Intersection of Dataset Size and Number of Variables in Bayesian Network Structure Learning

Written by James Mann Supervised by Dr V. Anne Smith in School of Biology at the University of St Andrews

1 Introduction

Bayesian Networks (BNs) are probabilistic graphical models that can be applied to a wide range of fields and complex systems. With applications in understanding neural information flow, political science, modelling Parkinson's risk, and more [1-3]. Bayesian Networks are often learned from data using a machine learning approach. A common question when utilising this approach is how much data is needed to accurately train a Bayesian Network. There has been some research into this however no detailed work on how variable number and training sample size interact and influence training effectiveness [4]. This study aims to remedy this through a simulation approach in which the behaviour of random Bayesian Networks with synthetic data is used to model how variable number and training sample size would affect discrete Bayesian Network learning on real world datasets. This paper will also consider the effectiveness of a bootstrapping technique that expands the training sample size.

1.1 Primer on Bayesian Networks

Learning discrete Bayesian Networks from data is a two-step process, first the structure of the network must be found, which is what this study focuses on, determining which variables are directly dependent on each other. Then for discrete Bayesian Networks, which this study considers, conditional probability tables (CPTs) can be fitted to describe the specific distributions relating linked variables. Learning Bayesian Network structures from data is an NP-hard problem, while the number of possible network structures grows super exponentially with variable number [5]. This means that for anything larger than a few variables it is not practical to use deterministic methods that find the best network given the data with 100% accuracy. Instead, heuristic search methods are used, this paper only considers score-based search methods. The training sample is used to score network structures with structures that fit the data better given higher scores, this turns the search into an optimisation problem. There are several different score functions however this paper only uses the Bayesian Dirichlet equivalence score (BDe) which is popular for discrete Bayesian Networks. Bayesian Networks have directed acyclic graph (DAG) structures. Some DAG structures are statistically indistinguishable, they encode the same conditional independences, these structures are known as I-equivalent. When comparing Bayesian Network structures often the skeleton of the graph is considered, its undirected structure, so that when I equivalent graphs are compared, they are considered the same.

2 Methods

2.1 Simulation Methodology

The general simulation process is:

1. Generate a random BN of given number of variables
2. From the random BN generate a random dataset of a given number of datapoints
3. Learn a BN from created dataset
4. Compare learned BN to original BN

The simulation was made using a combination of R, with the bnlearn package and the Java application Banjo [1,6]. Initial setup and data creation is done through bnlearn while searches are done through Banjo due to the ready access to search settings Banjo provides. The initial BN structure is generated using ic-dag generation with a maximum in-degree of 5 through the bnlearn package [6]. Ic-dag generation produces connected networks which with maximum in-degree of 5 have densities which are similar to real world networks.

Conditional probability tables (CPTs) are then created for the network structure, 2 level variables were used across all the simulations. The CPTs were generated by sampling from a Dirichlet distribution with hyperparameter alpha=0.5 for a parent and alpha=5 for non-parents [7].

From this fitted BN the rbn function in bnlearn is used to generate a dataset. In practice as datasets are generated in batches, the largest dataset size needed is generated, and subsets are taken to create smaller dataset sizes. Within banjo a simulated annealing search using bde scoring is used to ensure effective searching.

2.2 Analysis Methodology

When analysing the results of the simulation, two key metrics are considered, precision and recall.

$$\text{Precison} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Equation 1: Precision

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Equation 2: Recall

Precision is a metric showing how likely any given link in the learned network is a true link, while recall shows what proportion of all true links are found in the learned network. Both precision and recall are calculated for the skeleton of the BN, i.e. the direction of edges is not considered, this means full recovery only requires finding an I-equivalent network rather than the exact DAG. Comparison of the original and learned network is done through the bnlearn library in R.

For each variable number and dataset size multiple simulations are run and the mean precision and recall of the runs are taken. With the number of variables constant logistic curves, where precision/recall is a function of log(number of datapoints), were fitted for both precision and recall. Two different parameterisations of the logistic curve were used, one that fixes the maximum value to one (Equation 3), and another that allows the maximum value to vary (Equation 4). Both curves were fitted within R using non-linear least squares curve fitting.

$$\text{Recall or Precision} = \frac{1-d}{1+e^{-k(log(n)-c)}} + \text{d}$$

Equation 3: Logistic Fit with Maximum Fixed at 1

$$\text{Recall or Precision} = \frac{L}{1 + e^{-k(log(n)-c)}} + \text{d}$$

Equation 4: Logistic Fit with Maximum Free to Vary

3 Results

3.1 Search Constraints

To understand the effect of the number of networks searched before stopping on search quality, the bde score of the learned network and the original networks were calculated. The percentage difference in bde score between the learned and original networks was used as a metric for whether results were search or score constrained. A perfect simulation that recovers the original network completely would give a percentage bde difference of 0. A search constrained simulation would have a positive value indicating the search was not finding the global optimum. While a score constrained search would give a negative value indicating more data is needed to score networks effectively and recover the original network. Increasing the number of networks searched has a major impact on CPU time and makes some search constraints unavoidable.

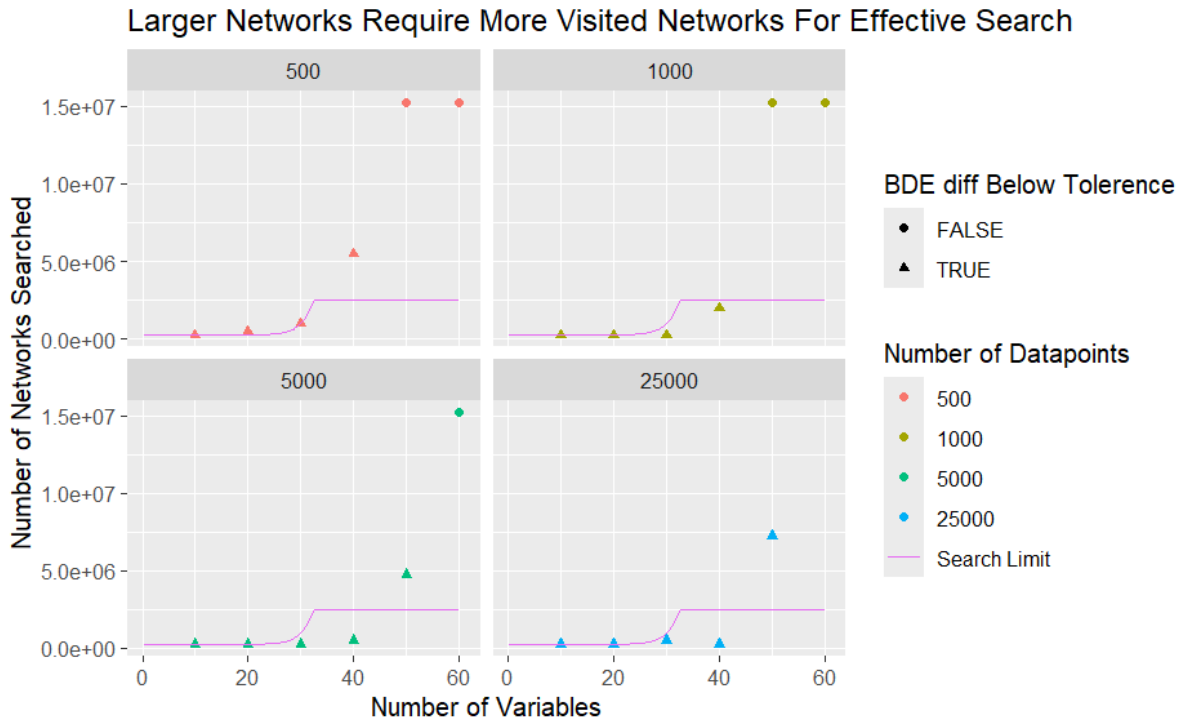

Figure 1: Scatter showing number of networks needed for effective search

To determine the number of networks to search and examine the effect of dataset size and variable number on search, a simulation was run. For an array of variable numbers and dataset sizes a network would be learned using 250,000 x *i* networks searched, where *i* is the run id; if

the bde difference was <=1% the sim would stop and that number of networks would be considered the threshold for effective search.

From the simulation we can see that effective search is more difficult for low numbers of datapoints, but the key factor is the number of variables. This is likely due to the super exponential increase in search space size. From this simulation work the number of networks to search was set as a function of number of variables with a cap of 2.5 million (See pink line on Fig 1). Searches that require more than the given number of networks for effective search are considered search constrained. For the search limit used, networks with 30 or more variables are search constrained.

3.2 Base Case

We ran 10 simulations for each number of variables over multiple training sample sizes. Ranging from 10 to 50 000 datapoints. In the base case with no bootstrapping, we can see how the behaviour of search constrained simulations differs from those without search constraints. In Figure 2 the low variable number searches show a sigmoid curve on a log scale for recall with top asymptotes at 1, indicating full recovery. While for search constrained runs the top asymptote does not reach 1 indicating that even with increasing numbers of datapoints we do not get full recovery due to search constraints. Precision increases to an asymptote quicker than recall, consequently small sample size searches can provide high precision despite low recall. Precision shows similar behaviour to recall under search constraints with the top asymptote level sinking below one. For large networks, >=40 variables, precision drops off slightly despite recall staying constant, indicating searches are recovering networks with too many links, possibly due to the more constrained searches at high variable counts.

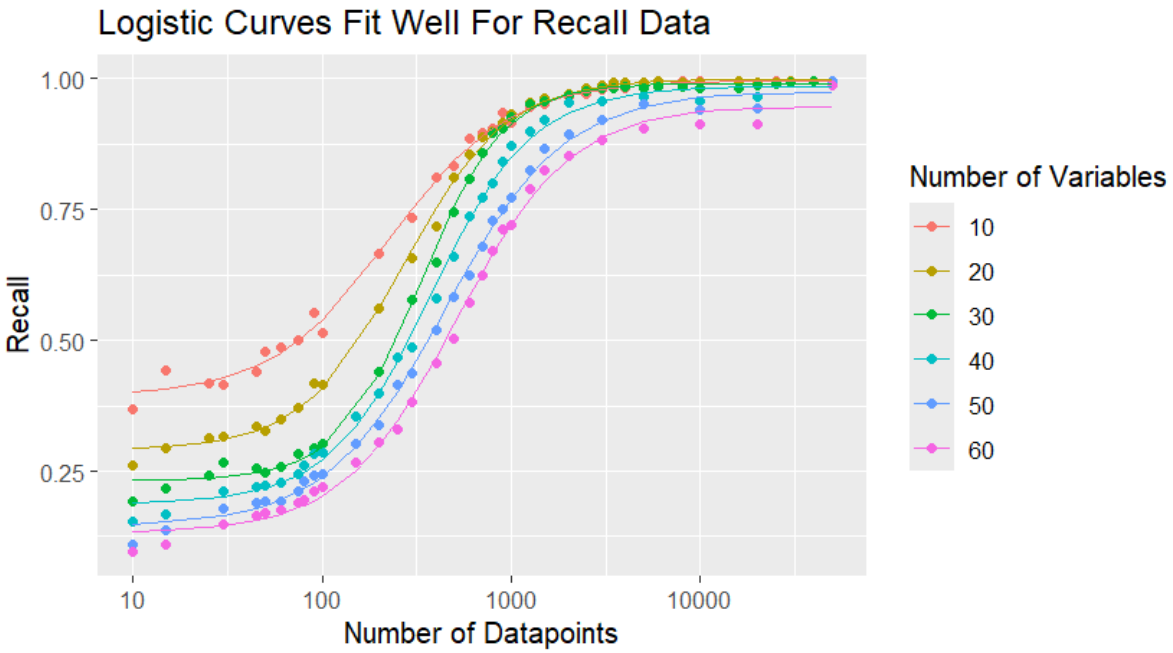

Figure 2: Scatter of mean recall data for base case with fitted curves (Equation 2)

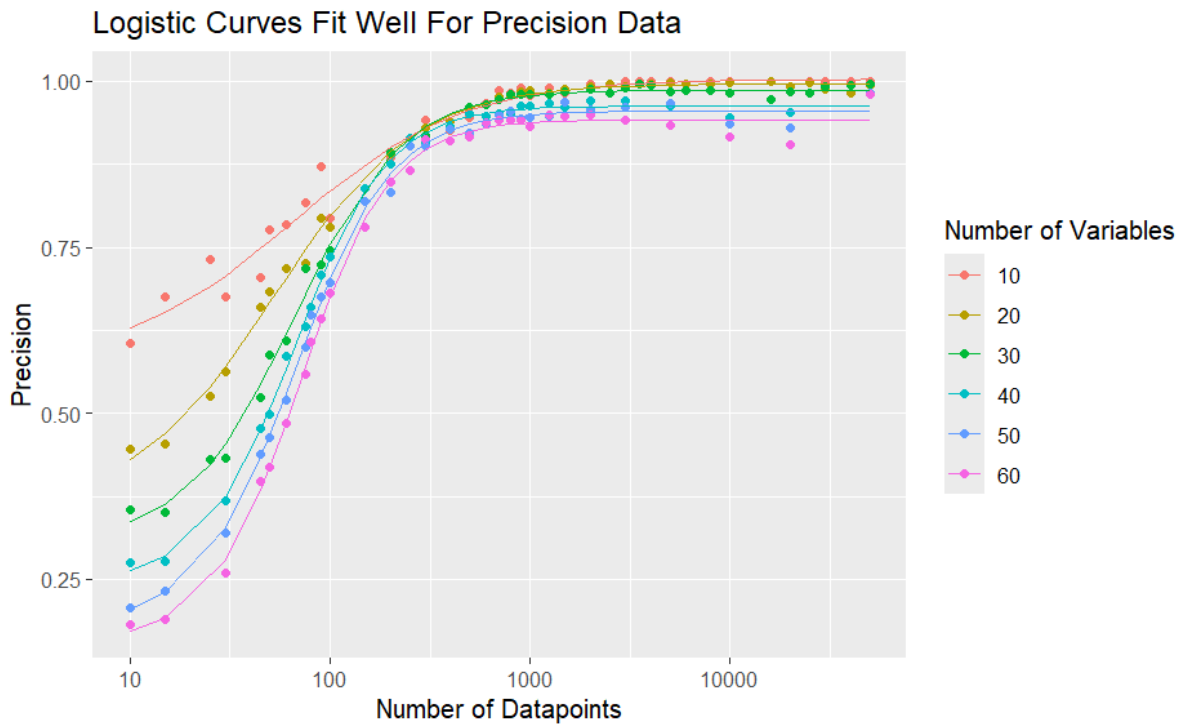

Figure 3: Scatter of mean precision for base case with fitted curves (Equation 2)

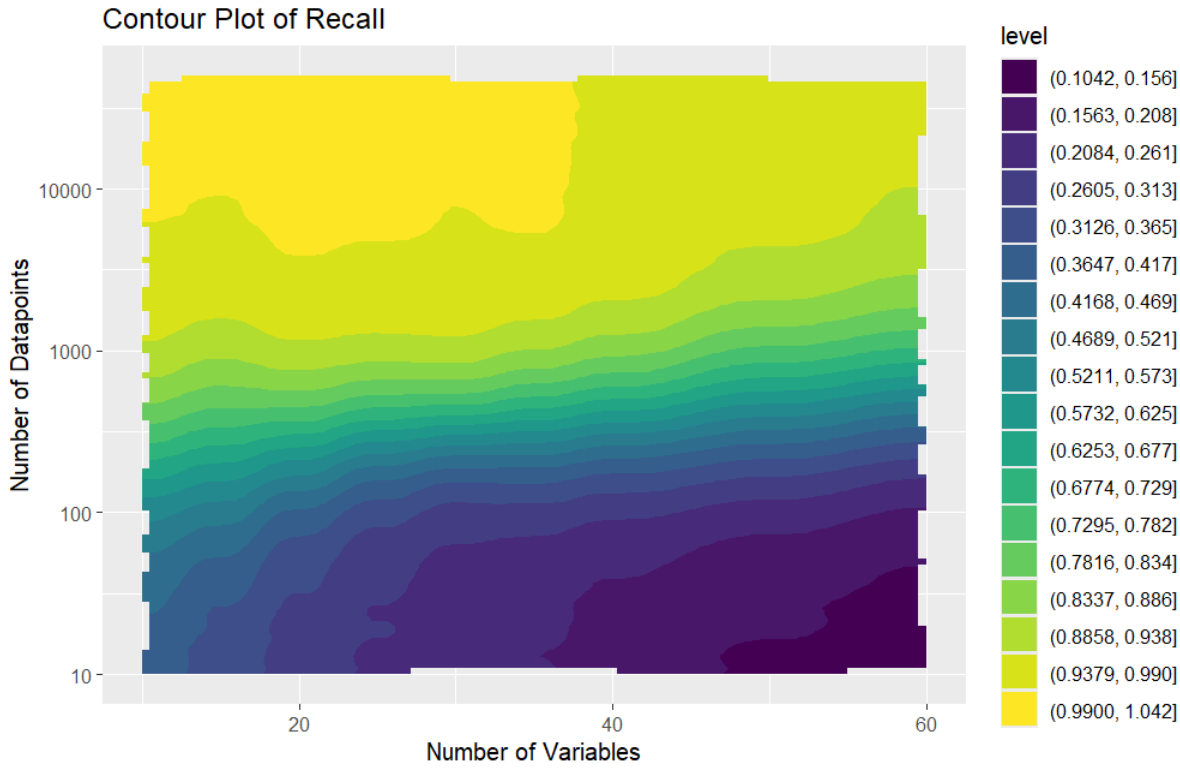

Figure 4: Contour plot of recall, interpolated using splines, colour corresponds to recall

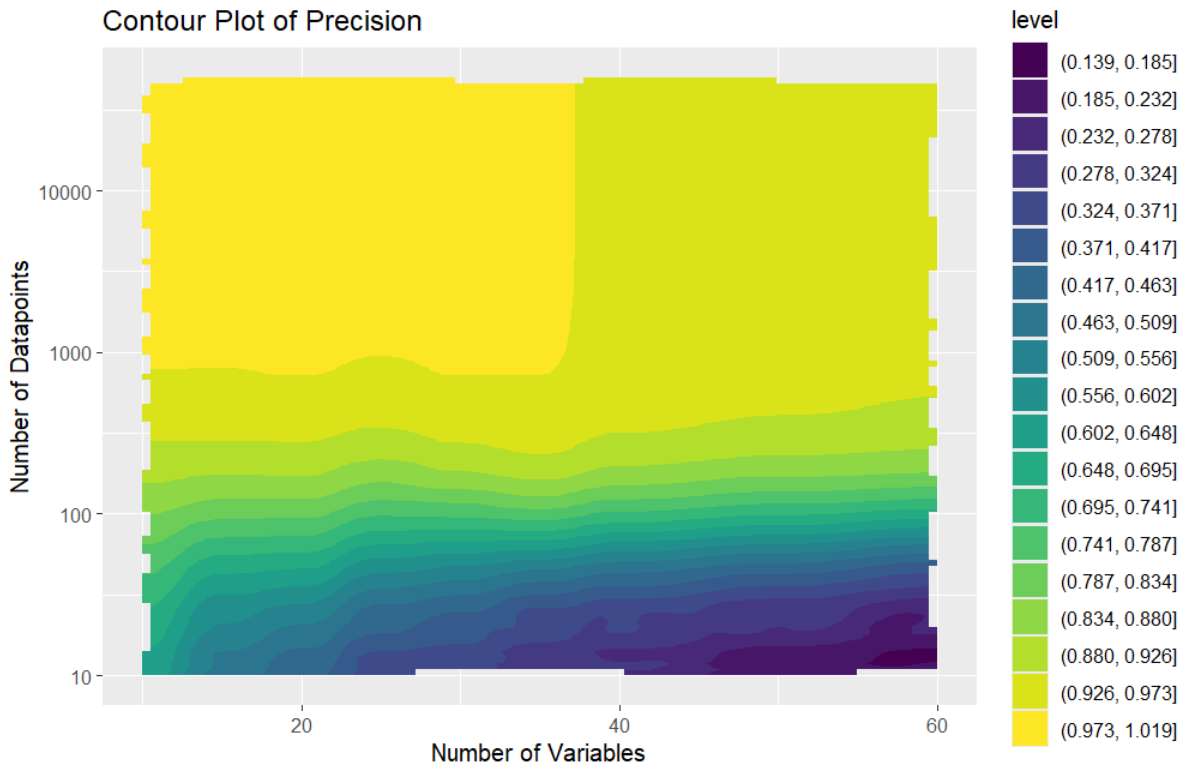

Figure 5: Contour plot of precision, colour corresponds to expected precision from spline interpolation

To quantify trends in the behaviour of precision and recall, and allow for extrapolation to different variable, counts full plots of parameter values of both sigmoid fits for the base case can be found in the appendices. Figures 6 and 7 show the max precision and recall for the

unfixed sigmoid fits which is calculated from the L and d parameters in Equation 4. Here again we can see different behaviours for higher variable counts where search constraints exist.

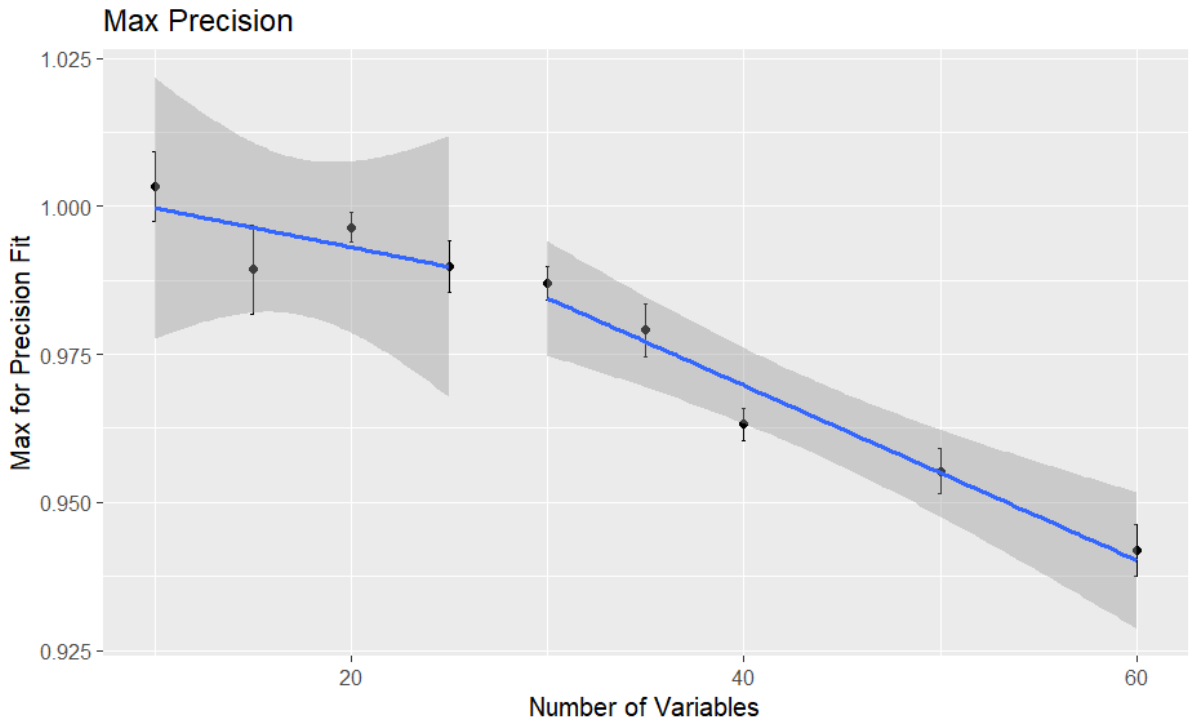

Figure 6: Plot of max precision for unfixed log sigmoid with 2 best fit lines for search constrained/unconstrained and 95% confidence interval of the fit, bars indicate errors from fitted parameters

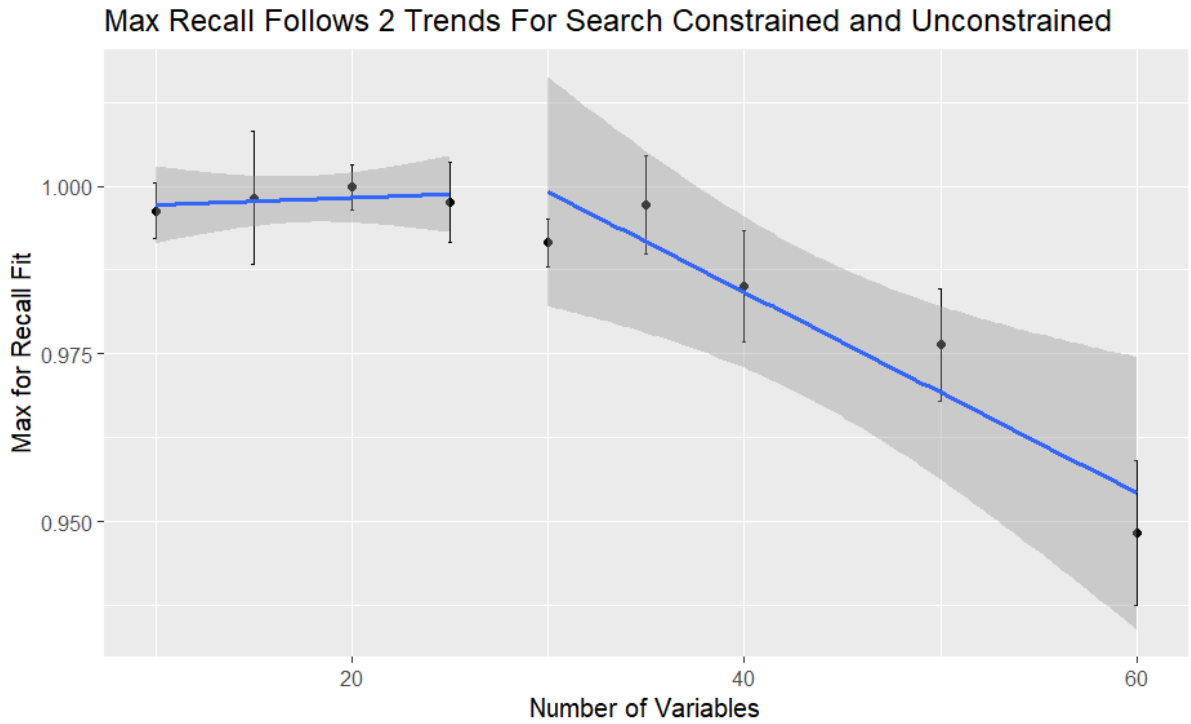

Figure 7: Plot of max recall for unfixed log sigmoid with 2 best fit lines for search constrained/unconstrained and 95% confidence interval of the fit, bars indicate errors from fitted parameters

*Table 1: Table showing minimum sample sizes for recall and precision above various thresholds. Calculated from fixed sigmoid fit parameters. *High thresholds, like 0.99, have high errors associated with them. **Higher variable count networks are increasingly search constrained and as a result have more uncertainty in the quality of fixed sigmoid fit.*

Number of Variables	Sample Size Needed For 0.8 Recall	Sample Size Needed For 0.9 Recall	Sample Size Needed For 0.99 Recall*	Sample Size Needed For 0.8 Precision	Sample Size Needed For 0.9 Precision	Sample Size Needed For 0.99 Precision*
10	390	767	4793	72	194	2073
20	485	835	3827	102	214	1648
30**	582	912	3270	126	231	1256
40**	791	1365	6518	141	252	1349
50**	1172	2239	14567	162	302	1853
60**	1513	2978	21357	179	331	1984

3.3 Bootstrapped Samples

To determine the effects of using bootstrapping to expand the training sample size to one larger the original sample, 10 simulations were run for each number of variables over an array of dataset sizes. This was done three times for 3 different scales of bootstrapped sample. As a percentage of the original sample size: 100%, 150%, and 200%. The bootstrapping was done within R, sampling with replacement. Figure 8 shows the effect of the different bootstrap sample sizes on precision. There is no noticeable difference for precision between bootstrap sizes, but the non-bootstrapped base has better precision outside of the asymptote regions. In contrast Figure 9 shows that increasing the sample size through bootstrapping does lead to an

increase in recall when outside the asymptote regions .

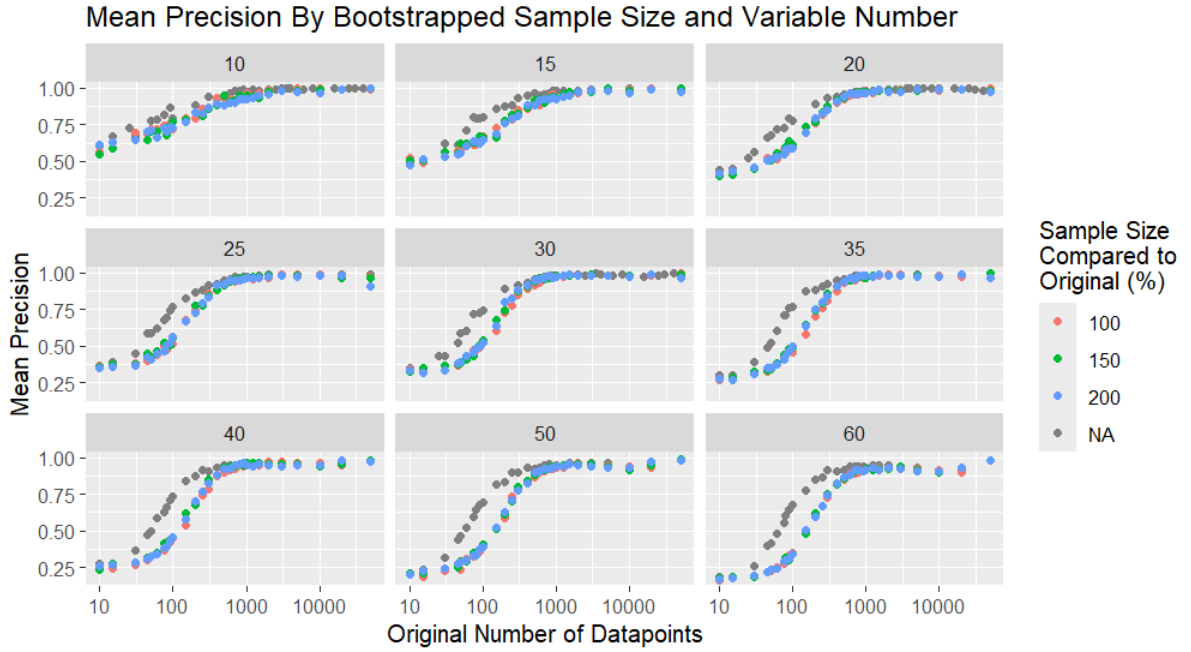

Figure 8: Scatter of mean precision, grouped by number of variables coloured by relative sample size, with base case data in grey

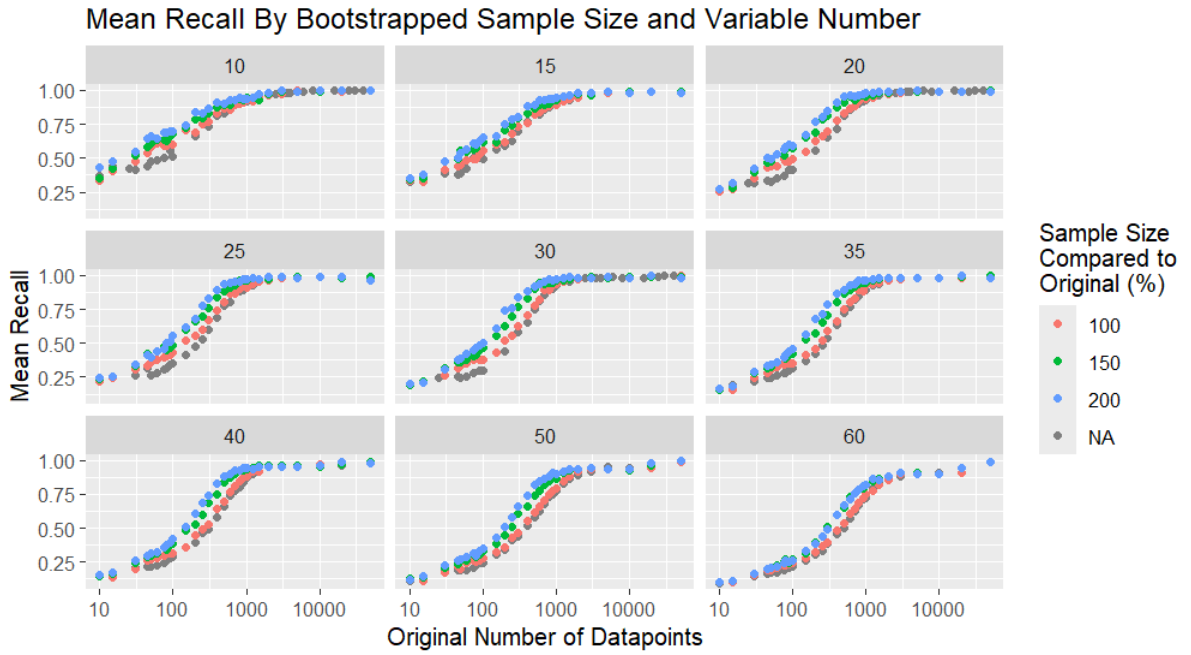

Figure 9: Scatters of mean recall, grouped by number of variables coloured by relative sample size, with the base case data in grey

4 Discussion

4.1 Practical Usage

This research examined the relationship between number of variables and training sample size for random Bayesian Networks. Through this work applications of Bayesian Networks can

understand how large of a sample size is needed for a given level of accuracy or precision. It also enables researchers with fixed training samples to understand how appropriate a Bayesian Network analysis of their data would be given the number of variables in consideration and sample size. Figures 4 and 5 as well as Table 1 are the best references for practical use.

4.2 Applications for Bootstrap Expansions of Sample Size

By testing different increases to sample size through bootstrapping this work has shown where this technique is applicable. For most situations due to the extra CPU time needed to run searches with larger dataset sizes; alongside the mixed effects on precision and recall from using bootstraps make this technique a detriment. However, if the training sample size is in the asymptote range for the relevant number of variables and the goal is to maximise the recall with no consideration of precision; this could be a useful technique.

4.3 Limitations

The simulated data used does not necessarily map on to real world data that is used in Bayesian Network applications one to one. It lacks the added complexity and noise of real-world data. This simulation also exclusively used 2 level variables with CPTs from a Dirichlet distribution, however there are a multitude of very different dependencies a Bayesian Network can encode. Applying these results to real data comes with the tacit assumption that the system of interest can be represented by a Bayesian Network which is not necessarily true. However, simulation work like this serves as a good basis for further work that could take some of these factors into account.

4.4 Possible Areas for Future Research

This research only considered small networks, the results here cannot necessarily be extrapolated to large networks with 100+ variables, or to different techniques for structure learning in large networks such as divide and conquer methods [8].

The question of search constraints was naturally thrown up in the process of working on this project. Future simulation studies could evaluate the effect of networks searched on recall and precision. There is also space for work that investigates the structure and scoring of search spaces and its interaction with variable number and training sample size.

There are minor details such as the decrease in precision when bootstrapped and the slight fall of precision for high variable networks as dataset size increases. Which could both be jumping off points for further research.

5 Conclusion

This study has evaluated the intersection of dataset size and number of variables for Bayesian Networks through simulation methods. Higher numbers of datapoints have produced better recovery, and high variable networks have been shown to require more datapoints for effective recovery. High precision was found to require smaller sample sizes than high recall, producing useful references for future applications of Bayesian Networks. Bootstrapping to increase

sample size has also been considered and areas this technique may be useful in have been identified. The work done here can be a useful tool in future applications of Bayesian Networks, and can be built on in further research into the properties of BNs.

Acknowledgements

I would like to thank Lord Laidlaw and the Laidlaw Foundation for the funding and support that made this work possible.

Thank you to my supervisor Dr. V Anne Smith for her insights and encouragement.

Thank you to Xuejia Ke for providing the R code which formed the foundation for the simulation work done in this paper.

References

1. Smith VA, Yu J, Smulders TV, Hartemink AJ, Jarvis ED (2006) Computational inference of neural information flow networks. PLoS Comput Biol 2(11): e161. doi:10.1371/journal.pcbi.0020161
2. Kopacheva E (2021) Predicting online participation through Bayesian network analysis. PLoS ONE 16(12): e0261663. https://doi.org/10.1371/journal.pone.0261663
3. Sood M, Suenkel U, von Thaler A-K, Zacharias HU, Brockmann K, Eschweiler GW, et al. (2023) Bayesian network modeling of risk and prodromal markers of Parkinson's disease. PLoS ONE 18(2): e0280609. https://doi.org/10.1371/journal.pone.0280609
4. Marco Scutari, Catharina Elisabeth Graafland, José Manuel Gutiérrez (2019) Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. International Journal of Approximate Reasoning, Volume 115. https://doi.org/10.1016/j.ijar.2019.10.003
5. Chickering DM. Learning Bayesian networks is NP-complete. In: Learning From data: artificial intelligence and statistics V. New York: Springer; 1996. p. 121–30.
6. Scutari M (2010). "Learning Bayesian Networks with the bnlearn R Package." *Journal of Statistical Software*, **35**(3), 1–22. doi:10.18637/jss.v035.i03
7. Ke, X., Keenan, K. & Smith, V.A. Evaluation of Bayesian network scoring functions in polychotomous data analysis. *Discov Data* **3**, 18 (2025). https://doi.org/10.1007/s44248-025-00033-7
8. Gu J, Zhou Q. Learning Big Gaussian Bayesian Networks: Partition, Estimation and Fusion (2020) Journal of Machine Learning Research, volume 21, pg1—31. http://jmlr.org/papers/v21/19-318.html