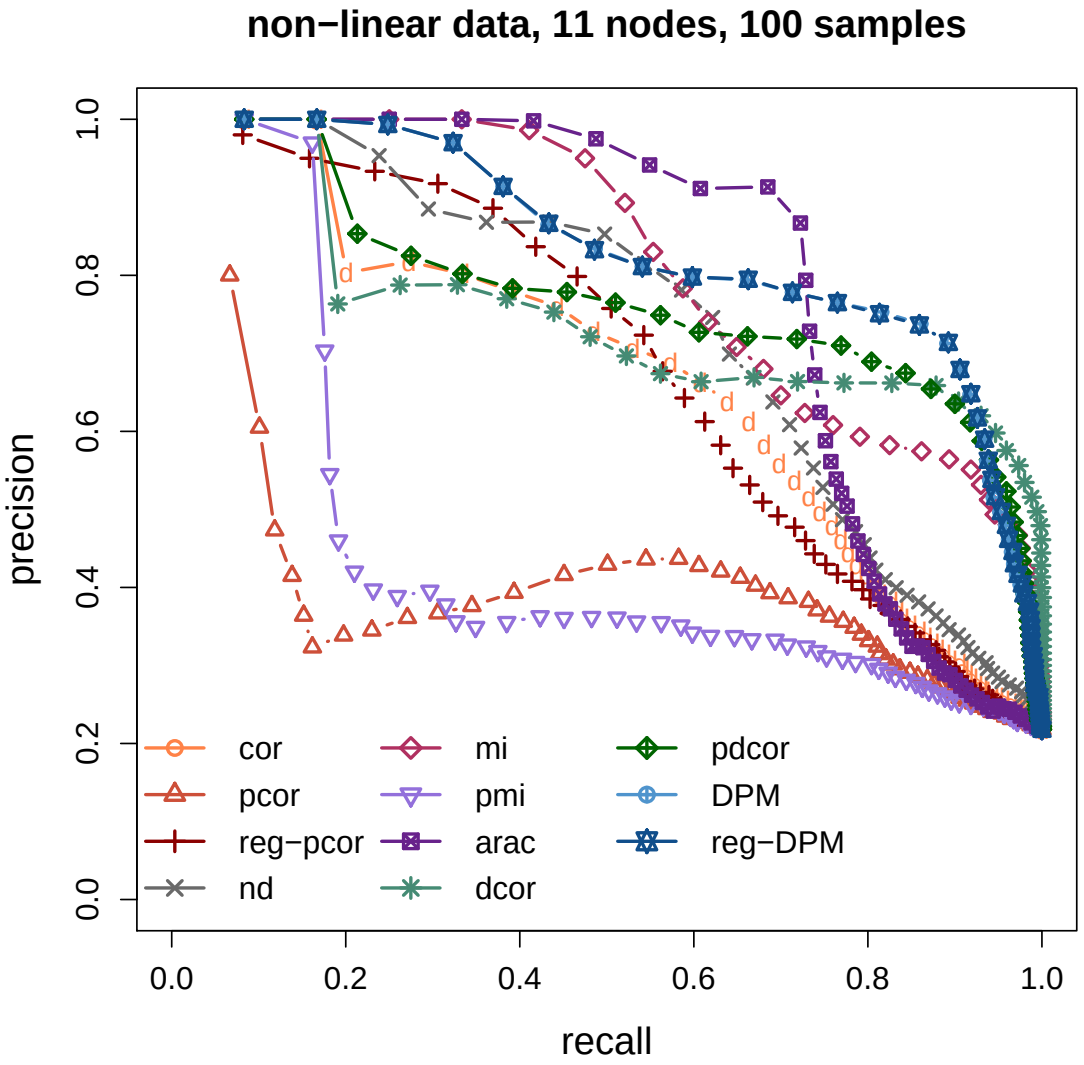
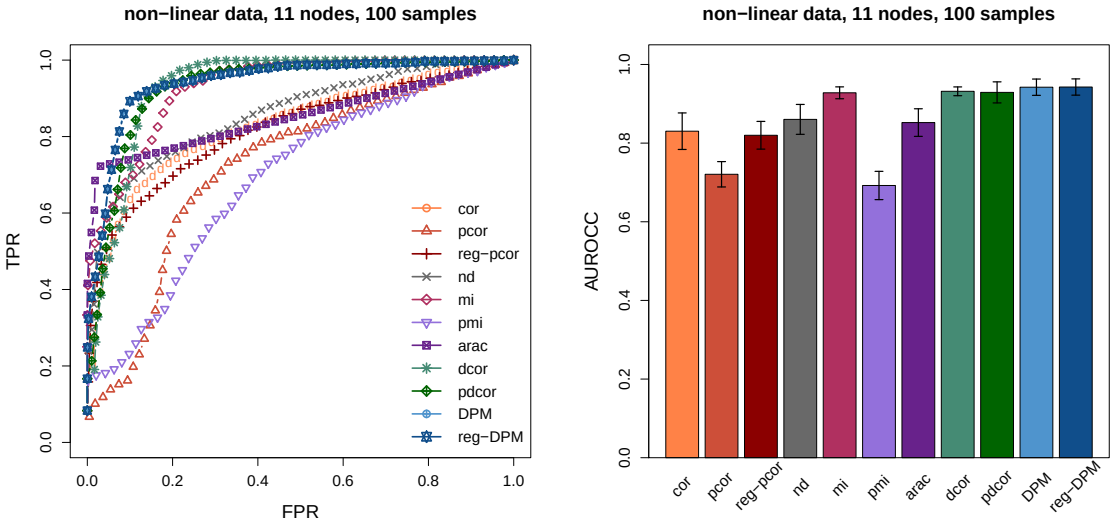


Figure 8: **Performance on simulated non-linear data.** The plots show the PR curves (averaged over 100 replicates with 100 samples per replicate) of all selected methods for the reconstruction of the network H in Main Document Figure 1. The expected PR curve on this task is a horizontal line at precision=0.2. mi, arac, DPM, and reg-DPM perform best.

pcor and pmi, which are geared towards Gaussian data and require a larger number of samples, perform worst on this task. DPM, reg-DPM (overlapping), and arguably mi and arac, perform best, followed by pdcor. Note that arac seems to drop down very quickly in precision. This is due to the candidate edges deemed absent by arac receiving the same score 0 and therefore being ranked randomly. Here, adding samples as done in Figure 10 helps neither pcor nor reg-pcor, due to the nature of the data. Again, while regularization dramatically improves the performance of pcor, there is no visible difference between

DPM and reg-DPM. Regarding ROC curves and AUROCCs (see Figure 9), DPM, reg-DPM, pdcor, dcor and mi perform best.

Figure 9 shows the ROC counterpart of Main Document Figure 2b and Figure 8 above.

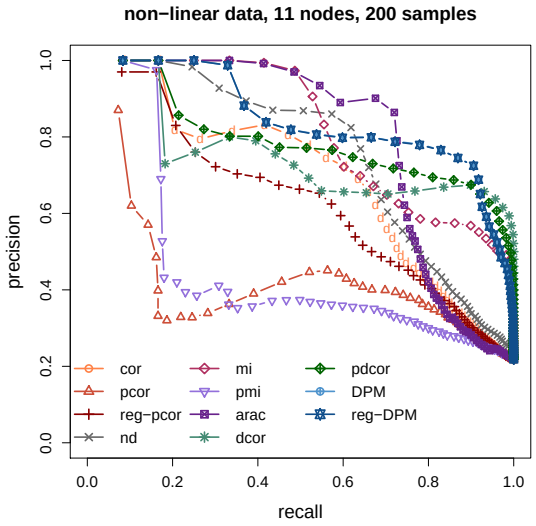


(a) Average ROC curves.

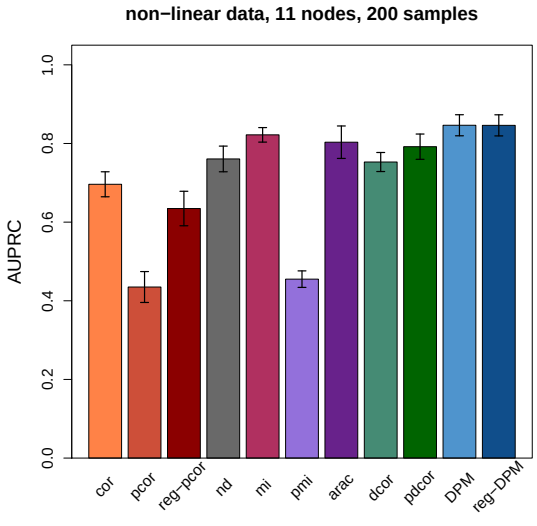
(b) Average AUROCCs with error bars showing one standard deviation.

Figure 9: **ROC-based performance on simulated non-linear data.** The plots show the performance (averaged over 100 replicates with 100 samples per replicate) of all selected methods for the reconstruction of the network H in Main Document Figure 1.

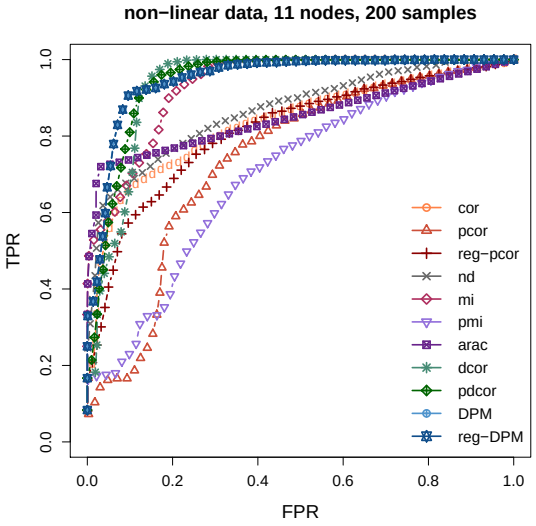
We run the same set of experiments as in Main Document Figure 2b but with 200 samples. Results are shown in Figure 10 and provide a similar message, unless discussed in Main Document. The expected PR curve on this task is a horizontal line at precision=0.2 and the expected AUPRC is 0.2.



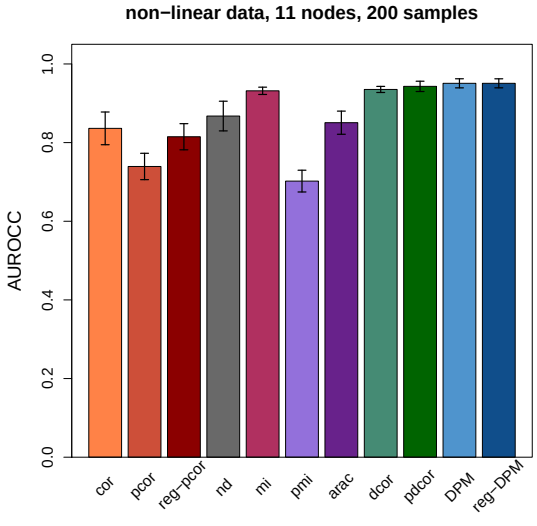
(a) Average PR curves. The expected PR curve on this task is a horizontal line at precision=0.2.



(b) Average AUPRCs with error bars showing one standard deviation. The expected AUPRC on this task is 0.2.



(c) Average ROC curves.

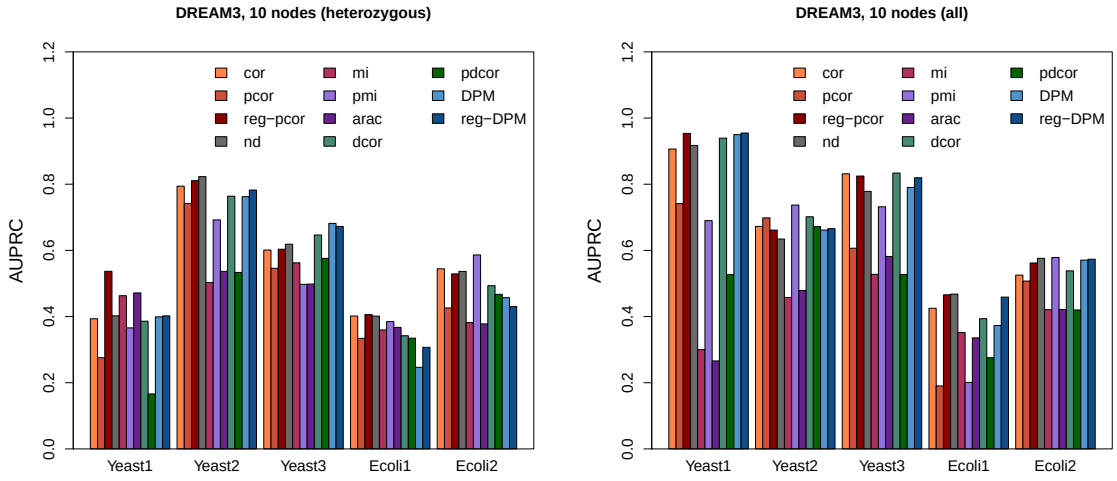


(d) Average AUROCCs with error bars showing one standard deviation.

Figure 10: **Performance on simulated non-linear data with 200 samples.** The plots show the performance (averaged over 100 replicates with 200 samples per replicate) of all selected methods for the reconstruction of the network H in Main Document Figure 1.

8 DREAM3 results using different data

Figure 11 shows the AUPRCs obtained on DREAM3 using various subsets of data for the 10 nodes networks. Wild-type is always included. The expected AUPRCs on these tasks are 0.22 for Yeast1, 0.56 for Yeast2, 0.49 for Yeast3, 0.24 for Ecoli1 and 0.33 for Ecoli2.

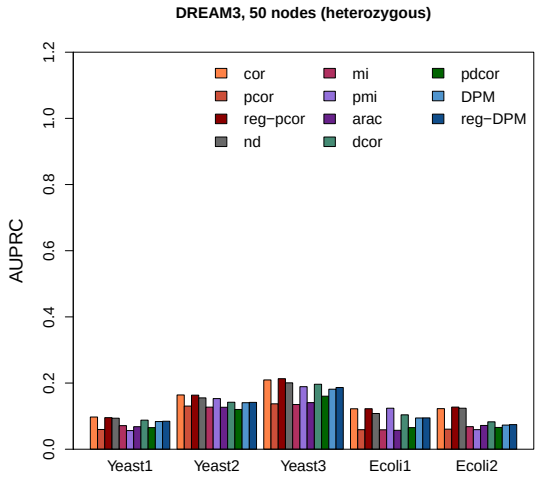


(a) Heterozygous data.

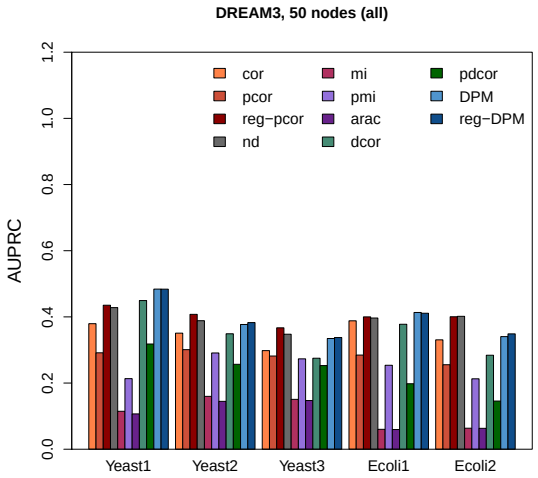
(b) All (null-mutants and heterozygous) data.

Figure 11: **AUPRCs on DREAM3 challenge using various subsets of data for the 10 nodes network.** Wild-type is always included and therefore not mentioned in sub-captions. The expected AUPRCs on these tasks are 0.22 for Yeast1, 0.56 for Yeast2, 0.49 for Yeast3, 0.24 for Ecoli1 and 0.33 for Ecoli2.

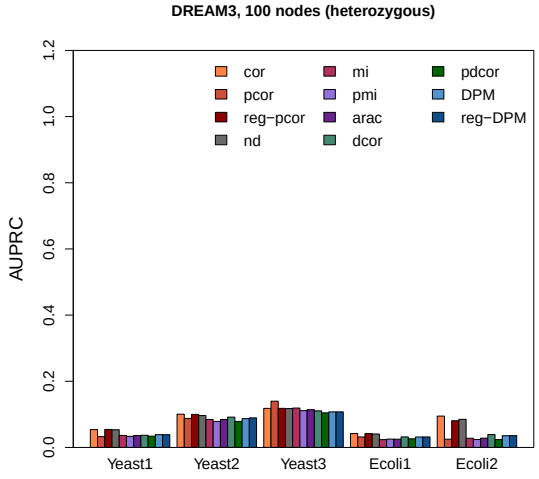
Figure 12 shows the AUPRCs obtained on DREAM3 using various subsets of data for the 50 nodes and 100 nodes networks. Wild-type is always included. The expected AUPRCs on these tasks for 50 nodes networks are 0.06 for Yeast1, 0.13 for Yeast2, 0.14 for Yeast3, 0.05 for Ecoli1 and 0.07 for Ecoli2, and for 100 nodes networks 0.03 for Yeast1, 0.08 for Yeast2, 0.11 for Yeast3, 0.03 for Ecoli1 and 0.02 for Ecoli2.



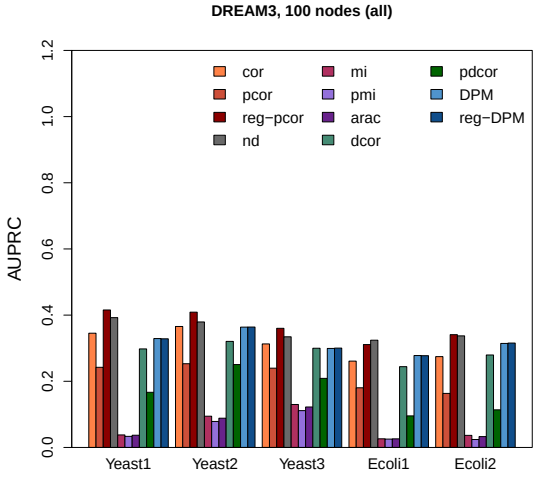
(a) 50 nodes networks, heterozygous data.



(b) 50 nodes networks, all (null-mutant and heterozygous) data.



(c) 100 nodes networks, heterozygous data.

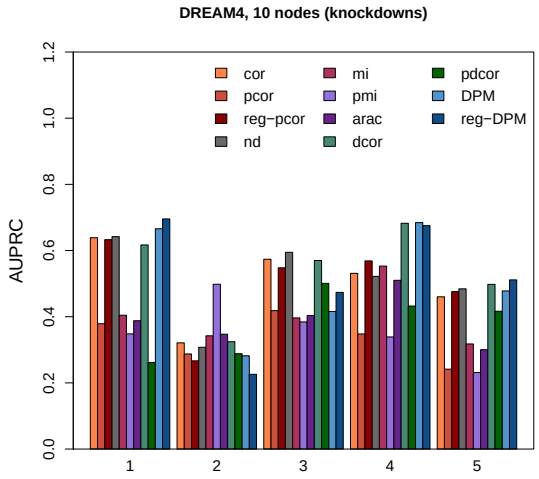


(d) 100 nodes networks, all (null-mutant and heterozygous) data.

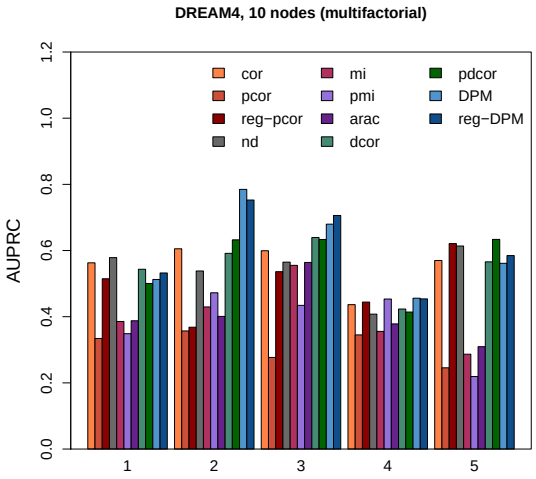
Figure 12: **AUPRCs on DREAM3 challenge using various subsets of data for the larger networks.** Wild-type is always included and therefore not mentioned in subcaptions. **(a, b): 50 nodes networks.** The expected AUPRCs on these tasks are 0.06 for Yeast1, 0.13 for Yeast2, 0.14 for Yeast3, 0.05 for Ecoli1 and 0.07 for Ecoli2. **(c, d): 100 nodes networks.** The expected AUPRCs on these tasks 0.03 for Yeast1, 0.08 for Yeast2, 0.11 for Yeast3, 0.03 for Ecoli1 and 0.02 for Ecoli2.

9 DREAM4 results using different data

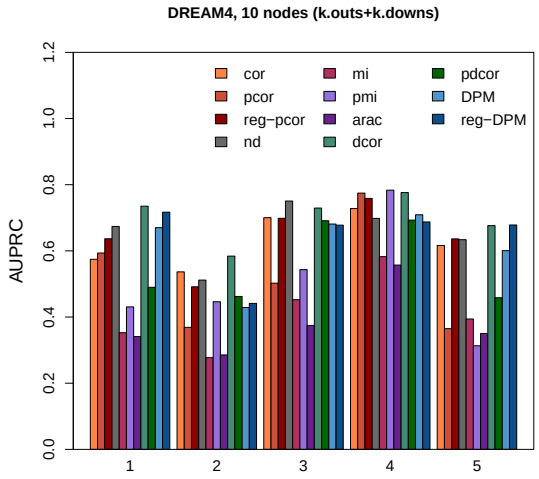
Figure 13 shows the AUPRCs obtained on DREAM4 using various subsets of data for the 10 nodes network. Wild-type is always included. The expected AUPRCs on these tasks are 0.29 for organism 1, 0.27 for organism 2, 0.33 for organism 3, 0.29 for organism 4 and 0.27 for organism 5.



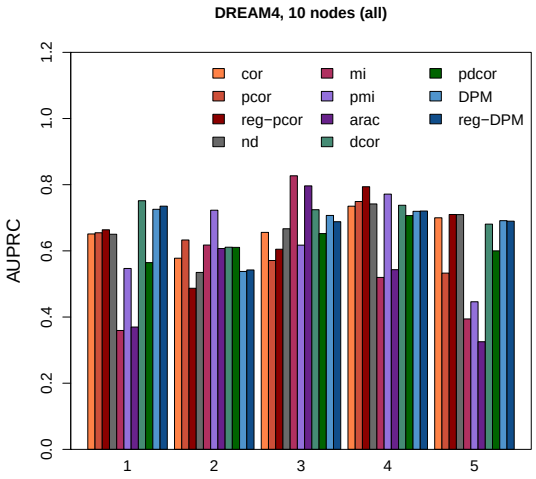
(a) Knockdown data.



(b) Multifactorial data.



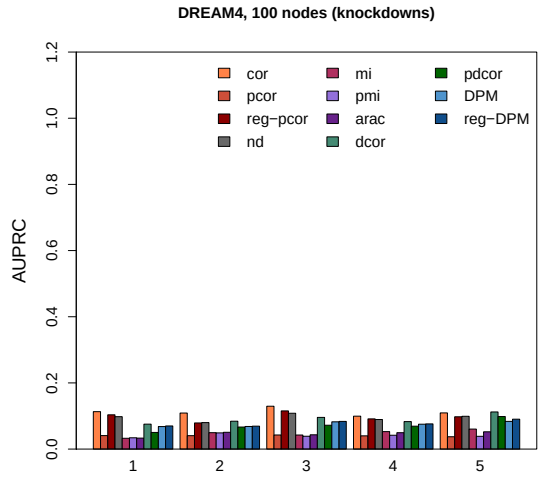
(c) Knockout and knockdown data.



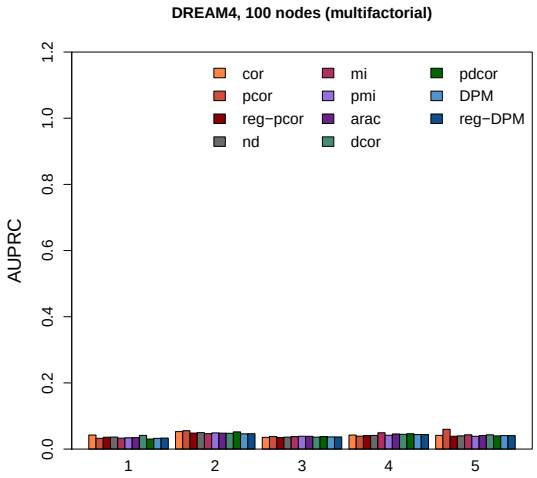
(d) All (knockout, knockdown and multifactorial) data.

Figure 13: **AUPRCs on DREAM4 challenge using various subsets of data for the 10 nodes networks.** Wild-type is always included and therefore not mentioned in sub-captions. The expected AUPRCs on these tasks are 0.29 for organism 1, 0.27 for organism 2, 0.33 for organism 3, 0.29 for organism 4 and 0.27 for organism 5.

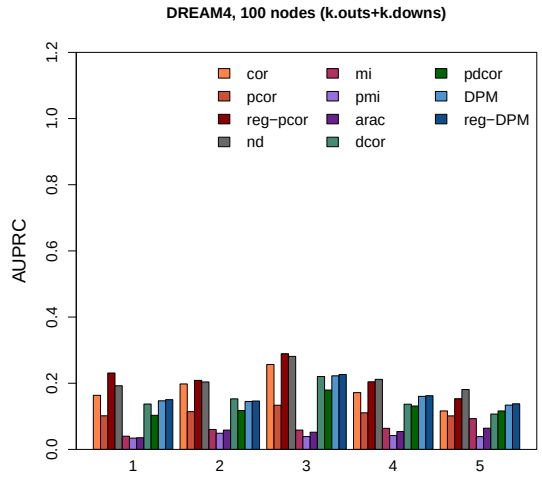
Figure 14 shows the AUPRCs obtained on DREAM4 using various subsets of data for the 100 nodes network. Wild-type is always included. The expected AUPRCs on these tasks are 0.03 for organism 1, 0.05 for organism 2, 0.04 for organism 3, 0.04 for organism 4 and 0.04 for organism 5.



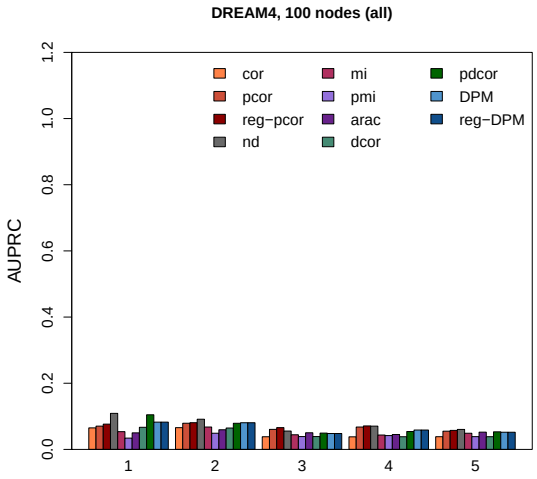
(a) Knockdown data.



(b) Multifactorial data.



(c) Knockout and knockdown data.



(d) All (knockout, knockdown and multifactorial) data.

Figure 14: **AUPRCs on DREAM4 challenge using various subsets of data for the 100 nodes networks.** Wild-type is always included and therefore not mentioned in sub-captions. The expected AUPRCs on these tasks are 0.03 for organism 1, 0.05 for organism 2, 0.04 for organism 3, 0.04 for organism 4 and 0.04 for organism 5.

10 DREAM5 results

Figure 15 shows the AUROCCs obtained on DREAM5 data. Performance is reported only on edges between regulators and transcription factors (noted TFs in captions). The simulated data has 1643 nodes (including 195 transcription factors) and 805 samples. The *E. coli* data has 4511 nodes (including 334 transcription factors) and 805 samples. The *S. cerevisiae* data has 5959 nodes (including 333 transcription factors) and 536 samples.

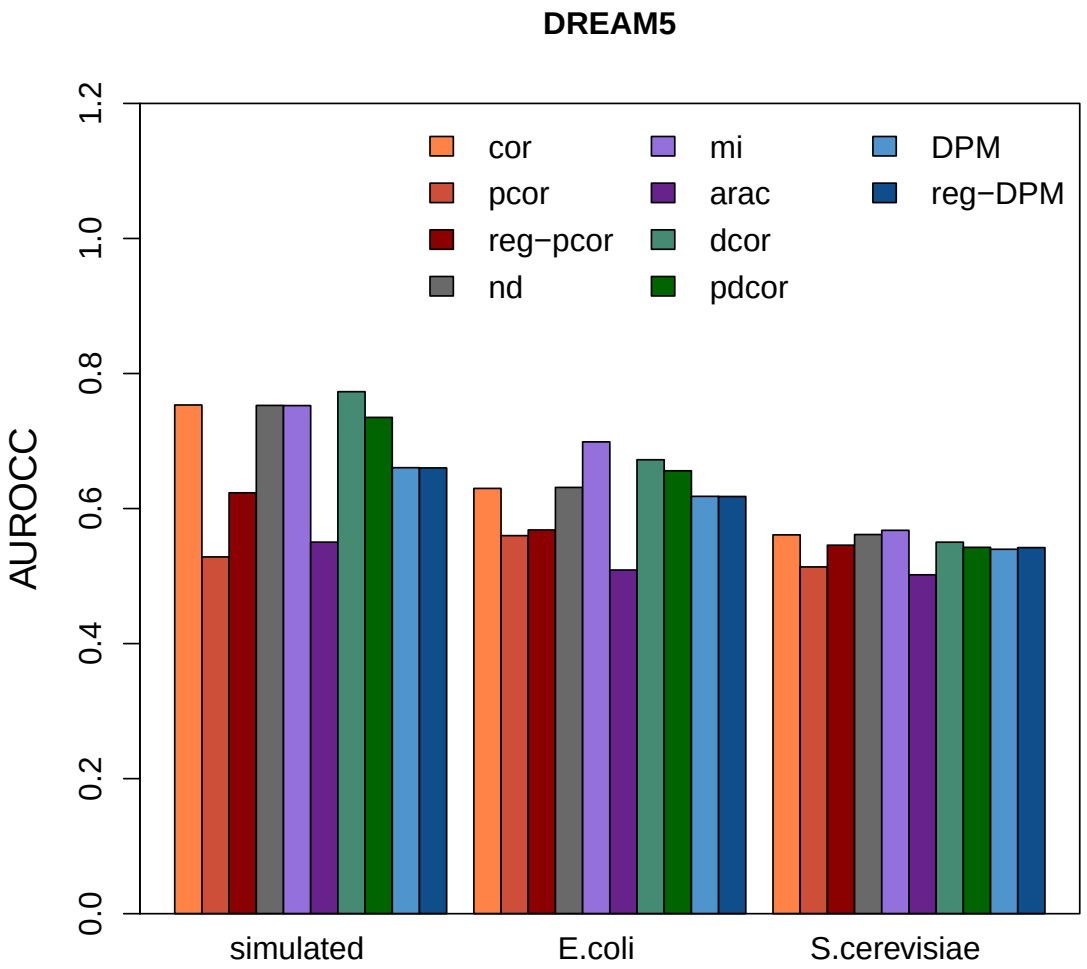


Figure 15: **AUROCCs on DREAM5 challenge data.** The simulated data has 1643 nodes (including 195 TFs), and 805 samples. The *E. coli* data has 4511 nodes (including 334 TFs) and 805 samples. The *S. cerevisiae* data has 5950 nodes (including 333 TFs) and 536 samples.

11 Setting a threshold

PR and ROC curves are useful to compare methods in the presence of a gold standard, but provide little guidance when the method is applied to a novel dataset. In the classical setting of Gaussian data, statistical tests can be applied with multiple testing correction. However, for many other measures including our DPM and reg-DPM, no statistical tests are available (yet). In principle, permutation tests can be applied [14, 15] but these can quickly become computationally prohibitive.

Recall that our method gives a score to all candidate edges. A scalable heuristic is to split the scores in the DPM matrix using k-means with $k = 2$ or a Gaussian mixture with $k = 2$ and unequal variances. The respective candidate edges can then be assigned to one of the two groups, one representing absent edges, the other present edges. In spite of its simplicity, this heuristic works reasonnably well, both on linear and non-linear data. We tested this for various number of samples: each particular number of samples yields a different score matrix and a different threshold. For statistical tests, we use a significance level $\alpha = 0.05$. We use the obtained threshold to reconstruct the network and compute the corresponding precision, recall and F1-score. The F1-score is a (balanced) trade-off between precision and recall and is therefore of particular interest. All results reported are averaged over 20 replicates.

Figure 16 shows for 10 nodes Gaussian networks and for varying numbers of samples the performance of thresholds resulting from various tests chosen according to the method. The black curves (best) show what could be achieved if we knew where to set the correct threshold to optimise this particular score. Unlike other thresholds, the "best" threshold is computed for each score independently. The best curves trivially reach 1 for both precision and recall, but not the F1-score. All subsequent plots are organised according to the following:

- The first row shows thresholds for DPM resulting from k-means (kmeans) and from a Gaussian Mixture Model (GMM).
- The middle row shows thresholds for pcor resulting from k-means (kmeans) and from the standard test based on Fisher transformation using $\alpha = 0.05$ (Fisher).
- The third row shows thresholds for pdcor resulting from k-means (kmeans) and from the test provided with pdcor using 499 permutations and $\alpha = 0.05$ (pdcor.test).

Both GMM and k-means perform relatively well for DPM, with a preference for GMM if precision and recall are of equal importance (F1-score), for k-means if precision is more important. Note that in general, precision is more important for network reconstruction. For pcor and pdcor, k-means performs just as well or better than their respective tests, again with a preference for k-means if precision is more important.

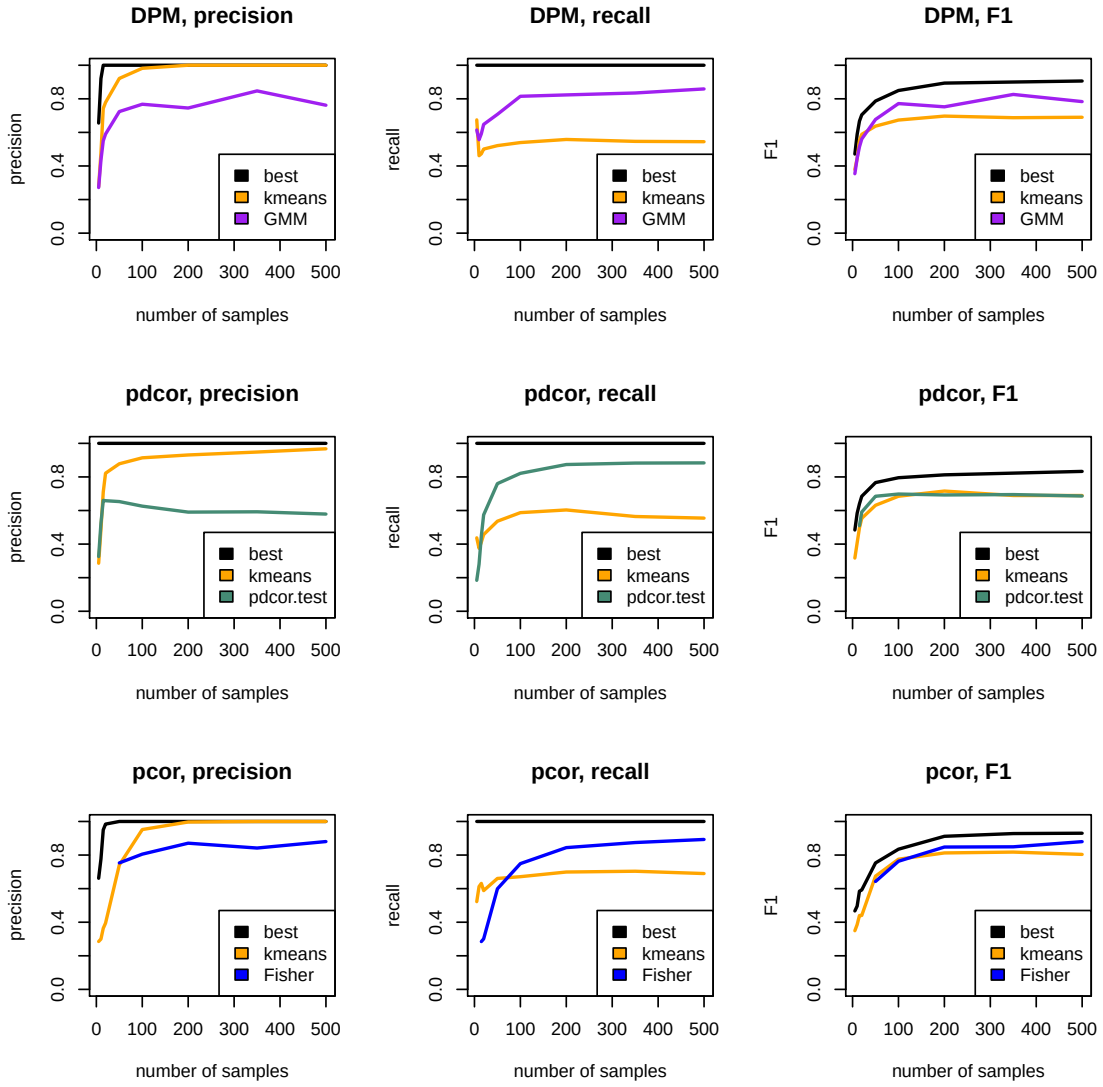


Figure 16: **Threshold selection using clustering on 10 nodes Gaussian networks.** The plots show for various scores the performance achieved (averaged over 20 replicates) using a particular threshold selection method. best refers to the top performance achievable, which is of course 1 for both precision and recall, but not for F1 which combines precision and recall. kmeans refers to the results given by the threshold computed from 2-means, GMM to the results given by the threshold computed from a bimodal Gaussian mixture with unequal variances, pdcor.test to the test supplied for pdcor with $\alpha = 0.05$ and 199 replicates, and Fisher to the standard partial correlation test based on the Fisher transformation with $\alpha = 0.05$.

Figure 17 shows the same plots as Figure 16, also on linear data, but for 50 nodes networks. For DPM, both GMM and k-means perform relatively well, with a preference for k-means, especially if precision is favoured. For pdcor and pcor, k-means behaves very well. In contrast, the statistical test for pdcor fails once the number of samples has reached the number of nodes, while k-means performs at any sample size, particularly when it comes to precision.

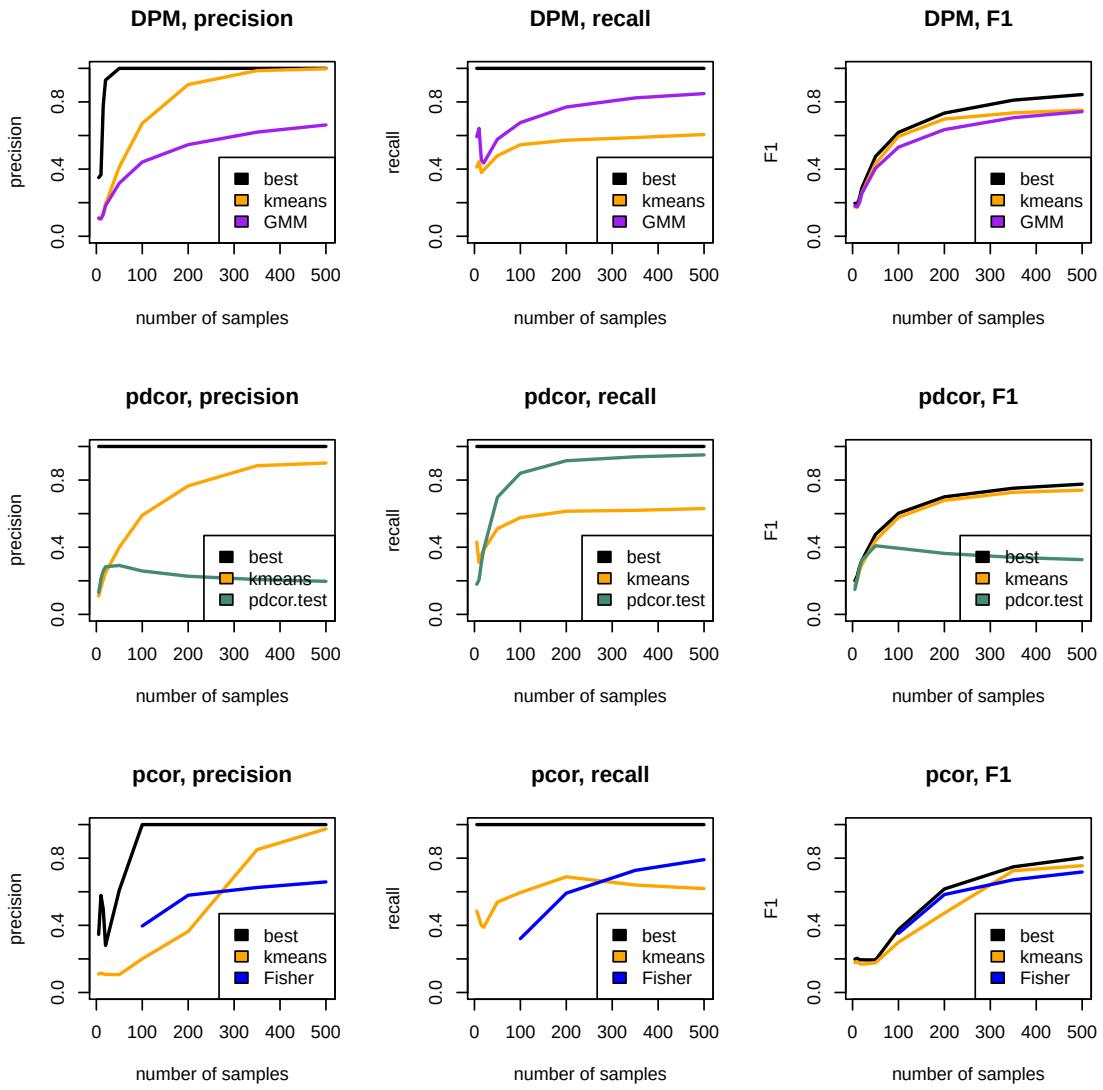


Figure 17: **Threshold selection using clustering on 50 nodes Gaussian networks.** The plots show for various scores the performance achieved (averaged over 20 replicates) using a particular threshold selection method. best refers to the top performance achievable, which is of course 1 for both precision and recall, but not for F1 which combines precision and recall. kmeans refers to the results given by the threshold computed from 2-means, GMM to the results given by the threshold computed from a bimodal Gaussian mixture with unequal variances, pdcor.test to the test supplied for pdcor with $\alpha = 0.05$ and 199 replicates, and Fisher to the standard partial correlation test based on the Fisher transformation with $\alpha = 0.05$.

Figure 18 shows the same plots as Figure 16 but on non-linear data generated using the network H in Main Document Figure 1. There again k-means behaves well for DPM.

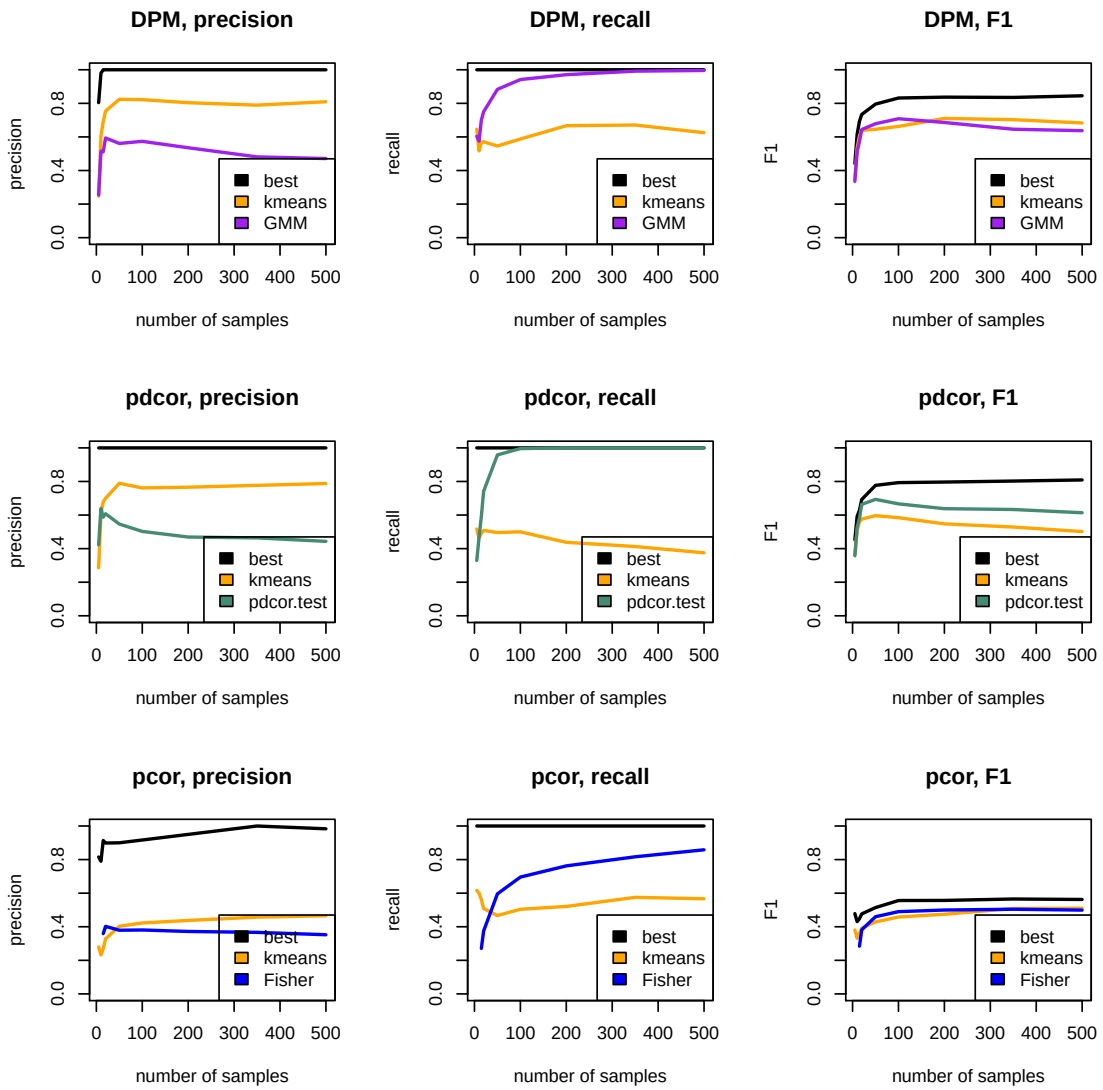


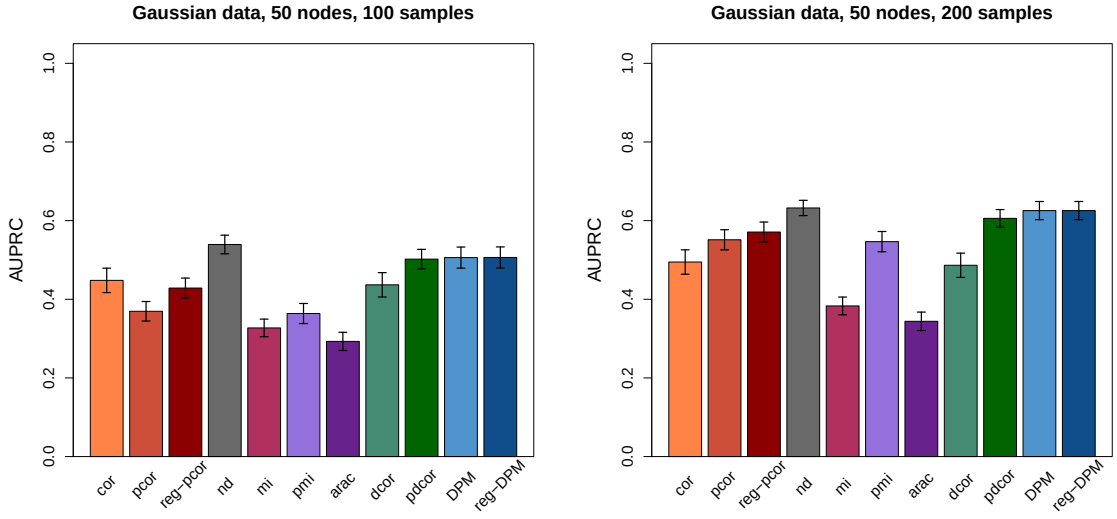
Figure 18: **Threshold selection using clustering on non-linear data simulated using the network H in Main Document Figure 1.** The plots show for various scores the performance achieved (averaged over 20 replicates) using a particular threshold selection method. best refers to the top performance achievable, which is of course 1 for both precision and recall, but not for F1 which combines precision and recall. kmeans refers to the results given by the threshold computed from 2-means, GMM to the results given by the threshold computed from a bimodal Gaussian mixture with unequal variances, pdcor.test to the test supplied for pdcor with $\alpha = 0.05$ and 199 replicates, and Fisher to the standard partial correlation test based on the Fisher transformation with $\alpha = 0.05$.

12 Detection of direct edges

Many graph reconstruction methods are based on standard partial correlation or covariance matrix inversion in order to focus on direct edges. pdcor, DPM and reg-DPM provide possible extensions to non-linear data and should therefore be able to distinguish direct edges from indirect ones. In this his section, we verify that real edges get a higher score indeed than indirect ones. We repeat the simulations for Gaussian and non-linear data described in their respective subsections, however here performance is computed using as positives the edges in the ground truth network (direct edges) and as negatives the pairs of nodes that are not connected but that are still marginally dependent in the ground truth network (indirect edges). Note that the positive set of edges has not changed, only the negative one.

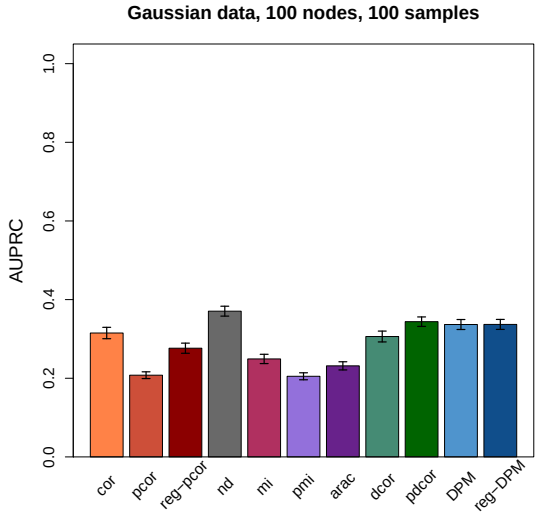
To identify the indirect edges, we use the marginal d-separation criterion by applying to the ground truth DAG the function dsep from the R package pcalg [1, 2]. Two nodes X and Y are marginally d-separated (and therefore marginally independent) if and only if all paths between X and Y contain a collider [16].For our experiments, all pairs of nodes that do not satisfy the d-separation criterion and that are not directly connected form indirect edges.

Figure 19 shows the average AUPRCs of all selected methods for distinguishing direct from indirect edges in larger networks for linear data. The expected AUPRCs on these tasks (averaged over the 100 simulated networks) are 0.21 for 50 nodes networks and 0.2 for 100 nodes networks.

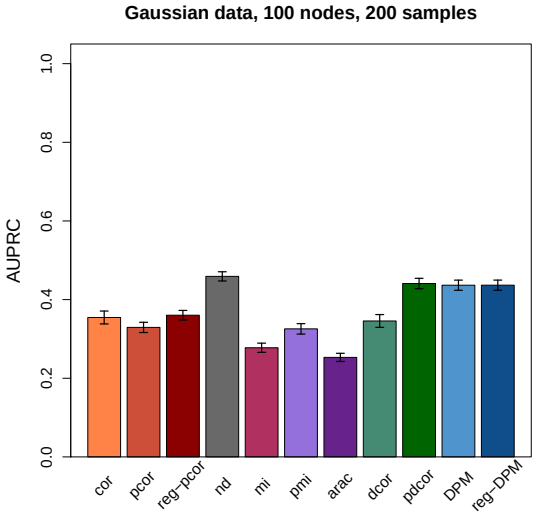


(a) 50 nodes networks, 100 samples.

(b) 50 nodes networks, 200 samples.



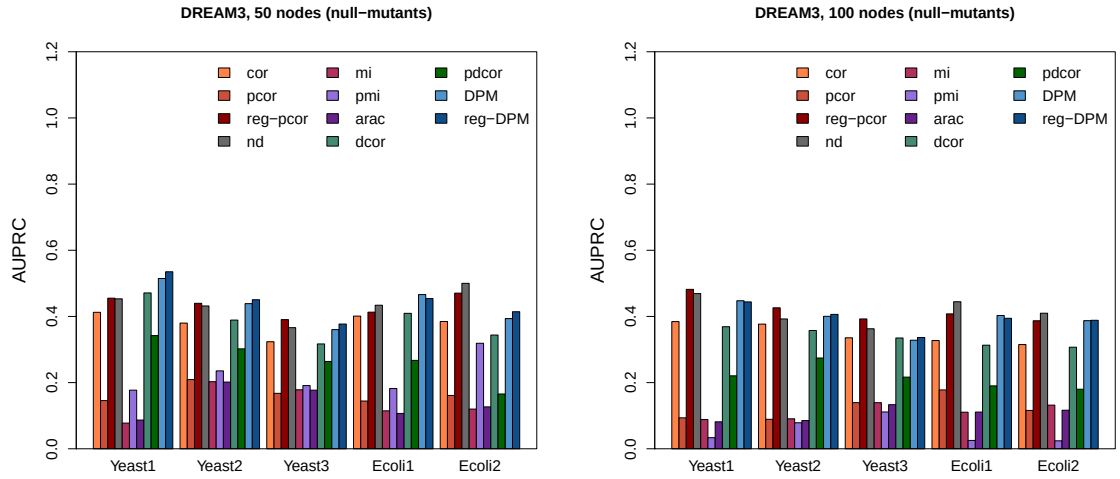
(c) 100 nodes networks, 100 samples.



(d) 100 nodes networks, 200 samples.

Figure 19: **Performance on direct vs indirect edges on Gaussian data for larger networks.** The plots show the AUPRCs (averaged over 100 replicates) of all selected methods for distinguishing direct from indirect edges in Gaussian networks. Note that networks are fully reconstructed, i.e. all edges are given a score, but the performance is computed exclusively on edges that are direct or edges whose nodes are not d-separated. **(a, b): 50 nodes samples.** The expected AUPRC on this task is 0.21. **(c, d): 100 nodes samples.** The expected AUPRC on this task is 0.2.

Figure 20 shows the AUPRCs of all selected methods for distinguishing direct from indirect edges in larger networks for DREAM3 data.



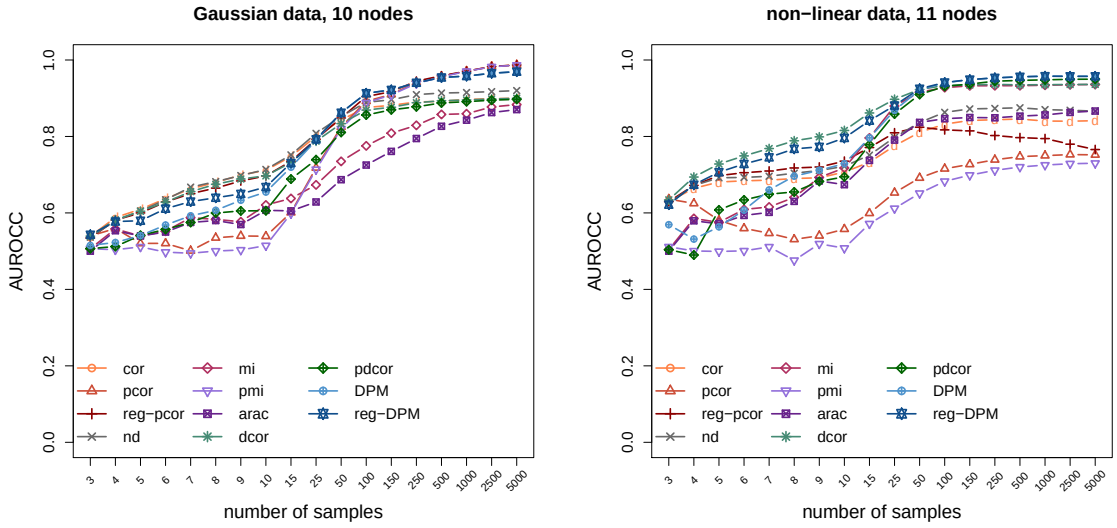
(a) 50 nodes networks. The expected AUPRC on these tasks are 0.09 for Yeast1, 0.18 for Yeast2, 0.15 for Yeast3, 0.11 for Ecoli1 and 0.13 for Ecoli2.

(b) 100 nodes networks. The expected AUPRCs on these tasks are 0.079 for Yeast1, 0.084 for Yeast2, 0.124 for Yeast3, 0.108 for Ecoli1 and 0.11 for Ecoli2.

Figure 20: **Performance on direct vs indirect edges on DREAM3 (wild-type and null-mutant) data for larger networks.** The plots show the AUPRCs of all selected methods for distinguishing direct from indirect edges in the network in DREAM3 networks. Note that networks are fully reconstructed, i.e. all edges are given a score, but the performance is computed exclusively on edges that are direct or edges whose nodes are not d-separated.

13 Effect of sample size

Figure 21 shows the ROC counterpart of Main Document Figure 5.



(a) Gaussian data.

(b) Non-linear data simulated using the network H in Main Document Figure 1.

Figure 21: **ROC-based effect of sample size on performance.** The plots show the evolution of the AUROCCs (averaged over 100 replicates) as sample size increases.

14 Effect of noise

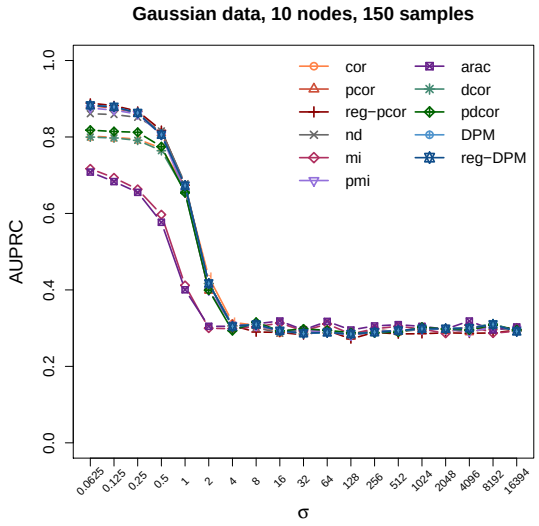
We repeat the simulations for Gaussian and non-linear data described in Main Document Sections "Performance comparison on Gaussian data" and "Performance comparison on non-linear data", and vary the variance of the univariate Gaussian noise that is added to the data drawn from the networks.

Figure 22 shows the evolution of the AUPRCs (averaged over 100 replicates with 150 samples per replicate) for all selected methods as noise increases. The expected AUPRCs on these tasks are 0.27 for Gaussian data (averaged over the 100 simulated networks) and 0.2 for non-linear data.

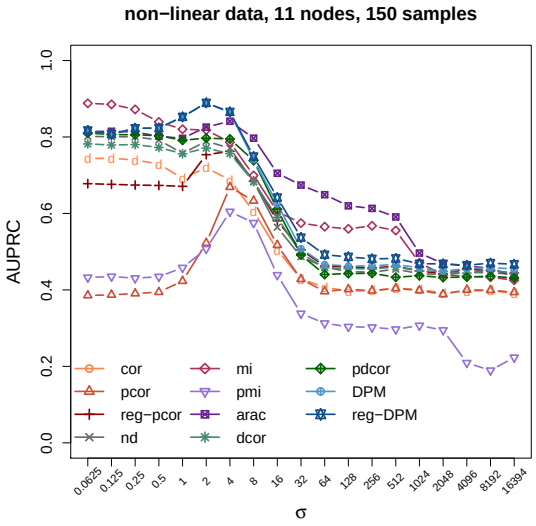
For Gaussian data (Figure 22a), DPM, reg-DPM, pcor, reg-pcor and nd perform best. All methods consistently get worse as noise increases until they almost reach the expected average AUPRC (0.27). Note that all methods start from various heights but all follow each other down, i.e. seem equally sensitive to noise.

For non-linear data (Figure 22b), mi is best for very small amounts of noise (here up to $\sigma = 0.5$). DPM and reg-DPM take over for small amounts of noise (up to $\sigma = 4$), and finally arac for large amounts of noise. Note that, below arac and mi, DPM and reg-DPM are above all other methods for large amounts of noise. Again, all curves go down around the same amount of noise, here $\sigma = 4$, and seem to follow each other down except for arac which decreases more slowly and for pmi which decreases more quickly. pcor, reg-pcor and pmi improve before they get worse, perhaps because relationships can appear more linear with the right amount of noise. When increasing the number of samples to 500 (Figure 23b), DPM and reg-DPM consistently perform better than mi but stay below arac for large amounts of noise. Note that, even with large amounts of noise, all methods seem to stabilise above random performance (0.2) except for pmi. ROC performance is reported in Figures 22c and 22d, and show similar trends. Results using 500 samples can be found in Figure 23 and show a similar hierarchy.

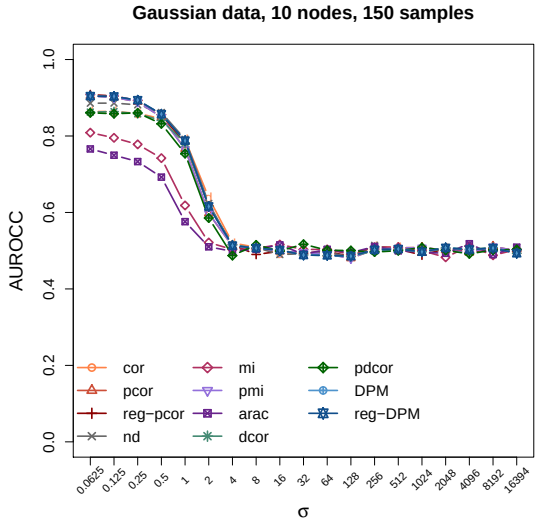
In conclusion, DPM and reg-DPM are methods of choice for small amounts of noise, but do not perform as well as arac for large amounts of noise on non-linear data. However mi and arac do not perform well on Gaussian data, while DPM and reg-DPM are reliable on both types of data.



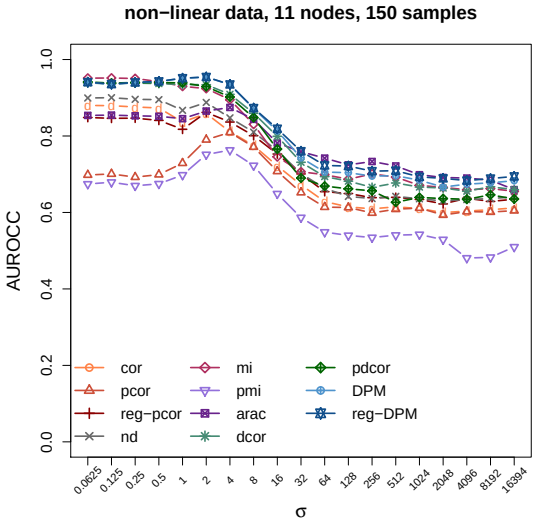
(a) Evolution of AUPRCs, Gaussian data. The expected AUPRC on this task is 0.27.



(b) Evolution of AUPRCs, non-linear data simulated using the network H in Main Document Figure 1. The expected AUPRC curve on this task is 0.2.



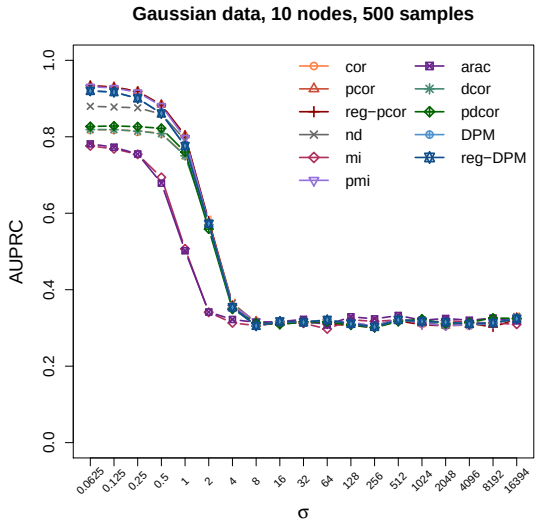
(c) Evolution of AUROCCs, Gaussian data.



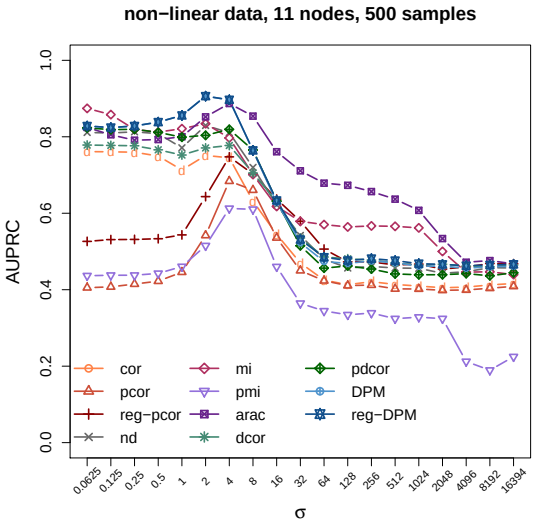
(d) Evolution of AUROCCs, non-linear data simulated using the network H in Main Document Figure 1.

Figure 22: **Effect of noise on performance with 150 samples.** The plot shows the evolution of the average performance as noise increases over 100 replicates with 150 samples per replicate. Each replicate contains additive Gaussian noise ϵ $N(0, \sigma^2)$.

We have run the same set of experiments as in Figure 22 but with 500 samples. Results are shown in Figure 23 and provide a similar message, unless discussed in Main Document. The expected AUPRC on this task is 0.29 for Gaussian data and 0.2 for non-linear data.

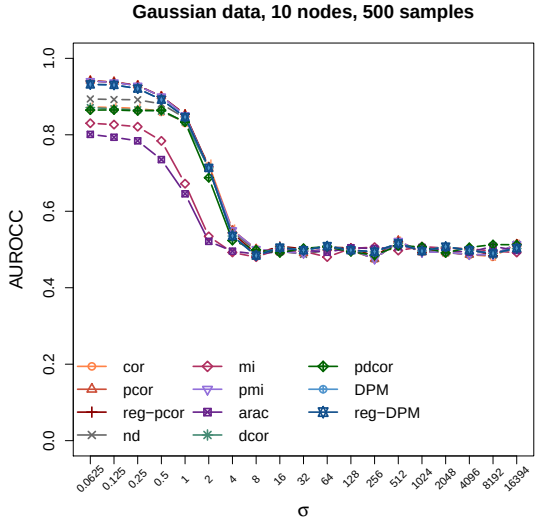


(a) Evolution of AUPRCs, Gaussian data. The expected AUPRC curve on this task is 0.29.

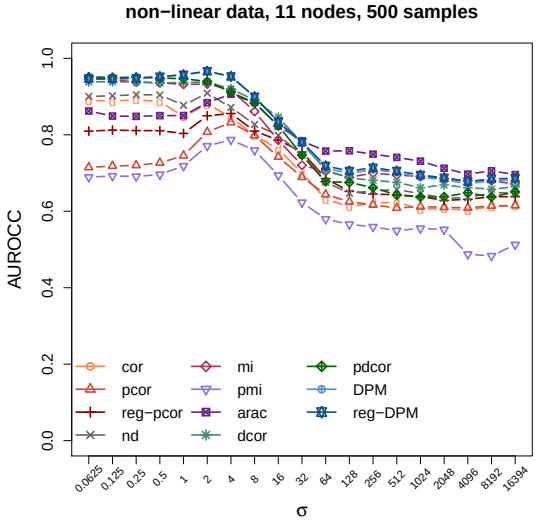


f

(b) Evolution of AUPRCs, non-linear data simulated using the network H in Main Document Figure 1. The expected AUPRC curve on this task is 0.2.



(c) Evolution of AUROCCs, Gaussian data.



(d) Evolution of AUROCCs, non-linear data simulated using the network H in Figure 1 in Main Document.

Figure 23: **Effect of noise on performance with 500 samples.** The plot shows the evolution of the average performance as noise increases over 100 replicates with 500 samples per replicate. Each replicate contains additive Gaussian noise ϵ $N(0, \sigma^2)$.

15 Differences between DPM and pdcor

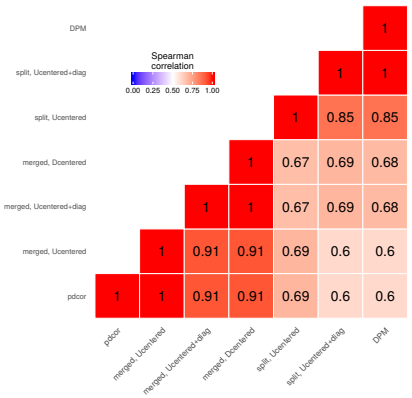
The experiments presented in Main Document show that, though similar in essence, pdcor and DPM produce different results. To understand better where the difference may come from, we deconstruct our method and compare each stage with pdcor using Spearman correlation between the respective scores. The intermediate stages are:

- merged, U-centered: DPM with distance covariance as defined in [15] and all control variables merged as in [15], this should be equivalent to pdcor.
- merged, U-centered+diag: DPM with distance covariance as defined in [15] but also taking the diagonal of distance matrices into account, and all control variables merged as in [15].
- merged, D-centered: DPM with distance covariance as defined in [14] and all control variables merged as in [15].
- split, U-centered: DPM with distance covariance as defined in [15] and control variables separated.
- split, U-centered+diag: DPM with distance covariance as defined in [15] but also taking the diagonal of distance matrices into account, and control variables separated.

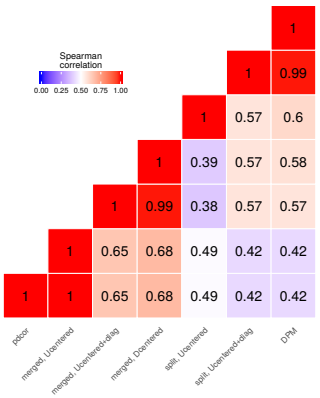
Reminder: D-centered vs U-centered. Let us assume we are computing the sample distance correlation for two random variables X and Y with n given samples $X_i, Y_i, i = 1, ..., n$. The entries of the distance matrices $\mathbf{A}^0$ and $\mathbf{B}^0$ are obtained using $a^0_{ij} = \|X_i - X_j\|_2$ and $b^0_{ij} = \|Y_i - Y_j\|_2$.

- D-centered [14]. The D-centered distance matrices $\mathbf{A}$ and $\mathbf{B}$ are obtained from $\mathbf{A}^0$ and $\mathbf{B}^0$ using $a_{ij} = a^0_{ij} - \frac{1}{n}\sum_{k=1}^{n} a^0_{ik} - \frac{1}{n}\sum_{k=1}^{n} a^0_{kj} + \frac{1}{n^2}\sum_{k,l=1}^{n} a^0_{kl}$. The sample distance covariance is then defined as the square root of $\mathrm{dcov}^2(X,Y) = \frac{1}{n^2}\sum_{i,j=1}^{n} a_{ij}b_{ij}$.
- U-centered [15]. The U-centered distance matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are obtained from $\mathbf{A}^0$ and $\mathbf{B}^0$ using $\tilde{a}_{ij} = a^0_{ij} - \frac{1}{n-2}\sum_{k=1}^{n} a^0_{ik} - \frac{1}{n-2}\sum_{k=1}^{n} a^0_{kj} + \frac{1}{(n-1)(n-2)}\sum_{i,j=1}^{n} a^0_{ij}$. The sample distance covariance is then defined as the square root of $\mathrm{dcov}^2(X,Y) = \frac{1}{n(n-3)}\sum_{i\neq j} \tilde{a}_{ij}\tilde{b}_{ij}$. Note the missing diagonal.

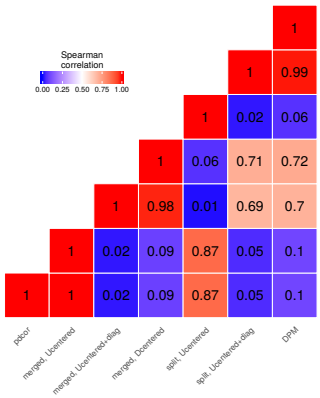
Figure 24 shows the Spearman correlation coefficients between the results of pdcor and various stages of DPM, for linear and non-linear with low sample size (left), for linear and non-linear data with high sample size (center) and for DREAM data (right).



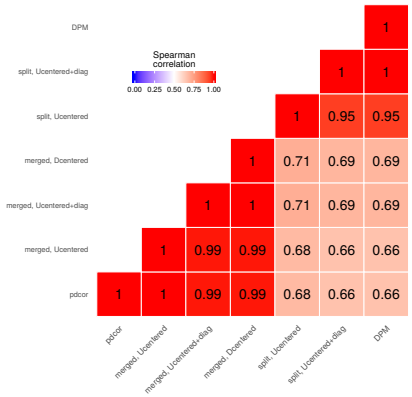
(a) 200 samples, Gaussian data, 10 nodes networks.



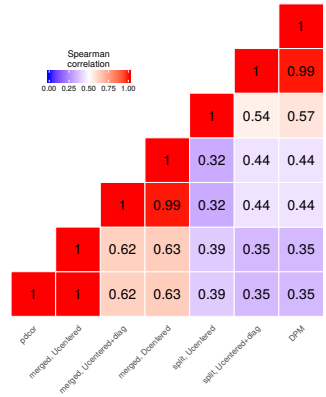
(b) 10 samples, Gaussian data, 10 nodes networks.



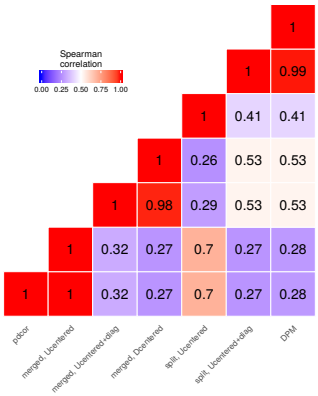
(c) DREAM3 (Ecoli1) wild-type and null-mutant data, 10 nodes networks, 11 samples.



(d) 200 samples, non-linear data.



(e) 10 samples, non-linear data.



(f) DREAM4 (1) wild-type and knockout data, 10 nodes networks, 11 samples.

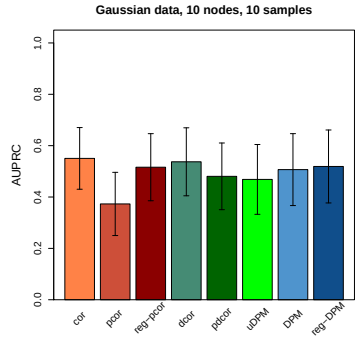
Figure 24: **From pdcor to DPM.** The plots show the average Spearman correlation coefficients between the results of pdcor and various stages of DPM. **(a, d): high sample size data.** Results averaged over 100 replicates with 200 samples per replicate. Merging control variables together when computing the distance matrices makes the most difference. **(b, e): low sample size data.** Results averaged over 100 replicates with 10 samples per replicate. The normalization has a strong impact too. **(c, f): DREAM data.** The use of the diagonal in the distance matrices has a strong impact.

Coefficients are high, which is to be expected, except for DREAM data where results tend to vary more across methods because of the low number of samples. Note that in these cases, DPM and reg-DPM do generally better than pdcor. As anticipated, pdcor and "DPM merged, U centered" have a full correlation. All other variations differ. Interestingly, these differences depend on the situation.

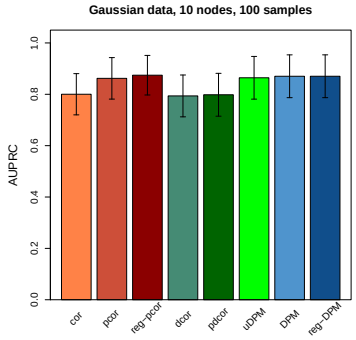
When sample size is large (left), merging control variables together when computing the distance matrices makes the most difference. The variations can practically be organized between "merged" and "split" methods. When sample size is low however (center), both the merging and the use of the diagonal make a clear difference. DREAM data (right) is interesting, as it seems that the use of the diagonal in the distance matrices when computing the dot products makes the largest difference. The variations can practically be

organized between "use diagonal" and "do not use diagonal" methods. Perhaps because the signal in the off-diagonal elements in DREAM data is very weak. The effect is less prominent in DREAM4 data.

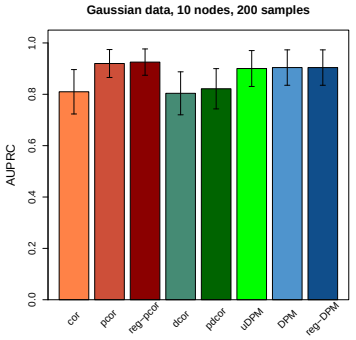
We investigated in greater depth the unbiased centering. Let us call uDPM the "split, U-centered" variant, i.e. DPM with U-centered vectors (excluding the diagonal) instead of D-centered vectors. Figure 25 shows a comparison of uDPM and DPM for different types of settings. uDPM performs worse than DPM on low sample sizes and about the same on large sample sizes (see figures below). In fact, at high sample size, there is barely any difference between the two, as expected from the theory, and as already observed in Figure 24.



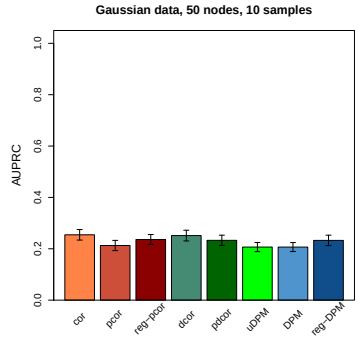
(a) Gaussian data, 10 nodes networks, 10 samples.



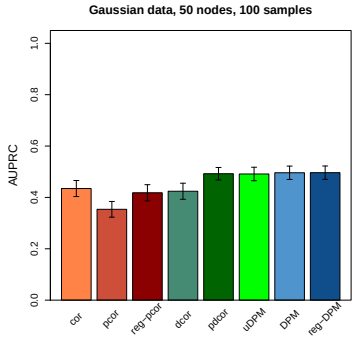
(b) Gaussian data, 10 nodes networks, 100 samples.



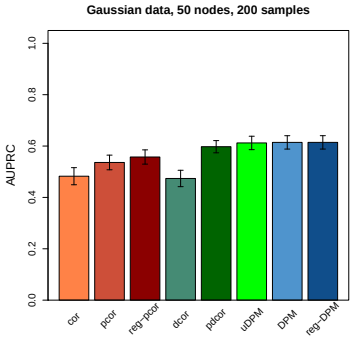
(c) Gaussian data, 10 nodes networks, 200 samples.



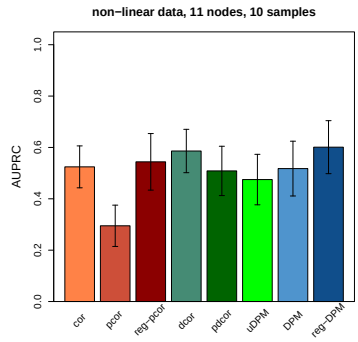
(d) Gaussian data, 50 nodes networks, 10 samples.



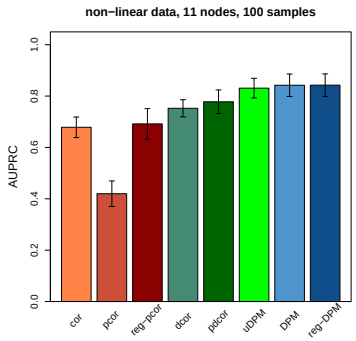
(e) Gaussian data, 50 nodes networks, 100 samples.



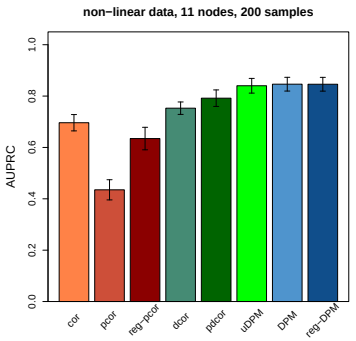
(f) Gaussian data, 50 nodes networks, 200 samples.



(g) Non-linear data, 10 samples.



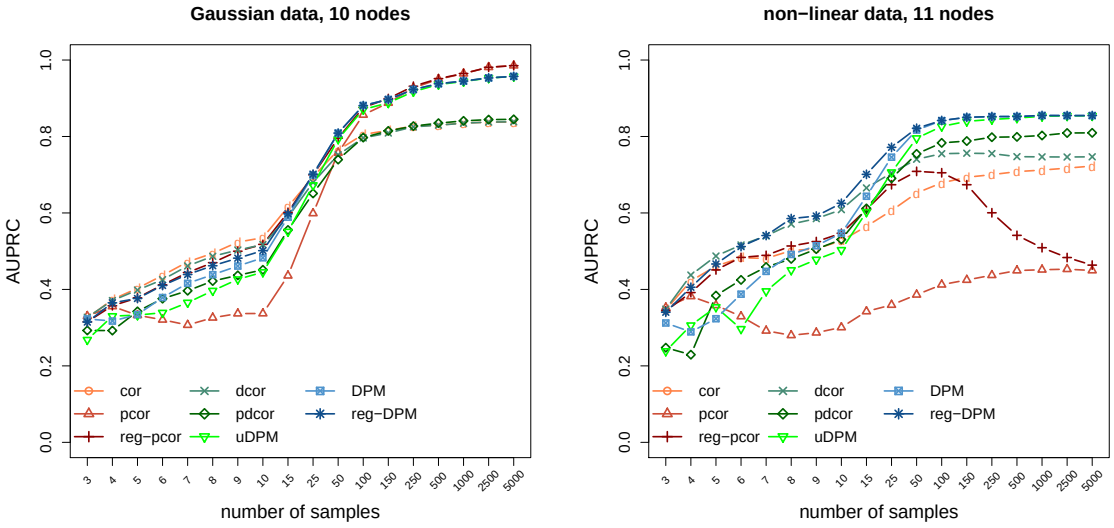
(h) Non-linear data, 100 samples.



(i) Non-linear data, 200 samples.

Figure 25: **Average AUPRCs on simulated data.** The plots show the AUPRCs (averaged over 100 replicates) of a few selected methods for the reconstruction of Gaussian networks for the first two rows, and of the network H in Main Document Figure 1 for the last row.

Figure 26 shows a comparison of uDPM and DPM as sample size varies.



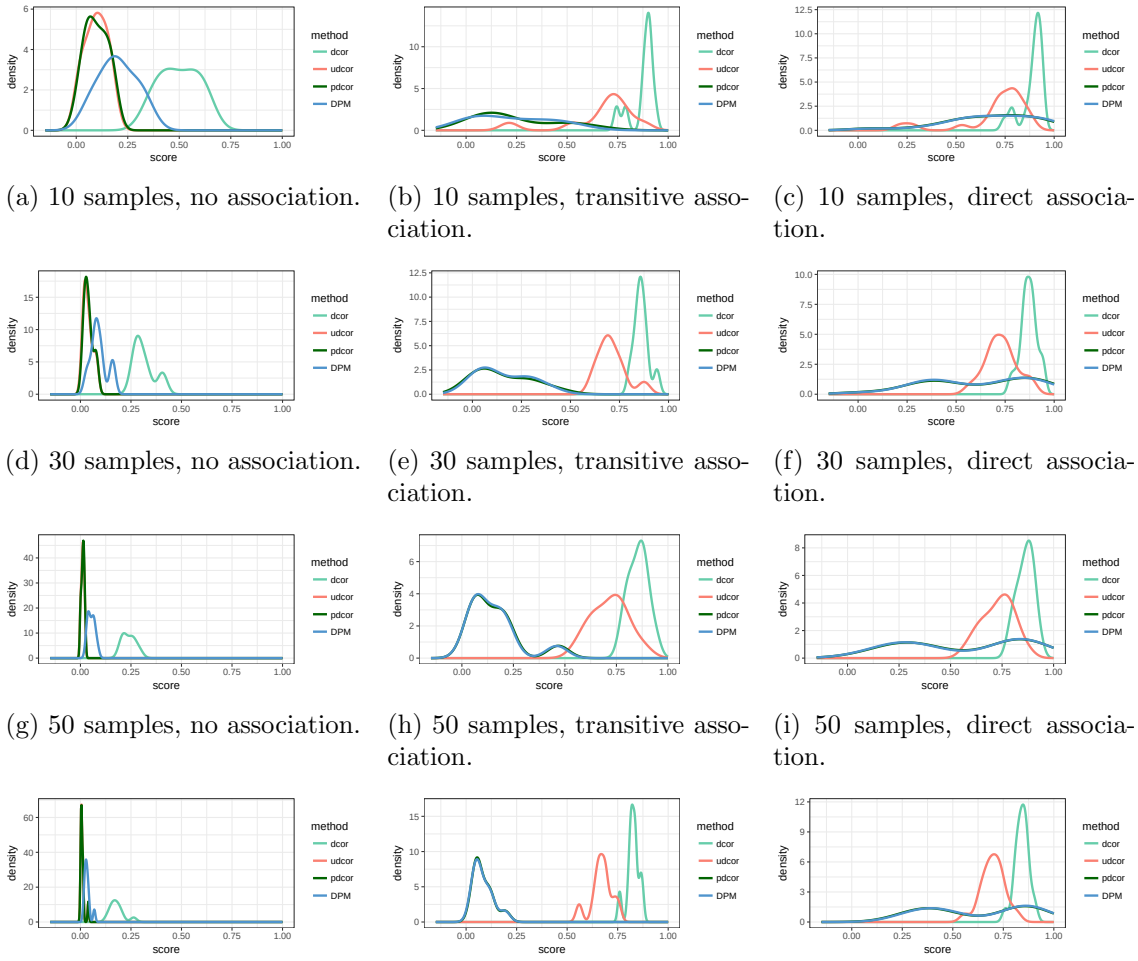
(a) Gaussian data, 10 nodes networks.

(b) Non-linear data simulated from the network H in Main Document Figure 1.

Figure 26: **Effect of sample size on performance.** The plots show the evolution of the AUPRCs (averaged over 100 replicates) as sample size increases.

16 Bias in DPM

We generate 2 kinds of networks with 3 variables: one kind with no connections, the other with two edges, i.e. two direct associations, and a transitive association (the missing edge). Figure 27 shows (for dcor, udcor, pdcor and DPM) the distributions of scores for non-edges in the left column, for transitive associations in the middle column, and for direct associations in the right column. All experiments are based on 10 replicates, and the various rows show results for various sample sizes. Ideally, scores should be around 0 in both the left and middle columns.



(a) 10 samples, no association. (b) 10 samples, transitive association. (c) 10 samples, direct association.

(d) 30 samples, no association. (e) 30 samples, transitive association. (f) 30 samples, direct association.

(g) 50 samples, no association. (h) 50 samples, transitive association. (i) 50 samples, direct association.

(j) 100 samples, no association. (k) 100 samples, transitive association. (l) 100 samples, direct association.

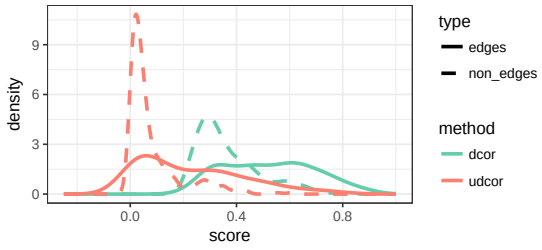
Figure 27: **Distribution of scores for pdcor and DPM on 3-nodes Gaussian networks** for different edge types and sample sizes. The set of scores is obtained by concatenating 10 realisations of the networks. **Top to bottom**: increasing sample size. **Left:** no association (3-nodes network with no edges). **Middle:** transitive or indirect associations. **Right:** direct associations.

For all methods, the bias reduces as the number of samples grows, with a bias close to 0 for udcor and pdcor. While the bias in dcor is rather large, note that the bias in DPM is much smaller, i.e. it does not inherit the bias of dcor.

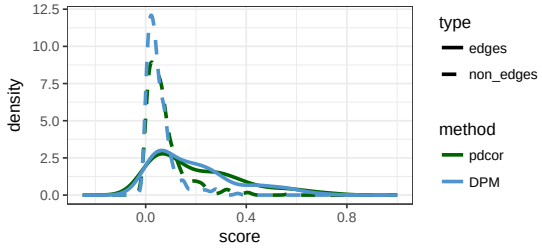
Interestingly, the mode bias of DPM does not change much between non-edges and transitive associations, which suggests that transitive associations are indeed recognised as non-edges. In contrast, the mode score of pdcor for transitive associations is notably higher than for non-edges, to the level of DPM's bias, which indicates that pdcor struggles more to discard transitive associations. This is consistent with our results in Main Document Figure 4.

It is worth noting that, for such 3-nodes networks, the conditioning is only on one input variable for both pdcor and DPM, so the difference between the two methods reduces to producing the double-centered vectors with udcor (for pdcor) or dcor (for DPM). In Figures 27(h,i,k,l), the dcores distributions for pdcor and DPM overlap, indicating that when enough samples are provided, using dcor or ucor makes very little numerical difference to the scores assigned to associations and to indirect associations.

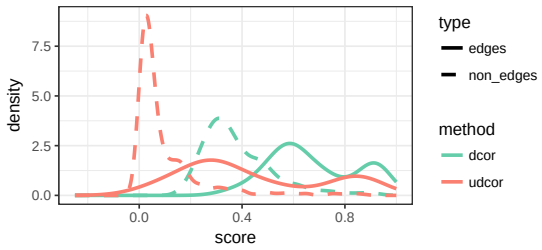
Figure 28 shows the distributions of scores for dcor and udcor (left column) and for pdcor and DPM (right column) for 10-nodes Gaussian networks (top) and our non-linear network (bottom), based on 30 samples and 10 replicates. The distributions are split between edges (solid lines) and non-edges, which include indirect associations (dashed lines). While dcor shows a clear bias, the bias of DPM is much lower, and the distribution of DPM scores is comparable to pdcor.



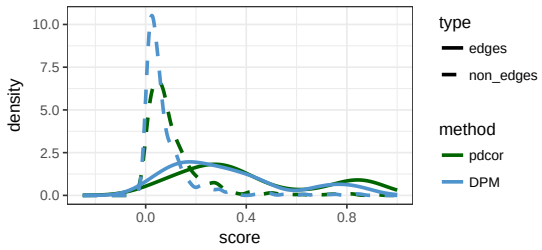
(a) 10-nodes Gaussian networks, dcor and udcor.



(b) 10-nodes Gaussian networks, pdcor and DPM.



(c) Non-linear network H in Main Document Figure 1, dcor and udcor.



(d) Non-linear network H in Main Document Figure 1, pdcor and DPM.

Figure 28: **Distribution of scores** for different edge types. The set of scores is obtained by concatenating 10 realisations of the network of 30 samples. **Top:** 10-nodes Gaussian networks. **Bottom:** non-linear network H in Main Document Figure 1. **Left:** dcor and udcor. **Right:** pdcor and DPM.

17 Runtime

Let d be the number of variables and n the number of samples. The algorithm consists of 3 phases.

- Computing the D-centered matrices takes $\mathcal{O}(d \times n^2)$ operations.
- Computing the distance covariance matrix takes $\mathcal{O}(n \times d^2)$ steps.
- Computing the matrix inversion is of complexity $\mathcal{O}(d^3)$.

Thus the overall complexity is $\mathcal{O}(n^2 \times d + d^3)$.

Figures 29 and 30 show the running times of several algorithms as sample size, respectively network size, increases. DPM is not the fastest, but it is reasonnable, and it is much faster than pdcor.

If speed is an issue, D-centered matrices can now be computed in $\mathcal{O}(d \times n \log n)$ [17] and the partial correlation matrix in $\mathcal{O}(n \times d^2)$ [18]. In general, in bioinformatics, n is rarely much larger than d and the dominant operation will be the computation of the partial correlation matrix, so we can do what partial correlation does, and the authors of FastGGM claim they can deal with 10.000 dimensions in about 20 hours, therefore we think that DPM is usable in practice.

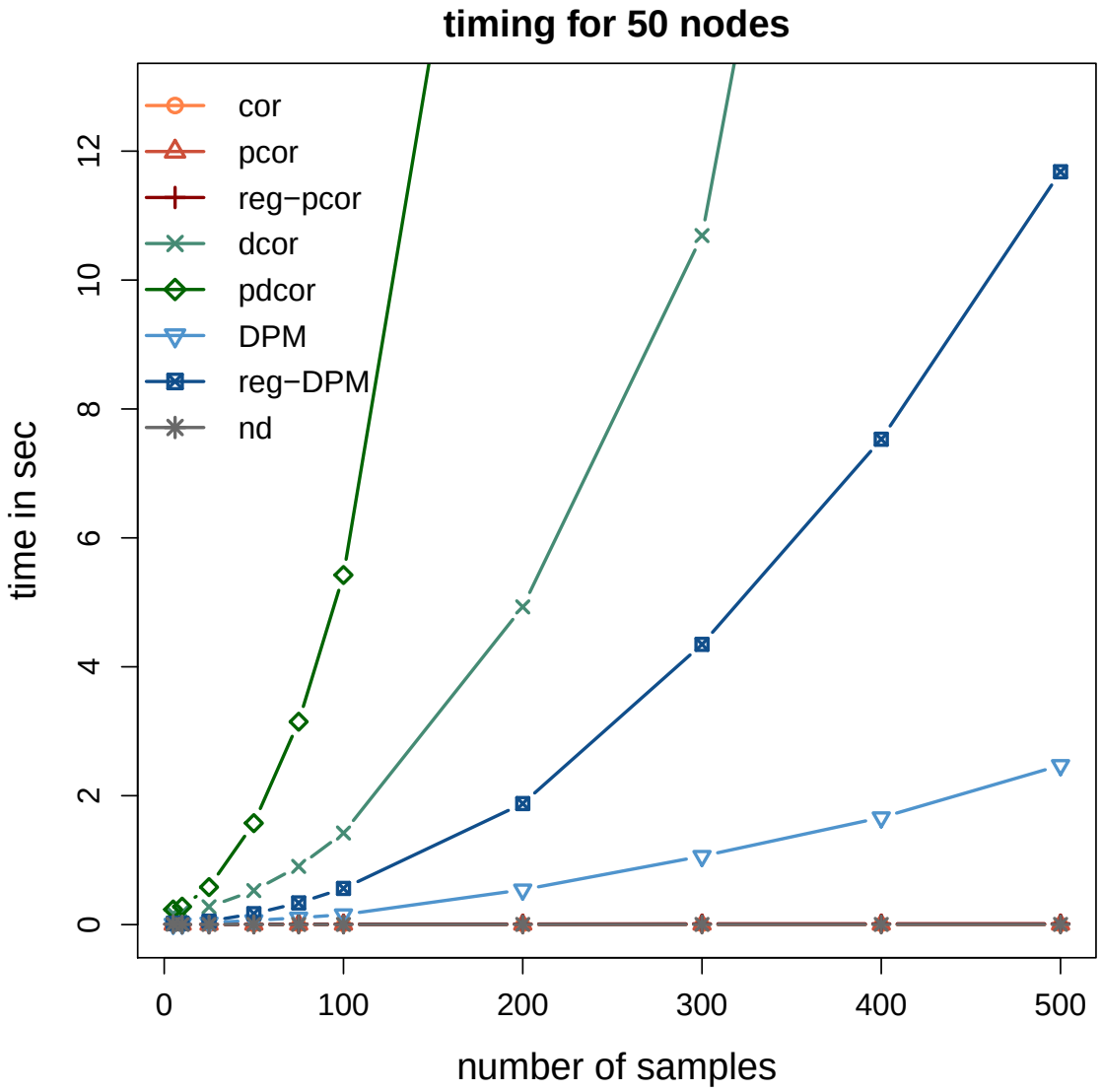


Figure 29: **Running times of various algorithms for 50 nodes networks as sample size increases.** These running times are averaged over 10 replicates. All algorithms were run on one CPU only.

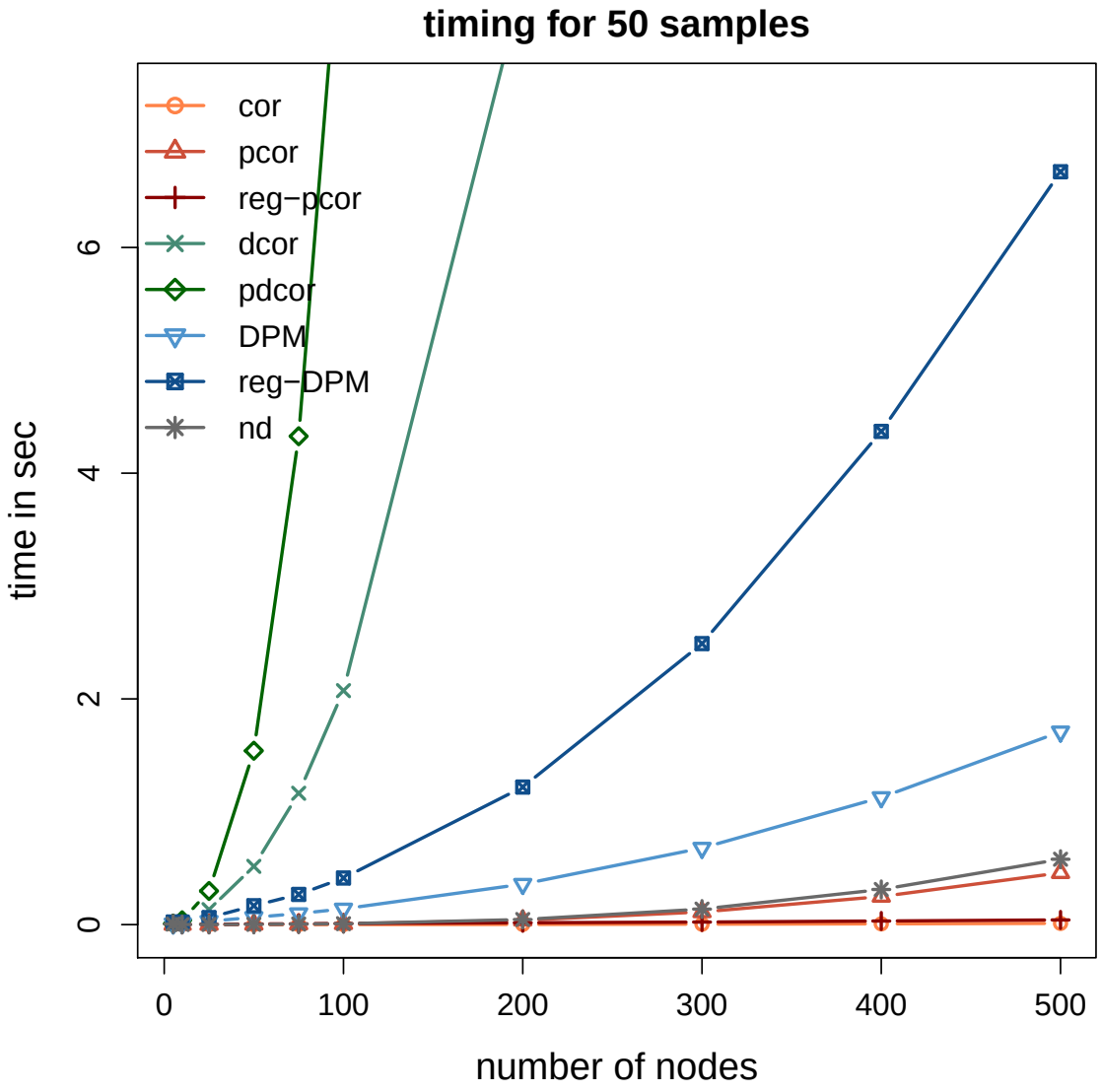


Figure 30: **Running times of various algorithms for 50 samples as network size increases.** These running times are averaged over 10 replicates. All algorithms were run on one CPU only.

References

[1] Kalisch M, Mächler M, Colombo D, Maathuis MH, Bühlmann P (2012) Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software* 47(11):1–26.

[2] Hauser A, Bühlmann P (2012) Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research* 13:2409–2464.

[3] Castelo R, Roverato A (2006) A robust procedure for gaussian graphical model search from microarray data with p larger than n. *Journal of Machine Learning Research* 7:2621–2650.

[4] Castelo R, Roverato A (2009) Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *Journal of Computational Biology* 16(2):213–227.

[5] Tur I, Castelo R, Roverato A (2014) Mapping eqtl networks with mixed graphical markov models. *Genetics* 198(4):1377–1393.

[6] Genz A et al. (2016) *mvtnorm: Multivariate Normal and t Distributions*. R package version 1.0-5.

[7] Genz A, Bretz F (2009) *Computation of Multivariate Normal and t Probabilities*, Lecture Notes in Statistics. (Springer-Verlag, Heidelberg).

[8] Saito T, Rehmsmeier M (2015) The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE* 10(3):e0118432.

[9] Margolin AA et al. (2006) Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics* 7(S-1).

[10] Schäfer J, Strimmer K (2005) A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics. *Statistical Applications in Genetics and Molecular Biology, The Berkeley Electronic Press* 4(1).

[11] Feizi S, Marbach D, Medard M, Kellis M (2013) Network deconvolution as a general method to distinguish direct dependencies in networks. *Nature Biotechnology* 31(8):726–733.

[12] Meyer P, Lafitte F, Bontempi G (2008) minet: A R/Bioconductor Package for Inferring Large Transcriptional Networks Using Mutual Information. *BMC Bioinformatics* 9(1):461+.

[13] J Zhao, Y Zhou XZ, Chen L (2016) Part mutual information for quantifying direct associations in networks. *Proceedings of the National Academy of Sciences* 113(18):e1003525.

[14] Székely GJ, Rizzo ML, Bakirov NK (2007) Measuring and testing dependence by correlation of distances. *Ann. Statist.* 35(6):2769–2794.

[15] Székely GJ, Rizzo ML (2014) Partial distance correlation with methods for dissimilarities. *Ann. Statist.* 42(6):2382–2412.

[16] Geiger D, Verma T, Pearl J (1990) Identifying independence in Bayesian Networks. *Networks* 20:507–534.

[17] Huo X, Székely GJ (2016) Fast computing for distance covariance. *Technometrics* 58(4):435–447.

[18] Wang T et al. (2016) Fastggm: An efficient algorithm for the inference of gaussian graphical model in biological networks. *PLOS Computational Biology* 12(2):1–16.