Master Thesis Computing Science

Radboud University Nijmegen

Structure Learning on an Increasing Amount of Available Data

Author:
M.R.H. Swanenberg (Maurice)
s4331095

Internal Supervisor:
Dr. J.C. Textor (Johannes)

External Supervisor:
D.A.S. Knoope MSc (Daan)

Second Reader:
Prof. Dr. Ir. D. Hiemstra (Djoerd)

August 26, 2021

Acknowledgement

I would like to thank De Volksbank for providing me the opportunity to write my thesis and perform my research while having access to their Spark cluster. My day-to-day supervisor, Daan Knoope, is for a great deal responsible for planting the seed of the research I have performed. Furthermore, he was there to help whenever I got stuck on the technicalities of the servers. For this I am extremely grateful. I would especially like to thank Dr. Johannes Textor for guiding me in the right direction. For without his intervention my research would have contributed much less to science than it does today. Moreover, the weekly meetings with Dr. Johannes Textor in the past few months enabled me to ask questions more frequently, which resulted in a healthy workflow. I would like to thank Ankur Ankan for shining a light on the several struggles I initially had to understand the algorithm which I used for structure learning. I am grateful for Prof. Dr. Ir. Djoerd Hiemstra to be my second reader for he was my supervisor when I performed my internship at De Volksbank prior to this thesis, and therefore knows how this research all came about. Last but not least, I would like to thank the proofreaders for their suggestions and the spelling errors they noticed, sorted lexicographically on surname: Daan Knoope, Marjolein van Nuland, Hendrik Werner.

Abstract

Structure learning provides insight on the relations among variables in a dataset. The learned structures become more reliable as the amount of available data increases. In order to perform structure learning on big data, three conditional independence (CI) tests are implemented in PySpark. The three CI tests implemented are: Pearson's chi-squared test, Monte Carlo permutation test and Random Forest test. This thesis describes the performance of different CI tests in structure learning for binary data. Performance is measured for each of the three CI tests in terms of calibration, power and runtime. Since each of those CI tests has its own strengths and weaknesses, a hybrid approach is proposed. This hybrid approach can switch between several CI tests, in order to get the best results in feasible runtime. The proposed hybrid approach is based on calibration, power and runtime of each CI test. Performance of the structure learning algorithm is measured using precision, recall, F1-score and runtime. The Random Forest test is the best test to use when only interested in the F1-score. Pearson's chi-squared test is by far the fastest test and the only feasible test for big data. The hybrid approach consists of Pearson's chi-squared test and the Random Forest test. This approach is significantly faster compared to using only the Random Forest test and it achieves a higher F1-score than the chi-squared test if the true graph is dense. This way of structure learning can lead to a better final result in feasible runtime, compared to the use of a single test.

Keywords: Causal inference, big data, Pearson's chi-squared test, Monte Carlo Permutation test, Random Forest test, PySpark, nominal data

Contents

1 Introduction

Over the past few decades, the world has become more and more digitalized. Therefore, nearly every company has to deal with datasets. They come in any form or shape. When companies have datasets, they would like to extract information from them. This information can be a simple query, like "does person A work here?", or "was product B sold last month?". Even more interesting are the relations among the variables in the dataset. For example, "if I change A, will this have an effect on B?". This process is an example of structure learning in causal inference. Say we have a dataset with a fixed amount of variables. The more data available for each of the variables, the more reliable the conclusions we draw, since we have more data to back it up. In other words, if variable A is a cause of variable B, we are more likely to detect this if we have more data on both variable A and B, since the probability that the detected relation is due to chance becomes smaller as the sample size increases. To illustrate this, we make use of the Asia dataset [17][6], a synthetic dataset often used for structure learning. The ground truth is illustrated in a directed acyclic graph (DAG) in Figure 1.

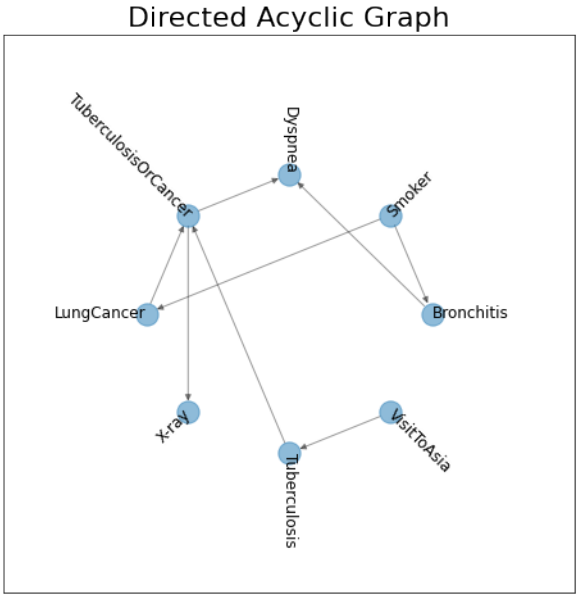


Figure 1: The DAG representing the ground truth of the structure in the Asia dataset. An edge between two variables implies that those variables are dependent on one another. If there is no edge connecting two variables, then those variables are (conditionally) independent.

A structure learning algorithm, called PC-stable [9][3], is applied to this dataset. The amount of taken samples n is varied, with $n \in \{100, 1000, 10\ 000\}$. The quality of the resulting DAG is measured using the F1-score, which can obtain values in $[0, 1]$. The higher the F1-score is, the better. The resulting DAGs and the corresponding F1-scores are depicted in Figures 2, 3 and 4.

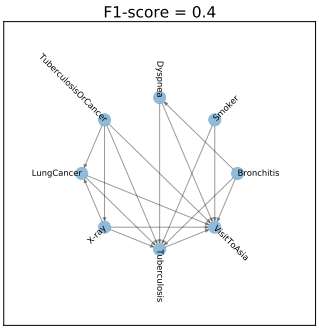


Figure 2: Resulting DAG for the Asia dataset, with $n = 100$, $\alpha = 0.01$.

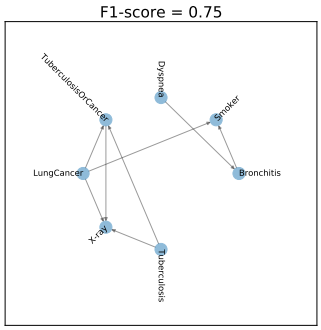


Figure 3: Resulting DAG for the Asia dataset, with $n = 1000$, $\alpha = 0.01$.

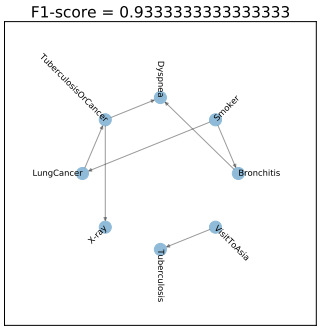


Figure 4: Resulting DAG for the Asia dataset, with $n = 10\ 000$, $\alpha = 0.01$.

As can be concluded by the increasing F1-scores in figures 2, 3 and 4 respectively, the more samples available, the closer we get to the true DAG. Since more data is not only beneficial to structure learning but to performing statistics in general, both the amount of datasets, as well as the size of datasets, continues to grow. Nowadays, some datasets are so large or complex that it is difficult or impossible to process using traditional methods [15]. This is what is called *big data* [15]. Big data can allow us to draw conclusions about the relations among variables which we were not able to draw before. This is indicated by the DAGs found for the Asia example, where an increasing sample size leads to more correctly detected independencies. To find those relations, conditional independence (CI) tests are utilized. More specifically, constraint-based algorithms use CI tests to detect relations. An often used constraint-based algorithm for causal inference is the PC algorithm [9][29][30], or variants thereof [10][26][14][18][35]. The PC algorithm returns a directed acyclic graph (DAG), in which an arrow from A to B indicates that it is assumed that a relation exists between A and B. We say assumed, because initially all variables are assumed to be dependent on one another, until proven otherwise. This process of trying to prove independence is done with CI tests. There is a variety of CI tests. Each of which has an advantage over the others. Unfortunately, not many CI tests are implemented yet to handle big data, since causal inference from observational big data is a relatively new research area, compared to the timespan in which causal inference has been researched. The main difference between big data and "normal" data, is that big data is stored in a distributed manner, because it is too big to be stored in local memory. Therefore, we turn to PySpark [25]. PySpark allows us to write Python code while using Apache Spark [5], an engine specifically designed to handle large-scale data processing.

In this thesis, we implement and compare three already existing CI tests in PySpark, in order to handle big data. The three CI tests used are: Pearson's chi-squared test [13][34] (from now on referred to as chi-squared test), the Monte Carlo permutation test [34] (MCPT) and the Random Forest test [31] (RFT), of which the latter two did not yet have an implementation to handle big data, to the best of our knowledge. We substitute those CI tests in the PC-stable algorithm [3], which is already implemented in a Python package designed to create probabilistic graphical models, called `pgmpy` [4]. We show the strengths and weaknesses of each CI test. Based on those strengths and weaknesses, we propose a hybrid approach in which several CI tests are used to learn a network structure. This allows us to to use each CI test when we expect it to perform best, given a certain situation.

The three mentioned CI tests are applicable when the data is categorical. Hence, we only consider datasets consisting of nominal variables in this paper. Given the scope of the thesis and the time it takes to obtain sufficient results, only binary variables are considered. Furthermore, we assume that no dataset has missing values.

1.1 Research Question

The research question we try to answer is as follows: How does an increasing amount of available data influence the performance of different CI tests in structure learning for binary data? We evaluate the conditional independence tests in terms of calibration, power and runtime. The performance of the structure learning algorithm is measured using the F1-score.

1.2 Hypotheses

As we gather more data, we can obtain a more accurate representation of the true relations among the variables via structure learning. Each of the CI tests has its own strengths and weaknesses. Therefore, we expect to observe a shift in performance as we increase the sample size or the amount of variables that we condition on. We expect the chi-squared test and the MCPT to lose power as we increase the size of our conditioning set, since those CI tests compute statistics for each unique combination of states in the conditioning set. This results in a summation of multiple statistics, each of which has been computed on a subsample of the provided data. We expect the chi-squared test to outperform the MCPT and the RFT in terms of runtime in all scenarios. We expect the MCPT to outperform the chi-squared test and the RFT in terms of calibration and power in small sample cases, but to become infeasible for larger sample sizes. We expect the RFT to outperform the chi-squared test and the MCPT in terms of calibration and power, when there is a low ratio between sample size and size of the conditioning set, due to its assumed characteristic that it is robust to noise [31]. We expect that using a hybrid approach leads to a better F1-score, compared to using the fastest CI test, while maintaining feasible runtime. Furthermore, we expect that using a hybrid approach performs significantly faster compared to the CI test that leads to the highest F1-score, while preserving comparable F1-scores.

1.3 Thesis Structure

This paper is structured as follows: We first provide relevant background information, where some definitions are explained (Section 2.1) and related literature is discussed (Section 2.2). We then provide intuition on the workings and result of the PC algorithm (Section 2.3). Afterwards, we go into the workings of each of the CI tests used, by providing low-level mathematical information (Section 2.4). Then our experimental setup is explained (Section 3), i.e., the network structures of our datasets we used to validate our CI tests (Sections 3.1, 3.2 and 3.3). We then state the necessary specifications to reproduce the results (Section 3.4). The following section (Section 4) is dedicated to providing empirical evidence that our implementations are valid as CI tests, by showing the calibration and power of each CI test in various situations. Here we also show the runtimes for each CI test in various scenarios in order to check if a CI test has feasible runtime. This section also contains the structure learning results, for each of the CI tests. A hybrid approach is proposed based on those results (Section 5). Here we immediately show the performance of the hybrid approach compared to the CI tests that make up the hybrid approach. We then conclude with our conclusions and discussion (Section 6).

2 Background

Before we can go into the theory, we first state relevant definitions, in order to avoid ambiguity.

2.1 Definitions

Two variables X and Y are said to be *independent* (notation: $X \perp\!\!\!\perp Y$), if $P(x) \cdot P(y) = P(x, y)$. Two variables X and Y are said to be *conditionally independent* (notation: $X \perp\!\!\!\perp Y | \boldsymbol{Z}$), if $P(x|\boldsymbol{Z} = z) \cdot P(y|\boldsymbol{Z} = z) = P(x, y|\boldsymbol{Z} = z)$, for all z with $P(z) > 0$ [12]. A k-level variable is a categorical variable which can obtain k different states. $T(X, Y|\boldsymbol{Z})$ is the statistical test, if X is independent of Y, given observations in $\boldsymbol{Z}$. If $\boldsymbol{Z} = \emptyset$, then it is a test of independence, rather than a test of conditional independence. A *contingency table* is a table containing the observed frequencies of certain combinations of states for multiple categorical variables. When we talk about a *stratum*, we refer to a "layer" of the contingency table. So, if we have a contingency table of 3 variables X, Y and Z, then we split the 3-dimensional contingency table in seperate 2-dimensional contingency tables, one for each state in Z. Each of those 2-dimensional contingency tables is then called a stratum of the 3-dimensional contingency table. A *graph* $G = (\boldsymbol{V}, \boldsymbol{E})$ consists of a set of vertices $\boldsymbol{V}$ and a set of edges $\boldsymbol{E}$. A *path* is a sequence of vertices, which are connected by edges. A *directed edge* is an edge between vertices that is oriented towards exactly one of those vertices (notation: $X_i \rightarrow X_j$ for a directed edge from vertex X_i to vertex X_j). A *directed path* is a path which follows the direction of directed edges. A *cycle* is a directed path in which a variable can reach itself, by following outgoing directed edges. A *skeleton graph* is a graph of which the edges are undirected. A *partially directed acyclic graph* (PDAG) is a graph which contains directed edges and undirected edges, in which there is no path forming a cycle. A *directed acyclic graph* (DAG) is a graph where all edges are directed, in which there is no path which forms a cycle. If $X_i \rightarrow X_j \leftarrow X_k$ and vertices X_i and X_k are not directly connected by an edge, X_j is a *collider* of X_i and X_k. The next two definitions are identical to the way Colombo et al. defined them [10]. Three vertices $\langle X_i, X_j, X_k \rangle$ are called an *unshielded triple*, if X_i and X_j are adjacent, X_j and X_k are adjacent, but X_i and X_k are not adjacent [10]. An unshielded triple $\langle X_i, X_j, X_k \rangle$

is called a *v-structure*, if X_j is a collider on the path $\langle X_i, X_j, X_k \rangle$ [10]. Our null hypothesis (H0) for $T(X, Y|\boldsymbol{Z})$ is that X and Y are independent given $\boldsymbol{Z}$. A *significance level* is a cut-off value for which we decide if a p-value is significant. Throughout this paper, a significance level will be denoted by the greek symbol α. If p-value $< \alpha$, we will reject H0. Two vertices X_i and X_j are said to be *d-separated* by a subset of vertices in $\boldsymbol{V} \backslash \{X_i, X_j\}$, if there is not an edge directly between X_i and X_j in the graph $G = (\boldsymbol{V}, \boldsymbol{E})$. The power of a CI test is the probability that H0 is correctly rejected. A calibrated test incorrectly rejects H0 with a ratio equal to the significance level. *Selection variables* are variables that determine whether or not a measured unit is included in the data [10].

2.2 Shoulders of Giants

In this thesis, we heavily rely on former research. The constraint-based PC algorithm introduced in 1993 by Spirtes et al. [30] and the PC-stable variant thereof, presented by Colombo et al. [9] in 2014, form the very foundation of this thesis. The working and intuition behind it is elaborated in the next subsection, Section 2.3. We make use of three different existing CI tests: Pearson's chi-squared test, the Monte Carlo Permutation test and the Random forest test. The chi-squared test, introduced in 1900 by Karl Pearson [13], is often used as a test of independence when categorical data is considered. The test statistic of the chi-squared test asymptotically approaches the $\mathcal{X}^2$ distribution [34], hence it is correct for an infinite sample size. Even for big data the amount of samples is limited. Therefore, we will also consider other CI tests, which provide reliable results with a smaller sample size. One approach could be exact testing, but as pointed out by Agrasti, this quickly becomes infeasible as the runtime increases exponentially with the sample size [1]. Instead, a Monte Carlo based approach is used. The Monte Carlo Permutation test was introduced in the late 1940s by Ulam et al. [21]. The third CI test, called the Random Forest test, was first presented in 2021 by Textor et al. [31]. This test was shown to be effective in small sample cases when conditioning on multiple variables. The calibration and power of the RFT seem unperturbed by an increasing amount of variables that is conditioned on. Hence, it seems robust to noise. There is no reason to assume that RFT's property to discard irrelevant information diminishes as we increase the available sample size. Therefore, the RFT is expected to be a good alternative to the chi-squared test, when there are multiple variables in the conditioning set. Each of those CI tests is substituted in the PC-stable algorithm. This algorithm is already implemented in a Python package called `pgmpy` [4], which is written by Ankur Ankan and was first published in 2015. The way it is implemented allows us to easily substitute our CI tests. In this thesis, we create numerous synthetic datasets. The structures of the datasets to measure calibration and power are identical to the structures proposed by Textor et al. [31]. The synthetic datasets used to measure the quality of our DAGs after structure learning are simulated with `Dagitty` [32]. The research performed in this thesis is inspired by, and built upon, the knowledge provided by formerly mentioned publications.

2.3 PC Algorithm

In order to create a DAG which represents the relations among variables in a dataset, we need to detect the independencies among the variables. For this, we make use of a variation of the constraint-based algorithm called the PC algorithm [30], namely PC-stable [9], as implemented in `pgmpy` [4][3]. The stable variant, PC-stable, is used to get more robust results [9], in the sense that the resulting skeleton graph is independent of the ordering of the CI tests. We substitute `pgmpy`'s CI tests with our custom CI tests; chi-squared test [34], Monte Carlo Permutation test [34], and Random Forest test [31]. Note that `pgmpy` already contains the chi-squared test. However, this implementation does not work for big data, since it relies on dataframes stored in local memory. The Random Forest test is valid for $T(X, Y|\boldsymbol{Z})$ if $\boldsymbol{Z} \neq \emptyset$, while the chi-squared test and Monte Carlo Permutation test are valid both as a test of independence as well as a test of conditional independence. The pseudocode of the PC-stable algorithm implemented in `pgmpy` is as follows:

Algorithm 1: PC-stable [3]

Result: DAG
Start with the fully connected undirected graph $G = (V, E)$ on all variables;
$l = 0$;
`/* Step 1, obtain the skeleton graph and the separating sets */`
while $l \leq d$ **and** $\exists v \in V : |N(v)| \geq l$ **do**
  Compute the set of neighbours $N(v)$ for each $v \in V$;
  for $(u, v) \in E$ **do**
    for $Z \subset \{N(v) \setminus \{u\} : |Z| = l\}$ *or* $Z \subset \{N(u) \setminus \{v\} : |Z| = l\}$ **do**
      obtain $p(u, v|Z)$ from $T(u, v|Z)$;
      if $p(u, v|Z) \geq \alpha$ **then**
        Remove edge (u, v) from graph G;
        Store the separating set Z in S;
        break;
      end
    end
  end
  $l = l + 1$;
end
`/* By now, G is a skeleton graph */`
`/* Step 2, skeleton to pdag by orienting the edges */`
Direct all edges in the skeleton both ways;
for each $X \leftrightarrow Z \leftrightarrow Y$, if Z not in the separating set of X,Y, then orient edges as $X \rightarrow Z \leftarrow Y$;
progress = True;
while *progress* **do**
  num_edges = amount of edges in G;
  for each $X \rightarrow Z \leftrightarrow Y$, orient edges to $X \rightarrow Z \rightarrow Y$;
  for each $X \leftrightarrow Y$ with a directed path from X to Y, orient edges to $X \rightarrow Y$;
  for each $X \leftrightarrow Z \leftrightarrow Y$ with $X \rightarrow W$, $Y \rightarrow W$, and $Z \leftrightarrow W$, orient edges to $Z \rightarrow W$;
  progress = num_edges > amount of edges in G;
end
`/* By now, G is a PDAG */`
`/* Step 3, PDAG to DAG by orienting remaining undirected edges */`
if *A faithful extension exists* **then**
  Orient edges in PDAG such that the resulting DAG has the same v-structures as the PDAG;
else
  Orient remaining undirected edges arbitrarily;
end
`/* Step 4, return the DAG */`
Return DAG

where

- $V \subset G$ is the set of variables of the provided dataset.
- d is the maximum allowed size of sets to condition on.
- S is the set of separating sets.
- $p(u, v|Z)$ is the p-value of the independence test, which tests if u and v are independent conditioned on Z.
- α is the significance level.
- $N(v)$ is the set of neighbours for $v \in V$.

The quality of the resulting DAG relies both on the chosen significance level α as well as on the quality of the used CI test. For a predetermined significance level α, an edge A - B will be removed if the p-value of $T(A, B|Z) \geq \alpha$.

This implies that as long as we can reject our null hypothesis of independence, i.e., we find a p-value $< \alpha$, then we still assume dependence. As soon as we do not find enough evidence to reject H0, the edge will be removed and the variables are said to be independent. For this to work, we need a CI test that is well calibrated and has high power. If a dependency exists, we need to detect it, since once an edge is removed, it will not come back. Incorrectly removing an edge is therefore an irreversible error. On the other hand, if two variables are indeed independent, so the null hypothesis is true, we expect to find uniformly distributed p-values, for a well calibrated test. This means that for two independent variables, we correctly remove the edge with probability 1 - α. Failing to remove an edge is not as bad as incorrectly removing an edge. This is due to the fact that an edge is tested multiple times, since the conditioning sets vary. Furthermore, when we increase d, we perform the same test with an extra variable in the conditioning set. Therefore, it is very likely that the edge will be rejected in one of the next tests performed, and the mistake will be rectified.

When all necessary CI tests are performed, the skeleton graph consists of vertices and undirected edges. The edges are then directed in such a way that the detected independencies in the data are equal to the dependencies implied by the DAG via d-separation, if such a DAG exists. See "Learning Bayesian Networks" by Neapolitan et al., Algorithm 10.2 (page 550) [22] for a more elaborate description.

The PC-stable algorithm returns a DAG. Multiple DAGs can fit the same data. To provide some intuition for this, imagine the DAG

$$A \xrightarrow{f} B \xrightarrow{f} C, \text{ where } f(x) = 2x$$

If this fits the data, then a DAG

$$A \xleftarrow{g} B \xleftarrow{g} C, \text{ where } g(x) = \frac{1}{2}x$$

fits the data equally well. Therefore we only focus on the first step of the PC-stable algorithm, i.e., finding the skeleton graph.

2.3.1 Assumptions

The PC(-stable) algorithm assumes *acyclicity* and that the provided dataset contains independent and identically distributed samples. Furthermore, the PC(-stable) algorithm is sound and complete under the assumptions of *causal sufficiency* and *faithfulness* [10].

- *Acyclicity*: The resulting DAG does not contain cycles.
- *Causal faithfulness*: The conditional independence relations among the variables are exactly equal to those implied by the DAG via d-separation [10][9].
- *Causal sufficiency*: There are no unmeasured common causes and no unmeasured selection variables [10].

In real datasets, causal sufficiency rarely holds. Therefore, for real data, PC(-stable) is not guaranteed to be sound and complete. However, we will show empirically that learning the structure of the skeleton graph still works when the assumption of causal sufficiency does not hold.

2.3.2 Measuring the Quality of the DAG

In order to determine the quality of a DAG, we make use of the F1-score, which is based on precision and recall. Only the presence or the absence of an edge is considered. Hence, the direction of the edges is irrelevant. If an edge is present in the DAG, we consider this the "positive" case. Likewise, if an edge is absent in the DAG, we consider this the "negative" case. The formulas for precision and recall are as follows:

$$\text{precision} = \frac{\text{Number of True Positives}}{\text{Number of True Positives} + \text{Number of False Positives}}$$

$$\text{recall} = \frac{\text{Number of True Positives}}{\text{Number of True Positives} + \text{Number of False Negatives}}$$

Which now allows us to define the F1-score

$$\text{F1-score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Each of those metrics returns values in $[0, 1]$, in which 1 is a perfect score and 0 is the worst possible outcome.

2.4 Conditional Independence Tests

Three CI tests are considered. The well known and widely used chi-squared test [13][34], a Monte Carlo permutation based approach to the chi-squared test [34] and a novel test called the Random Forest test recently proposed by Textor et al. [31].

Each test returns a p-value with which we determine whether we reject the null hypothesis or not. This p-value can lead to 4 different situations.

- True positive: An edge is present where it should be present. (H0 correctly rejected)
- False positive: An edge is present where it should be absent. (H0 incorrectly rejected)
- False negative: An edge is absent where it should be present. (H0 incorrectly not rejected)
- True negative: An edge is absent where it should be absent. (H0 correctly not rejected)

Remember that at the start of the PC algorithm we start with a fully connected graph. So initially our graph is full of true and false positives and our main aim is to remove as many false positives as possible (i.e., find true negatives), while leaving the true positives intact (i.e., avoid false negatives).

2.4.1 Chi-squared test

The chi-squared test makes use of a contingency table, and is valid as a test of independence as well as a test of conditional independence. The contingency table is used to compute a $\mathcal{X}^2$ value and the degrees of freedom (dof).

For an independence test $T(X, Y)$, the contingency table for any two categorical variables X and Y takes the following form:

		Y			
		Y_1	$Y_{..}$	Y_c	Total
X	X_1	$x_{1,1}$	$x_{1,..}$	$x_{1,c}$	$x_{1,total}$
	$X_{..}$	$x_{..,1}$	$x_{..,..}$	$x_{..,c}$	$x_{..,total}$
	X_r	$x_{r,1}$	$x_{r,..}$	$x_{r,c}$	$x_{r,total}$
	Total	$x_{total,1}$	$x_{total,..}$	$x_{total,c}$	$x_{total,total}$

Table 1: Contingency table of an r-level variable X and c-level variable Y. The value for *total* is obtained by summing over the corresponding axis.

Say we have the following test $T(X, Y|\boldsymbol{Z})$. Then we do not merely get a 2-dimensional contingency table, but a multidimensional contingency table. The amount of dimensions is equal to the amount of variables that is conditioned on plus two. So if we condition on 4 variables, we get a 6-dimensional contingency table, and if we condition on 0 variables, we get a 2-dimensional contingency table (as expected, since this is the simple independence case). Luckily, this multidimensional contingency table can be split up into t 2-dimensional contingency tables, stratified on $\boldsymbol{Z}$, where $t = \prod_{Z \in \boldsymbol{Z}} |Z|$, with $|Z|$ the amount of different states variable $Z \in \boldsymbol{Z}$ obtains.

For each of the t obtained contingency tables, the rows and colums are removed which only contain zeroes. If one of the resulting contingency tables does not have both length and width at least two, that specific contingency table will not be considered, i.e., for those tables a 0 is added to the total $\mathcal{X}^2$ and 0 is added to the total (dof).

Define

$$O_{i,j,k} = x_{i,j} \text{ in contingency table } k$$

and

$$E_{i,j,k} = \frac{x_{total,j} \cdot x_{i,total}}{x_{total,total}} \text{ in contingency table } k \text{ for } i \in \{1, ..., r\},\ j \in \{1, ..., c\} \text{ and } k \in \{1, ..., t\}.$$

To compute the p-value for $T(X, Y|\boldsymbol{Z})$, we compute the $\mathcal{X}^2$ value and the dof for each contingency table and then sum those values over all of the contingency tables.

$$\mathcal{X}^2 = \sum_k \sum_{i,j} \frac{(O_{i,j,k} - E_{i,j,k})^2}{E_{i,j,k}}$$

The degrees of freedom dof is also required to compute the p-value.

$$dof = \sum_k (|X_k| - 1) \cdot (|Y_k| - 1)$$

where $|X_k|$ and $|Y_k|$ represent the length and width of the contingency table k respectively. Then the p-value for $T(X, Y|\boldsymbol{Z})$ is computed as follows:

$$\mathrm{p} = 1 - F_{dof}(\mathcal{X}^2)$$

where F_{dof} is the cumulative distribution function of the $\mathcal{X}^2$ distribution, with dof degrees of freedom.

If the p-value is greater than or equal to a previously chosen significance level, independence is assumed. Otherwise, we can reject H0 and still assume dependence until proven otherwise. The chi-squared test becomes more reliable as the sample size increases while other attributes remain the same, i.e., it is asymptotically correct. When the sample size is low, it can provide unreliable results, as is shown in Section 4. In this case, another test might be more appropriate.

An important aspect of the chi-squared test is that the more we stratify our data, the more power the chi-squared test loses when the effect remains the same. See the example below, where we show the power of the test $X \perp\!\!\!\perp Y$ and $X \perp\!\!\!\perp Y|Z$, where Z is a uniformly distributed binary variable.

		Y		
		Y_1	Y_2	Total
X	X_1	12	6	18
	X_2	6	12	18
	Total	18	18	36

Table 2: Contingency table of a binary variable X and a binary variable Y

		Y		
Z_1		Y_1	Y_2	Total
X	X_1	6	3	9
	X_2	3	6	9
	Total	9	9	18

Table 3: Contingency table of a binary variable X and a binary variable Y conditioned on $Z = Z_1$

		Y		
Z_2		Y_1	Y_2	Total
X	X_1	6	3	9
	X_2	3	6	9
	Total	9	9	18

Table 4: Contingency table of a binary variable X and a binary variable Y conditioned on $Z = Z_2$

The $\mathcal{X}^2$ values of Tables 2, 3 and 4 are 4, 2 and 2 respectively. The degrees of freedom are 1 in each table.

In each of the tables, $\frac{2}{3}$ of the entries match for X and Y. For the first table, Table 2, we obtain a p-value of $1 - F_1(4) \approx 0.0455$, while combining the information in Table 3 and Table 4 gives a p-value of $1 - F_2(4) \approx 0.1353$, where $F_{dof}(\mathcal{X}^2)$ is the cumulative distribution function of the $\mathcal{X}^2$ distribution, with dof degrees of freedom. If we have a significance level $\alpha = 0.05$, then we would reject H0 based on the non-stratified test and still assume dependence, while based on the stratified data we would not reject H0 and assume independence. Hence, we lose power by stratifying.

2.4.2 Monte Carlo Permutation test

Rather than merely having reliable results for large sample sizes, it is also possible to compute the exact p-value which is correct for any sample size, if the assumptions under the null hypothesis hold.

Say we have a contingency table like Table 1. The marginals, present in the row and column called Total, are known. Under the null hypothesis, which states that X and Y are independent, each possible contingency table with those marginals is equally likely. To compute the exact p-value, one needs to sum the multivariate hypergeometric probabilities of all tables that have $\mathcal{X}^2$ at least as large as the one observed [1], for all contingency tables with the same marginals. This is feasible for small sample sizes and small contingency tables, but as the total sample size

increases, the amount of possible permutations grows exponentially [2]. If n is fixed, but the width or length of the contingency table increases, the amount of possible permutations again grows rapidly [2]. See table 5 for a few examples Agrasti [2] provided.

		Sample size n	
		20	100
Size of the contingency table	4 by 4	$4.0 \cdot 10^4$	$7.2 \cdot 10^9$
	5 by 5	$2.1 \cdot 10^6$	$9.2 \cdot 10^{14}$

Table 5: Maximum cardinality [2]

As depicted in table 5, for a 5 by 5 contingency table with only a 100 samples, we can already obtain nearly a quadrillion (!!!) tables with the same margins. To ensure that each test we perform is tractable, we can use Monte Carlo sampling to take a subset of contingency tables out of all the possible contingency tables with those fixed marginals. For each of those permutations, a $\mathcal{X}^2$ statistic is computed. This approach is no longer exact, but merely another way to approximate the p-values.

Following the pseudocode presented by Tsamardinos et al. in Algorithm 1 (page 327) [34], the p-value that results from the MCPT can be obtained as follows:

$$\mathrm{p} = \sum_{i}^{n_{\text{perm}}} \frac{\mathbb{1}_{\{\mathcal{X}^2_{\text{observed}} \leq \mathcal{X}^2_i\}}}{n_{\text{perm}}}$$

where $\mathcal{X}^2_{\text{observed}}$ is the $\mathcal{X}^2$ value of our original contingency table, $\mathcal{X}^2_i$ is the $\mathcal{X}^2$ value of permutation i, $\mathbb{1}$ is the indicator function and n_{perm} is the total amount of permutations.

To save runtime for easy to determine cases, we use the same heuristic as presented by Tsamardinos et al. [34], where we accept independence if the p-value of the chi-squared test for $\mathcal{X}^2_{\text{observed}}$ and the observed degrees of freedom is greater than or equal to 0.5. The intuition behind this is that the chi-squared test often returns small p-values, i.e., < 0.1. If the chi-squared test returns a p-value ≥ 0.5, it is often safe to assume independence. This leads to the chi-squared test and MCPT to be equal when the chi-squared test returns a p-value greater than or equal to 0.5.

If the p-value is greater than or equal to a previously determined significance level, independence is assumed. Otherwise, we still assume dependence until proven otherwise.

Like the chi-squared test, the Monte Carlo permutation test (MCPT) is also valid both as a test of independence as well as a test of conditional independence.

2.4.3 Random Forest test

The implementation of the Random Forest test is entirely based on the description in the original paper by Textor et al. [31] and their implemented R code.

Unlike the previously mentioned tests, the Random Forest test (RFT) is only valid as a conditional independence test. This is due to the fact that the forests used in the RFT are trained on the observations in the conditioning set. If there is no conditioning set, there is no data to train any forests.

For a test of conditional independence $T(X, Y|\boldsymbol{Z})$, the RFT works as follows: Train two random forests using only the features in $\boldsymbol{Z}$. One forest to predict X given $\boldsymbol{Z}$, and one forest to predict Y given $\boldsymbol{Z}$. Each forest is trained on all samples. Then a sample probability distribution $\hat{p}$ is computed for both forests, by predicting all samples. Since training and testing is performed on the same samples, one might worry about overfitting. But since we are trying to find the sample distribution, rather than creating a forest that is generalizable to new data, this does not matter.

In order to get the test statistic, we first need to perform a lot of computations.

First, define the conditional residual $R_{x_i|z_i}$ as follows [19][31]:

$$R_{x_i|z_i} = \hat{p}(X < x_i|Z = z_i) - \hat{p}(X > x_i|Z = z_i)$$

Since we will merely use this on indicator variables, which are binary, this can be simplified to

$$R_{x_i|z_i} = x_i - \hat{p}(X = 1|Z = z_i)$$

where $x_i = 1 \leftrightarrow x_i$ is the true label.

To illustrate the calculations, let us assume we have a small dataset consisting of three variables: X, Y and Z, with just 5 entries.

X	Y	Z
1	2	small
2	1	big
1	1	big
3	2	small
4	3	small

Table 6: Small dataset for illustrative purposes.

Assume we want to test $T(X, Y|Z)$. We start by dummy encoding [8] each dependent variable. So X becomes (1, 0, 1, 0, 0), (0, 1, 0, 0, 0) and (0, 0, 0, 1, 0), and Y becomes (0, 1, 1, 0, 0), (1, 0, 0, 1, 0). Notice that there is no vector representing the indices where the original vector X has value 4, nor where Y has value 3. This is because that information is already incorperated in the combination of the other vectors.

In order to compute our test statistic, we first need a vector d, which is the pairwise dot product of our residuals. In order to obtain d, we take the dummy encoded labels of X and Y and substract the probabilities for each outcome of those vectors, given by $\hat{p}$. For brevity, only a calculation to obtain the first row will be shown here. Say that by conditioning on Z we found these probability distributions:

$$\hat{p}_{X|Z=\text{small}} = (0.2, 0.1, 0.3, 0.4)$$
$$\hat{p}_{X|Z=\text{big}} = (0.3, 0.4, 0.2, 0.1)$$
$$\hat{p}_{Y|Z=\text{small}} = (0.2, 0.5, 0.3)$$
$$\hat{p}_{Y|Z=\text{big}} = (0.7, 0.2, 0.1)$$

Then the first row of d will be the pairwise dot product of R_{X_1} and R_{Y_1}, where

$$R_{X_1} = (1, 0, 0) - (0.2, 0.1, 0.3) = (0.8, -0.1, -0.3)$$

and

$$R_{Y_1} = (0, 1) - (0.2, 0.5) = (-0.2, 0.5)$$

Again, note that we do not need the probabilities of obtaining value 4 and 3 for X and Y respectively. This is due to the fact that the residuals should add up to 1, and therefore all information is captured by this vector.

The first row of d then becomes

$$(0.8 \cdot -0.2, 0.8 \cdot 0.5, -0.1 \cdot -0.2, -0.1 \cdot 0.5, -0.3 \cdot -0.2, -0.3 \cdot 0.5)$$
$$= (-0.16, 0.4, -0.02, -0.05, 0.06, -0.15)$$

This is done for each sample in our dataset, so we end up with a matrix d of shape $(n, (r-1) \cdot (c-1))$, where n is the amount of samples in our dataset, and r and c are the amount of levels of our dependent variables. In our running example, we would end up with a shape of $(5, 6)$ for d.

Finally, the test statistic S is defined as

$$S(\boldsymbol{X}, \boldsymbol{Y}) = \frac{d \cdot \hat{\Sigma}_d^{-1} \cdot d^T}{n}$$

where $\hat{\Sigma}_d$ is the estimated covariance matrix of d [31].

$S(\boldsymbol{X}, \boldsymbol{Y})$ is $\mathcal{X}^2$ distributed with $(c-1)\cdot(r-1)$ degrees of freedom, as is shown by Textor et al. [31]. As it was for the chi-squared test, the p-value for $T(X, Y|\boldsymbol{Z})$ is computed as follows:

$$\mathrm{p} = 1 - F_{dof}(S(\boldsymbol{X}, \boldsymbol{Y}))$$

where F_{dof} is the cumulative distribution function of the $\mathcal{X}^2$ distribution, with dof degrees of freedom and $S(\boldsymbol{X}, \boldsymbol{Y})$ is the test statistic.

If the p-value is greater than or equal to a previously determined significance level, independence is assumed. Otherwise, we still assume dependence until proven otherwise.

3 Experimental Setup

We now have a total of three CI tests, of which only the RFT is unusable if we have no observed variables. So in the case of $T(X, Y)$, we need to determine whether we want to perform the chi-squared test, or the MCPT or no test at all. If we have a CI test $T(X, Y|\boldsymbol{Z})$, we need to determine whether it is best to perform the chi-squared test, the MCPT or the RFT.

Before using any CI tests, it is important to know if those CI tests even work. Two important notions of CI tests are calibration and power. The calibration of a test represents the amount of null hypotheses that are wrongfully rejected, which should be equal to the significance level by definition of the p-value, when the test is well-calibrated. The power of a test represents the ratio of correctly rejected null hypotheses. In order to measure calibration or power, a ground truth has to be known. Therefore, synthetic datasets are constructed in which the ground truths are known.

The network structures used to generate synthetic datasets for calibration and power are equal to the network structures proposed by Textor et al. [31].

3.1 Synthetic Dataset Calibration

In order to show how well the tests are calibrated in different scenarios, we will show 16 distinct cases. For each of those cases, 400 datasets are generated using the structure depicted in Figure 5.

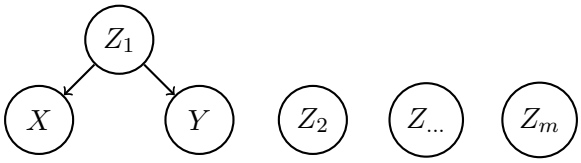


Figure 5: Network structure used for calibration. Values generated for X and Y depend on values in Z_1. All other variables are independent of one another.

For the calibration plots, only binary variables are considered. For each Z_i, we draw n samples from the discrete uniform distribution on the interval $[1, 2]$. X and Y are both generated using only the values in Z_1. For each $x_i \in X$, with $i \in \{1, ..., n\}$, we draw from the Bernoulli distribution $Bernoulli(Z_{1_i}/3)$ and 1 is added to this value. The same holds for each $y_i \in Y$. With those initialisations, X and Y are expected to match Z_1 in $\frac{2}{3}$ of the entries.

For our calibration plots, we will both vary the amount of samples n present in the dataset ($n \in \{20, 50, 200, 10\ 000\}$), as well as the amount of variables k that is conditioned on ($k \in \{0, 1, 3, 5\}$). Remember that the RFT is only valid when $k \geq 1$, so in the case $k = 0$, the RFT will not be performed. Furthermore, when $n = 10\ 000$ and $k = 0$, a MCPT of 5000 simulations takes about 12 minutes per test. This runtime is obtained using the specifications specified in the upcoming section, Section 3.4, with the only difference that 2 executor instances were used instead of 4. Performing 400 tests is not feasible, and therefore it will not be performed. Note that since the MCPT takes about 12 minutes in such cases, we will not perform it in real world applications. Hence, how well it is calibrated is not interesting for practical reasons. The same holds for performing the MCPT when $n = 10\ 000$ and $k \in \{1, 3, 5\}$, where a single MCPT takes about 10 minutes in each case.

The conditional independence test $T(X, Y|\boldsymbol{Z})$ is performed on different combinations of nodes when the amount of variables k in $\boldsymbol{Z}$ varies, see Table 7. Note that X and Y are only dependent on Z_1, so X and Y are independent when we condition on Z_1. When we add Z_i with $i \in \{2, ..., k\}$ to the conditioning set, the independence still holds, since Z_1 is still in the conditioning set, but it will be harder to detect due to the noise introduced by the irrelevant variables.

k	X	Y	$\boldsymbol{Z}$
0	Z_1	Z_2	$\emptyset$
1	X	Y	$\{Z_1\}$
3	X	Y	$\{Z_1, Z_2, Z_3\}$
5	X	Y	$\{Z_1, Z_2, Z_3, Z_4, Z_5\}$

Table 7: Variables used for the test $T(X, Y|\boldsymbol{Z})$ for various levels of k.

3.2 Synthetic Dataset Power

To measure the power of the tests, a network structure similar to the one used for calibration is used. The only difference is that there is an edge from X to Y, see figure 6. Again, this structure is identical to the one Textor et al. used [31].

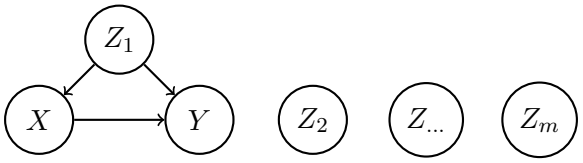


Figure 6: Network structure used for power. Values generated for X are dependent on values in Z_1. Values generated for Y depend both on values in X as well as values in Z_1. All other variables are independent of one another.

The values present in the datasets to measure power do differ from the values present in the datasets to measure calibration. Again, only binary values are considered, but now the entries for each Z_i are from the set $\{0, 1\}$, rather than $\{1, 2\}$. For each Z_i, we draw from the discrete uniform distribution on the interval $[0, 1]$, which is the same as flipping a fair coin or sampling from the Bernoulli distribution with $p = \frac{1}{2}$. We then first generate values for X, using the values of Z_1, by employing the logistic function with an integrated effect size, again based on the paper introducing the Random Forest test [31].

$$P(X_i = 1) = \frac{e^{\beta \cdot Z_{1_i}}}{e^{\beta \cdot Z_{1_i}} + 1}$$

This way, the effect size β can be controlled. If $\beta = 0$, the values in X are expected to match the values in Z_1 in $\frac{e^0}{e^0 + 1} = \frac{1}{2}$ of the cases. Similarly put, it is completely random (no effect).

If $\beta = 1$

$$P(X_i = 1|Z_{1_i} = 1) = \frac{e}{e + 1} \approx 0.73$$

and

$$P(X_i = 1|Z_{1_i} = 0) = \frac{e^0}{e^0 + 1} = \frac{1}{2}$$

This means that for entry i in Z_1 where $Z_{1_i} = 1$, the entry i in X is expected to match with probability approximately 0.73, while for entry j where $Z_{1_j} = 0$, X_j still is randomly generated.

After X is generated, Y is generated, which depends both on X and Z_1. Y is generated in the same fashion as X, however it now both depends on the values present in X and Z_1.

$$P(Y_i = 1) = \frac{e^{\beta \cdot (Z_{1_i} + X_i)}}{e^{\beta \cdot (Z_{1_i} + X_i)} + 1}$$

Since Z_{1_i} and X only contain values in the set $\{0, 1\}$, $Z_{1_i} + X_i$ has values in $\{0, 1, 2\}$.
If $Z_{1_i} + X_i = 0$:

$$P(Y_i = 1|Z_{1_i} + X_i = 0) = \frac{e^{\beta \cdot 0}}{e^{\beta \cdot 0} + 1} = \frac{1}{2}$$

If $Z_{1_i} + X_i = 1$:

$$P(Y_i = 1|Z_{1_i} + X_i = 1) = \frac{e^{\beta \cdot 1}}{e^{\beta \cdot 1} + 1} = \frac{e^{\beta}}{e^{\beta} + 1}$$

If $Z_{1_i} + X_i = 2$:

$$P(Y_i = 1 | Z_{1_i} + X_i = 2) = \frac{e^{\beta \cdot 2}}{e^{\beta \cdot 2} + 1} = \frac{e^{2\beta}}{e^{2\beta} + 1}$$

For our power plots, we used $\beta \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$, where $\beta = 0$ represents no effect and $\beta = 1$ represents a strong effect. We perform the conditional independence test $X \perp\!\!\!\perp Y | Z$ with $Z \in \{\emptyset, \{Z_1\}, \{Z_1, Z_2, Z_3\}, \{Z_1, Z_2, Z_3, Z_4, Z_5\}\}$.

3.3 Synthetic Datasets Structure Learning

After the performance for each of the three CI tests is known in terms of calibration, power and runtime, we also put the CI tests to practice. We substitute each of the CI tests in the PC-stable algorithm, as implemented in `pgmpy` [3]. We will use two synthetic datasets for structure learning. The first synthetic dataset is described by Schipf et al. [27]. The true network structure of this dataset consists of 7 nodes and 14 edges and is depicted in Figure 7. Our generated data which follows this network structure will be referred to as dataset A. The second synthetic dataset is described by Polzer et al. [16]. The original network structure consists of 14 nodes and 69 edges. However, after generating the data, we removed the last 5 variables when lexicographically sorted. By removing those 5 variables, we purposely violated the causal sufficiency assumption, since we know that there are exactly 5 hidden variables. The network structure we worked with consists of 9 nodes and 24 edges and is depicted in Figure 8. We will refer to this generated dataset as dataset B.

For both dataset A and B with its original structure we generated 10 million samples using `Dagitty`'s `simulateLogistic` [11]. For dataset A, we used an effect size `b.default` $\in \{0.1, 0.4\}$. For dataset B, we used an effect size `b.default` of 0.1. For each of those three situations, we only generated a single dataset with 10 million rows. When performing structure learning on a subset of the dataset, we used a random sample of this dataset of size n.

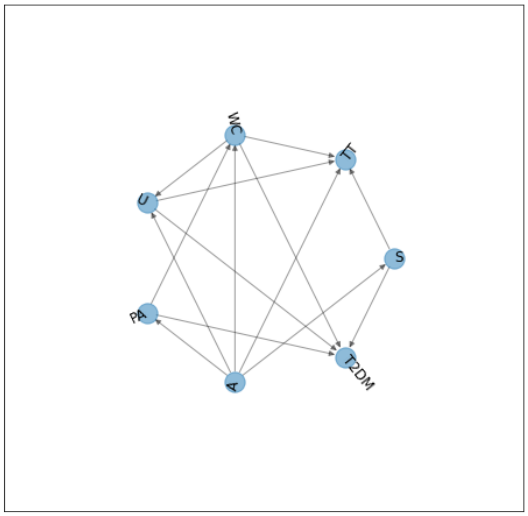


Figure 7: True DAG of dataset A.

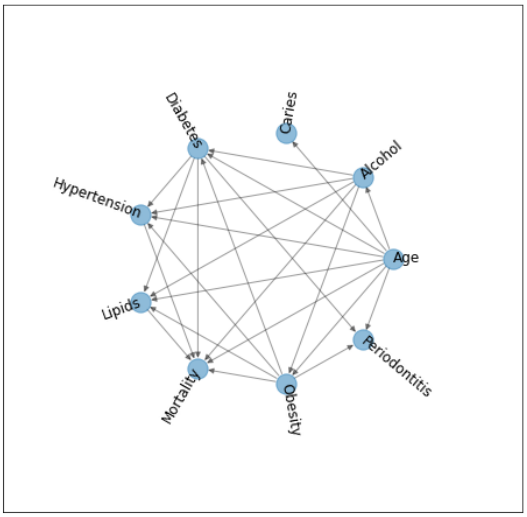


Figure 8: True DAG for the first 9 lexicographically sorted variables of dataset B.

Both graphs have the same density: $\frac{14}{\binom{7}{2}} = \frac{14}{21} = \frac{2}{3}$ for dataset A and $\frac{24}{\binom{9}{2}} = \frac{24}{36} = \frac{2}{3}$ for dataset B. However, the graph of dataset B contains a very dense subgraph. If "Caries" is removed, the graph consists of 8 nodes and 23 edges, which results in a density of $\frac{23}{\binom{8}{2}} = \frac{23}{28} \approx 0.82$. Therefore, the graph of dataset B is considered to be denser than the graph of dataset A. Since the graph of dataset B contains a dense subgraph on more variables than the total amount of variables in dataset A, we expect the PC-algorithm to reach a deeper depth d, which in turn leads to a loss in power for stratified CI tests like the chi-squared test and the MCPT.

3.4 Specifications

All simulations were performed using a PySpark session, which was instantiated with the same resources available every time. Below some information on the available resources, relevant hardware and versions for reproducibility:

- 4 executor instances
- 2 cores per executor
- 8G memory per executor
- Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
- DDR4 32GB 2400MHz
- SUSE Linux Enterprise Server 12 SP2
- Python version 3.6.13
- RStudio version 1.2.1335
- R version 3.6.1
- Spark version 2.4.3
- Pgmpy version 0.1.13
- Dagitty version 0.3-1

4 Results and Analysis

4.1 Calibration

The plots present in Figure 9 were created using the network structure explained in Section 3.1.

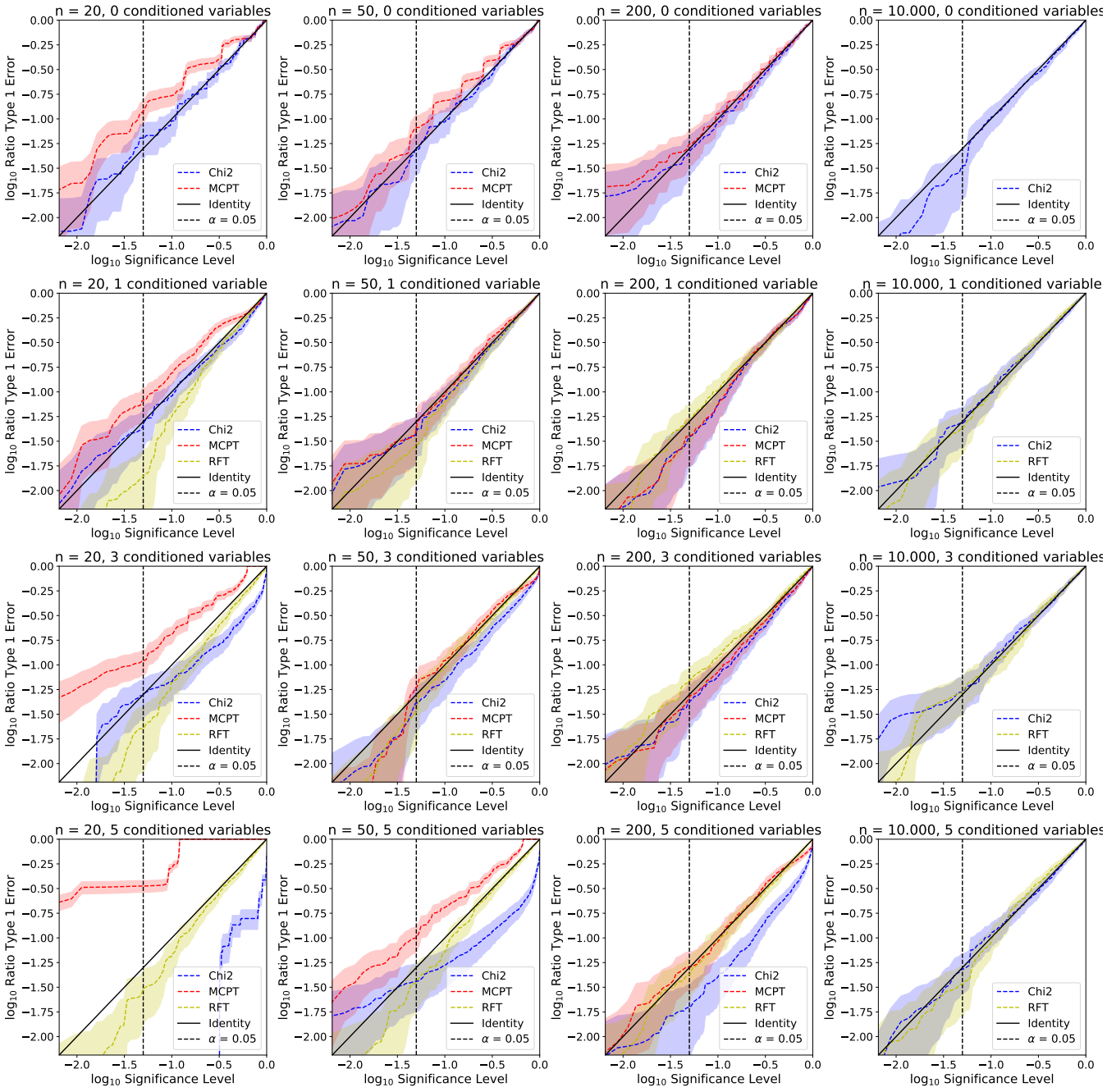


Figure 9: Calibration plots on log-log scale for the chi-squared test, MCPT and RFT, when applicable. All plots are based on 400 simulated datasets, with varying sample sizes n and conditioned set size k. n varies from left to right with $n \in \{20, 50, 200, 10\,000\}$. k varies from top to bottom with $k \in \{0, 1, 3, 5\}$. A 95% confidence interval is present in the plots to statistically substantiate our conclusions.[1]

[1]Note that the plots are on a log-log scale. This scale better represents the calibration on the interval of interest. Conditional independence tests are often performed with significance levels ≤ 0.1, therefore it is most interesting how the tests are calibrated for those values. On the x-axis, values of -2.0 and -1.0 represent significance levels of 0.01 and 0.1 respectively.

In the calibration plots, depicted in Figure 9, the ratio of Type 1 errors is plotted against various levels of significance, for certain combinations of sample sizes and conditioning set sizes. A Type 1 error occurs when the null hypothesis, i.e., independence, is wrongfully rejected. If a test is perfectly calibrated, we expect the distribution of p-values to be uniform, hence they should form or approach the diagonal. For reference, the diagonal is also plotted.

The following statements are made with 95% certainty, since 95% confidence intervals are plotted. From the calibration plots, it can be deduced that the chi-squared test becomes less calibrated as the conditioned size k increases if n stays the same, while the RFT seems unperturbed by the amount of variables that is conditioned on, hence it is more robust to noise (i.e., irrelevant variables in the conditioned set). Since the RFT is on the right hand side of the diagonal when $n = 20$ or $n = 50$, it is slightly more conservative than the other two tests when the sample size is low. For $n = 20$, the MCPT seems to reject the null hypothesis too often, given the 95% confidence interval around $\alpha = 0.05$. None of the tests are calibrated when $n = 20$ and $k = 5$. This is no surprise, since conditioning on 5 binary variables leads to 32 contingency tables of which at least $32 - 20 = 12$ are completely empty, while the remaining 20 contingency tables contain 1 entry on average. Hence, there is little to no information on dependence available. For the other cases, the chi-squared test seems to be calibrated, with the exception of the two center plots in the last row ($n \in \{50, 200\}$ and $k = 5$). When inspecting those two plots, we see that when $n = 50$ and $k = 5$, the RFT seems better calibrated compared to the chi-squared test and the MCPT. When $n = 200$ and $k = 5$, both the MCPT and the RFT are well calibrated.

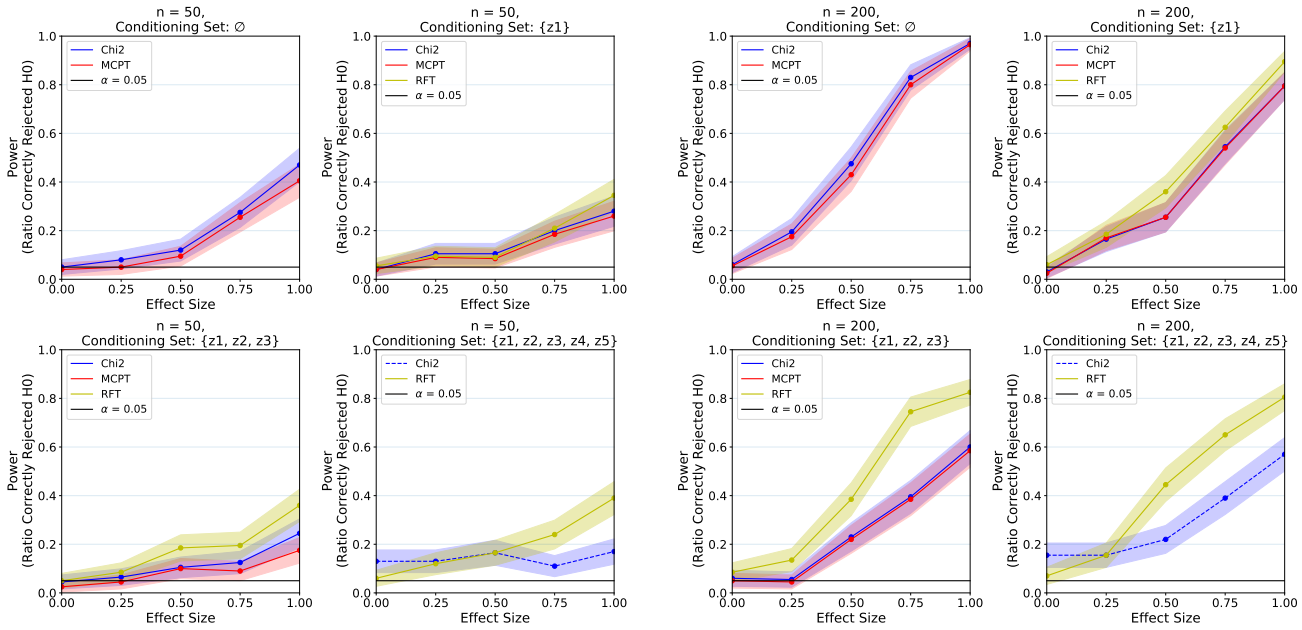
If calibration was the only important aspect, we could use any of the three CI tests when there is a high ratio between sample size and the amount of combinations of states in the conditioning set. The chi-squared test seems best calibrated for low sample cases when conditioned on little to no variables. The RFT seems better calibrated for low sample cases, where the conditioning set contains multiple variables.

Apart from finding different situations each test might be better calibrated than the other tests, the main goal was to show that all tests are well calibrated in situations where they should work. This is the case, as can be concluded from the 4 center plots in Figure 9, where $n \in \{50, 200\}$ and $k \in \{1, 3\}$.

Calibration is not the only important aspect of a test, since randomly drawing from the uniform distribution on the interval $[0, 1]$ will also lead to a diagonal line in those plots. Another important attribute we are interested in is power.

4.2 Power

Since the calibration plot depicted in figure 9 shows that each of the CI tests is well calibrated for $n = 50$ and $n = 200$ for all of the cases except when $k = 5$, we are interested in the power for those cases.



Power plots for $n = 50$ Power plots for $n = 200$

Figure 10: Various power plots for $n \in \{50, 200\}$ and $k \in \{0, 1, 3, 5\}$. The 95% confidence interval is also depicted. 200 datasets were simulated per scenario.

Note that for each power plot there also is an effect size of 0. In this case, there is no relation to detect, hence "Ratio Correctly Rejected H0" does not apply here. The reason to include this effect size in the power plots is to again emphasize the calibration of the CI tests. Since no relation is present, we hope to see that we incorrectly reject H0 with a ratio equal to the significance level. A significance level of 0.05 was used to generate these power plots, so each line indicating power should start at around 0.05, for well calibrated tests. For $k = 5$, the chi-squared graph does not start around 0.05, but signficantly higher. This does not mean that it is better to use. On the contrary, it means that the chi-squared test is not calibrated in this case, hence should not be used, or at least not be preferred when other calibrated tests are available. This is emphasized by the use of a dashed line, rather than a solid line. The reason to still plot the power of the chi-squared test when it is not well calibrated is to show that, even in that situation, a well calibrated test like the RFT still achieves a higher power.

As can be deduced from Figure 10, the CI tests have similar power for low effect sizes when $k \in \{0, 1\}$. But as the effect size increases, the power of the RFT starts to distinguish from the powers of the chi-squared test and the MCPT, especially in the cases where there are multiple noisy variables. This is due to the fact that the power of the chi-squared test and the MCPT decrease as noise is introduced, while the RFT again seems unperturbed by irrelevant variables in the conditioning set, much like it was for the calibration plots. Hence, the power of the RFT mostly relies on the effect size of the relation to detect and the sample size. The power of both the chi-squared test and the MCPT rely on effect size, sample size and the amount of combinations of states present in the conditioning set.

The observation that the power of the chi-squared test and the MCPT seem to rely on the amount of unique states in the conditioning set, while again the RFT seems unperturbed, is no surprise. The chi-squared test and the MCPT both use stratifications of the data. Since a strata of sample size n can provide more evidence against a null hypothesis than 4 strata of sample size $\frac{n}{4}$, it is only logical that the power decreases as we increase the amount of strata, while retaining the total sample size, i.e., the combined sample size of all strata. This was already illustrated with an example in Section 2.4.1.

Furthermore, the power of the CI tests grows with $\sqrt{n}$ [33], as is clearly visible in Figure 10 for an effect size of 1.

If only the effect size would be known beforehand, we could now predict what test is best to use given the calibration and power of the CI test in a similar situation.

However, say one of our CI tests takes one second to perform, while another one takes an entire day to perform in the same situation, we would rather take the first mentioned CI test if the difference in power is not too big and it is still calibrated. This brings us to the last attribute of our CI tests.

4.3 Runtime

If a CI test is perfectly calibrated and has high power, the only thing left that we are interested in is the runtime. Since a CI test which is well calibrated and has high power but has infeasible runtime is completely useless. In order to measure our runtimes, we ran the tests at night. The cluster of De Volksbank, on which we ran all of our Spark sessions, had more jobs to schedule during the day than at nighttime. Since important processes of De Volksbank have a higher priority on the server than our tests, we wanted to make sure that enough resources were available. The runtime plots were created using the resources stated in Section 3.4.

An important note on our runtimes is that the chi-squared test and the MCPT are implemented in such a way that the frequencies present in the contingency table are gathered before the testing begins. This is depicted in our runtime plots as "Obtaining Counts". The RFT does not require these counts, so if the graph of the RFT lies above the one of the chi-squared test or the MCPT, it does not directly imply that only using the chi-squared test and MCPT results in faster runtimes, compared to only using the RFT.

We varied our sample size $n \in \{100, 1000, 10\,000, 100\,000, 1\,000\,000, 10\,000\,000, 100\,000\,000\}$ and our conditioning set size $k \in \{0, 1, 3, 5\}$. The structure used to generate our data is the same one we used for our calibration plots, see Figure 5. The variables we conditioned on are also the same ones as for our calibration plots, see Table 7.

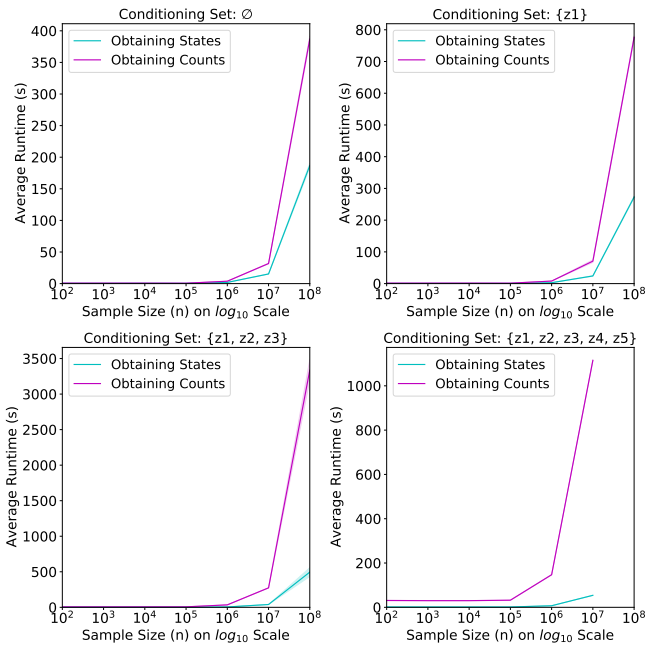


Figure 11: Runtimes to obtain necessary statistics to perform our CI tests. The sample size is on a log scale. Obtaining States refers to detecting the different states all of our variables obtain. Obtaining Counts refers to the instantiation of a dictionary, where the frequency for each combination of states relevant to the CI tests we are expected to perform, is stored. Each graph is based on 20 runs. The 95% confidence interval is also plotted.

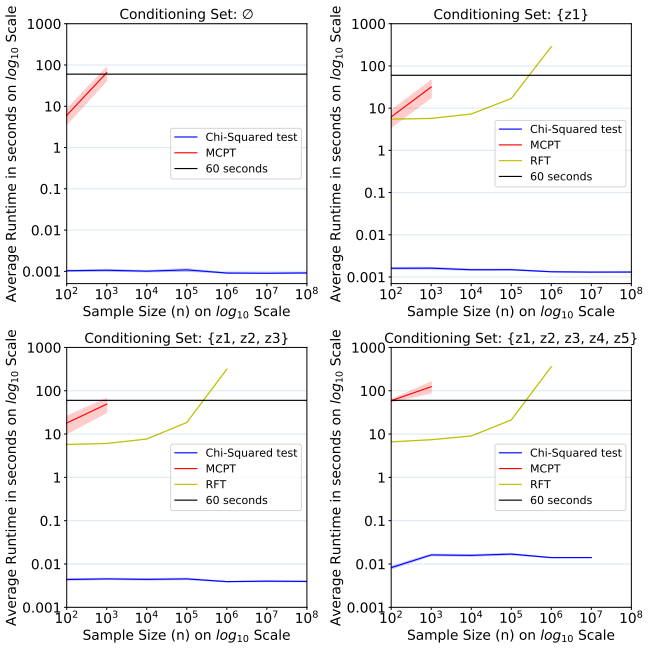


Figure 12: Runtimes of the chi-squared test, MCPT and RFT depicted on a log-log scale. Only tests with feasible runtime are performed. Each graph is based on 20 runs. The 95% confidence interval is also plotted.

When we obtained the runtimes, we only provided dataframes with the relevant columns. So, when we did not condition on any variables, our dataframe contains 2 columns. If we condition on 5 variables, our dataframe contains 7 columns. Due to time constraints, we were unable to obtain a sufficient amount of results for the runtimes when $n = 100\ 000\ 000$ and $k = 5$. The first results indicated that obtaining counts takes 4 times as long for $k = 5$ as it did for $k = 3$. This follows the trend that for every binary variable added to the conditioning set, the runtime doubles.

Note again that the RFT is not present in the runtime plot, Figure 12, when we condition on the empty set. This is again due to the fact that the RFT is not applicable when there are no variables in the conditioning set. Furthermore, we only computed runtimes for the MCPT when $k \in \{100, 1000\}$, due to the fact that we observed long runtimes for the small sample cases combined with the observation that the MCPT does not outperform the chi-squared test and the RFT in terms of calibration and power. It was therefore decided not to measure the runtimes of the MCPT for bigger sample sizes.

In Figure 11, the runtimes to obtain the various states for each variable are depicted. Obtaining these various states is required for all of our CI tests and is only required to be done once, since the set of states each variable can obtain is immutable. Note that this step only takes a matter of seconds, even for a sample size of 1 000 000. The other graph, representing the runtimes to obtain the relevant counts, is the more interesting one. At first sight, the plots look very similar. However, note that the values on the y-axis differ. Figure 11 depicts that each time a variable is added to the dataset, the runtime increases by a factor of 2. This is in line with our expectations, since these synthethic datasets consist of binary variables.

From Figure 12, it can be deduced that the chi-squared test is by far the fastest CI test, if all of the frequencies are already stored locally, since a single test takes only several milliseconds. The MCPT is the only test for which the 95% confidence interval is clearly visible. This is due to the heuristic presented in Section 2.4.2. A single test either takes a lot of time, or is performed in a matter of milliseconds, since it is then based on a single chi-squared test. Even with this heuristic applied, it still resulted in the longest runtimes. Unlike the two previously mentioned tests, the RFT does not require the counts of each combination of states, so the total runtime of the RFT can directly be deduced from Figure 12.

4.4 Structure Learning

As was shown in Section 4.3, for an increasing sample size the MCPT and the RFT eventually end up with infeasible runtimes. Therefore, we had to limit the sample size of our datasets in structure learning. Various statistics are gathered to measure the performance for the use of a single CI test in the PC-stable algorithm.

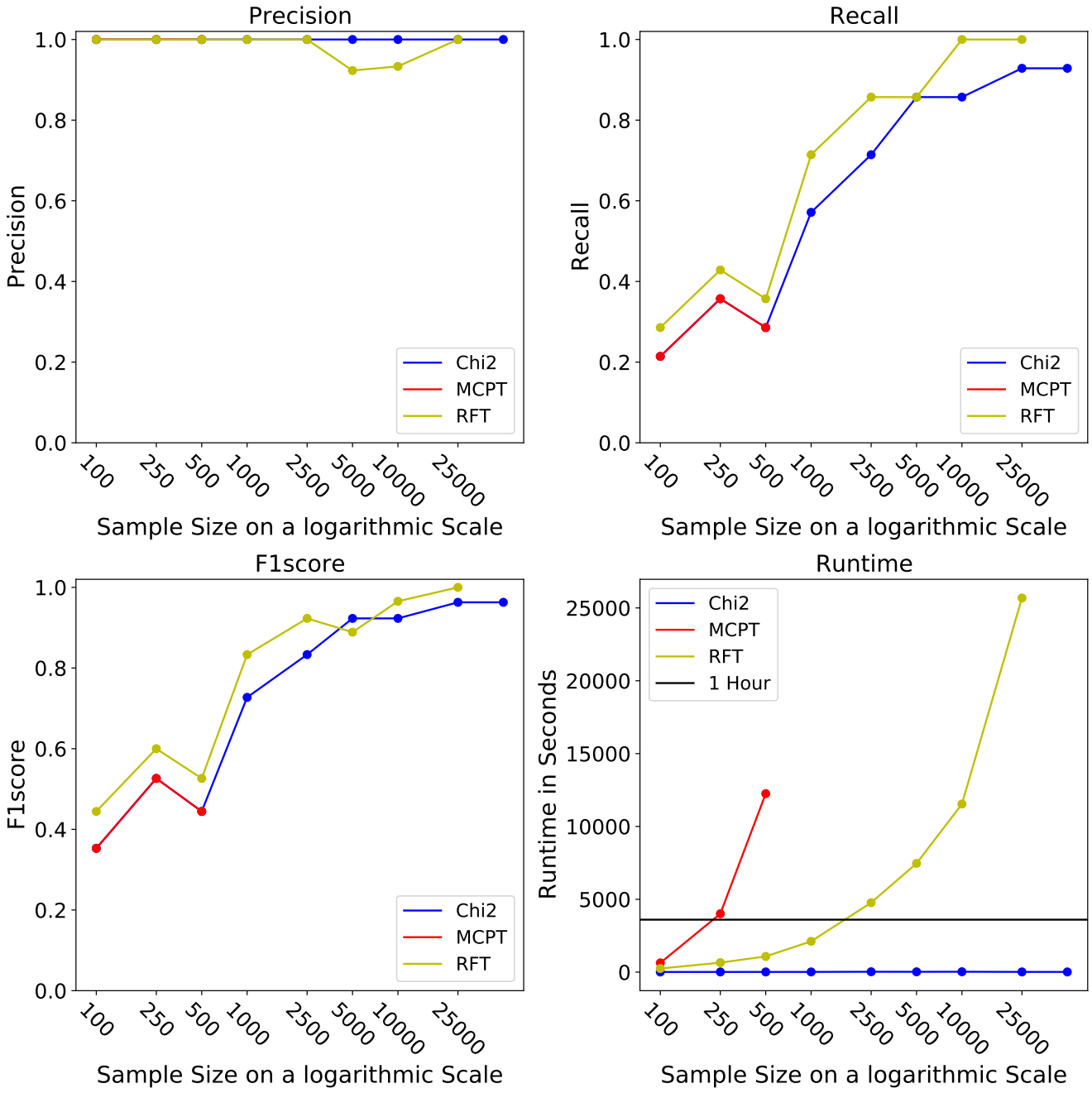


Figure 13: Several metrics are used to measure the performance for structure learning using a single CI test. The metrics precision, recall and F1-score are defined in 2.3.2. Runtime is the total runtime of the PC-stable algorithm, using only that CI test.

Due to time constraints, each dot in the graph is based on a single run of the PC-stable algorithm on the synthetic dataset A, of which the true network structure is depicted in Figure 7. Therefore, any observations based on these plots are merely indications.

For the chi-squared test, we used sample sizes $\{100, 250, 500, 1000, 2500, 5000, 10\ 000, 25\ 000, 50\ 000\}$. For MCPT sample sizes $\{100, 250, 500\}$ were used. For RFT we used sample sizes $\{100, 250, 500, 1000, 2500, 5000, 10\ 000, 25\ 000\}$.

Note that the chi-squared test and the MCPT have identical performance in terms of precision, recall and F1-score, while the runtime of the MCPT is, as we already concluded before, much higher than the runtime of chi-squared. These observations are substantiated by our findings in sections 4.1, 4.2 and 4.3. Therefore, from now on we will focus on the analysis of the comparison of the chi-squared test and the Random Forest test.

Each of the tests has perfect precision in each case, except for two cases for the RFT. For a sample size of 5000 and 10 000, the Random Forest test failed to remove a single edge, which should have been removed. This resulted in a single false positive for both cases.

It can be deduced that for 8 different sample sizes, the plot indicates that the RFT does not have a worse recall than the chi-squared test. It is even better in 7 out of 8 cases. However, we want to stress again that these remarks are merely indications. Nonetheless, these observations are in line with our expectations, since the RFT was shown to be a more powerful test than the chi-squared test in Figure 10. The higher the power of a CI test, the more often it detects a relation, which in turn results in less removed edges.

Due to the distinctive performance of the RFT in terms of recall, the F1-score also favours the RFT over the chi-squared test for 7 out of 8 sample sizes.

Unfortunately, even though using only the RFT obtains an F1-score of 1 for sample sizes 10 000 and 25 000, the runtimes were 3 hours 12 minutes and 7 hours 8 minutes respectively, with the specifications noted in 3.4, with the only difference that there were 2 executor instances instead of 4. For those same sample sizes, using only the chi-squared test resulted in runtimes of 25.6 and 14.0 seconds respectively.[2]

Besides stating metrics which are only relevant to the final result, it is also important to know what happens during the execution of the algorithm. Therefore, we show for each CI test what types of edges were removed at a certain depth d of the PC-stable algorithm. The types of edges can be true negatives (TN) and false negatives (FN). Again, if an edge is incorrectly removed, it is a false negative, and if an edge is correctly removed, it is a true negative. Figure 14 depicts an overview of the removed edges.

[2] 593 and 597 (conditional) independence tests were performed, respectively. Therefore, the descending runtime is most likely due to the occupation of the cluster.

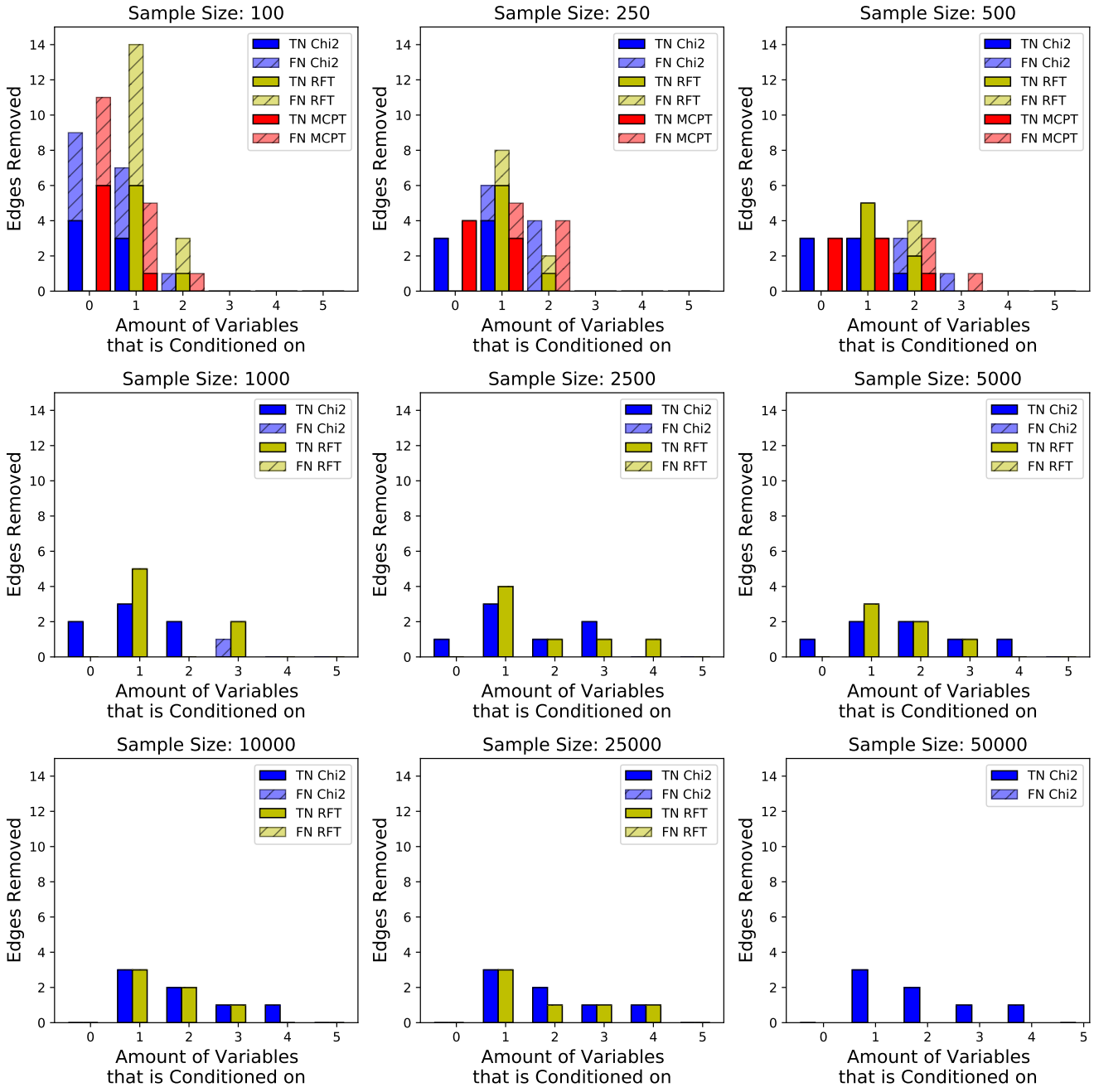


Figure 14: The amount and types of edges removed by each CI test, at each depth d of the PC-stable algorithm.

When inspecting the first 4 bar plots, i.e., the bar plots for sample sizes $\{100, 250, 500, 1000\}$, it can be deduced that for the chi-squared test the first false negatives seem to occur following a trend, see Table 8.

Sample Size	100	250	500	1000
Max. Strata Sizes No Error	-	$\frac{250}{1} = 250$	$\frac{500}{2^1} = 250$	$\frac{1000}{2^2} = 250$
Strata Sizes First Error	$\frac{100}{1} = 100$	$\frac{250}{2^1} = 125$	$\frac{500}{2^2} = 125$	$\frac{1000}{2^3} = 125$

Table 8: Trend in first occurrences of false negatives for the chi-squared test.

Unfortunately, this trend does not only depend on the size of each stratum, but also on the effect size. Since in real world settings the effect size is not known beforehand, this is not enough to draw up criteria to be certain which CI test is best in a specific case for real world data. However, in Figure 10, it is illustrated that RFT does have more power than the chi-squared test. Therefore, if the RFT is calibrated and has feasible runtime, switching from the chi-squared test to the RFT can result in less false negatives in the final DAG. From Figure 9 it can be deduced that the RFT is calibrated, for binary variables, if the sample size is at least 50. If a runtime is feasible highly depends on your personal definition of feasible. Given the scope of this thesis, we interpret feasible as a maximum of 30 seconds per CI test. Therefore, the RFT will only be performed for a maximum sample size of 100 000.

5 Hybrid Approach

Given the observed trend in Table 8, combined with the observation that the RFT outperforms the chi-squared test in terms of power and is well calibrated if the sample size is greater than or equal to 50, we propose the following hybrid approach.

If the expected sample size of each stratum is less than 1000, we apply the Random Forest test, otherwise the chi-squared test is applied. The choice of the threshold 1000 is based on the observed trend that, for our specific dataset in which we knew the effect size, the chi-squared test performed well if the expected size of the biggest stratum was at least 250. As illustrated in figure 10, the power decreases as the effect size decreases. We take a threshold of 1000, to still get a reliable result for slighty lower effect sizes. However, it must be noted that this threshold for the hybrid approach is not a criterium that should hold for every situation. We mostly want to show that a hybrid approach can be a good alternative to the use of a single CI test.

5.1 Performance on Dataset A

We first present structure learning results on the dataset A. See Figure 7 for the correct network structure. We vary our sample sizes $n \in \{2500, 5000\}$ and the effect size $\beta \in \{0.1, 0.4\}$ used to generate the data using `Dagitty`'s `simulateLogistic` [32][11]. Each beeswarm boxplot presented is based on 10 runs.

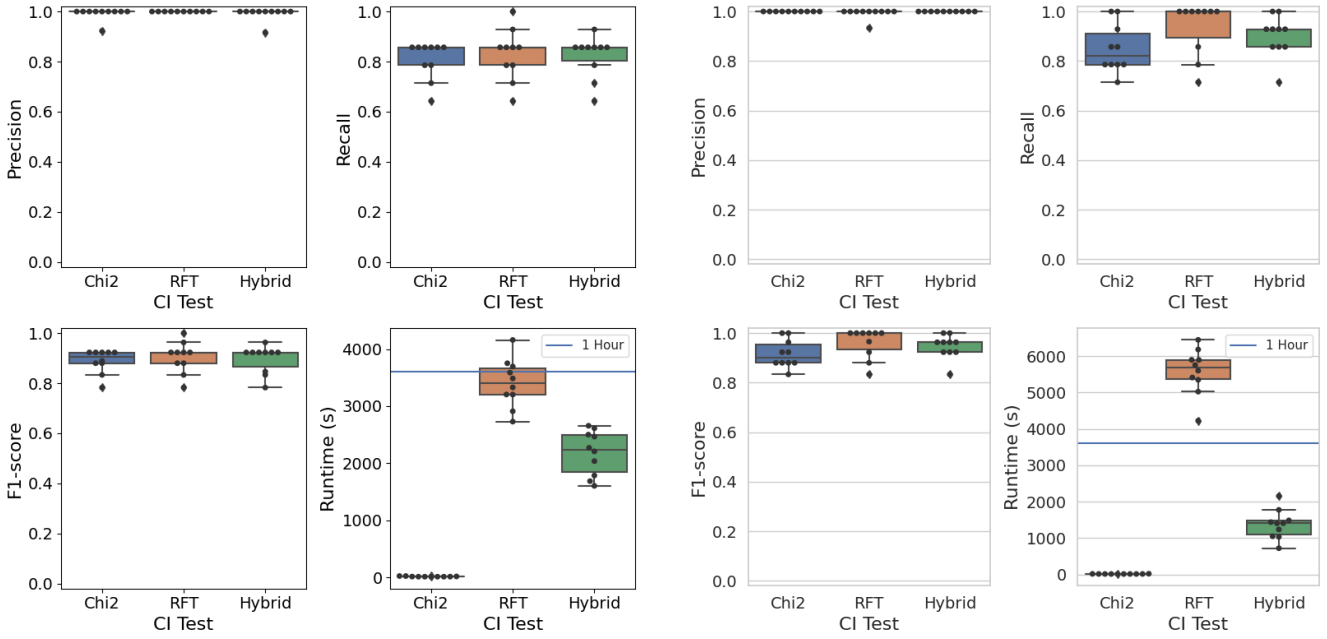


Figure 15: Precision, recall, F1-score and runtimes of structure learning using PC-stable are depicted. Either only the chi-squared test was used, or only the Random Forest test, or the hybrid approach. The dataset was generated with `Dagitty`'s `simulateLogistic` [32][11] with an effect size $\beta = 0.4$. Sample sizes were varied with $n \in \{2500, 5000\}$.

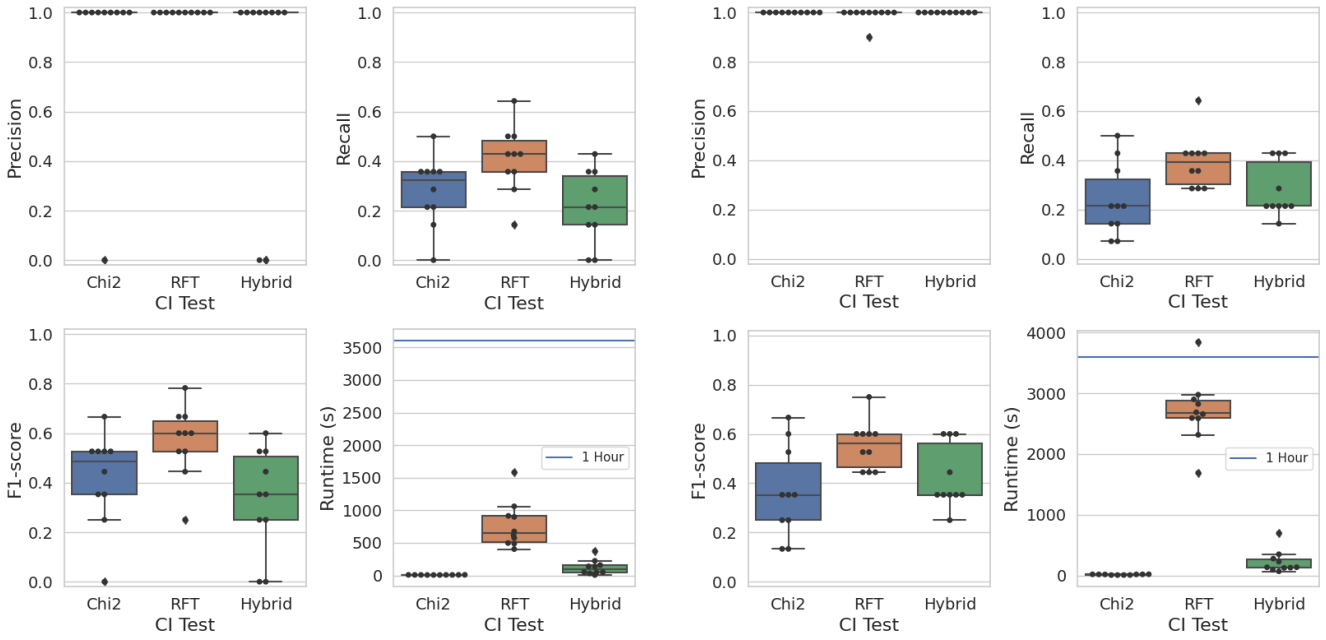


Figure 16: Precision, recall, F1-score and runtimes of structure learning using PC-stable are depicted. Either only the chi-squared test was used, or only the Random Forest test, or the hybrid approach. The dataset was generated with `Dagitty`'s `simulateLogistic` [32][11] with an effect size $\beta = 0.1$. Sample sizes were varied with $n \in \{2500, 5000\}$.

Sample Size	Effect Size	CI Test	Precision	Recall	F1-score	Runtime (s)
$n = 2500$	$\beta = 0.4$	Chi2	0.9923	0.8071	0.8880	13.95
		RFT	1.0	0.8286	0.9031	3404.49
		Hybrid	0.9917	0.8214	0.8964	2181.61
$n = 5000$	$\beta = 0.4$	Chi2	1.0	0.85	0.9162	12.68
		RFT	0.9933	0.9357	0.9602	5578.71
		Hybrid	1.0	0.9	0.9454	1369.34
$n = 2500$	$\beta = 0.1$	Chi2	0.9	0.2786	0.4172	8.85
		RFT	1.0	0.4071	0.5663	771.46
		Hybrid	0.8	0.2143	0.3303	122.52
$n = 5000$	$\beta = 0.1$	Chi2	1.0	0.2357	0.3618	16.32
		RFT	0.99	0.3929	0.5536	2707.37
		Hybrid	1.0	0.2786	0.4259	225.86

Table 9: Means for each metric for each CI test on samples of dataset A.

Sample Size	Effect Size	Compared Tests	Precision	Recall	F1-score	Runtime
$n = 2500$	$\beta = 0.4$	Chi2 vs. RFT	0.3434	0.6010	0.5509	$\mathbf{1.1 \cdot 10^{-9}}$
		Chi2 vs. Hybrid	0.9555	0.6946	0.7241	$\mathbf{2.8 \cdot 10^{-8}}$
		RFT vs. Hybrid	0.3434	0.8665	0.8013	$\mathbf{2.7 \cdot 10^{-6}}$
$n = 5000$	$\beta = 0.4$	Chi2 vs. RFT	0.3434	0.0807	0.1111	$\mathbf{4.7 \cdot 10^{-10}}$
		Chi2 vs. Hybrid	1.0	0.2361	0.2303	$\mathbf{2.1 \cdot 10^{-6}}$
		RFT vs. Hybrid	0.3434	0.4225	0.5574	$\mathbf{1.2 \cdot 10^{-11}}$
$n = 2500$	$\beta = 0.1$	Chi2 vs. RFT	0.3434	0.0515	0.0641	$\mathbf{8.1 \cdot 10^{-5}}$
		Chi2 vs. Hybrid	0.5566	0.3306	0.3427	**0.0104**
		RFT vs. Hybrid	0.1679	**0.0068**	**0.0098**	**0.0002**
$n = 5000$	$\beta = 0.1$	Chi2 vs. RFT	0.3434	**0.0146**	**0.0117**	$\mathbf{7.4 \cdot 10^{-8}}$
		Chi2 vs. Hybrid	1.0	0.4690	0.3810	**0.0063**
		RFT vs. Hybrid	0.3434	**0.0298**	**0.0228**	$\mathbf{2.5 \cdot 10^{-8}}$

Table 10: p-values of the two-tailed Welch's t-test [20] whether the observed difference is significant. Bold values are significant p-values, with a significance level $\alpha = 0.05$. The p-values are not corrected for multiple testing.

From Figures 15, 16 and Table 10, it can be deduced that using the chi-squared test is significantly faster than both the RFT and the hybrid approach. The hybrid approach is significantly faster than using only the RFT. When the effect size is relatively high, e.g., $\beta = 0.4$, we do not see any significant difference in performance in terms of precision, recall and F1-score. For a lower effect size, $\beta = 0.1$, using only the RFT is shown to perform significantly better than the hybrid approach for both sample sizes in terms of recall and F1-score. Furthermore, when $n = 2500$ and $\beta = 0.1$, the RFT performs significantly better than the hybrid approach in terms of recall and F1-score, while the difference with the chi-squared test is not significant.

If these were the only results, we would conclude that it is best to perform either only the chi-squared test or only the Random Forest test. However, the structure we tried to learn has relatively few variables. This favours the chi-squared test, since conditioning on extra variables leads to another stratification of the contingency tables, which in turn leads to a lower power for the chi-squared test. The lower power will in turn lead to extra removal of edges. Therefore, we are also interested in the performance of the chi-squared test, the hybrid approach and the Random Forest test of learning a denser structure on more variables.

5.2 Performance on Dataset B

The Polzer dataset [16] originally has 14 variables. Due to the exponential increase in runtime of the PC algorithm for each additional variable in a dataset, we removed 5 variables. The 5 variables removed were the last variables when sorted lexicographically. The variables were removed after the dataset was generated, which means that causal sufficiency does not hold, since we know that there are exactly 5 hidden variables. The resulting dataset is named dataset B. In order to get our results, we upscaled our resources mentioned in Section 3.4. We now have 4 executor instances, with 8 cores each. The other specifications remain the same.

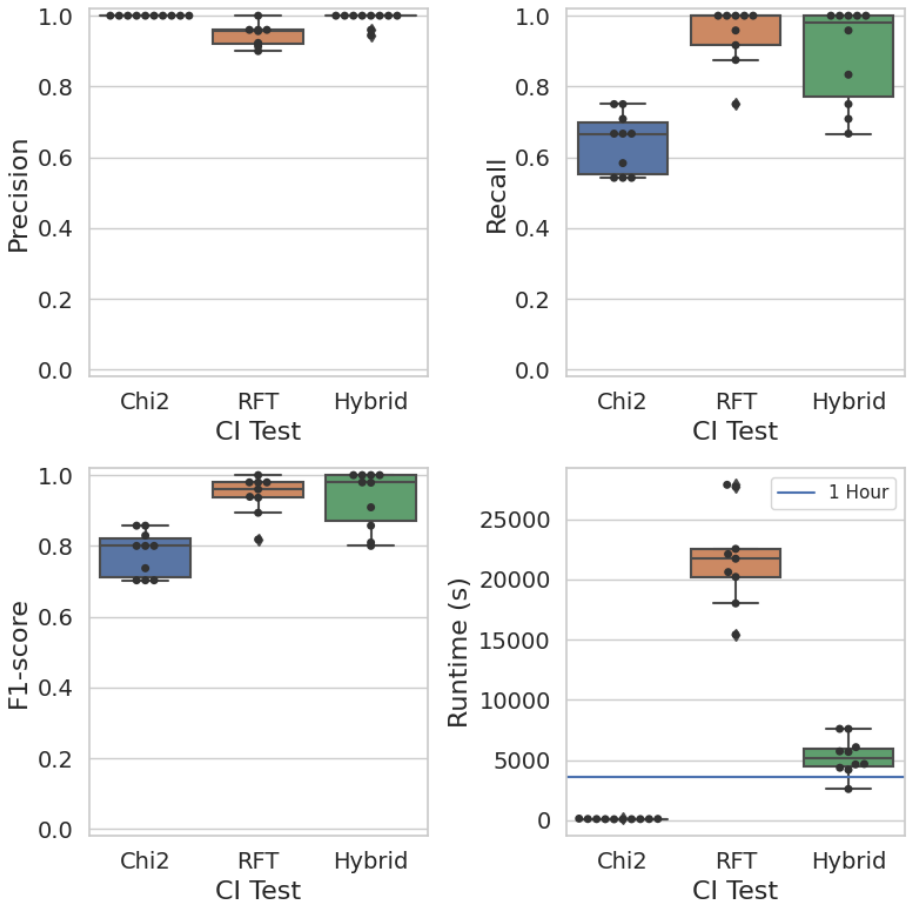


Figure 17: Precision, recall, F1-score and runtimes of structure learning using PC-stable are depicted. Either only the chi-squared test was used, or only the Random Forest test, or the hybrid approach.[3]

CI Test	Precision	Recall	F1-score	Runtime (s)
Chi2	1.0	0.6417	0.7789	111.47
RFT	0.9436	0.9444	0.9428	21785.59
Hybrid	0.9904	0.8917	0.9334	5316.12

Table 11: Means for each metric for each CI test on samples of dataset B.

Compared Tests	Precision	Recall	F1-score	Runtime
Chi2 vs. RFT	**0.0007**	$\mathbf{6.2 \cdot 10^{-7}}$	$\mathbf{1.5 \cdot 10^{-5}}$	$\mathbf{2.2 \cdot 10^{-7}}$
Chi2 vs. Hybrid	0.1741	**0.0002**	**0.0002**	$\mathbf{2.2 \cdot 10^{-6}}$
RFT vs. Hybrid	**0.0021**	0.3272	0.7737	$\mathbf{4.1 \cdot 10^{-7}}$

Table 12: p-values of the two-tailed Welch's t-test [20] whether the observed difference is significant. Bold values are significant p-values. The p-values are not corrected for multiple testing.

From Figure 17 and Table 12, we can deduce that the RFT performs significantly worse compared to the chi-squared test and hybrid approach in terms of precision. When inspecting the recall, the chi-squared test performs significantly

[3]Only 9 datasets were generated for the Random Forest test.

worse than the hybrid approach and the RFT. This poor performance in recall also leads to the chi-squared test being significantly worse than the hybrid approach and the RFT in terms of the F1-score. As always, structure learning with the chi-squared test is significantly faster than the hybrid approach or the RFT and the hybrid approach is significantly faster than the RFT.

Here, the hybrid approach has better precision than the RFT, better recall than the chi-squared test and a better F1-score than the chi-squared test. Even though the runtime of the hybrid approach is significantly worse than the chi-squared test, it still has feasible runtime, even when we would not have such a massive cluster available.

6 Conclusion

We investigated the performance of Pearson's chi-squared test, the Monte Carlo Permutation test and the Random Forest test for an increasing sample size. Therefore, each of the three CI tests is implemented to be able to handle big data. The chi-squared test already had an implementation for big data, but the Monte Carlo based approach and the RFT did not, to the best of our knowledge. Nonetheless, we used our own custom implementation of the chi-squared test. The reason for the lacking implementation of the Monte Carlo based approach is most likely that the available hardware is not even close to handle a MCPT on a large amount of data. This is due to the computationally expensive complexity of obtaining permutations of contingency tables for big data that follow the hypergeometric distribution. So, even though the MCPT works for big data in theory, its runtime is not feasible and will therefore not be used for big data since other CI tests, like the chi-squared test, perform equally well, or better, in terms of calibration or power in such cases. The reason that the Random Forest test did not yet have an implementation for big data, was that 1) it was only introduced this year, and 2) the authors were mostly interested in small sample size cases with multiple variables in the conditioning set. This is due to the property that the RFT is effective at discarding irrelevant information from high-dimensional conditioning sets [31]. For big data, this becomes interesting when conditioning on a large amount of variables, which at this time is not yet feasible due to the resulting runtimes.

The Monte Carlo Permutation test with the extra heuristic does not show promising results in terms of calibration, power or runtime. Furthermore, the Monte Carlo Permutation test quickly becomes infeasible as the sample size increases. The poor performance of the Monte Carlo Permutation test might be due to the heuristic we used. The Monte Carlo Permutation test was considered as an alternative to the chi-squared test, when we expect the chi-square test to be unreliable. However, we now accept independence for the Monte Carlo Permutation test if the p-value of the chi-squared test is > 0.5. This possibly negates the benefits the Monte Carlo Permutation test might bring along, compared to the chi-squared test. Unfortunately, there was not enough time to get new results when removing this heuristic.

Pearson's chi-squared test is the fastest CI test out of the three considered CI tests. The more variables we condition on, the less powerful the chi-squared test becomes, as is depicted in Figure 10. This leads to sparser DAGs compared to the use of a conditional independence test which does not lose power when conditioning on more variables. Given the high scores on precision after structure learning, see Figures 15, 16 and 17, we can conclude that it is best to use the chi-squared test if one wants to learn a network structure which does not contain false positives. Minimalizing the amount of false positives is especially important when each edge in the resulting DAG will extensively be researched. Research costs both time and money, so it is best to avoid investigating a relation which does not exist.

The Random Forest test is a more powerful test than both the chi-squared test and the Monte Carlo Permutation test. It does not lose power as we increase the amount of variables in our conditioning set, in contrast to the other two CI tests, see Figure 10. Since the Random Forest test has more power, it rejects the null hypothesis of independence more often, which in turn leads to less false negatives in the resulting DAG. This is substantiated by the high observed recalls in our experiments, see Figures 15, 16 and 17. The Random Forest test should be used if minimalizing the amount of false negatives in the final DAG is most important. Minimalizing the amount of false negatives is always desirable for a structure learning algorithm, since the resulting DAG is an independence map. That is, if in the DAG an edge is not present between two variables, those variables are (conditionally) independent. False negatives are especially detrimental in fields like the medical field. Say, we learn a DAG of a dataset that includes whether someone smokes and if that person has cancer. If there is a false negative between smoker and cancer, i.e., there is no edge directly connecting those variables, one could conclude that smoking is independent of cancer given some other variables. One could incorrectly attribute the correlation of smoking and cancer to confounding variables like age or weekly alcohol consumption. A mistake like this can pose a health risk.

The proposed hybrid approach provides the option to achieve higher precision than the Random Forest test alone and higher recall than the chi-squared test alone, see Figure 17 and Table 12. It provides the fast and still powerful use of the chi-squared test in the early stage of the PC-algorithm, while switching to the slower, yet more reliable, Random Forest test later on.

For big data, the chi-squared test is the only conditional independence test which results in feasible runtimes out of the three conditional independence tests considered.

7 Future Research

Unlike the chi-squared test and the Random Forest test, the Monte Carlo Permutation test can be implemented in several ways. Our conclusions might not hold for other implementations of the Monte Carlo Permutation test. The `bnlearn` package [28], which is an R package for learning Bayesian networks, already contains 3 Monte Carlo based implementations for several tests [7]; The basic Monte Carlo permutation test, the sequential Monte Carlo permutation test and the semiparametric test. Further research should be performed to check the calibration, power and runtimes for those implementations of the Monte Carlo based approach. Alternative implementations might outperform the approach we used throughout this thesis. This could lead to a hybrid approach which would include some sort of permutation based approach.

We tried our best to implement the Random Forest test as efficient as possible for big data. However, I am convinced that there is still room left for improvement. This can result in faster runtimes. Furthermore, we only used multiple CPUs to generate our results. Using GPUs might also improve the runtimes of the Random Forest test significantly, due to the large matrices which are multiplied.

The hybrid approach has a hard threshold of 1000 at this time. This approach now still has an unpredictable runtime. We propose the following variation to our hybrid approach. For this approach an acceptable runtime is provided. Start with a threshold of 250. So when the largest stratum has at least 250 samples, perform the chi-squared test, otherwise perform the RFT. At the end, return the result. If there is time left, increase this threshold by a factor, of say, 2. Again, perform structure learning. This will increase the amount of performed Random Forest tests, which in turn leads to an increase in runtime, while getting a more reliable result. As long as there is runtime left, keep increasing the threshold and perform structure learning.

The chi-squared test loses power as we increase the size of the conditioning set, while the Random Forest test seems unperturbed by this, see Figure 10. The Random Forest test with its current implementation eventually results in infeasible runtimes if the sample size becomes too large, see Figure 12. Another solution to tackle this problem with a hybrid approach can be to provide a subsample of the data to the Random Forest test, in order to still get feasible runtimes, while outperforming the chi-squared test in terms of power. Further research can provide insight if there exists a direct link between the power of the chi-squared test and the power of the Random forest test. Where the power of the chi-squared test is dependent on the sample size of the data and the amount of strata in the conditional independence test and the power of the Random Forest test is dependent on the size or the ratio of the subsample.

It is important to remember that we only used our implementations which can handle big data. If it is known beforehand that the dataset is relatively small, it might be better not to use PySpark which distributes the data. For relatively small datasets, it is better to use datastructures which are based on in-memory analytics like Pandas DataFrames [23][24]. This most likely improves the runtimes significantly.

Due to the scope of this thesis and the time it took to obtain all the data for the calibration plots, power plots, runtime plots and structure learning plots, it was not possible to test every situation we wanted. Therefore, we focused on binary variables, both for the conditioning set as well as the variables we tested independence of. The conclusion that an increasing sample size will lead to infeasible runtimes for the Random Forest test and the Monte Carlo Permutation test will still apply for k-level categorical variables. For the conditioning set, we are confident that this translates to multi-level categorical variables, since stratifying to 4 strata can be done by 2 binary variables, or by the use of a single 4-level variable, i.e., a categorical variable which can obtain 4 different states. But for the variables we test for independence, multiple levels were not taken into account, so it is not possible to directly transfer our findings to the multi-level case. Additional research should be performed to see whether this is the case.

In this thesis we focused on three CI tests: Pearson's chi-squared test, Monte Carlo Permutation test with an heuristic and the Random Forest test. The proposed hybrid approach shows promosing results compared to the use of a single CI test. Research on the performance of other CI tests can lead to a new hybrid approach consisting of multiple CI tests, which in turn can outperform the hybrid approach proposed here.

References

[1] A. Agresti. *Categorical Data Analysis*. Wiley Series in Probability and Statistics. Wiley, 2012. ISBN: 9780470463635. URL: https://books.google.nl/books?id=UOrr47-2oisC.

[2] Alan Agresti. "A Survey of Exact Inference for Contingency Tables". In: *Statistical Science* 7.1 (1992), pp. 131–153. DOI: 10.1214/ss/1177011454. URL: https://doi.org/10.1214/ss/1177011454.

[3] Ankur Ankan. *Implementation of PC-stable*. URL: https://pgmpy.org/_modules/pgmpy/estimators/PC.html (visited on 2021-06-29).

[4] Ankur Ankan and Abinash Panda. "pgmpy: Probabilistic graphical models using python". In: *Proceedings of the 14th Python in Science Conference (SCIPY 2015)*. Citeseer. 2015.

[5] *Apache Spark*. URL: https://spark.apache.org/ (visited on 2021-06-29).

[6] *Asia dataset 10k samples*. URL: http://www.openmarkov.org/learning/ (visited on 2021-08-25).

[7] *bnlearn - Conditional independence tests*. URL: https://www.bnlearn.com/documentation/man/conditional.independence.tests.html (visited on 2021-07-14).

[8] J. Cohen et al. "Dummy encoding". In: *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences (3rd ed.)* Routledge, 2002. URL: https://doi.org/10.4324/9780203774441.

[9] Diego Colombo and Marloes H. Maathuis. "Order-Independent Constraint-Based Causal Structure Learning". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 3741–3782. ISSN: 1532-4435.

[10] Diego Colombo et al. "Learning high-dimensional directed acyclic graphs with latent and selection variables". In: *The Annals of Statistics* 40.1 (Feb. 2012), pp. 294–321. ISSN: 0090-5364. DOI: 10.1214/11-aos940. URL: http://dx.doi.org/10.1214/11-AOS940.

[11] *Dagitty - simulateLogistic*. URL: https://rdrr.io/cran/dagitty/man/simulateLogistic.html (visited on 2021-07-28).

[12] A. P. Dawid. "Conditional Independence in Statistical Theory". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 41.1 (1979), pp. 1–31. ISSN: 00359246. URL: http://www.jstor.org/stable/2984718.

[13] Karl Pearson F.R.S. "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50.302 (1900), pp. 157–175. DOI: 10.1080/14786440009463897. eprint: https://doi.org/10.1080/14786440009463897. URL: https://doi.org/10.1080/14786440009463897.

[14] Zhifeng Hao et al. "Causal Discovery on High Dimensional Data". In: *Applied Intelligence* 42.3 (Apr. 2015), pp. 594–607. ISSN: 0924-669X. DOI: 10.1007/s10489-014-0607-0. URL: https://doi.org/10.1007/s10489-014-0607-0.

[15] *History of big data*. URL: https://www.sas.com/nl_nl/insights/big-data/what-is-big-data.html (visited on 2021-05-03).

[16] Polzer I et al. "The association of tooth loss with all-cause and circulatory mortality. Is there a benefit of replaced teeth? A systematic review and meta-analysis." In: *Clinical Oral Investigations* (Apr. 2012), pp. 333–351. DOI: 10.1007/s00784-011-0625-9.

[17] S. Lauritzen and D. Spiegelhalter. "Local computations with probabilities on graphical structures and their application to expert systems". In: *Journal of the royal statistical society series b-methodological* 50 (1988), pp. 415–448.

[18] Thuc Duy Le et al. "A Fast PC Algorithm for High Dimensional Causal Discovery with Multi-Core PCs". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 16.5 (2019). ISSN: 2374-0043. DOI: 10.1109/tcbb.2016.2591526. URL: http://dx.doi.org/10.1109/TCBB.2016.2591526.

[19] Chun Li and Bryan E. Shepherd. "A new residual for ordinal outcomes". In: *Biometrika* 99.2 (Mar. 2012), pp. 473–480. ISSN: 0006-3444. DOI: 10.1093/biomet/asr073. eprint: https://academic.oup.com/biomet/article-pdf/99/2/473/17461362/asr073.pdf. URL: https://doi.org/10.1093/biomet/asr073.

[20] Zhenqiu Lu and Ke-Hai Yuan. "Welch's t test". In: Jan. 2010, pp. 1620–1623. DOI: 10.13140/RG.2.1.3057.9607.

[21] Nicholas Metropolis and S. Ulam. "The Monte Carlo Method". In: *Journal of the American Statistical Association* 44.247 (1949). PMID: 18139350, pp. 335–341. DOI: 10.1080/01621459.1949.10483310. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1949.10483310. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1949.10483310.

[22] Richard Neapolitan. *Learning Bayesian Networks*. Jan. 2003. ISBN: 9780123704771. DOI: 10.1145/1327942.1327961.

[23] *Pandas DataFrame*. URL: https://pandas.pydata.org/pandas-docs/stable/reference/frame.html (visited on 2021-06-29).

[24] *Pandas: Scaling to large datasets*. URL: https://pandas.pydata.org/pandas-docs/stable/user_guide/scale.html (visited on 2021-04-26).

[25] *PySpark: Apache Spark with Python*. URL: https://intellipaat.com/blog/tutorial/spark-tutorial/pyspark-tutorial/ (visited on 2021-02-23).

[26] K. Vineet Raghu et al. "Comparison of strategies for scalable causal discovery of latent variable models from mixed data". In: *I. J. Data Science and Analytics* (2018), pp. 33–45.

[27] Sabine Schipf et al. "Low total testosterone is associated with increased risk of incident type 2 diabetes mellitus in men: Results from the Study of Health in Pomerania (SHIP)". In: *The aging male : the official journal of the International Society for the Study of the Aging Male* 14 (Nov. 2010), pp. 168–75. DOI: 10.3109/13685538.2010.524955.

[28] Marco Scutari. "Learning Bayesian Networks with the bnlearn R Package". In: *Journal of Statistical Software* 35.3 (2010), pp. 1–22. DOI: 10.18637/jss.v035.i03.

[29] Peter Spirtes and Clark Glymour. "An Algorithm for Fast Recovery of Sparse Causal Graphs". In: *Social Science Computer Review* 9.1 (1991), pp. 62–72. DOI: 10.1177/089443939100900106. eprint: https://doi.org/10.1177/089443939100900106. URL: https://doi.org/10.1177/089443939100900106.

[30] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Vol. 81. Jan. 1993. ISBN: 978-1-4612-7650-0. DOI: 10.1007/978-1-4612-2748-9.

[31] Johannes Textor and Ankur Ankan. "A Simple Unified Approach to Testing High-Dimensional Conditional Independencies for Categorical and Ordinal Data". In: (2021). Manuscript in preparation.

[32] Johannes Textor et al. "Robust causal inference using directed acyclic graphs: the R package 'dagitty'". In: *International Journal of Epidemiology* 45.6 (2016), pp. 1887–1894. DOI: 10.1093/ije/dyw341.

[33] *The Beauty of Sampling*. URL: https://online.stat.psu.edu/stat100/lesson/2/2.2 (visited on 2021-07-12).

[34] Ioannis Tsamardinos and Giorgos Borboudakis. "Permutation Testing Improves Bayesian Network Learning". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by José Luis Balcázar et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 322–337. ISBN: 978-3-642-15939-8.

[35] Ruibo Tu et al. "Causal Discovery in the Presence of Missing Data". In: *Proceedings of Machine Learning Research*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, 16–18 Apr 2019, pp. 1762–1770. URL: http://proceedings.mlr.press/v89/tu19a.html.