G-RIPS Sendai 2025

Fujitsu Group

Final Report

Authors:
Jamin Kochman[1]
Nicole Lacey[2] ☺
Joseph Nyingi[3]
Jia Hui Sim[4]

Mentors:
Molly Noel[5] +
Jorge Gutierrez*
Hiroyuki Higuchi*
Kei Fujita*

[1] University of Kentucky
[2] University of Utah
[3] Gifu University
[4] Tohoku University
[5] Rensselaer Polytechnic Institute
[+] Academic Mentor
[*] Fujitsu Industry Mentors
[☺] Project Manager

Abstract

For much of human history, science has sought to explain the world we live in by understanding causal relationships. The development of causal inference algorithms has opened an avenue to expedite this process. Fujitsu Causal Discovery is an AI tool based on the DirectLiNGAM algorithm which generates causal graphs from a multivariate table of data. In this project, we improve the discoverability of Fujitsu Causal Discovery by evaluating the tool's reliability under various conditions and creating systems to detect those conditions. We also develop a user-friendly interface, based on our results, to provide feedback on data quality and potential avenues for improvement.

August 7, 2025

「守破離」*(shu-ha-ri)*

First, follow the rules.
Then, break the rules.
Finally, transcend the rules.

—The path to mastery in Japanese arts

"The road to wisdom? — Well, it's plain and simple to express:
Err and err and err again but less and less and less."

—Piet Hein

Contents

1 Introduction

With computational tools widely available, data analysis has become a streamlined process, enabling the general public to analyze virtually any dataset in record time. However, to truly understand our data, it is necessary to not only identify the correlation between two variables, but also the causation. Previously, causal relationships have been found through experiments, but these are often impractical or unethical. For example, longitudinal studies of diseases can take several decades to produce results. Fujitsu Causal Discovery AI can instead find these relationships using DirectLiNGAM, an algorithm designed to find causal relationships in data. We propose enhancements to the discoverability of this causal discovery tool by providing a measure of data quality. We define discoverability as the ability to find previously unknown reliable causal relationships, or lack thereof, in correlational data. Reliability can be measured by the statistical likelihood that a suggested relationship holds true. These enhancements will quantify the reliability of Fujitsu Causal Discovery AI results in a way that is currently lacking. We believe there is a difference between a possible causal relationship and a true causal relationship. By providing this measure of reliability we aim to certify the veracity of results, enhancing the ability to discover true relationships.

This is significant because misinterpretation of data patterns can result in flawed policy decisions and inefficient allocation of resources. Correlation alone cannot tell us what will happen if we change something in the system, but causation can. In fields such as health, education, environmental science, and crime, understanding cause and effect is important in the design of effective solutions and in the mitigation of misinformation. By developing robust methods to distinguish true causal relationships from mere correlations, we can make informed and impactful decisions while achieving a more scientifically literate society.

There are two key questions to consider when uncovering new causal relationships without access to conclusive experimentation: can we find convincing causal insights from just observational data? This was the topic of the 2024 G-RIPS Fujitsu project [1]. Furthermore, how do we evaluate the quality of these results? This process is always affected by real-world constraints, such as biased or incomplete data, which limit the accuracy of causal inferences. This prompts the need for robust evaluation metrics and benchmarks. We address this need by combining two central ideas. One, evaluate adherence to the assumptions of DirectLiNGAM for an arbitrary dataset. Two, analyze the effects of violating these assumptions on the accuracy of data with known ground truths. Combine these results into a single score representing reliability of DirectLiNGAM predictions.

1.1 Background

DirectLiNGAM (Direct Linear Non-Gaussian Acyclic Model) is a causal discovery algorithm that recovers the structure of a causal graph from observational data [10]. This algorithm operates on the principle that, if a variable causes other variables, when applying Gaussian noise to the 'effect' variable, we will be able to recover the original data. Or, at least more closely than if we had applied noise to the 'cause'.

The main idea of DirectLiNGAM is to identify the most independent variable (the root cause) in a dataset. This is done by iteratively searching for a variable that cannot be expressed linearly by any of the other variables. Once the most independent variable is found, its effect is removed from the remaining variables. This process is repeated to identify the next variable in the causal order until the complete ordering is determined.

Then, DirectLiNGAM performs linear regression on each variable with every variable that comes after it, according to the causal ordering, to determine direct causal effects. An adjacency matrix of this causal graph is produced based on the presence of direct causal effects between variables in the dataset. More details can be found in section A.1.

It has been proven that if a dataset meets the following assumptions, then DirectLiNGAM is guaranteed to converge to the correct solution.

- Linearity: Each variable is a linear combination of its cause(s) and an independent noise variable.
- Non-Gaussianity: Each noise term of variables in a causal relationship is non-Gaussian and statistically independent of the others.
- Non-confoundedness: All causes of the variables are included in the dataset.

- Acyclicity: There are no feedback loops or reciprocal causal relationships in the causal graph of the data.
- Infinite data: The data is infinite, and therefore sufficient to identify causal relationships within the data.

1.2 Project overview

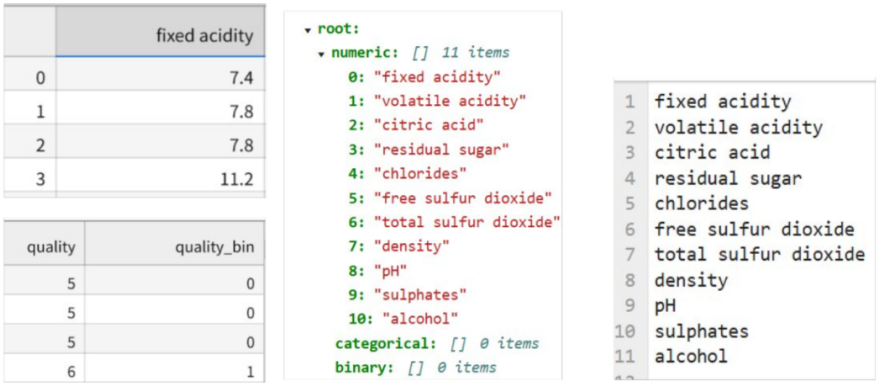
We quantify the reliability of DirectLiNGAM results for any observational dataset. To do this, we measure the impact of violating DirectLiNGAM assumptions on the accuracy of results using generated data. Then, we develop methods for detecting these violations in an arbitrary dataset, and validate these methods with the aforementioned generated data. These results then inform our measure of reliability on DirectLiNGAM results for any dataset. This reliability score is how we propose to enhance discoverability.

To accomplish this, we do the following. First, section 2 is a preprocessing step, mainly for the ease of the user. Next, we investigate the severity of DirectLiNGAM assumption violations in section 3 using flawed data and various accuracy metrics. Then we predict those accuracy metrics with a linear model, using the violations of the datasets as the predictors. Following this, we develop original methods to detect these violations in an arbitrary dataset in 4. Combining all previous steps, we formalize a measure of reliability in section 5. Last, we extend this work, in section 7.6, to a wider user base by developing an interface.

2 Data preparation

The causal discovery tool requires three input files to run: a clean `.csv` file that contains the dataset, a `.txt` file that contains a list of the numerical variables of the dataset, and a `.JSON` file that contains a list of all variables, sorted by variable type. The clean dataset must have an index column, headers, and a binarized target variable, which is a copy of any one variable, but binarized. We create an automated pipeline to generate the required files from a raw `.csv` file, see figure 1. This includes the creation of a clean version of the data. If it does not contain headers, the user is asked to name the columns in the dataset. This tool also requires a target variable to begin the binarizer process, so the user is prompted to choose a target variable and define its cutoff value. For categorical variables, the user selects one of the categories as the positive class. For numerical variables, the user specifies a threshold to binarize the variable. The binarized target variable is not used in our project, but is necessary to express the full functionality of the tool.

The information from all user inputs is saved, and if the same `.csv` file is used, then users have the choice to use previous configurations or to overwrite them with new information. This preprocessing pipeline ensures that the user only needs to provide information once per dataset which improves efficiency. It also ensures that the Fujitsu Causal Discovery tool receives correctly formatted input. One last addition is the automation of all Fujitsu Conditional Causal Discovery code. More details on this can be found here: A.2.



	fixed acidity
0	7.4
1	7.8
2	7.8
3	11.2

quality	quality_bin
5	0
5	0
5	0
6	1


```
root:
  numeric: [] 11 items
    0: "fixed acidity"
    1: "volatile acidity"
    2: "citric acid"
    3: "residual sugar"
    4: "chlorides"
    5: "free sulfur dioxide"
    6: "total sulfur dioxide"
    7: "density"
    8: "pH"
    9: "sulphates"
    10: "alcohol"
  categorical: [] 0 items
  binary: [] 0 items
```


```
fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
```

Figure 1: Example data preparation files

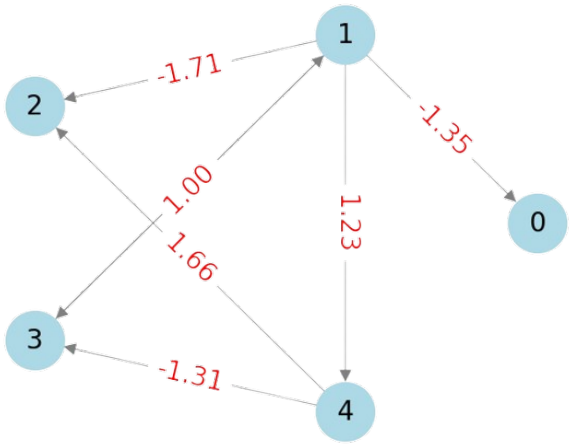


Figure 2: Ground truth graph generated by `causally`

3 Determining severity of assumption violations

In this section, we investigate the repercussions of violating assumptions of DirectLiNGAM. To do so, we generate data that intentionally violate these assumptions to varying degrees. Then, we use DirectLiNGAM to infer their causal structures and evaluate the accuracy of the predictions. We analyze these accuracy scores to determine the effects of violating different DirectLiNGAM assumptions.

3.1 Data generation

To evaluate causal discovery using DirectLiNGAM under assumption violations, we generate datasets with ground truth causal structures that explicitly violate DirectLiNGAM assumptions. First, to generate acyclic datasets, we use the Python package `causally` [6]. We begin by generating a ground truth adjacency matrix that represents the causal relationship between variables.

Using `causally`, we call causal mechanisms to simulate linear and nonlinear relationships between causally-related variables. This process also applies Gaussian or uniform noise, denote this ϵ. For example, consider the variable x_1, randomly initialized within a preset range. Then, for x_1 and x_2 related linearly with slope a, we calculate $x_2 = ax_1 + \epsilon$. The nonlinear model uses a neural network with one hidden layer. We can represent transformations to and from the hidden layer with adjacency matrices, say A_1 and A_2. Then, $x_2 = A_2(A_1x_1) + \epsilon$.

To introduce latent confounders we expand the ground truth matrix with additional nodes to the dataset. This introduces hidden common causes, i.e. unobserved variables that have an impact on the observed data, without explicitly revealing them to the discovery algorithms.

To generate cyclic datasets, we bypass `causally` and manually construct cyclic adjacency matrices by injecting 2-cycles. See figure 2. We use the same causal mechanisms as `causally` to generate linear and nonlinear data. We introduce confounders to the data in the same way, as well. For data that comes from nodes in 2-cycles, we allow data to propagate through multiple time steps, which simulate feedback loops. We do this by overwriting a randomly selected proportion of rows for four time steps, each step going from one variable to the other. We only consider 2-cycles because the complexity of creating cycles of length 3+, and the difficulties of detecting these cycles in data put this option beyond the scope of our project. Once the 2-cycle relationships were generated, the rest of the dataset was produced as normal.

We generate synthetic datasets by combining different configurations as shown in the chart below (figure 3). For each unique combination of these settings, we used 20 different seeds for a diverse set of data for analysis. This results in a total of 3,840 datasets. The elements under (Rows) are column multipliers, that is, for 2 from (Cols) and 100 from (Rows), there would be 2 columns and 200 rows in the dataset. Thus, that number of rows are added to the dataset for each addition of a column. Additionally, all files necessary for DirectLiNGAM and post-processing analysis are generated. See figure 4.

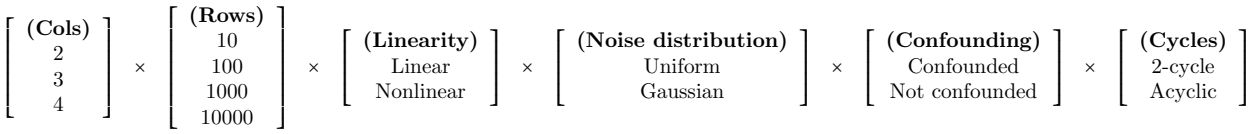


Figure 3: Configurations for data generation

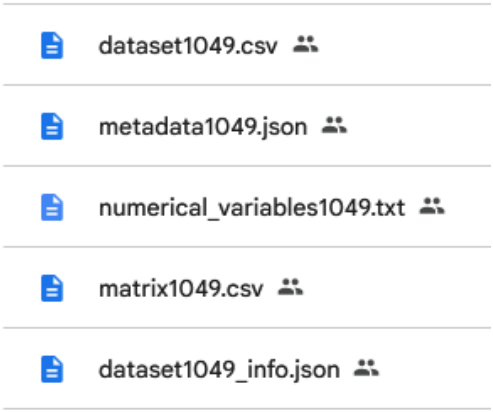


Figure 4: Dataset 1049 and associated files

3.2 Accuracy testing

After generating datasets that incorporate various violations of the DirectLiNGAM assumptions, we aim to evaluate how these violations affect the algorithm's performance. Specifically, we want to quantify the extent to which each type of violation degrades the accuracy of the recovered causal structure.

We consider three accuracy measures which are commonly used in causal discovery: the F1-score, Structural Hamming Distance, and Frobenius norm [13]. We also develop a modification to the F1-score specialized to our context. We use the F1 score because of its performance under sensitivity analysis. Details can be found here: A.6.

The F1-score is a measure of predictive performance. It is given by

$$F1 = \frac{TP}{TP + 0.5(FP + FN)}$$

where

- TP: True positive, when the algorithm correctly predicts an edge.
- FP: False positive, when the algorithm predicts an edge when there isn't one.
- FN: False negative, when the algorithm does not predict an edge when there is supposed to be one.

The F1-score lies in the range $[0, 1]$, where a score of 1 indicates a perfect prediction, while a score of 0 indicates that DirectLiNGAM did not predict any edges correctly. However, the typical F1-score does not consider edge weights. Thus, we create a modified F1-score, where edge weights *near* 0 are counted as 0, small differences between predicted edge weight and the actual edge weight is counted as a true positive, and differences in sign (+/-) are observed. Out modified F1-score is given by:

$$F1 = \frac{TP}{TP + 0.5(FP + IFP + FN + IFN + I + ME)}$$

where

Dataset Name	Confounders	Cycles	Seed	F1	Noise Type	Linearity
dataset1	1	1	0	0.60	0	1
dataset2	1	0	0	0.20	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
dataset8339	0	1	19	0.05	1	0
dataset8340	0	0	19	0.29	1	0

Table 1: Example dataset used for the model predicting F1

- IFP: Inverse false positive, where the algorithm predicts a false positive, but with the opposite sign (+/-).
- IFN: Inverse false negative, where the algorithm predicts a false negative, but with the opposite sign (+/-).
- I: Inverse, where the algorithm correctly predicts the existence of an edge, but the edge weights have an opposite signs (+/-).
- ME: Magnitude error, where the algorithm correctly predicts the existence of an edge, but the magnitudes of the edge weights are much different.

This modified F1-score gives us a more detailed depiction of the algorithm's prediction performance. We not only return the score, but also the number of such cases, which gives us more insight into how different LiNGAM violations affect prediction performance. Similar to the F1-score, the modified F1-score also lies in the range $[0, 1]$, where a score of 1 indicates a perfect prediction, while a score of 0 indicates the opposite. Pseudocode: A.3

The following two metrics measure the distance between the ground truth adjacency matrix and the predicted adjacency matrix from DirectLiNGAM. First, we consider Structural Hamming Distance (SHD) [12] as an accuracy test. SHD is a metric used to quantify the distance between two graphs by the difference of their adjacency matrices. For two adjacency matrices A and B, their SHD is given by

$$d_H(A, B) = \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij} - b_{ij}|.$$

A lower SHD indicates better performance from the predictive algorithm, and vice versa.

We also consider the Frobenius norm as an accuracy measure. This is defined as the square root of the sum of the absolute squares of its elements. To find the distance between two matrices, $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times n}$, we calculate

$$\|A - B\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij} - b_{ij}|^2}.$$

A lower Frobenius norm indicates a smaller difference between the predicted matrix and the true matrix, which means a more accurate prediction.

3.3 Sensitivity analysis

We consider a dataset where each observation is a dataset and its assumption violations, or lack thereof, and its accuracy under DirectLiNGAM. Table 1 shows an example dataset. We fit a linear model using cyclicity, Gaussianity, and nonlinearity as predictors. To predict the F1 score, we calculate the following linear model:

$$F1 = .3141(\text{acyclic}) + .2552(\text{non-Gaussian}) + .1602(\text{linear}),$$

where the p-values are $5.2613e^{-244}$, $2.4403e^{-169}$ and $5.6562e^{72}$, respectively, with an R^2 value of 0.649. See Appendix A.8 for details on this calculation. Figure 5 visualizes these coefficients. The existence of assumption violations follows this rule:

$$\begin{cases} 0 : \text{violates assumption} \\ 1 : \text{does not violate assumption} \end{cases} \tag{1}$$

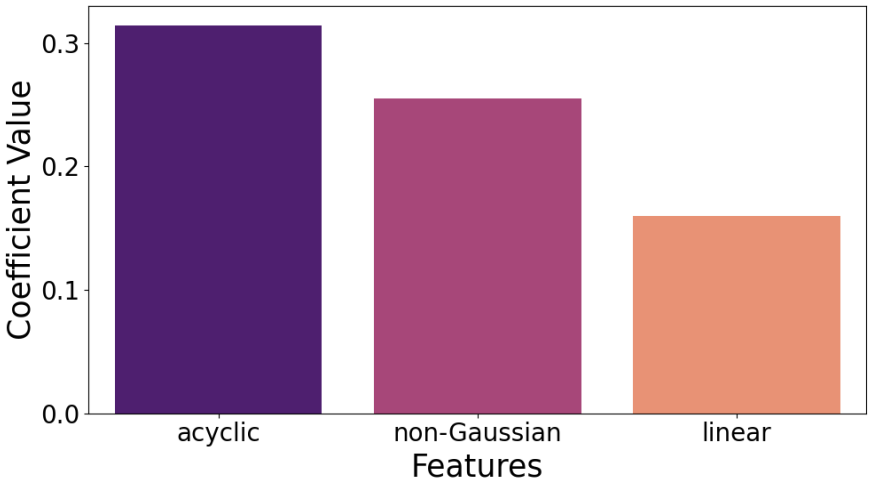


Figure 5: Coefficients for the linear model predicting F1 score

While the coefficients tell us how much the variables contribute to the overall score, we also care about the significance values. Each p-value comes from a statistical test where the null-hypothesis assumes that the variable is not in the model. If the model with the variable in it is a significant improvement, that would mean a low p-value below a certain threshold, you would reject the null hypothesis that the model does not have that variable. We choose a threshold of 0.05 without loss of generality. Because each variable has a p-value less than 0.05, this tells us that these relationships are unlikely to have occurred by chance and individually they all significantly contribute to predicting the dependent variable (F1). Thus, all of our variables are meaningful in predicting the F1 score. Because we are predicting F1 score as the measure for reliability, a score of 1 is ideal.

Note the absence of confounders as a predictor. Unfortunately, the coefficient value for the presence of confounders is the opposite sign of what we would expect. When we generate data in the 'vanilla' case, meaning the data has no other DirectLiNGAM violations, confounders contribute in the expected way. This implies that the problem might come from generating confounded data in a specific case of another variable not being vanilla. More research is needed here. There are additional reasons for this removal mentioned in section 4.3.

4 Detecting assumption violations

Given that we know the impact of DirectLiNGAM assumption violations on accuracy, we now aim to detect these violations in an arbitrary dataset. Then, we will know the expected accuracy, i.e., the reliability of results, for any dataset. The following are our methods for finding violations of linearity, non-Gaussianity, non-confoundedness, acyclicity, and infinite data. There are various parameters associated with each of these tests, which are tuned. Hyperparameter tuning results are located in the appendix, section A.5. Given these optimal parameters, we evaluate success rate on the generated data for each method. All of these methods additionally return the afflicted variables, which may be an avenue for graph generation in the future. This means either removing variables, adding relationships, or looking at specific cut points for conditional causal discovery depending on the context. These suggestions will bring us closer to the true causal graph, thus enhancing discoverability.

4.1 Nonlinearity

The linearity requirement of DirectLiNGAM indicates that when variables are related, they must be so in a linear manner. However, unrelated variables might show linear or nonlinear relationships as well. It is impossible to know a priori which relationships are supposed to be linear, and which do not matter. To mitigate this, we consider particular cases 'unrelated'. With this caveat acknowledged, we develop the following model for detecting nonlinear relationships in a dataset.

We use successive polynomial fitting via orthogonal polynomials, see A.4, to fit lines to every possible pair of variables in a dataset. For example, choose variable x_1 as the independent variable and x_2 as the dependent variable. Then fit a linear model, e.g., $x_2 = mx_2 + b$, to this data. The null hypothesis is that the constant model, $x_2 = b$, explains the data. If the linear model is a statistically significant improvement over the constant model, reject the null hypothesis. Otherwise, fail to reject. Repeat this process with with increasing degree polynomials. Figure 6 shows an example of a linear fit versus a quadratic fit. Hyperparameter testing

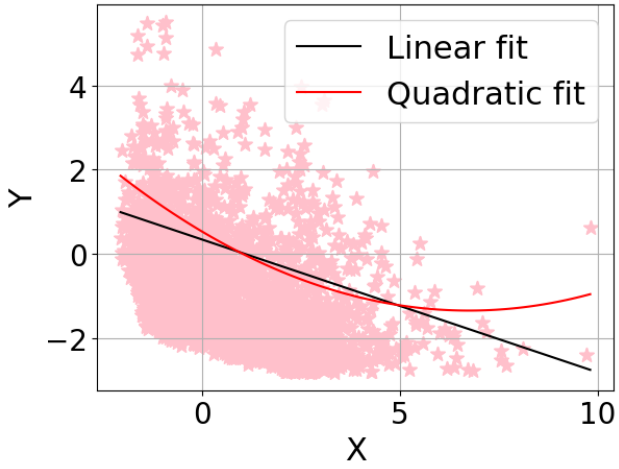


Figure 6: Example of fitting a linear and quadratic line to data

shows that max degree = 9 and p-value threshold = 0.001 performs the best, with a success rate equaling 68.4%.

To classify a relationship as linear, the linear model must reject the null hypothesis, and all higher degree polynomials must fail to reject. To classify a relationship as nonlinear, at least one degree 2 through degree 8 polynomial must reject, and the highest degree, i.e., degree 9, must fail to reject. All other cases are considered unrelated. This is a clear limitation in our method because it cannot identify higher degree or non-polynomial functional relationships. Our test would be improved by allowing for higher degree polynomials, but that is computationally expensive.

This process is then repeated for all pairs of variables. If at least one pair is classified as nonlinear, the dataset is nonlinear. Otherwise, it is linear. By including the 'unrelated' category, we try to account for the fact that variables that are not related to each other might have nonlinear relationships. However, it is still possible that datasets are incorrectly classified as having broken this linearity criterion by unrelated variables being related in exactly a quadratic manner. This should be considered when extending this idea to higher powers.

Another area of extension applies to conditional causal discovery. While fitting curves to any pair of variables, recording either local maximums and minimums or points of highest curvature to serve as cut points could emulate more linear data and thus provide more accurate DirectLiNGAM results. The fitted polynomials are in an orthogonalized basis and should be transformed back to the original space before identifying cut points.

	Linear	Nonlinear	Total
Predicted linear	1450	744	2194
Predicted nonlinear	470	1176	1646
Total	1920	1920	3840

Table 2: Results of linearity test on generated data

4.2 Gaussianity

Recall the requirement of DirectLiNGAM that requires causal relationships between variables to be related with non-Gaussian noise. The difference between non-Gaussian noise and a non-Gaussian distribution should be noted. Thus, we check, for every pair of variables, the distribution of the residuals that come from predicting a linear relationship between them. This is determined using the Shapiro-Wilk test because Razali and Yap [5] found it to be the best in all scenarios. Our parameter tuning reveals that this test performs best when checking every set of variables regardless of the correlation coefficient, i.e., regardless of whether there is a causal relationship or not.

An obvious limitation to this is that we fit a line to each pair of variables, so this test will not be as accurate for nonlinear data. Using residuals to the fitted polynomial from the nonlinearity test instead of a line could improve this metric.

	Non-Gaussian	Gaussian	Total
Predicted non-Gaussian	1096	149	1295
Predicted Gaussian	824	1771	2595
Total	1920	1920	3840

Table 3: Results of Gaussianity test on generated data

4.3 Confoundedness

Unlike the other DirectLiNGAM assumption violations, it does not matter if the pair of variables afflicted are actually related. Instead, we care if confoundedness is present in the dataset as a whole. To detect confoundedness, we consider the case where two variables, say x_1 and x_2, have a high correlation coefficient but no link is found between them via DirectLiNGAM. If no other variable causes both x_1 and x_2, then we say there is $+1$ confounder. This is then repeated for every pair of variables. If this count is non-zero, the dataset is classified as confounded.

It is apparent that this metric does not contribute in a meaningful way during sensitivity analysis, and in addition to this, its best success rate on generated data is 48.6% correct. Hence, we remove this as a predictor of reliability.

Although this test is not helpful within the scope of this project, we have ideas to improve it in future work. The primary technique we might consider is called bootstrapping. Our current approach only gives us one chance to see whether DirectLiNGAM will find a relationship. The new approach would involve taking overlapping subsets of the data so we can run DirectLiNGAM multiple times for the same variables. Then we can evaluate the proportion of time a particular relationship holds. This will give us more opportunities to find confounders which we currently severely under predict as per table 4.

	Not confounded	Confounded	Total
Predicted not confounded	1712	1799	3511
Predicted confounded	208	121	329
Total	1920	1920	3840

Table 4: Results of confoundedness test on generated data

4.4 Cyclicity

Recall that DirectLiNGAM starts by assigning a hierarchy of variables which immediately eliminates the possibility of finding cycles. To circumvent this, we can use prior knowledge. Our test for finding cycles starts by running DirectLiNGAM once for each variable where that variable is forced to be the most independent in the hierarchy. We also run it once without prior knowledge to serve as a baseline. Next we count, of the times a given relationship is allowed by the hierarchy, how often does it appear in the result. This ratio is then compared to a threshold. If the ratios for both A causes B and B causes A are above the threshold for a given pair of variables, then we say (A, B) is a cycle. More details are provided in section A.7.

This test performs so poorly that its inverse may be useful. The lowest accuracy we achieved was 36.4% with the tendency to over-predict the existence of cycles. See table 5 for more details. Perhaps the real explanation here is that on the data we generate with cycles, DirectLiNGAM is just less likely to find relationships than on the acyclic data. Unfortunately, our sensitivity analysis showed that cyclicity is the most significant factor in F1-score, so we would like to improve this test in future work. Similar to confoundedness, bootstrapping may be a valuable technique here.

	Acyclic	Cyclic	Total
Predicted acyclic	38	561	599
Predicted cyclic	1882	1359	3241
Total	1920	1920	3840

Table 5: Results of cyclicity test on generated data

4.5 Finite data

We investigate the metric of how much a dataset violates the infinite data assumption of DirectLiNGAM. Our research shows that the number of rows and the number of columns is insignificant in predicting any accuracy metrics. However, the proportion of rows to columns was significant, but not significant enough to include in our final model. We include these details for the reader, regardless. The ratio is

$$P = \frac{\text{number of columns}}{\text{number of rows}},$$

which means a smaller number fits the assumption better, and vice versa.

5 Creating a dataset scoring metric

We combine all of this information to create a metric for reliability. We use 'reliability' and 'predicted accuracy' synonymously, and variables refer to violations of DirectLiNGAM. Using the coefficients from the linear model, we plug in the information on insufficiencies in the data to predict accuracy. Recall the labels for the existence of violations from equation 1.

Consider a dataset that is found to have the following DirectLiNGAM assumption violations.

$$\text{example dataset traits} = \begin{cases} \text{acyclic: } 0 \\ \text{non-Gaussian: } 1 \\ \text{linear: } 0 \end{cases}$$

The linear model that was found in section 3.3 with this input is:

$$\begin{aligned} R &= .3141(\text{acyclic}) + .2552(\text{non-Gaussian}) + .1602(\text{linear}), \\ &= .3141(0) + .2552(1) + .1602(0) \\ &= .2552, \end{aligned}$$

where R is the measure of reliability.

Observe that the highest possible score is 0.7295, so we normalize each score by this. Given there are eight configurations for violating/non violating assumptions of DirectLiNGAM, we choose the scoring metric shown in table 6. With this scoring metric, the previous example now has a reliability score of .3498, i.e., an F. We can see that for any dataset, violating the assumptions in order of importance, while not violating any others, leads to grades of A, B, C, etc. For examples of use, see Appendix **??**.

A further application of this uses conditional linear models to find how much each variable individually contributes. There are two routes possible for this. First, consider the metadata dataset with saved information on datasets. An example of this is located here: Table 1. First, split up the metadata dataset into two parts, one that satisfies the most significant condition and one that does not. Again, fit a linear model

acyclic	non-Gaussian	linear	reliability (normalized)	grade
0	0	0	0.0000	F
0	0	1	0.2196	F
0	1	0	0.3498	F
0	1	1	0.5694	D
1	0	0	0.4306	F
1	0	1	0.6502	C
1	1	0	0.7804	B
1	1	1	1.0000	A

Table 6: Possible reliability scores and associated grades

to both of these. Do this process until for every configuration of variables, there is a smaller linear model to predict accuracy. Then, we can see how well the remaining variable is able to predict the reliability score. Alternatively, this hierarchy can be created from how confident we are on our metrics for detecting violations, instead of significance.

6 Conclusion

We set out to enhance discoverability by uncovering the reliability of DirectLiNGAM. This was achieved by first generating large amounts of artificial data. These are datasets with known ground truth and known adherence to or violations of DirectLiNGAM assumptions. Then we assessed the output of DirectLiNGAM when run on these datasets. Separately we developed tests for finding violations of the DirectLiNGAM assumptions in arbitrary datasets. Finally, we put these results together to create a scoring metric. This enhances discoverability by establishing a confidence level to support the varacity of DirectLiNGAM results.

7 Future work

7.1 Previously mentioned

The following are areas in which our work could be extended that we have previously mentioned. Full details are included in the text and cited here.

- 3.3 Correct the coefficient for confoundedness in the linear model predicting accuracy.
- 4 Graph modification to more accurately represent your data.
- 4.1 Include higher-power polynomials and other functions in nonlinearity detection.
- 5 Conditional linear models for finding specific violation contributions to the reliability metric.
- 7.6 Integrate our work with the user interface.

7.2 Literature review

Due to the nature of this project, we did not do extensive literature review. Most of our work was based around Fujitsu Causal Discovery rather than the greater body of work concerning DirectLiNGAM. We are aware of other papers evaluating the performance of DirectLiNGAM under various conditions [14, 8]. We would like to compare our results with theirs. This may also help us gain insight into some of our concerns about the data generation process mentioned above.

Further work exists comparing DirectLiNGAM to other causal discovery algorithms [11, 7]. This work could serve as a valuable starting point for our future direction 7.4 below.

7.3 Conditions

As mentioned above, the inspiration for this project was Fujitsu Causal Discovery. This tool enhances DirectLiNGAM by allowing for conditional causal discovery. Each condition amounts to selecting a subset of the input data before running DirectLiNGAM. Our work should extend to this context in a straightforward way. Once conditions are chosen we can run all of our violation tests on the appropriate subsets of data and produce a corresponding reliability score for the resulting conditional causal graph.

7.4 Choosing an algorithm

This project focused on DirectLiNGAM because that is the basis of Fujitsu Causal Discovery. But there are dozens of other causal inference algorithms [3], including several variations on LiNGAM [9] designed around a similar goal. Each algorithm has its own assumptions, strengths, and weaknesses. Another future direction for this project would be to repeat our process for many other algorithms. This way, after evaluating a dataset, we can tell the user which algorithm is expected to give the best results based on the nature of the data.

7.5 Machine learning

In section 3, we created datasets with known ground truth structures that explicitly violate DirectLiNGAM assumptions. We applied DirectLiNGAM to each dataset and compared its output to the true causal graph. This allows us to assess to what degree different types of assumption violations affect its predictive accuracy.

In principle, we would be able to use machine learning algorithms to identify patterns among an arbitrary dataset based on the nature of its assumption violations. Given an arbitrary dataset, we could assess its similarity to our generated datasets and their ground truth structures, and then infer the most likely ground truth structure.

This would be helpful for datasets that violate DirectLiNGAM assumptions, and would therefore produce an inaccurate prediction. We would not rely solely on DirectLiNGAM's output, but by learning from past cases where their true causal structure is already known.

Causal discovery itself is an interesting problem for machine learning. Since we have developed a process for generating data with known ground truth, we could create arbitrarily large training sets. In this way, we could train a model to generate causal graphs from on arbitrary datasets. The team members of the G-RIPS Fujitsu group, 2025, have strong interest in studying this.

7.6 User experience and interface design

Our model provides the following workflow:

1. Upload CSV
2. Receive causal graph and reliability grade
3. Receive DirectLiNGAM assumption violations
4. See variables that caused the violations

Users upload a `.csv` file of a dataset into the interface and the data is preprocessed, as in section 2. The user then receives the causal graph that DirectLiNGAM has inferred, together with the corresponding reliability grade, according to table 6 in section 5. To help users understand the reliability grade, the interface provides a list of DirectLiNGAM assumption violations detected in the uploaded dataset. For each violation, the user is provided with information on which variables are responsible.

7.6.1 Visualization and results presentation

Visualizations are developed using the `Plotly` and `PyVis` libraries, offering outputs suitable for academic, business, or policy applications. Color schemes and visual hierarchy guide users through the interface, while clear status indicators (e.g., success, warnings, and errors) enhance understanding.

The platform adapts language based on user type, such as researchers, business users, or students, and includes contextual help for every step.

7.6.2 Technical architecture

The system is built on a modular architecture. The frontend manages user interaction and visualization, while the backend handles the computational logic using Python's scientific stack.

Session tracking ensures continuity, allowing users to pause and resume work without data loss. Export functionality supports various formats, facilitating professional report generation.

We also hope to include instructions on how to integrate our project and Fujitsu Causal Discovery AI into this interface.

7.6.3 Summary

In summary, the platform will successfully bridge the gap between advanced statistical modeling and practical usability. It will empower technical users to perform reliable causal analysis without requiring deep expertise.

References

[1] John Forde, Gaspar Mendez, Akane Okubo, Daniel Quigley, and Renji Sakamoto. G-RIPS Sendai 2024 Fujitsu Goup Final Report, 2024. URL https://www.mccs.tohoku.ac.jp/g-rips/report/2024/pdf/fujitsu_final_report.pdf.

[2] Davar Khoshnevisan. Linear Statistical Models, 2011. URL http://www.math.utah.edu/œdavar.

[3] D. Malinsky and D. Danks. Causal discovery algorithms: A practical guide. *Philosophy Compass*, 13 (1), 2018. doi: 10.1111/phc3.12470.

[4] Max Planck Institute for Intelligent Systems. Cause-effect database. URL https://webdav.tuebingen.mpg.de/cause-effect/.

[5] Nornadiah Mohd Razali and Bee Yap. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *J. Stat. Model. Analytics*, 2, 01 2011.

[6] Francesco Montagna, Atalanti A. Mastakouri, Elias Eulig, Nicoletta Noceti, Lorenzo Rosasco, Dominik Janzing, Bryon Aragam, and Francesco Locatello. Assumption violations in causal discovery and the robustness of score matching. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=IyTArtpuCK. https://github.com/francescomontagna/causally?tab=readme-ov-file and https://causally.readthedocs.io/en/latest/usage.html.

[7] Francesco Montagna, Atalanti A. Mastakouri, Elias Eulig, Nicoletta Noceti, Lorenzo Rosasco, Dominik Janzing, Bryon Aragam, and Francesco Locatello. Assumption violations in causal discovery and the robustness of score matching, 2024. URL https://arxiv.org/abs/2310.13387.

[8] Shreya Prakash, Fan Xia, and Elena Erosheva. A diagnostic tool for functional causal discovery, 2024. URL https://arxiv.org/abs/2406.07787.

[9] S. Shimizu. URL https://sites.google.com/view/sshimizu06/lingam?

[10] Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvarinen, Yoshinobu Kawahara, Takashi Washio, Patrik Hoyer, and Kenneth Bollen. DirectLiNGAM: A Direct Method for Learning a Linear Non-Gaussian Structural Equation Model. *Journal of Machine Learning Research*, 12, 01 2011.

[11] Tatsuya Tashiro, Shohei Shimizu, Aapo Hyvarinen, and Takashi Washio. ParceLiNGAM: A causal ordering method robust against latent confounders. 2013. URL https://arxiv.org/abs/1303.7410.

[12] I. Tsamardinos, L.E. Brown, and C.F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Mach Learn 65, 31–78*, 2006. URL https://doi.org/10.1007/s10994-006-6889-7.

[13] Gene Yu, Ce Guo, and Wayne Luk. Robust time series causal discovery for agent-based model validation, 2024. URL https://arxiv.org/abs/2410.19412.

[14] A. Zamanian, L. Mareis, and N. Ahmidi. Partially specified causal simulations, 2023. URL https://arxiv.org/abs/2309.10514.

A Appendix

A.1 DirectLiNGAM algorithm

The following is pseudo-code for the DirectLiNGAM algorithm.

1. Initialize:
   - x: a p-dimensional random vector
   - U: variable subscripts of x
   - X: $p \times n$ data matrix of x's
   - $\mathbf{K} := \varnothing$ (ordered list of variables)
2. Repeat until $p-1$ subscripts are appended to K:
   (a) Perform least squares regression of x_i to x_j $\forall i \in U \setminus K (i \neq j)$.
   (b) Compute residual vectors $r^{(j)}$ and residual data matrix $R^{(j)}$ $\forall j \in U \setminus K$
   (c) Find x_m: the most independent variable

$$x_m = \arg \min_{j \in U \setminus K} T_{kernel}(x_j; U \setminus K)$$

   where $T_{kernel} = \sum_{i \in U, i \neq j} \widehat{MI}(x_j, r_i^{(j)})$.
   $\widehat{MI}$ is a measure of mutual information.
   (d) Append m to k.
3. Append remaining variable to end of K
4. Construct lower triangular matrix, B, estimating connection strengths, b_{ij}, using covariance-based regression. (LS or MLE)

A.2 Automated Fujitsu Causal Discovery

Previously, Fujitsu Causal Discovery was separated into three Jupyter notebooks. Each cell was run one at a time and it was difficult to consolidate the output. Our new process initializes a class, figure 7, which creates a folder based on a keyword that you input, and then creates an input and output folder in it. See figure 8. This class takes in all parameters for steps for our reliability code as well as Fujitsu code. This includes s-min, o-max, r-value, k, modes for binarizer, thresholds etc. Now, each step is a class function, so they can be run independently from a single command. This also means that class variables are defined, like the dataframe. i.e., the dataset and other parameters can be accessed anytime, and across different models easily. This eliminates as much manual work as possible.

```
import packages

model = Class(dataset)

model.create_file_directory
model.prepare_dataset
model.run_fujitsu_causal_discovery
```

Figure 7: Example of automated pipeline

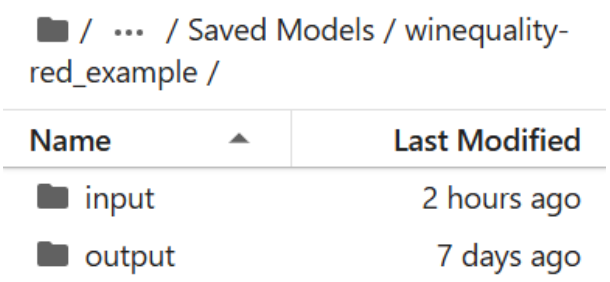


Figure 8: File structure example

Algorithm 1 Modified F1-score

Initialize:
$TP, TN, FP, IFP, FN, IFN, I, ME \leftarrow 0$
$\epsilon \leftarrow 0.5 \times \max(|\texttt{matrix1}|, |\texttt{matrix2}|)$
$\delta \leftarrow 0.5 \times \epsilon$
$x_1 \leftarrow \texttt{matrix1}[i, j]$
$x_2 \leftarrow \texttt{matrix2}[i, j]$

if $|x_1| \bigwedge |x_2| < \delta$ **then**
  $TN \leftarrow TN + 1$
else if $|x_1| \bigwedge |x_2| > \delta$ **then**
  if $|x_1 - x_2| < \epsilon$ **then**
    $TP \leftarrow TP + 1$
  else if $|x_1 - x_2| > \epsilon$ **then**
    if same sign **then**
      $ME \leftarrow ME + 1$
    else if diff sign **then**
      $I \leftarrow I + 1$
else if $|x_1| < \delta \bigwedge |x_2| > \delta$ **then**
  if $|x_1 - x_2| < \epsilon$ **then**
    $TN \leftarrow TN + 1$
  else if $|x_1 - x_2| > \epsilon$ **then**
    if same sign **then**
      $FP \leftarrow FP + 1$
    else if diff sign **then**
      $IFP \leftarrow IFP + 1$
else if $|x_1| > \delta \bigwedge |x_2| < \delta$ **then**
  if $|x_1 - x_2| < \epsilon$ **then**
    $TN \leftarrow TN + 1$
  else if $|x_1 - x_2| > \epsilon$ **then**
    if same sign **then**
      $FN \leftarrow FN + 1$
    else if diff sign **then**
      $IFN \leftarrow IFN + 1$

- **TP (True Positive)**: Both matrices have strong, similarly signed edges.
- **TN (True Negative)**: Both matrices have near-zero (insignificant) edges.
- **FP (False Positive)**: Prediction shows a significant edge where none exists.
- **IFP (Inverse False Positive)**: Prediction shows a reversed-sign edge that shouldn't exist.
- **FN (False Negative)**: Ground truth has a strong edge that was missed.
- **IFN (Inverse False Negative)**: Ground truth edge exists but is predicted in the wrong direction.
- **I (Inverse)**: Both have strong edges but in opposite directions and with large differences.
- **ME (Magnitude Error)**: Both have edges with the same sign but vastly different magnitudes.

$$\text{Modified F1-score} = \frac{\text{TP}}{\text{TP} + 0.5 \cdot (\text{FP} + \text{IFP} + \text{FN} + \text{IFN} + \text{I} + \text{ME})}$$

A.3 Modified F1-score

We develop a modified F1-score, to account for edge magnitude and direction, that was not considered by the original F1-score. Algorithm 1 shows pseudo-code for our modified F1-score algorithm.

A.4 Successive polynomial fitting

To detect nonlinearity, we do successive polynomial fitting using orthogonal polynomials. The following is pseudo-code to find a polynomial of degree d where x and y have n observations. This was created by Nicole Lacey with guidance from [2].

Initialize orthogonal basis for polynomial of degree d: ϕ.

Transform x to the orthogonal basis.

$$x_\phi = [\phi_1(x)\ \phi_1(x)\ \cdots\ \phi_d(x)]^T$$

Find the polynomial coefficients in this space.

$$\hat{\beta}_\phi = x_\phi^T y$$

Find the residual squared error.

$$\text{RSS} = \|y - x_\phi \hat{\beta}_\phi\|^2$$

Define a matrix, A, to test the null hypothesis that the leading coefficient is 0.

$$A = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}$$

Find the coefficients under the null hypothesis.

$$\beta_{H_0} = \hat{\beta}_\phi + A^T(AA^T)^{-1}(-A\hat{\beta}_\phi)$$

Find RSS under the null hypothesis.

$$\text{RSS}_{H_0} = (y - x_\phi \beta_\phi)^T (y - x_\phi \beta_\phi)$$

Check if the model with the degree d term is a statistically significant improvement.

$$\text{f}_{\text{stat}} = \frac{|\text{RSS}_{H_0} - \text{RSS}|}{\text{RSS}}(n - d)$$

Calculate the p-value from the f-stat and degrees of freedom of the model. If it is below the chosen threshold, i.e., p = .01, reject the null hypothesis and thus accept the model with the added degree. Failing to reject the null hypothesis is different from accepting it.

A.5 Hyperparameter tuning

The parameters we tune for each of the violation tests are as follows. For the nonlinearity test we vary the p-value threshold and max polynomial degree. For the Gaussianity test we vary the correlation coefficient threshold which determines whether we check the noise in a given relationship. For the confoundedness test we vary the correlation coefficient threshold in the same way. And for the cyclicity test we vary the threshold

for what proportion of causal relationships need to appear to count a cycle. For each parameter choice we run the tests on all of our generated datasets and measure the percentage of successful guesses. The figures below show the results of this process. From those results we choose a final parameter, and a further break down of the accuracy at the chosen parameter can be found in section 3. In the case of nonlinearity the results improve as we increase max degree, but run time also increases exponentially, so we do not test beyond degree 9.

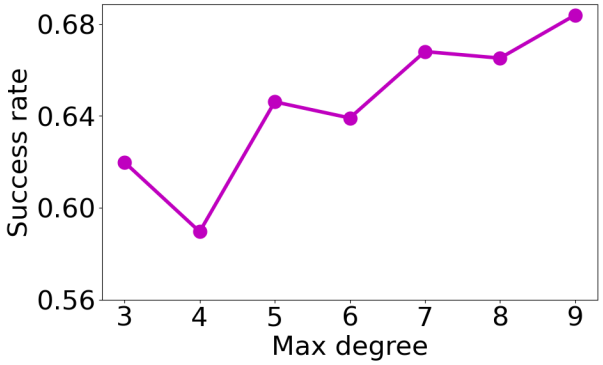


Figure 9: Nonlinearity test with 'p-value threshold' = 0.001 vs. success rate

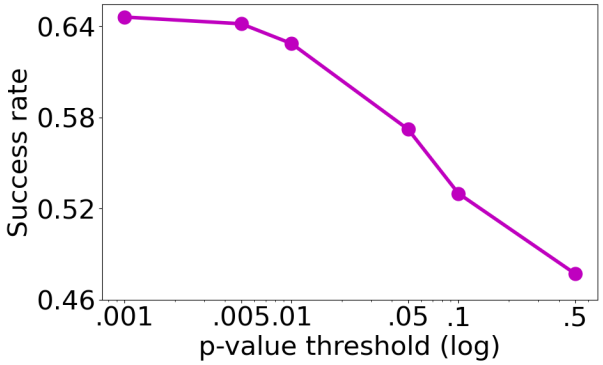


Figure 10: Nonlinearity test with max degree = 5 vs. success rate

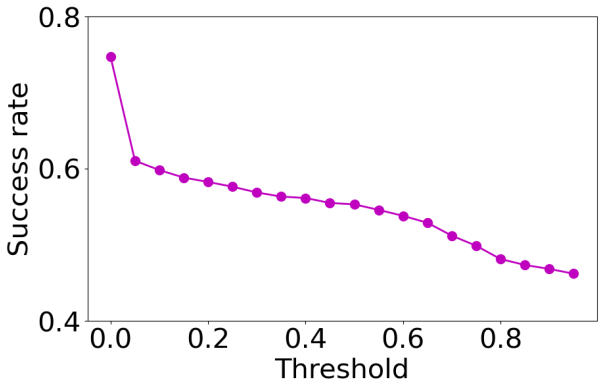


Figure 11: Gaussianity test success rate vs. threshold

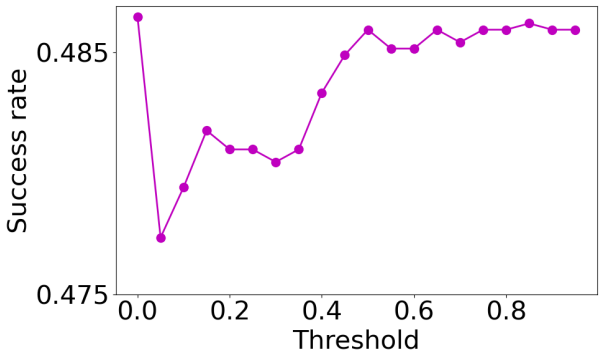


Figure 12: Confoundedness test success rate vs. threshold

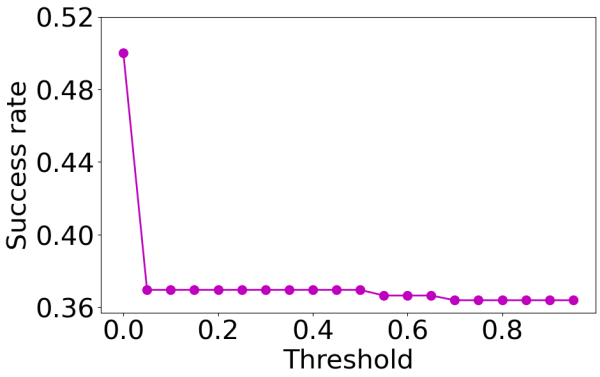


Figure 13: Cyclicity test success rate vs. threshold

A.6 Choosing an accuracy score

We consider F1, modified F1, Structural Hamming Distance, and Frobenius norm as metrics for accuracy. We find that fitting a linear model to predict F1 has an R^2 value of 0.649, while SHD and Frobenius norm are 0.166 and 0.121, respectively. See figure 14. This value shows us the extent to which the predictors–acyclicity, non-Gaussianity, and linearity–are able to explain dependent variable. Because of SHD and the Frobenius norm's low R^2 values, we do not consider them for measures of reliability. Thus, we select predicted F1 score as a measure for reliability. Additionally, the modified F1 score was not explained well by the predictors.

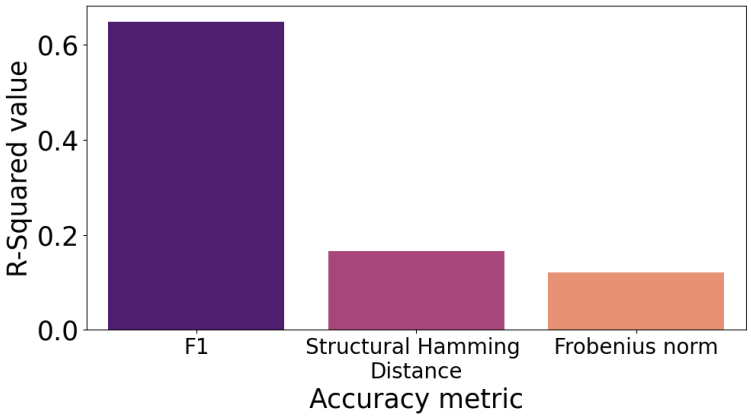


Figure 14: R^2 values for accuracy metrics

A.7 Cyclicity Test

Algorithm 2 shows pseudo-code for our cycle finding test.

Algorithm 2 FindCycles

Initialize:
outputList, cyclesList ← [ ]
for all col ∈ columns of dataset **do**
  hierarchy ← [col] + [c | c ∈ columns of dataset, $c \neq$ col]
  result ← RunDirectLiNGAM(dataset, hierarchy)
  outputList.append([col, result])
outputList.append(RunDirectLiNGAM(dataset)) ▷ Run without hierarchy for control
for all a ∈ columns of dataset **do**
  for all b ∈ columns of dataset **do**
    $R1_{\text{num}} \leftarrow 0, \quad R1_{\text{den}} \leftarrow 0$
    $R2_{\text{num}} \leftarrow 0, \quad R2_{\text{den}} \leftarrow 0$
    for all k ∈ outputList **do**
      if $k[a][b] \neq 0$ **then**
        $R1_{\text{num}} \leftarrow R1_{\text{num}} + 1$
        $R1_{\text{den}} \leftarrow R1_{\text{den}} + 1$
      else if $k[b][a] \neq 0$ **then**
        $R2_{\text{num}} \leftarrow R2_{\text{num}} + 1$
        $R2_{\text{den}} \leftarrow R2_{\text{den}} + 1$
      else
        if $k[0] \neq a$ **then**
          $R2_{\text{den}} \leftarrow R2_{\text{den}} + 1$
        if $k[0] \neq b$ **then**
          $R1_{\text{den}} \leftarrow R1_{\text{den}} + 1$
    $R1 \leftarrow \frac{R1_{\text{num}}}{R1_{\text{den}}}, \quad R2 \leftarrow \frac{R2_{\text{num}}}{R2_{\text{den}}}$
    if min($R1$, $R2$) ≥ threshold **then**
      cyclesList.append([a, b])

- **outputList**: List of adjacency matrices produced by DirectLiNGAM.
- **cyclesList**: List of expected cycles.
- $R1_{\textbf{num}}$: Number of times a causes b appears in a causal graph.
- $R1_{\textbf{den}}$: Number of times a causes b was possible.
- $R2_{\textbf{num}}$: Number of times b causes a appears.
- $R2_{\textbf{den}}$: Number of times b causes a was possible.
- **threshold**: Minimum ratio to count as a cycle.

A.8 Linear Model

This section provides a description of how to calculate a linear model for a generic dataset. Let predictor variables $x_1, \cdots, x_p \in \mathbf{R}^{n\times 1}$, where $X = [x_1 \cdots x_p]$, and a dependent variable $Y \in \mathbf{R}^{n\times 1}$. Then, observe the desired model with the coefficient $\beta = \beta_1, \cdots, \beta_p$:

$$Y = X\beta + \epsilon$$

We wish to know the values of β. Observe:

$$Y = X\beta$$
$$X^T Y = X^T X\beta$$
$$(X^T X)^{-1} X^T Y = \beta$$

Thus, we have found the coefficients, i.e., β.

A.9 Examples

The following two datasets are from an online repository, cited here: [4]. Our first example is a two-variable dataset, recording altitude versus temperature. This dataset received a score of F, or 43%, due to its Gaussian noise and nonlinear relationships. We can see that these impacted the accuracy of the results of DirectLiNGAM. The resulting graph in Figure 15 shows that temperature causes altitude, when we know that the opposite is true. Our reliability metric tells us not to trust the DirectLiNGAM results, and indeed, the results were not correct.

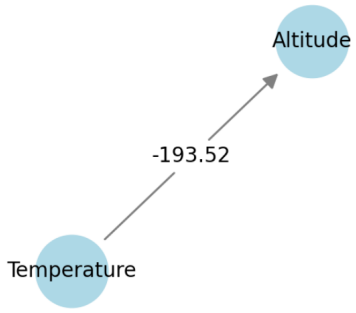


Figure 15: DirectLiNGAM prediction: altitude vs. temperature dataset

We can see, in a dataset investigating horsepower versus MPG (miles per gallon), that our reliability metric tells us how true our results are. The predicted score is a B, or 78%, and the results align with what we know to be true – that horsepower does cause MPG. We can expect reliable results because this dataset only violated the nonlinearity condition. See Figure 16.

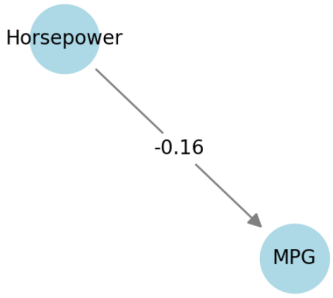


Figure 16: DirectLiNGAM prediction: horsepower vs. mpg dataset

Last, consider the canonical wine quality dataset. DirectLiNGAM will output that 'quality' is a cause of other variables, which we know is not true. If given a reliability score, users will then know that they should, or should not, trust the results. This dataset received a score of 78%, or a B because it violated the nonlinearity condition. So, we know that the results of this dataset are not perfect, and indeed our score reflects that. These show that for any dataset we can find the necessary information to reasonably tell if results are expected to be true i.e., the veracity of the results of DirectLiNGAM.

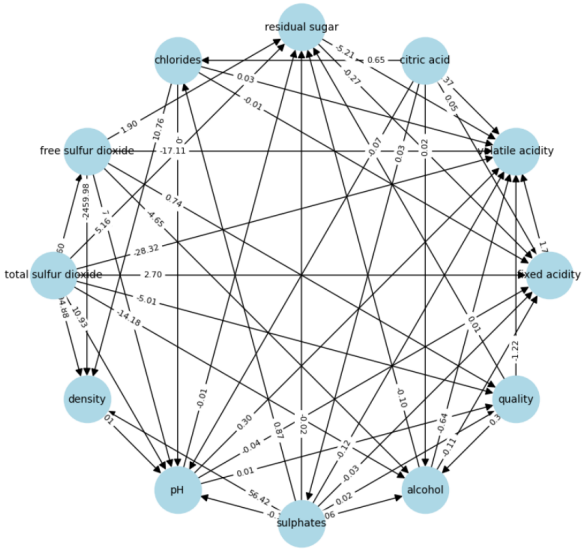


Figure 17: DirectLiNGAM prediction: wine quality dataset

B Contributions

B.1 Jamin Kochman

Jamin's primary role in this project was in code integration. Do to the secure nature of this project it was difficult to directly collaborate on code. Early on Jamin worked with Nicole on automating the Fujitsu process. From that point Jamin became the team member most familiar with the Fujitsu code, and integrated the functions created by Nicole and Jia Hui into the final working project. Jamin was responsible for running large scale data generation and testing after organizing the group's code in a cohesive way. Furthermore, they assisted in a consulting role on most aspects of the project framework and code debugging. They also used their experience with LaTeX and rhetoric to streamline editing of all deliverables. Finally, they developed the algorithm for cycle detection in a dataset and the code to implement it.

B.2 Nicole Lacey

Nicole contributed to this project as a team member, but also a project manager. We begin with her contributions as a team member. Early on, she worked with Jamin to automate Fujitsu's code. She contributed file structure automation code from her personal research to help do this, and advised on the importance of a class to hold all of the functions and variables. Then, when we needed to find accuracy metrics, Nicole wrote python scripts to calculate F1 score and structural hamming distance. She also researched alternative accuracy metrics, but did not find anything helpful. Nicole also devised a way to create cyclic data from a cyclic adjacency matrix, which Jia Hui then implemented. For the "finding assumption violations in data" part of our project, Nicole provided code from her graduate coursework to find nonlinearity in datasets, and adapted it to our specific needs. She also created the method to find confoundedness in data, with input from Jamin, after extensively researching possible alternatives and finding nothing concrete. She and Jamin also developed the method to find Gaussian noise in data. Finally, she developed the linear model for predicting the F1 score. Throughout the summer, she helped Jia Hui and Jamin debug various parts of the project and significantly contributed to all deliverables.

As project manager, Nicole continuously compiled and submitted questions to the industry mentors, and assigned tasks to members needing work or direction. She continuously checked in with team members to minimize miscommunication and ensure understanding. She also consulted with the academic mentor in a timely manner to address issues with team members, resulting in morning meetings that better addressed our needs.

B.3 Joseph Nyingi

At the beginning of the project, Joseph collaborated with Jia Hui to design and implement a data automation codebase for the data processing pipeline. He contributed to the team's literature review by exploring metrics such as Structural Hamming Distance (SHD) for graph comparison, as well as conducting preliminary research on nonlinearity tests, including polynomial fitting and the Ramsey RESET test. Joseph also identified appropriate tests to see data comes from a Gaussian distribution. Later in the project, he led the development of the user interface starting with wireframe design in Notion and moving to a functional prototype using Streamlit to align with the team's project goals. The resulting interface provided a user-friendly, three-step workflow upload, analyze, and discover featuring an analysis dashboard with tabs for detecting DirectLiNGAM assumption violations, visualizing causal relationships, and a Discoverability section that surfaced novel patterns and assessed reliability dynamically.

B.4 Jia Hui Sim

Jia Hui contributed to the project in terms of theoretical research and software implementation. An initial literature review was conducted on the assumptions underlying the DirectLiNGAM causal discovery algorithm, with particular attention paid to common assumption violations and how they may affect algorithmic performance. In addition, she explored various accuracy metrics used in the causal inference literature.

Jia Hui worked closely with Joseph at the beginning of the project to design and implement a codebase that automated the data processing pipeline. This greatly streamlined the experimental workflow and allowed the team to easily run experiments across multiple datasets with minimal manual intervention.

Another contribution was developing a Python-based data generation loop using the `causally` package. This script automated the creation of synthetic causal graphs and corresponding datasets. Jia Hui modified this generation loop to accommodate different combinations of graphs that violated the assumptions of DirectLiNGAM. In collaboration with Nicole, she helped debug issues that arose in generating confounded datasets and co-developed a method for injecting cycles into the generated graphs, since the `causally` package does not support cyclic structures. She worked with generating datasets for Jamin to use for the testing of Fujitsu Causal Discovery.

Regarding evaluation, Jia Hui contributed to metric design by working with the team to develop a modified F1-score tailored to causal discovery tasks. She wrote the corresponding implementation in Python to enable its use in empirical evaluations. Additionally, she reviewed literature on Network Portrait Divergence as an alternative accuracy metric and implemented preliminary tests, although this method was not ultimately used in the final analysis.

Finally, Jia Hui assisted with Nicole's implementation for evaluating linearity and nonlinearity in arbitrary datasets.