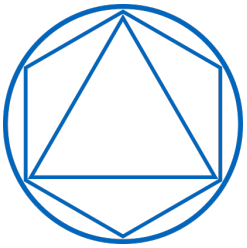


Technische Universität München

Fakultät für Mathematik

Learning sparse Gaussian graphical models with few covariance queries

Masterarbeit

von

Tom Hochsprung

Aufgabensteller:	Prof. Dr. Mathias Drton
Betreuer:	Dr. Carlos Améndola Cerón
Abgabetermin:	01.10.2021

Ich erkläre hiermit, dass ich diese Arbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe.

München, 01.10.2021

Danksagung

I would like to thank Professor Dr. Mathias Drton and Dr. Carlos Améndola Cerón for giving me the opportunity to write my thesis under their supervision. Both have been very generous with their time and provided numerous helpful insights. Moreover, they have been interested in my personal well-being and designed the best possible learning experience for me. I really appreciate the whole supervision process.

I would also like to thank my parents for their enduring support throughout my life. They encouraged me to study and worked hard to give me all opportunities.

Zusammenfassung

Graphische Modelle sind nützlich, um multivariate Abhängigkeiten zu modellieren; das Lernen ihrer Struktur basierend auf Daten ist hierbei ein wichtiges Problem. Klassische Methoden, um Gaußsche Graphische Modelle zu lernen, erfordern das Abspeichern der gesamten Kovarianzmatrix. In hochdimensionalen Fällen kann dies zu aufwendig sein. Wir untersuchen einen Algorithmus von Lugosi et al. (2021), der Gaußsche Graphische Modelle, insbesondere partielle Korrelationsgraphen, in kurzer Zeit mit wenigen Einträgen aus der Kovarianzmatrix lernen kann. Wir erweitern diesen Algorithmus auf empirische Anwendungsfälle mithilfe von Hypothesentests. Wir betrachten die Wahl von passenden Signifikanzniveaus aus zwei unterschiedlichen Perspektiven, einmal aus der Perspektive des multiplen Testens und einmal aus der Pespektive, dass jede Hypothese mit gleichem Signifikanzniveau getestet wird. Wir beweisen analoge Resultate zur Korrektheit und Zeitkomplexität und zeigen, dass diese Resultate auch gelten, wenn nur wenige Einträge aus der empirischen Kovarianzmatrix benutzt werden. Wir beweisen diese Resultate, indem wir das Verhältnis des empirischen und des nicht-empirischen Falls untersuchen und dann diese Resultate mit denen von Lugosi et al. (2021) verknüpfen. Wir erweitern den Algorithmus auch auf binäre Verteilungen und leiten analoge Resultate her. Zum Schluss untersuchen wir jeden der Algorithmen in Simulationen mit synthetisch generierten Daten.

Abstract

Graphical models are useful in modelling multivariate dependencies; learning their structure based on data is a key problem. Classical methods for learning Gaussian graphical models rely on storing the entire covariance matrix, in high-dimensional settings this is often too expensive. We investigate an algorithm of Lugosi et al. (2021) that is able to learn Gaussian graphical models, in particular partial correlation graphs, in a short time using only a few entries of the covariance matrix. We extend that algorithm to an empirical setting using the notion of hypothesis testing. We discuss choosing appropriate significance levels from two different points of view, multiple testing and testing each hypothesis at the same significance level. We prove analogous results on the correctness and time complexity of the algorithm and we show that this is possible using only a few entries of the sample covariance matrix. We usually do so by bridging the gap between the empirical and non-empirical case and combine these results with results of Lugosi et al. (2021). Finally, we extend the algorithm and the results from Gaussian to binary distributions and examine each algorithm in simulations using synthetically generated data.

Nomenclature

Sets

$\mathbb{N}$	Natural numbers ≥ 1
$\mathbb{Z}$	Integers
$\mathbb{R}$	Real numbers
$\mathbb{C}$	Complex numbers
$\lvert V \rvert$	Cardinality of the set V
$[p]$	Set of first p integers, i.e. $[p] = \{1, \ldots, p\}$

Linear algebra

x	Scalar in $\mathbb{R}$
$\boldsymbol{x}$	Vector in $\mathbb{R}^d$ with $d \geq 1$
$\boldsymbol{x}_A$	Subvector $\boldsymbol{x}_A$ of $\boldsymbol{x}$ with indices in some set A
$\boldsymbol{\Sigma}$	Matrix in $\mathbb{R}^{d \cdot d}$ with $d \geq 1$
$\boldsymbol{\Sigma}_{A,B}$	Submatrix of Σ with rows in A and columns in B

Graph theory

$G = (V, E)$	A graph G with vertex set V and edge set E; we may write T instead of G to indicate that G is a tree
uw	Abbreviation of edge {u,w} in an undirected graph $G = (V, E)$
$\overline{uw}$	Path between vertex u and vertex w
$V(G)$	The vertex set of graph G
$E(G)$	The edge set of graph G

$\mathrm{nb}(A)$	All vertices that are adjacent to a vertex in A, but not in A itself
$\mathrm{cl}(A)$	All vertices that are either a neighbor of some vertex in A or in A itself
$G \setminus S$	$G \setminus S := G[V \setminus S]$
$\mathcal{G}(\boldsymbol{\Sigma})$	Concentration graph, also conditional independence graph, for Gaussian distributions with covariance matrix $\boldsymbol{\Sigma}$
$\mathcal{G}(p)$	Conditional independence graph for binary random vectors with distribution p

Probability theory

$(\Omega, \mathcal{F}, P)$	A probability space with set Ω, sigma-algebra $\mathcal{F}$ and probability measure P on the probability space $(\Omega, \mathcal{F})$.
X	Random variable
$\boldsymbol{X}$	Random vector
$\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y} \mid \boldsymbol{Z}$	$\boldsymbol{X}$ independent of $\boldsymbol{Y}$ given $\boldsymbol{Z}$
$\boldsymbol{X} \not\perp\!\!\!\perp \boldsymbol{Y} \mid \boldsymbol{Z}$	$\boldsymbol{X}$ not independent of $\boldsymbol{Y}$ given $\boldsymbol{Z}$
$\mathbb{E}_{\boldsymbol{X}}[\cdot]$	Expectation with respect to the (fixed) distribution of $\boldsymbol{X}$. Index is omitted if it is clear from the context.
$\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$	Independent and identically distributed (i.i.d.) sample of size n
$\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$	Realization of an i.i.d. sample of size n
$p_A(\boldsymbol{i}_A)$	Cell probability for cell i_A in the marginal table with columns in set A; indices are omitted if A contains all possible columns
$N_A(\boldsymbol{i}_A)$	Number of observations (random) for cell $\boldsymbol{i}_A$ in the marginal table with columns in the set A; indices are omitted if A contains all possible columns

$n_A(\boldsymbol{i}_A)$	Number of observations (not random) for cell $\boldsymbol{i}_A$ in the marginal table with columns in the set A; indices are omitted if A contains all possible columns
$\Phi(x)$	Cumulative distribution function of a standard normal random variable
$\Phi^{-1}_{1-\alpha}(x)$	$1-\alpha$-quantile of a standard normal distribution
$\Phi_2(x, y, \lambda)$	Cumulative distribution function of a bivariate normal random vector with mean 0 and correlation λ
$\chi^2_k(x)$	Cumulative distribution function of a chi-square random variable with k degrees of freedom
$(\chi^2)^{-1}_{1-\alpha,k}$	$1-\alpha$-quantile of a chi-square distribution with k degrees of freedom

Asymptotics

$f(n) = \mathcal{O}(g(n))$	There exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$
$f(n) = \Omega(g(n))$	There exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$
$f(n) = o(g(n))$	For any positive constant $c > 0$, there exists a constant n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$
$f(n) = \omega(g(n))$	For any positive constant $c > 0$, there exists a constant n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$

Contents

List of Figures

List of Tables

1. Introduction

Graphical models allow to represent complex multivariate dependencies of a given random vector $\boldsymbol{X}$ using the notion of graphs. Each component of $\boldsymbol{X}$ corresponds to a vertex in an underlying graph $G = (V, E)$ with vertex set V and edge set E. The edge structure of G encodes conditional independencies between the components of $\boldsymbol{X}$ and the family of all distributions that satisfy these conditional independencies is the graphical model associated to G (see Lauritzen (1996)).

A sparse graphical model is a graphical model such that the associated graph is sparse, i.e. the graph has only few edges. One usually aims for sparsity, as this allows to reduce noise that is encorporated into the model. Furthermore, sparse graphical models are computationally less demanding, especially in high-dimensional settings (see Meinshausen and Bühlmann (2006)). Moreover, sparsity is motivated by the scholastic principle of parsimony, also known as Occam's razor, which advocates to choose the simplest model that fits the data (see MacKay (2003), Chapter 28; Jefferys and Berger (1992) and Thorburn (1918)).
A popular class of sparse graphs, which then induce sparse graphical models, are trees. Trees are connected graphs with no cycles; they are considered sparse because they have only $|V| - 1$ edges compared to the complete graph that has $|V|(|V| - 1)/2$ edges.

The Gaussian distribution is particularly popular in the context of graphical modelling. For Gaussian distributions one can show that two components of $\boldsymbol{X}$ are conditionally independent given all other components if and only if the corresponding entry in the inverse covariance matrix $\boldsymbol{K}$, also called precision or concentration matrix, is zero (Lauritzen (1996), Proposition 5.2). This conditional independence can be encoded using concentration graphs; an edge in the concentration graph is present if and only if the respective entry of $\boldsymbol{K}$ is non-zero. For Gaussian graphical models, sparsity is thus also associated to few non-zero entries in the precision matrix $\boldsymbol{K}$.
Methods to identify conditional independencies using the precision matrix based on an i.i.d. sample $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$ were introduced by Dempster (1972) and are also known under the name covariance selection. Due to the aforementioned relation to concentration graphs, subsequent methods for covariance selection often included ideas from graph theory. Examples include greedy or stepwise search,

where one efficiently adds or deletes edges based on some criterion like the AIC or BIC (see Hojsgaard et al. (2012), Section 4.4; Foygel and Drton (2010)).

More recently, there has been some focus on sparse high-dimensional settings, as even greedy or stepwise search turns out to be too costly in these cases. One approach is neighborhood selection, which has been introduced by Meinshausen and Bühlmann (2006). In neighborhood selection, one tries to learn the neighborhood of each vertex by regressing it onto the remaining vertices using regularized regression methods like the lasso. It has been shown that neighborhood selection chooses the correct support, i.e. the set of all non-zero entries of $\boldsymbol{K}$ asymptotically in sparse settings.

Another lasso-type approach is the graphical lasso (see Banerjee et al. (2008); Friedman et al. (2008)). It tries to find a maximum of the penalized log-likelihood of the underlying Gaussian distribution and hence in some sense, exactifies neighborhood selection. The graphical lasso can also be used to preselect some models that can then be analyzed further using information criteria and stepwise search approaches (see Foygel and Drton (2010)).
There are also algorithms for special types of graphs; one example is the Chow-Liu algorithm that makes use of a particular factorization property of trees (see Chow and Liu (1968)). We explore the Chow-Liu algorithm briefly in Chapter 4.

A problem is that the aforementioned and other algorithms have a computational complexity of at least the squared or even cubed number of vertices (see Hsieh et al. (2012); Hsieh et al. (2013)). In large scale applications, e.g. in gene expression studies (see e.g. Thorne (2018); Zhang et al. (2012); Hwang et al. (2018); Chan et al. (2017)), calculating and storing the entire covariance matrix may be prohibitively expensive.

Lugosi et al. (2021) have developed a set of algorithms that learns the structure of a given Gaussian graphical model if the underlying graph is a tree (or some other special type of graph). We refer to these algorithms overall as the LTVZ-algorithm; this name is composed of the initials of the authors last names, Lugosi, Truszkowski, Velona and Zwiernik. The LTVZ-algorithm learns the support of the concentration matrix $\boldsymbol{K}$ of a given random vector $\boldsymbol{X}$ using only some entries of the covariance matrix $\boldsymbol{\Sigma}$. For that, Lugosi et al. (2021) imagine the abstract setting where the true underlying covariance matrix $\boldsymbol{\Sigma}$ is known and an oracle returns an entry $\Sigma_{i,j}$ given indices i, j. The LTVZ-algorithm only works for special types of graphs, because it uses inherent factorizations for the respective graph type.

In practical settings, one usually works with an estimate of $\boldsymbol{\Sigma}$. We work out how the particular adaption to the empirical setting can be done. We do this in the framework of hypothesis testing, which slightly deviates from the approach Lugosi

et al. (2021) foreshadow. We spend some time on choosing appropriate significance levels for the occuring hypotheses tests by approaches from multiple testing (motivated by Drton and Perlman (2007)) and by fixing some overall significance level at which each hypothesis is tested (motivated by Kalisch and Bühlmann (2007)). We prove consistency results for obtaining the correct tree structure and we derive complexity bounds for the empirical versions of the LTVZ-algorithm; in particular, tuning parameters can be chosen such that the time complexity and number of entries used from the covariance matrix is asymptotically of order $\mathcal{O}(|V|\ln(|V|)\max\{\ln(|V|)/(\varepsilon-\alpha), d\})$ with probability of at least $1-\varepsilon$. Here, α is a familywise error rate of particular hypotheses tests and d is the maximal degree of the underlying tree.

Gaussian graphical models received a lot of attention in the realm of graphical modelling. This relied on the fact that all conditional independencies of a Gaussian random vector $\boldsymbol{X}$ are encoded in the concentration matrix $\boldsymbol{K}$. It has only recently become more clear, whether similar results are true for non-Gaussian distributions as well. Liu et al. (2009) and Liu et al. (2012) have shown that similar results carry over to nonparanormal distributions; Loh and Wainwright (2013) studied discrete graphical models and showed that the inverses of generalized covariance matrices, i.e. covariance matrices that are augmented by further moments, encode the dependence structure. For binary distributions corresponding to a tree, the generalized covariance matrix is the usual covariance matrix $\boldsymbol{\Sigma}$.
It also turns out that a factorization property for trees, namely that the pairwise correlations factorize with respect to a tree, a property which the LTVZ-algoritm uses for Gaussians, is also true for binary distributions (see Zwiernik (2019)).
This motivates the extension of the LTVZ-algorithm to the binary case, which has only been mentioned in some remarks of Lugosi et al. (2021). We do this extension, again in the framework of hypothesis testing. For that, we rely on the conditional G^2-test. This test is a popular choice for other related algorithms (see e.g. Neapolitan (2004), Section 10.3.1); however, knowing all entries of the sample covariance matrix is not enough to calculate it. Due to this self-imposed problem, we imagine an oracle that stores all three-way tables and returns the respective three-way table given indices i, j, k. Note that storing three-way tables has a cubic order; storing covariance matrices has only quadratic order. Nevertheless, we show similar results as for Gaussians, but see in our simulations that this self-imposed complication is indeed a problem in practice.

The LTVZ-algorithm is in particular interesting for large scale applications, e.g. to infer a gene regulatory network from gene expression data. Datasets obtained from normalized microarrays are usually considered normally distributed, however, inference from other types of data, e.g. RNA-sequencing data, is more challenging

as the data does not follow a Gaussian distribution; in the case of RNA-sequencing data one usually looks at the Poisson or negative binomial distribution. The literature for these non-Gaussian distributions is not so well developed (see Thorne (2018)). Therefore, developing a binary version of the LTVZ-algorithm is a step forward, but not the end of the journey.

The overall structure of the thesis is as follows.

In **Chapter 2** we review some basic facts of graph theory, graphical models, Gaussian distributions, discrete distributions and structure learning.

In **Chapter 3** we discuss the LTVZ-algorithm in detail and derive sample versions of it in tandem; we begin with Gaussians and then move to the binary setting.

In **Chapter 4** we do simulations for the LTVZ-algorithm and compare it to the Chow-Liu algorithm. We see that the Chow-Liu algorithm performs better in terms of correctness in most settings, but for some settings it could be beaten. In any case, the LTVZ-algorithm is significantly better than the randomly guessing trees. We also see that the runtime of the Chow-Liu and the LTVZ-algorithm heavily depend on how much time a particular query costs and how large the graph is. We empirically show that for an increasing number of vertices, the LTVZ-algorithm can have a non-worsening level of correctness while at the same time the relative query complexity, i.e. the number of queries divided by the number of all unique entries in the sample covariance matrix, is decreasing. Moreover, we show that the LTVZ-algorithm is faster than the Chow-Liu algorithm for larger vertex sets or if a query is more time consuming.

2. Preliminaries

2.1. Graph theory

We introduce the basic notions of graph theory as presented in Lauritzen (1996), Harary (1969) and Lugosi et al. (2021).

A graph $G = (V, E)$ is a pair of finite sets $V = V(G)$ and $E = E(G)$. V is the vertex set and $E \subseteq V \times V$ is the edge set. We assume that the graph contains no self-loops, that is $(u, u) \notin E$ for all $u \in V$. A graph is called undirected if $(u, v) \in E$ implies $(v, u) \in E$. In this case, we write $\{u, v\}$ or even simpler uv and say that u and v are adjacent, neighbors or connected by a line. Henceforth, we use the notions graph and undirected graph interchangeably if not indicated otherwise.

The set of all neighbors of v is denoted by $\mathrm{nb}(v)$ and the number of neighbors is the degree $\deg(v)$. The maximal degree of a graph is $\Delta(G) := \max_{v \in V} \deg(v)$. The set of all neighbors of $A \subseteq V$ is

$$\mathrm{nb}(A) = \bigcup_{v \in A} \mathrm{nb}(v) \setminus A$$

and the closure of A is

$$\mathrm{cl}(A) = \mathrm{nb}(A) \cup A.$$

A graph $G' = (V', E')$ is a subgraph of G if and only if $V' \subseteq V$ and $E' \subseteq E$. If $E' = \{uv \in E \mid u, v \in V'\}$, then G' is called the induced subgraph $G[V']$ of G on V'. For $S \subseteq V$ we write $G \setminus S$ to denote $G[V \setminus S]$. A graph is complete, if all vertices are joined by an edge. A subset $V' \subseteq V$ is complete, if it induces a complete subgraph. A maximal complete subset of V is a clique.

A path from u to v is a sequence $v_0v_1, \ldots, v_{k-1}v_k$ of edges with $v_0 = u$ and $v_k = v$. We allow paths to consists of a single vertex. These paths are called empty paths. Two vertices u and v are connected if there is a path between them. A graph is connected if all of its vertices $u, v \in V$ are connected. A path is called a cycle if $v_0 = v_k$. An acyclic graph is a graph with no cycles. A tree is a connected acyclic graph.

A maximal connected subgraph of G, with respect to inclusion, is a connected component of G. For sets $A, B, S \subseteq V$ we say that S separates $A, B \subseteq V$ in G if every path between A and B contains a node in S. The set S is called a separator of A and B. We do not prohibit A and B to intersect. If they do, each separator of A and B needs to contain $A \cap B$. Let $\mathcal{C}^S$ be the set of all connected components of the graph $G \setminus S$ and assume that A and B do not intersect. In case S separates A and B, then u and v do not lie in the same components of $G \setminus S$ if $u \in A \setminus S$ and $v \in B \setminus S$.

In this thesis we focus on trees. So far, we have defined a tree as a connected acyclic graph. The following lemma states a well known equivalence between different definition of trees (Harary (1969), Theorem 4.1).

Lemma 2.1. *For a graph G, the following statements are equivalent:*

(a) G is a tree.

(b) Every two points of G are joined by a unique path.

(c) G is connected and $|E| = |V| - 1$.

2.2. Conditional independence

In this section we introduce the notion of conditional independence. We refer to Studený (2005), who gives a thorough treatment of conditional independence. Some outlines can be found in Studený (2019).

For each $v \in V$, let $(\Omega_v, \mathcal{F}_v)$ be some measure space. Here, we already foreshadow the connection to graphs, by indexing with a vertex $v \in V$. Let P be a probability measure on the Cartesian product of these measure spaces. For a nonempty proper subset A of V, let P^A denote the marginal of P for A and $\mathcal{F}_A$ the corresponding sigma-algebra. We adopt the following two conventions. In case $A = V$, we say that $P^A = P$, and in case $A = \emptyset$, we say that P^A is a probability measure on the trivial sigma-algebra.

Given two disjoint sets $A, C \subseteq V$ the conditional probability on Ω_A given Ω_C is a function of two arguments $P_{A|C} : \mathcal{F}_A \times \mathcal{F}_C \longrightarrow [0, 1]$ which maps an $\mathcal{F}_C$-measurable function $P_{A|C}(A|\cdot)$ to every $A \in \mathcal{F}_A$ such that

$$P^{AC}(A \times C) = \int_C P_{A|C}(A|x) dP^C(\boldsymbol{x})$$

for every $C \in \mathcal{F}_C$.

To define conditional independence, let $A, B, C \subseteq V$ be some disjoint subsets. We say that A is conditionally independent of B given C if for every $A' \in \mathcal{F}_A$ and for every $B' \in \mathcal{F}_B$,

$$P_{AB|C}(A' \times B'|\boldsymbol{x}) = P_{A|C}(A'|\boldsymbol{x}) \cdot P_{B|C}(B'|\boldsymbol{x})$$

for P^C almost every $\boldsymbol{x} \in \Omega_C$. In that case, we write $A \perp\!\!\!\perp B|C$.

This setup already includes the setup for random variables. To see this, let each X_v be a random variable with respect to the probability space $(\Omega, \mathcal{F}, P^v)$. The random vector $\boldsymbol{X}$ having components X_v is then defined on the probability space $((\bigtimes_{v\in V} \Omega, \bigtimes_{v\in V} \mathcal{F}, P)$, where P denotes the product measure of the P^v. Note that by formulating it this way, P is always a product measure. We are not interested in non-product-measure-cases.

Again, let P^A denote the marginal with respect to some set $A \subseteq V$. For some Borel set F_A (with respect to the Borel sigma-algebra of $\mathbb{R}^{|A|}$) we can write

$$P^A(\boldsymbol{X}_A \in F_A) = P^A(\boldsymbol{X}_A^{-1}(F_A)).$$

As each X_v is a random variable and thus, in particular measurable, $\boldsymbol{X}_A^{-1}(F_A)$ is a measurable set with respect to the product sigma algebra $((\bigtimes_{v\in A} \Omega, \bigtimes_{v\in A} \mathcal{F})$. For these sets, we have established the notion of conditional independence above. Therefore, we say that $\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B|\boldsymbol{X}_C$ for some disjoint sets $A, B, C \subseteq V$ if

$$\begin{aligned} &P_{AB|C}(\boldsymbol{X}_A \in F_A, \boldsymbol{X}_B \in F_B|\boldsymbol{X}_c = \boldsymbol{x}_c) \\ &\qquad = P_{A|C}(\boldsymbol{X}_A \in F_A|\boldsymbol{X}_c = \boldsymbol{x}_c) P_{B|C}(\boldsymbol{X}_B \in F_B|\boldsymbol{X}_c = \boldsymbol{x}_c) \end{aligned} \tag{2.1}$$

for all Borel sets F_A, F_B and for $P^{\boldsymbol{X}_c}$ almost every $\boldsymbol{x}_c$. Equivalently, we sometimes say $A \perp\!\!\!\perp B|C$.

As this thesis focuses on discrete and (regular) Gaussian random vectors, i.e. examples that have densities with respect to the count or Lebesgue measure, we henceforth use the characterization via densities. Chapter 3 of Lauritzen (1996) contains the following useful result.

Proposition 2.2. *Let $\boldsymbol{X}$ be a random vector and let f be the generic symbol for a density.*[1] *Then, the following statements are equivalent.*

(a) $\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B|\boldsymbol{X}_C$.

(b) $f(\boldsymbol{x}_A, \boldsymbol{x}_B, \boldsymbol{x}_C) = f(\boldsymbol{x}_A, \boldsymbol{x}_C) f(\boldsymbol{x}_B, \boldsymbol{x}_C)/f(\boldsymbol{x}_C)$.

[1] This is to avoid subscripts $f_{\boldsymbol{X}_A \boldsymbol{X}_B|\boldsymbol{X}_C=\boldsymbol{x}_C}$.

(c) $f(\boldsymbol{x}_A|\boldsymbol{x}_B, \boldsymbol{x}_C) = f(\boldsymbol{x}_A|\boldsymbol{x}_C)$.

(d) $f(\boldsymbol{x}_A, \boldsymbol{x}_C|\boldsymbol{x}_B) = f(\boldsymbol{x}_A|\boldsymbol{x}_C)f(\boldsymbol{x}_C|\boldsymbol{x}_B)$.

(e) $f(\boldsymbol{x}_A, \boldsymbol{x}_B, \boldsymbol{x}_C) = h(\boldsymbol{x}_A, \boldsymbol{x}_C)k(\boldsymbol{x}_B, \boldsymbol{x}_C)$ *for some measurable functions* h, k.

(f) $f(\boldsymbol{x}_A, \boldsymbol{x}_B, \boldsymbol{x}_C) = f(\boldsymbol{x}_A|\boldsymbol{x}_C)f(\boldsymbol{x}_B, \boldsymbol{x}_C)$.

Here, all equalities hold with probabilitiy 1.

There are 4 general properties of conditional independence.

(C1) Symmetry:
$\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B|\boldsymbol{X}_C \iff \boldsymbol{X}_B \perp\!\!\!\perp \boldsymbol{X}_A|\boldsymbol{X}_C$.

(C2) Decomposition:
$\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B|\boldsymbol{X}_C \implies h(\boldsymbol{X}_A) \perp\!\!\!\perp \boldsymbol{X}_B|\boldsymbol{X}_C$ for any measurable function h.

(C3) Weak union:
$\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B|\boldsymbol{X}_C \implies \boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B|(\boldsymbol{X}_C, h(\boldsymbol{X}_A))$ for any measurable function h.

(C4) Contraction:
$\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B|\boldsymbol{X}_C$ and $\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_D|(\boldsymbol{X}_B, \boldsymbol{X}_C) \implies \boldsymbol{X}_A \perp\!\!\!\perp (\boldsymbol{X}_D, \boldsymbol{X}_B)|\boldsymbol{X}_C$.

Studený (2005) also includes $\boldsymbol{X}_A \perp\!\!\!\perp \emptyset|\boldsymbol{X}_C$ and calls it triviality. We abbreviate it with (C0). There is also a fifth property (C5), which, however, is not always true. But if the joint density of $\boldsymbol{X}$ is positive and continuous with respect to a product measure, then (C5) is true. On the other hand, positivity of the density is not necessary for (C5).

(C5) Intersection:
If $\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B|\boldsymbol{X}_C$ and $\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_C|\boldsymbol{X}_B$, then $\boldsymbol{X}_A \perp\!\!\!\perp (\boldsymbol{X}_B, \boldsymbol{X}_C)$.

Note that the reverse implication of (C5) follows from (C4). Similarly, the reverse implication of (C4) follows from (C2) combined with (C3).

We conclude the discussion of these properties with a small remark. The notion of conditional independence can be generalized and studied from an algebraic perspective. In particular, the properties (C1)-(C5) become axioms that define algebraic structures. For example, if (C1)-(C4) holds, one speaks of semi-graphoids. For more details we refer to Pearl (1988) and Studený (2005).

2.3. Conditional independence and graphs

There are several ways of relating the notion of conditional independence to graphs. We present these approaches as done in Chapter 3 of Lauritzen (1996).

The first idea is to connect independence to missing edges. One may say that two nodes are independent given all the other nodes if they are not connected by an edge. The pairwise Markov property formalizes this idea.

Definition 2.3. Let $\boldsymbol{X}$ be a random vector with joint distribution $P^{\boldsymbol{X}}$. Then, $P^{\boldsymbol{X}}$ satisfies the pairwise Markov property relative to G if for every non-adjacent pair of nodes u and v,

$$X_u \perp\!\!\!\perp X_v | \boldsymbol{X}_{V \setminus \{u,v\}}.$$

Another intuitive idea is to define conditional independence via neighbors. One could say that u is independent of all non-neighbors given the neighbors. The local Markov property formalizes this approach.

Definition 2.4. Let $\boldsymbol{X}$ be a random vector with joint distribution $P^{\boldsymbol{X}}$. Then, $P^{\boldsymbol{X}}$ satisfies the local Markov property relative to G if for every vertex $u \in V$,

$$X_u \perp\!\!\!\perp \boldsymbol{X}_{V \setminus \mathrm{cl}(u)} | \boldsymbol{X}_{\mathrm{nb}(u)}.$$

Both ideas can be generalized to arbitrary separation statements. This gives rise to the global Markov Property.

Definition 2.5. Let $\boldsymbol{X}$ be a random vector with joint distribution $P^{\boldsymbol{X}}$. Then, $P^{\boldsymbol{X}}$ satisfies the global Markov property relative to G if for any triple (A, B, S) of disjoint subsets of V such that S separates A from B in G,

$$\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B | \boldsymbol{X}_S.$$

In cases where $P^{\boldsymbol{X}}$ is fixed, we also say that $\boldsymbol{X}$ satisfies the pairwise/local/global Markov property relative to G.

It is a classical result, e.g. Proposition 3.4 in Lauritzen (1996), that the global Markov property implies the local and pairwise Markov property and the local Markov property implies the pairwise Markov property. The contrary is not necessarily true. If, however, a distribution satisfies the intersection property (C5),

then these Markov properties are equivalent. This result has been proved by Pearl and Paz (1987).

As we have seen earlier, positivity and continuity of the density with respect to a product measure are sufficient for the intersection property (C5). Therefore, if a distribution has a positive and continuous density with respect to a product measure, all three Markov properties are equivalent.

As of now, we have seen that conditional independence can be related to graphs. In Proposition 2.2 we have seen that conditional independence has an intimate relationship to factorization. Unsurprisingly, we can relate graphs and factorization. Indeed, a probability measure $P^{\boldsymbol{X}}$ is said to factorize according to G, if for all complete subsets F of V there is a product measure $\boldsymbol{\mu}$ such that $P^{\boldsymbol{X}}$ has density f with respect to $\boldsymbol{\mu}$ and

$$f(\boldsymbol{x}) = \prod_{F \text{ complete}} \psi_F(\boldsymbol{x}).$$

Here, the so-called potential functions $\psi_F(\boldsymbol{x})$ are non-negative and only depend on $\boldsymbol{x}$ through $\boldsymbol{x}_F$. They do not have to be densities, but it is not forbidden that they can be. Obviously, this factorization is not unique. We may only look at cliques, i.e. maximally connected components, and set ψ_F equal to 1 for all complete subsets which are not a clique. Let $\mathcal{C}$ denote the set of all cliques of G. Then, we can alternatively write

$$f(\boldsymbol{x}) = \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}). \tag{2.2}$$

It can be shown that factorization implies the global Markov property (e.g. Lauritzen (1996), Proposition 3.8). Under some conditions, the reverse is also true.

Theorem 2.6 (Hammersley-Clifford)**.** *Let $P^{\boldsymbol{X}}$ be a probability distribution that has a positive and continuous density f with respect to some product measure $\boldsymbol{\mu}$. Then, $P^{\boldsymbol{X}}$ satisfies the pairwise Markov property if and only if it factorizes according to G.*

This theorem is very useful. On the one hand, we can check (under the assumptions) whether a given density satisfies the conditional independence statements of a particular graph. On the other hand, we can construct arbitrary joint densities that satisfy the conditional independence statements given by some graph as long as the ψ's are integrable, so we can normalize, and the other assumptions are met.

At this point, one important ingredient is still missing. All Markov properties are a deduction of conditional independence statements from separation statements. But they are not a deduction of separation statements from conditional

independence statements. In particular, the global Markov property ensures that separation statements imply conditional independence statements. However, we do not necessarily have the reverse implication, namely that conditional independence implies separation. A result known as strong completeness yields that we can always find a probability measure $P^{\boldsymbol{X}}$ such that C separates A and B if and only if $\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B | \boldsymbol{X}_C$, see e.g. Pearl and Paz (1987) or Pearl (1988). For this choice of $P^{\boldsymbol{X}}$ we also say that G is a perfect map of $P^{\boldsymbol{X}}$. If $P^{\boldsymbol{X}}$ is known and fixed, we may also say that G is perfect.

2.4. Gaussian graphical models

There are several introductions to the multivariate normal distribution and Gaussian graphical models. We focus on the ones given by Anderson (2003), Chapter 2, Uhler (2019) and Lauritzen (1996), Chapter 5 and Appendix.

A random vector $\boldsymbol{X} = (X_1, \ldots, X_p)$ follows a multivariate normal distribution $\mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ in p dimensions with mean $\boldsymbol{\mu} \in \mathbb{R}^p$ and positive definite covariance matrix $\boldsymbol{\Sigma} = [\Sigma_{i,j}]_{i,j \in \{1,\ldots,p\}}$, if the density is given by

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \det(2\pi\boldsymbol{\Sigma})^{-1/2} \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu}) \Big).$$

If the dimension is clear from context, we write $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ instead of $\mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The inverse of $\boldsymbol{\Sigma}$, which exists if we assume positive definiteness, is the concentration or precision matrix and we denote it by $\boldsymbol{K}$. In subsequent sections, we set $\boldsymbol{\mu} = \boldsymbol{0}$ as the mean plays no role in the analysis.

The multivariate normal distribution satisfies some helpful properties. It is closed under marginalization and conditioning as the following result shows.

Proposition 2.7. *Let $\boldsymbol{X} \in \mathbb{R}^p$ have distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and let $\boldsymbol{X}$ be partioned into two components $\boldsymbol{X}_A \in \mathbb{R}^a$ and $\boldsymbol{X}_B \in \mathbb{R}^b$ such that $a + b = p$. Similarly, write*

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{pmatrix} \text{ and } \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{A,A} & \boldsymbol{\Sigma}_{A,B} \\ \boldsymbol{\Sigma}_{B,A} & \boldsymbol{\Sigma}_{B,B} \end{pmatrix}$$

to denote the partition of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ with respect to A and B. Moreover, assume that $\boldsymbol{\Sigma}_{B,B}$ is symmetric and positive definite. Then,

(a) the marginal distribution of $\boldsymbol{X}_A$ is $\mathcal{N}(\boldsymbol{\mu}_A, \boldsymbol{\Sigma}_{A,A})$;

(b) the conditional distribution of $\boldsymbol{X}_A | \boldsymbol{X}_B = \boldsymbol{x}_B$ is $\mathcal{N}(\boldsymbol{\mu}_{A|B}, \boldsymbol{\Sigma}_{A|B})$, where

$$\boldsymbol{\mu}_{A|B} = \boldsymbol{\mu}_A + \boldsymbol{\Sigma}_{A,B}\boldsymbol{\Sigma}_{B,B}^{-1}(\boldsymbol{x}_B - \boldsymbol{\mu}_B) \text{ and } \boldsymbol{\Sigma}_{A|B} = \boldsymbol{\Sigma}_{A,A} - \boldsymbol{\Sigma}_{A,B}\boldsymbol{\Sigma}_{B,B}^{-1}\boldsymbol{\Sigma}_{B,A}.$$

We call the entries of $\boldsymbol{\Sigma}_{A,B}$ the conditional covariance $\boldsymbol{\Sigma}_{i,j|B}$. The partial correlation between X_i and X_j (with $i, j \in A$) holding $\boldsymbol{X}_B$ fixed, is defined by

$$\rho_{i,j|B} := \frac{\boldsymbol{\Sigma}_{i,j|B}}{\sqrt{\boldsymbol{\Sigma}_{i,i|B}}\sqrt{\boldsymbol{\Sigma}_{j,j|B}}}.$$

There exists a useful recursion formula, which is also true for other distributions with sufficiently many moments, to calculate the partial correlation coefficient $\rho_{i,j|B}$ (c.f. Yule and Kendall (1965), Section 12.15). Let $k \in B$ be arbitrary. Then,

$$\rho_{i,j|B} = \frac{\rho_{i,j|B\setminus\{k\}} - \rho_{i,k|B\setminus\{k\}}\rho_{j,k|B\setminus\{k\}}}{\sqrt{(1-\rho^2_{i,k|B\setminus\{k\}})(1-\rho^2_{j,k|B\setminus\{k\}})}}. \tag{2.3}$$

We can use this formula to estimate the partial correlation coefficient. We just need to calculate the respective pairwise sample correlations and apply the recursion formula to arrive at the estimate $\hat{\rho}_{i,j|B}$.

Proposition 2.7 spins off two corollaries, which are crucial for the remaining thesis.

Corollary 2.8. *Let $\boldsymbol{X} \in \mathbb{R}^p$ be a random vector with distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and let $i, j \in [p]$ with $i \neq j$. Then,*

(a) $X_i \perp\!\!\!\perp X_j$ if and only if $\Sigma_{i,j} = 0$;

(b) $X_i \perp\!\!\!\perp X_j | \boldsymbol{X}_{[p]\setminus\{i,j\}}$ if and only if $K_{i,j} = 0$ if and only if $\det(\boldsymbol{\Sigma}_{[p]\setminus\{i\},[p]\setminus\{j\}}) = 0$;

(c) The partial correlation coefficient $\rho_{i,j|[p]\setminus\{j\}} = 0$.

Remark 2.9. Note that property c) is basically the same as property b). Scaling $\boldsymbol{K}$ yields a matrix whose entries are given by the respective partial correlations.

The next Corollary refines the previous one.

Corollary 2.10. *Let $\boldsymbol{X} \in \mathbb{R}^p$ be a random vector with distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and let $i, j \in [p]$ with $i \neq j$. Furthermore, let $S \subseteq [p] \setminus \{i, j\}$. Then, the following statements are equivalent.*

(a) $X_i \perp\!\!\!\perp X_j | \boldsymbol{X}_S$;

(b) $\det(\boldsymbol{\Sigma}_{iS,jS}) = 0$, where $iS = \{i\} \cup S$;

(c) $\det(\boldsymbol{K}_{iR,jR}) = 0$, where $R = [p] \setminus (S \cup \{i, j\})$;

(d) The partial correlation coefficient $\rho_{i,j|S} = 0$.

Note that our reference for d), Proposition 5.2 in Lauritzen (1996), writes the statement not in the form given here. But the proof can be adapted, and other authors have used this result and reference as well (e.g. Kalisch and Bühlmann (2007)).

With these results, we are well-equipped to go back to graphical models. As above, let $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $p = |V|$ and let G be an undirected graph with vertex set V and edge set E. By the reasoning of the previous section and the fact that the multivariate normal distribution is positive and continuous with respect to a Lebesgue-product measure[2], it holds that

$$\text{Factorization} \iff \text{Global Mp.} \iff \text{Local Mp.} \iff \text{Pairwise Mp}$$

in the case of Gaussians. Due to this fact, we may say that $\boldsymbol{X}$ (or its distribution) is Markov relative to G if factorization or one of the Markov properties is satisfied.

Now, if $\boldsymbol{X}$ is Markov relative to G, we say that $\boldsymbol{X}$ satisfies the Gaussian graphical model or covariance selection model for $\boldsymbol{X}$ with graph G. A Gaussian graphical model $\mathcal{M}(G)$ of an undirected graph G is the set of all multivariate normal distributions that have a positive definite covariance matrix and are Markov relative to G. By definition and Corollary 2.8, all distributions in $\mathcal{M}(G)$ satisfy

$$\begin{aligned} \{u, v\} \notin E &\overset{\text{Pw. Markov Prop.}}{\implies} X_u \perp\!\!\!\perp X_v | \boldsymbol{X}_{V \setminus \{u,v\}} \\ &\iff K_{u,v} = 0 \\ &\iff \det(\boldsymbol{\Sigma}_{V\setminus\{u\}, V\setminus\{v\}}) = 0. \end{aligned} \tag{2.4}$$

Statement (2.4) illustrates the relation between missing edges, conditional independence, zeros in the concentration matrix and singular subdeterminants of the covariance matrix.

In this thesis, we focus on trees. There exists a very helpful factorization of pairwise correlations over trees. This result can be found in multiple papers, we refer to Lugosi et al. (2021) and Zwiernik (2019), Section 11.2.

Lemma 2.11. *Let G be a tree and $\boldsymbol{\Sigma} \in \mathcal{M}(G)$. Then, for any two vertices a, b of G, the correlation $\rho_{a,b} := Corr(X_a, X_b)$ can be written as the product*

$$\rho_{a,b} = \prod_{(u,v) \in \overline{ab}} \rho_{u,v}. \tag{2.5}$$

Here, $\overline{ab}$ denotes the unique path between a and b. The converse result is also true. If (2.5) is satisfied for some tree G, then $\boldsymbol{\Sigma} \in \mathcal{M}(G)$.

[2]This is true if $\boldsymbol{\Sigma}$ is positive definite, which we assume.

2.5. Structure learning for Gaussian graphical models

Given a Gaussian random vector $\boldsymbol{X}$, a natural goal is to learn the support of $\boldsymbol{K}$, i.e. which entries are non-zero, as this yields information about the dependence structure of $\boldsymbol{X}$. An overview of several methods to obtain information about the dependence structure is given by Drton and Maathuis (2017). In this thesis, we want to learn the dependence structure by using only some entries of $\boldsymbol{\Sigma}$ or a noisy version $\hat{\boldsymbol{\Sigma}}$. We exploit the intimate relationship of the support of $\boldsymbol{K}$ and the concentration graph

$$\begin{aligned}\mathcal{G}(\boldsymbol{\Sigma}) :&= (V, \{ij : \ X_i \not\perp\!\!\!\perp X_j \mid \boldsymbol{X}_{V\setminus\{i,j\}}\}) \\ &= (V, \{ij : \ K_{ij} \neq 0\}).\end{aligned}$$

This graph is also known under the name conditional independence graph. By definition, $\boldsymbol{\Sigma} \in \mathcal{M}(\mathcal{G}(\boldsymbol{\Sigma}))$ as the pairwise and hence all other Markov properties and factorization are satisfied. Furthermore, full equivalency holds in (2.4) for $G = \mathcal{G}(\boldsymbol{\Sigma})$. Thus, if we learn the edges of $\mathcal{G}(\boldsymbol{\Sigma})$ employing graph algorithms, we learn the entire support of $\boldsymbol{K}$.

Condition d) of Corollary 2.10 is helping us on that endeavor. It allow us to check whether $X_v \perp\!\!\!\perp X_u | \boldsymbol{X}_S$ for some $u, v \in V$ and $S \subseteq V$ assuming we know $\boldsymbol{\Sigma}$. In practice, however, we do not know $\boldsymbol{\Sigma}$ and only have some data $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ which are a realization of the i.i.d. random variables $\boldsymbol{X}^{(1)} \ldots, \boldsymbol{X}^{(n)}$ that have the same distribution as $\boldsymbol{X}$. Due to that, we cannot apply the result of Corollary 2.10 directly. Instead, we need to test whether condition d) of Corollary 2.10 is actually true. A natural choice for doing this is the hypothesis test

$$H_0 : \ \rho_{u,w|S} = 0 \text{ vs. } H_1 : \rho_{u,w|S} \neq 0. \tag{2.6}$$

Anderson (2003) discusses these kind of tests in Chapter 4 of his book. We briefly outline a result which is very useful to us. For that, let $\hat{\rho}_{u,w|S}$ be the sample partial correlation coefficient and let

$$z(x) := \frac{1}{2} \ln\left(\frac{1+x}{1-x}\right)$$

denote Fisher's z-transform. Furthermore, write $z_{u,w|S} := z(\rho_{u,w|S})$ and $\hat{z}_{u,w|S} := z(\hat{\rho}_{u,w|S})$. Since $x = 0 \iff z(x) = 0$, (2.6) is equivalent to

$$H_0 : \ \hat{z}_{u,w|S} = 0 \text{ vs. } H_1 : \hat{z}_{u,w|S} \neq 0.$$

We can construct an asymptotic level α test based on a result due to Fisher (1924), namely that

$$\sqrt{n-|S|-3}\big(z_{u,w|S}-\hat{z}_{u,w|S}\big) \xrightarrow{d} \mathcal{N}(0,1).$$

In particular, we reject H_0 at level $\alpha \in (0,1)$ if

$$\sqrt{n-|S|-3}\big|z_{u,w|S}\big| > \Phi^{-1}\left(1-\frac{\alpha}{2}\right). \tag{2.7}$$

Here, Φ denotes the cumulative distribution function of $\mathcal{N}(0,1)$.

We conclude this section with a small remark. Anderson (2003), Chapter 4, also discusses other asymptotic results without using the z-transform. But either the z-transform is easier to use or the convergence is much faster, as the z-transform exhibits variance stabilizing properties.

2.6. Graphical models for discrete data

For the following outline, we refer to La Rocca and Roverato (2019), Roverato (2017) and Chapter 4 of Lauritzen (1996). For discrete data, we assume that each component X_v of $\boldsymbol{X}$ is discrete. We denote the state space of X_v by $\mathcal{I}_v$ and the joint state space of $\boldsymbol{X}$ by

$$\mathcal{I} = \bigtimes_{v\in V} \mathcal{I}_v,$$

which is the Cartesian product of the component-state-spaces. In the discrete setting all state spaces $\mathcal{I}_v$ corresponding to some component X_v contain finitely many elements, therefore also the joint state space $\mathcal{I}$ of $\boldsymbol{X}$ contains finitely many elements. Because of this, we can write $\mathcal{I}_v = \{0,1,\dots,d_v-1\}$, whereas d_v denotes the number of elements in $\mathcal{I}_v$ and

$$\mathcal{I} = \{\boldsymbol{i} := (i_1,\dots,i_{|V|}) :\ \forall j \in \{1,\dots,|V|\} :\ i_j \in \mathcal{I}_j\}.$$

For some given $\boldsymbol{i} \in \mathcal{I}$ we express the joint probability that $\boldsymbol{X} = \boldsymbol{i}$ by

$$p(\boldsymbol{i}) := \mathrm{Prob}(X_1 = i_1,\dots,X_{|V|} = i_{|V|}).$$

By definition, $\sum_{i\in\mathcal{I}} p(\boldsymbol{i}) = 1$ and $p(\boldsymbol{i}) \geq 0$ for all $\boldsymbol{i} \in \mathcal{I}$.

From now on we assume that the distribution $p := (p(\boldsymbol{i}))_{\boldsymbol{i}\in\mathcal{I}}$ of $\boldsymbol{X}$ is positive, which means that $p(\boldsymbol{i}) > 0$ for all $\boldsymbol{i} \in \mathcal{I}$. Furthermore, to avoid degenerated cases, we assume that no components of $\boldsymbol{X}$ are linearly dependent. Then, given some

graph G, we say that the graphical model given by G, which we denote by $\mathcal{M}_+(G)$, is the family of all positive probability distributions for $\boldsymbol{X}$ such that $\boldsymbol{X}$ satisfies the global Markov property according to G. As we assume that $\boldsymbol{X}$ has a positive distribution, the theorem of Hammesley-Clifford (Theorem 2.6) applies and yields that the global Markov property is equivalent to the other Markov properties and factorization according to G.

The graphical model $\mathcal{M}_+(G)$ is sometimes also called the log-linear graphical model specified by G. The reason for this is as follows. From (2.2) we know that that $p = (p(\boldsymbol{i}))_{\boldsymbol{i}\in\mathcal{I}} \in \mathcal{M}_+(G)$ is equivalent to the existence of some potential functions $\psi_F : \mathcal{I} \to \mathbb{R}$, where $\psi_F(\boldsymbol{i})$ only depends on $\boldsymbol{i}_F$, such that

$$p(\boldsymbol{i}) = \prod_{F\subseteq V \text{complete}} \psi_F(\boldsymbol{i}),$$

for all $\boldsymbol{i} \in \mathcal{I}$. We can rewrite this product by applying the logarithm on both sides. By this, we obtain that

$$\ln(p(\boldsymbol{i})) = \sum_{F\subseteq V \text{complete}} \ln(\psi_F(\boldsymbol{i})).$$

This linear parametrization after applying the logarithm gives rise to the name of log-linear graphical model.

Henceforth, we restrict ourselves to binary graphical models. That means, each $\mathcal{I}_v = \{0, 1\}$ up to isomorphisms. We are interested in similar conditions that encode conditional independence statements and are easy to check, similarly as for Gaussians. For binary random vectors, there exists the following useful lemma (see Corollary 2 in Loh and Wainwright (2013)).

Lemma 2.12. *Let $\boldsymbol{X}$ be a binary random vector with strictly positive distribution p and inverse covariance matrix $\boldsymbol{K}$. Let $p \in \mathcal{M}_+(T)$ for a tree $T = (V, E)$. Then, $(u, w) \notin E$ implies*

(A) $K_{u,w} = 0$ *and*

(B) $\rho_{u,w|V\setminus\{u,w\}} = 0$.

Furthermore, it holds up to null sets that $(u, w) \in E$ implies

(a) $K_{u,w} \neq 0$ *and*

(b) $\rho_{u,w|V\setminus\{u,w\}} \neq 0$.

Here, null sets refer to Lebesgue null sets with respect to the set of all $p \in \mathcal{M}_+(T)$, which can be seen as a subset of $[0,1]^{|\mathcal{I}|-1}$. As a small remark, for general discrete distributions ibid. introduce generalized covariance matrices and obtain generalized results.

There is also a binary analog of Lemma 2.11, which is stated in the following; we again refer to Lugosi et al. (2021) and Zwiernik (2019), Section 11.2.

Lemma 2.13. *Let G be a tree and let $p = (p(\boldsymbol{i}))_{\boldsymbol{i}\in\mathcal{I}} \in \mathcal{M}_+(G)$. Then, for any two vertices a, b of G, the correlation $\rho_{a,b} := Corr(X_a, X_b)$ can be written as the product*

$$\rho_{a,b} = \prod_{(u,v)\in\overline{ab}} \rho_{u,v}. \tag{2.8}$$

Here, $\overline{ab}$ denotes the unique path between a and b.

2.7. Structure learning for discrete graphical models

Similarly as for Gaussians, we aim to learn the conditional independence structure of a binary random vector $\boldsymbol{X}$, given the true cell probabilities $p(\boldsymbol{i})$ or some noisy version $\hat{p}(\boldsymbol{i})$. For that, we again look at the conditional independence graph

$$\mathcal{G}(p) := (V, \{uw : X_u \not\perp\!\!\!\perp X_w \mid X_{V\setminus\{u,w\}}\}).$$

Note that in the binary case $\mathcal{G}(p) = (V, \{ij : K_{ij} \neq 0\})$ up to null sets if $\mathcal{G}(p)$ is a tree; this follows from Lemma 2.12. Hence, we call $\mathcal{G}(p)$ the concentration graph in the binary case as well. Therefore, by learning the edges of $\mathcal{G}(p)$, we obtain information about the dependence structure of $\boldsymbol{X}$, as $p \in \mathcal{M}_+(\mathcal{G}(p))$ by definition (the pairwise Markov property clearly holds).

Lemma 2.13 helps us to learn $\mathcal{G}(p)$, but is only sufficient if we know the underlying true cell probabilities $p(\boldsymbol{i})$ for all $\boldsymbol{i} \in \mathcal{I}$. In practice, we usually do not know the true cell probabilities, we need to estimate them by data. We outline some of the theory behind this, if not otherwise stated, based on Chapter 4 of Lauritzen (1996).

Discrete data, so the realization of random vectors $\boldsymbol{X}^{(k)}$ for $k = 1, \ldots, n$ with the same distribution as $\boldsymbol{X}$, is often available as a list of observations or as a contingency table of counts $\{N(\boldsymbol{i})\}_{\boldsymbol{i}\in\mathcal{I}}$. There are several sampling schemes one can assume. We focus on the sampling scheme, where the total number of observations is fixed, but all cell counts are otherwise random. We assume that there is a total of n independent "objects" to be classified, and each "object" belongs to a cell

$\boldsymbol{i}$ with probability $p(\boldsymbol{i})$. Thus in our setup, binary data is a realization of n i.i.d. random variables $\boldsymbol{X}^{(k)}$ with the same distribution as $\boldsymbol{X}$. Under this setting, the counts for a cell $\boldsymbol{i}$ follow a multinomial distribution, i.e.

$$\text{Prob}(N(\boldsymbol{i}) = n(\boldsymbol{i}),\ \boldsymbol{i} \in \mathcal{I}) = \frac{n!}{\prod_{\boldsymbol{j}\in\mathcal{I}} n(\boldsymbol{j})!} \prod_{\boldsymbol{j}\in\mathcal{I}} p(\boldsymbol{j})^{n(\boldsymbol{j})},$$

whereas $\sum_{\boldsymbol{i}\in\mathcal{I}} n(\boldsymbol{i}) = n$.

For some set $A \subseteq V$ let $N_A(\boldsymbol{i}_A)$ resp. $n_A(\boldsymbol{i}_A)$ denote the marginal number of counts in the cell $\boldsymbol{i}_A$ of the table with columns given by the elements of A. Similarly, let $p_A(\boldsymbol{i}_A)$ denote the corresponding marginal cell probability. It holds that the distribution for this marginal table is again multinomial, i.e.

$$\text{Prob}(N_A(\boldsymbol{i}_A) = n(\boldsymbol{i}_A),\ \boldsymbol{i}_A \in \mathcal{I}_A) = \frac{n!}{\prod_{\boldsymbol{j}_A\in\mathcal{I}_A} n(\boldsymbol{j}_A)!} \prod_{\boldsymbol{j}_A\in\mathcal{I}_A} p(\boldsymbol{j}_A)^{n(\boldsymbol{j}_A)},$$

and $\sum_{\boldsymbol{i}_A\in\mathcal{I}_A} n_A(\boldsymbol{i}_A) = n$.

Similarly as for Gaussians, we are interested in testing conditional independencies of the form $X_u \perp\!\!\!\perp X_w | X_v$ for $u \neq v \neq w$. We start by discussing global independences of the form $X_u \perp\!\!\!\perp X_w$, and later extend the discussion to testing the aforementioned conditional independence. For now, we assume that $u \neq w$, otherwise, testing independence is trivial. We lighten our notation by writing $n_{u,w}(i_u, i_w) := n_{\{u,w\}}((i_u, i_w))$ (and so on).

Ferguson (1996) (Section 8, Theorem 8) states the following general asymptotic result, which is applicable to any kind of distribution with finite fourth moments $\mathbb{E}[X_w^4]$ for all $w \in V$, and hence to binary random vectors. It states that

$$\sqrt{n}(\hat{\rho}_{u,w} - \rho_{u,w}) \xrightarrow{\text{d}} \mathcal{N}(0, \gamma^2).$$

Here,

$$\hat{\rho}_{u,w} = \frac{\sum_{k=1}^n (X_u^{(k)} - \overline{X}_u)(X_w^{(k)} - \overline{X}_w)}{\sqrt{\sum_{k=1}^n (X_u^{(k)} - \overline{X}_u)^2}\sqrt{\sum_{k=1}^n (X_w^{(k)} - \overline{X}_w)^2}}$$

and $\overline{X}_w = \frac{1}{n}\sum_{k=1}^n X_w^{(k)}$. For the precise form of γ^2 we refer to ibid., as the expression is rather long. Conceptually, γ^2 depends on the fourth and cross-second moments of X_u and X_w. In practice, γ^2 and hence these moments need to be estimated; and estimating these moments comes with large standard errors as ibid. notes. For the binary setting, this may be not very problematic, as the k-th moment is simply the first moment, but in general this is not the case. Moreover,

this approach seems not to be standard in the literature for testing these types of independence, hence we do not further discuss it.

A classical approach to test $X_u \perp\!\!\!\perp X_w$ is the χ^2-statistic (see Rice (2007), Sections 9.4, 13.4). For a given distribution p and a given sample size n we define

$$\chi^2(p_{u,w}) := \sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} \frac{n(p_{u,w}(i_u,i_w) - p_u(i_u)p_w(i_w))^2}{p_u(i_u)p_w(i_w)}$$

Usually, one estimates $p_{u,w}$ by the maximum likelihood estimator, which is given by $\hat{p}_{u,w} := N_{u,w}(i_u,i_w)/n$. The chi-square statistic is then given by

$$\begin{aligned}\chi^2(\hat{p}_{u,w}) &:= \sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} \frac{n(\hat{p}_{u,w}(i_u,i_w) - \hat{p}_u(i_u)\hat{p}_w(i_w))^2}{\hat{p}_u(i_u)\hat{p}_w(i_w)} \\ &= \sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} \frac{(N_{u,w}(i_u,i_w) - N_u(i_u)N_w(i_w)/n)^2}{N_u(i_u)N_w(i_w)/n}.\end{aligned}$$

Asymptotically, the χ^2-statistic is chi-square distributed with 1 degree of freedom in the binary case (and $(|d_u|-1)(|d_w|-1)$ degrees of freedom in the general discrete case) if $X_u \perp\!\!\!\perp X_w$ is indeed true. Therefore, we can construct an asymptotic level α test, i.e. we reject $X_u \perp\!\!\!\perp X_w$ at some level $\alpha \in (0,1)$ if $\chi^2(\hat{p}_{u,w}) > (\chi^2)^{-1}_{1-\alpha,1}$. Here, $(\chi^2)^{-1}_{1-\alpha,q}$ denotes the $1-\alpha$-quantile of a chi-square distribution with q degrees of freedom.

A related approach to the χ^2-statistic is the G^2-statistic (see again Rice (2007), Sections 9.4, 13.4 and Neapolitan (2004), Section 10.3.1). For a given distribution p and a given sample size n we define

$$G^2(p_{u,w}) := 2 \sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} np_{u,w}(i_u,i_w) \ln\left(\frac{p_{u,w}(i_u,i_w)}{p_u(i_u)p_w(i_w)}\right).$$

The G^2-statistic is then given by

$$\begin{aligned}G^2(\hat{p}_{u,w}) &:= 2 \sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} n\hat{p}_{u,w}(i_u,i_w) \ln\left(\frac{\hat{p}_{u,w}(i_u,i_w)}{\hat{p}_u(i_u)\hat{p}_w(i_w)}\right) \\ &= 2 \sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} N_{u,w}(i_u,i_w) \ln\left(\frac{N_{u,w}(i_u,i_w)n}{N_u(i_u)N_w(i_w)}\right).\end{aligned}$$

We use the convention that $0\ln(0/0) = 0$ and $0\ln(0/\text{positive term}) = 0$ (c.f. Cover and Thomas (2006), Section 2.3). The G^2-statistic is equal to the generalized likelihood ratio test statistic for testing $X_u \perp\!\!\!\perp X_w$ (see Section 9.4 in Rice (2007)). To see this, we define the likelihood ratio by

$$\Lambda_{u,w} := \frac{\max_{q_{u,w}\in\Omega_0}(\text{lik}(q_{u,w}))}{\max_{q_{u,w}\in\Omega}(\text{lik}(q_{u,w}))}.$$

Here, Ω contains all probability distributions $q_{u,w}$ on $\mathcal{I}_{u,w}$, and Ω_0 contains all probability distributions on $\mathcal{I}_{u,w}$ such that $q_{u,w}(i_u, i_w) = q_u(i_u)q_w(i_w)$ for all $(i_u, i_w) \in \mathcal{I}_{u,w}$. Furthermore, $\text{lik}(q_{u,w})$ denotes the likelihood function. Here, we parametrize $q_{u,w}$ by itself. The denominator is maximized by the maximum likelihood estimator $\hat{p}_{u,w}(i_u, i_w)$. The numerator is maximized by $\hat{p}_{0;u,w} := \hat{p}_u(i_u)\hat{p}_w(i_w)$, where $\hat{p}_u$ and $\hat{p}_w$ are the marginals of $\hat{p}_{u,w}$ (see Exercise 10 of Chapter 13 in Rice (2007)). Therefore,

$$\begin{aligned}
\Lambda_{u,w} &= \frac{\frac{n!}{\prod_{(i_u,i_w)\in\mathcal{I}_{u,w}} N_{u,w}(i_u,i_w)!}\prod_{(i_u,i_w)\in\mathcal{I}_{u,w}} \hat{p}_{0;u,w}(i_u, i_w)^{N(i_u,i_w)}}{\frac{n!}{\prod_{(i_u,i_w)\in\mathcal{I}_{u,w}} N_{u,w}(i_u,i_w)!}\prod_{(i_u,i_w)\in\mathcal{I}_{u,w}} \hat{p}_{u,w}(i_u, i_w)^{N_{u,w}(i_u,i_w)}} \\
&= \prod_{(i_u,i_w)\in\mathcal{I}_{u,w}} \left(\frac{\hat{p}_{0;u,w}(i_u, i_w)}{\hat{p}_{u,w}(i_u, i_w)}\right)^{N_{u,w}(i_u,i_w)} \\
&= \prod_{(i_u,i_w)\in\mathcal{I}_{u,w}} \left(\frac{\hat{p}_u(i_u)\hat{p}_w(i_w)}{\hat{p}_{u,w}(i_u, i_w)}\right)^{N_{u,w}(i_u,i_w)}.
\end{aligned}$$

Applying the logarithm and multiplying both sides by -2 yields the likelihood ratio test statistic, which is

$$\begin{aligned}
-2\ln(\Lambda_{u,w}) &= 2\sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} N_{u,w}(i_u, i_w)\ln\left(\frac{\hat{p}_{u,w}(i_u, i_w)}{\hat{p}_u(i_u)\hat{p}_w(i_w)}\right) \\
&= 2\sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} N_{u,w}(i_u, i_w)\ln\left(\frac{N_{u,w}(i_u, i_w)n}{N_u(i_u)N_w(i_w)}\right).
\end{aligned}$$

It can be shown that the G^2-statistic (under some technical assumptions) is asymptotically chi-square distributed with 1 degree of freedom in the binary case (and $(|d_u|-1)(|d_w|-1)$ in the general discrete case) if $X_u \perp\!\!\!\perp X_w$ is true. Therefore, we can again construct an asymptotic level α test, i.e. we reject $X_u \perp\!\!\!\perp X_w$ at some level $\alpha \in (0,1)$ if $G^2(\hat{p}_{u,w}(i_u, i_w)) > (\chi^2)^{-1}_{1-\alpha,1}$. Pearson's chi-square statistic and the G^2-statistic are very similar, and are even asymptotically equivalent under the null hypothesis $X_u \perp\!\!\!\perp X_w$. However, due to the connection of the G^2-statistic with the likelihood ratio test, we prefer that one. We remark that there are other

types of tests, e.g. Fisher's exact tests or certain rank tests. However, there are also problems with some of these. For example, Fisher's exact test collides with our sampling scheme, because it assumes fixed row and column totals. For further discussions of these tests and some recommendations we point the reader to Sokal and Rohlf (1995), Sections 17.4 and 17.5.

We now define the G^2-statistic for testing the conditional independence $X_u \perp\!\!\!\perp X_w | X_v$ for $u \neq v \neq w$. We again assume that $u \neq w$ as testing conditional is otherwise trivial. In equation (2.1) and Lemma 2.2 we have seen that $X_u \perp\!\!\!\perp X_w | X_v$ is equivalent to the fact that the conditional densities factorize, i.e. $p_{u,w|X_v=i_v}(i_u, i_w|i_v) = p_{u|X_v=i_v}(i_u|i_v)p_{w|X_v=i_v}(i_w|i_v)$, for all $(i_u, i_w, i_v) \in \mathcal{I}_{u,w,v}$. This gives rise to define

$$\begin{aligned}
G^2(p_{u,w,v}; p_v) &:= \sum_{i_v \in \mathcal{I}_v} p_v(i_v) G^2(p_{u,w|X_v=i_v}) \\
&= 2n \sum_{i_v \in \mathcal{I}_v} p_v(i_v) \sum_{(i_u,i_w) \in \mathcal{I}_{u,w}} p_{u,w|X_v=i_v}(i_u, i_w|i_v) \\
&\quad \ln\left(\frac{p_{u,w|X_v=i_v}(i_u, i_w|i_v)}{p_{u|X_v=i_v}(i_u|i_v) p_{w|X_v=i_v}(i_w|i_v)} \right) \\
&= 2n \sum_{i_v \in \mathcal{I}_v} \sum_{(i_u,i_w) \in \mathcal{I}_{u,w}} p_{u,w,v}(i_u, i_w, i_v) \ln\left(\frac{p_{u,w,v}(i_u, i_w, i_v) p_v(i_v)}{p_{u,v}(i_u, i_v) p_{w,v}(i_w, i_v)} \right)
\end{aligned}$$

and the G^2-statistic for conditional independence (see e.g. Neapolitan (2004), Section 10.3.1, Sokal and Rohlf (1995), Section 17.5) by

$$\begin{aligned}
G^2(\hat{p}_{u,w,v}; \hat{p}_v) &:= \sum_{i_v \in \mathcal{I}_v} \hat{p}_v(i_v) G^2(\hat{p}_{u,w|X_v=i_v}) \\
&= 2n \sum_{i_v \in \mathcal{I}_v} \hat{p}_v(i_v) \sum_{(i_u,i_w) \in \mathcal{I}_{u,w}} \hat{p}_{u,w|X_v=i_v}(i_u, i_w|i_v) \\
&\quad \ln\left(\frac{\hat{p}_{u,w|X_v=i_v}(i_u, i_w|i_v)}{\hat{p}_{u|X_v=i_v}(i_u|i_v) \hat{p}_{w|X_v=i_v}(i_w|i_v)} \right) \\
&= 2n \sum_{i_v \in \mathcal{I}_v} \sum_{(i_u,i_w) \in \mathcal{I}_{u,w}} \hat{p}_{u,w,v}(i_u, i_w, i_v) \ln\left(\frac{\hat{p}_{u,w,v}(i_u, i_w, i_v) \hat{p}_v(i_v)}{\hat{p}_{u,v}(i_u, i_v) \hat{p}_{w,v}(i_w, i_v)} \right) \\
&= 2 \sum_{i_v \in \mathcal{I}_v} \sum_{(i_u,i_w) \in \mathcal{I}_{u,w}} N_{u,w,v}(i_u, i_w, i_v) \ln\left(\frac{N_{u,w,v}(i_u, i_w, i_v) N_v(i_v)}{N_{u,v}(i_u, i_v) N_{w,v}(i_w, i_v)} \right).
\end{aligned}$$

We may use of any of these expressions, depending on what we need in a particular context. It can be shown that the conditional G^2-statistic is asymptotically chi-square distributed (under some technical assumptions) with 2 degrees of freedom in

the binary case (and $(d_u-1)(d_w-1)d_v$ in the general discrete case) if $X_u \perp\!\!\!\perp X_w|X_v$ is indeed true.

3. Lugosi-Truszkowski-Velona-Zwiernik-algorithm

3.1. Algorithm for Gaussian graphical models

Lugosi et al. (2021) have developed an algorithm that learns the support of a concentration matrix $\boldsymbol{K}$ of a Gaussian random vector $\boldsymbol{X}$. It does so by only querying a few entries of the covariance matrix $\boldsymbol{\Sigma}$. Formally, we may think of a covariance oracle which takes a pair of indices u, v as input and returns the entry $\Sigma_{u,v} = \mathbb{E}[(X_u - \mathbb{E}X_u)(X_v - EX_v)]$ as output. The number of entries of $\boldsymbol{\Sigma}$ that the algorithm queries during one run is the query complexity. Due to symmetry of $\boldsymbol{\Sigma}$, there are at most $|V|(|V|+1)/2$ unique entries which the algorithm can query.

Of course, querying a covariance matrix is an idealiazed scenario, in practice we do not know the entries $\Sigma_{u,v}$; we have to estimate them by some $\hat{\Sigma}_{u,v}$ based on data. We are therefore interested in the query complexity with respect to that estimate.

In this chapter, we will discuss this particular algorithm, which we call the LTVZ-algorithm. The LTVZ-algorithm consists of several parts, later we may refer to these individual parts instead. We will explain these parts and develop sample versions for them in tandem for trees.

The underlying paper is Lugosi et al. (2021). In the following chapter, we refer by default to that particular paper. We have included several theorems and propositions of that paper, however, we do not present the proofs. We only show proofs that do not appear in the original paper. It should be noted that these "new" proofs build upon the old proofs; we usually derive some bound for the difference between the empirical and non-empirical version and then combine this bound with results of Lugosi et al. (2021).

3.1.1. Finding central vertices

The LTVZ-algorithm is a divide-and-conquer algorithm. It learns the concentration graph of an underlying Gaussian random vector $\boldsymbol{X}$ and hence, the dependence

structure of $\boldsymbol{X}$. To do this, the LTVZ-algorithm cuts out some vertex in the underlying concentration graph $\mathcal{G}(\Sigma)$ and splits the remaining graph into new subgraphs. Each time doing so, it uncovers some part of the original tree structure. Ideally, the cut vertex is a "central vertex" in the original concentration graph, as this corresponds to smaller computational complexity[1]. The "most central" vertex in our context is a centroid. A centroid has the property that each new subtree in the remaining graph has a size of at most $|V|/2$. A non-centroid does not have this property and is less suitable as a good cut vertex. But it turns out later, that nodes close enough to a centroid, nodes that sit at a balanced spot, are good cut vertices as well.

In the following we will formalize the notion of a centroid. Let $G = (V, E)$ be a tree and let $\boldsymbol{\Sigma} \in \mathcal{M}(G)$ be arbitrary but fixed. An example would be an arbitrary covariance matrix $\boldsymbol{\Sigma}$ and the induced graphical model of its concentration graph $\mathcal{G}(\boldsymbol{\Sigma})$. For $v \in V$ let $\mathcal{C}^v$ be the set of connected components in $G \setminus v$ and define

$$c(v) := \frac{1}{|V| - 1} \max_{C \in \mathcal{C}^v} |C|.$$

A centroid v^* miminizes $c(v)$, i.e. $v^* := \arg\min_{v \in V} c(v)$. The factor $1/(|V| - 1)$ in the definition is arbitrary, however, this particular scaling allows for comparison between graphs of different sizes. Each tree has either 1 or 2 centroids (see Harary (1969), Theorem 4.3). Furthermore, if v^* is a centroid, then $c(v^*) \leq 1/2 \cdot |V|/(|V| - 1)$.

Figure 3.1 shows two graphs, one with one centroid and the other with two centroids. Note how the intuitive idea of centrality corresponds to the concept of a centroid.

[1] "Computational complexity" is our generic expression for both time and query complexity. If we care about the distinction between time and query complexity, we make that distinction.

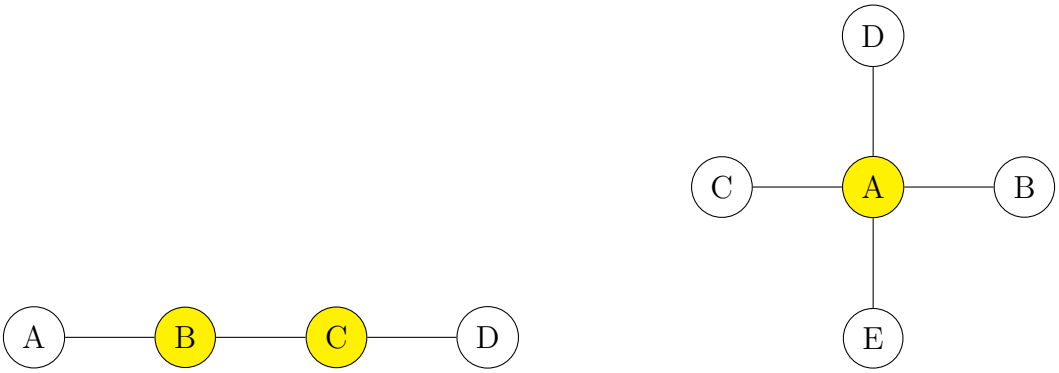


Figure 3.1.: Left: B and C are the only centroids. Both cut the graph into two subtrees of size at most 2. Choosing A and D as a cut vertex would result in a subtree of size 3. Right: Only A is a centroid. For each other node, there would be a subtree of size 3.

The LTVZ-algorithm does not work with $c(v)$ directly. Instead of calculating $c(v)$, a surrogate function $s(v)$ efficiently approximates it. The surrogate is also called s-centrality and defined by

$$s(v) := \frac{1}{(|V|-1)^2} \sum_{C \in \mathcal{C}^v} |C|^2.$$

S-centrality behaves similar as $c(v)$, but there are slight differences. Let v^o denote the minimizer of $s(v)$, i.e. $v^o = \arg\min_{v \in V} s(v)$. Then there may be more than two minimizers v^o. It may even be the case that a centroid is not a minimizer of $s(v)$. Figure 3.2 shows "problematic" examples. The tree on the left hand side has only one centroid, but there are 3 vertices that minimize $s(v)$. In contrast, the tree on the right hand side has exactly 2 nodes that minimize $s(v)$, but neither of them is a centroid. Examples in this fashion are possible for arbitrary large trees. We could not, however, construct a tree such that more than 3 nodes minimize $s(v)$. We could also not proof that the upper bound is indeed 3.

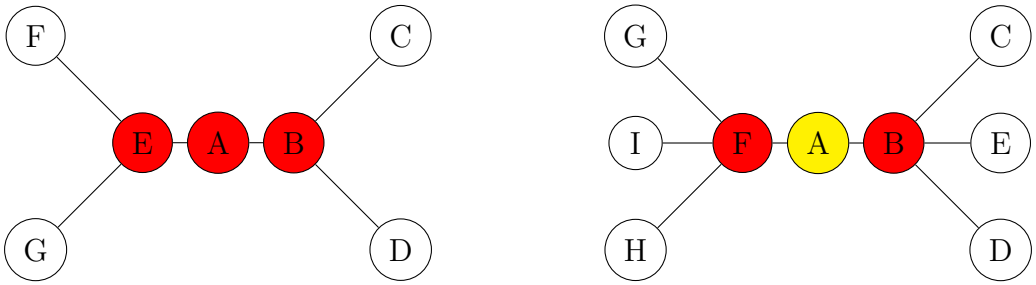


Figure 3.2.: Left: A is the only centroid. But, $s(A) = 1/36 \cdot (3^2 + 3^2) = 18/36 = 1/2$ and $s(B) = s(E) = 1/36 \cdot (4^2 + 1^2 + 1^2) = 18/36 = 1/2$. Hence, A, B and E minimize $s(v)$.
Right: Again, A is the only centroid. We have $s(A) = 1/64 \cdot (4^2 + 4^2) = 32/64 = 1/2$ and $s(B) = s(F) = 1/64 \cdot (3 \cdot 1^2 + 5^2) = 28/64 = 7/16$. Hence, A does not minimize $s(v)$. Note that adding more neighbors to B and F in a symmetric fashion will not make A a minimizer of $s(v)$.

However, these problems do not worry us. Bounding $c(v)$ with the help of $s(v)$ is helpful enough, and this can be achieved as the following lemma shows.

Lemma 3.1. *For every vertex $v \in V$, $s(v) \leq c(v) \leq \sqrt{s(v)}$. In addition, $s(v^o) \leq c(v^*)$.*

The algorithm `sCentralDeterministic` calculates $s(v)$ for each $v \in V$ iteratively. It does the following. For each vertex $v \in V$ and each set of vertices $\{u, w\} \in V \setminus \{v\}$, it checks whether v separates u and w in the graph G. Checking $\det(\mathbf{\Sigma}_{uv,vw}) = 0$ is equivalent by Corollary 2.10 or Lemma 2.11. If v does not separate u and w, `sCentralDeterministic` adds $1/\kappa$ to the current value of $s(v)$. Here, $\kappa := (|V| - 1)^2/2$. Loosely speaking, the more often v separates some pair of nodes, the less often $s(v)$ is increased by $1/\kappa$.

Algorithm 1: sCentralDeterministic(V)

```
κ := (|V|−1)²/2;
s(v) := 0;
for all v ∈ V do
    for all {u, w} ⊆ V \ {v} do
        if det(Σ_{uv,vw}) ≠ 0 then
            s(v) := s(v) + 1/κ;
        end
    end
end
return arg min_v ŝ(v)
```

Before we continue, we want to make a small remark. The case $u = w$ occurs as well. But $v \neq u$ and therefore, X_v is not linearly dependent of X_u since $\boldsymbol{X}$ is multivariate Gaussian. In case of no linear dependence, the Cauchy-Schwarz inequality is strict (see e.g. Klenke (2020), Theorem 5.8) and yields $\Sigma_{u,v}^2 < \Sigma_{u,u}\Sigma_{v,v}$, which implies $\det(\boldsymbol{\Sigma}_{uv,vu}) \neq 0$.

The reason why this procedure exactly calculates $s(v)$ is as follows. For each $v \in V$ there are $(|V|-1)^2/2$ subsets $\{u, w\}$ of $V \setminus \{v\}$. By definition, v does not separate u and w if and only if $u, w \in C$ for some $C \in \mathcal{C}^v$. For each $C \in \mathcal{C}^v$ there are $|C|^2/2$ subsets. Thus, the number of times `sCentralDeterministic` increases $\hat{s}(v)$ by $1/\kappa$ is exactly $\sum_{C \in C^v} |C|^2/2$, i.e. the total number of subsets with size at most 2 for each connected component in $\mathcal{C}^v$. Then, instead of applying the algorithm `sCentralDeterministic`, we can directly calculate

$$\begin{aligned} s(v)_{\text{sCentralDeterministic}} &= \frac{1}{\kappa} \sum_{C \in C^v} \frac{|C|^2}{2} \\ &= \frac{2}{\big(|V|-1\big)^2} \sum_{C \in C^v} \frac{|C|^2}{2} \\ &= \frac{1}{\big(|V|-1\big)^2} \sum_{C \in C^v} |C|^2 \\ &= s(v). \end{aligned}$$

Here, we called the output of `sCentralDeterministic` $s(v)_{\text{sCentralDeterministic}}$.

Iterating over all $(|V|-1)^2/2$ requires querying each entry of $\boldsymbol{\Sigma}$ and is also not very efficient timewise. That is the reason why the authors suggest a randomized procedure, called `sCentral`. Instead of iterating over all pairs, u and w are uniformly picked at random in $V \setminus \{v\}$. This is done κ-times for each vertex $v \in V$, whereas κ has become a tuning parameter. The actual algorithm `sCentral` then looks as follows.

Algorithm 2: sCentral(V)

```
Parameter κ;
ŝ(v) := 0;
for all v ∈ V do
    for i = 1, ..., κ do
        Pick u, w uniformly at random in V \ {v};
        if det(Σ_{uv,vw}) ≠ 0 then
            ŝ(v) := ŝ(v) + 1/κ;
        end
    end
end
return arg min_v ŝ(v)
```

Of course, $\hat{s}(v)$ in `sCentral` only approximates $s(v)$. But as the u, w are picked uniformly at random, we can at least say that $\kappa\hat{s}(v) \sim \text{Bin}(\kappa, s(v))$.

For the sample version we consider the following setup. Let $n \in \mathbb{N}$ be the sample size. Moreover, let $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$ be i.i.d. and have the same distribution as $\boldsymbol{X}$ and let $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ be a realization of $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$. We call the sample-analog of algorithm 2 `sCentralSample`. To derive it, we need to turn the abstract condition $\det(\boldsymbol{\Sigma}_{uv,vw}) = 0$, or equivalently $X_u \perp\!\!\!\perp X_w | X_v$, into a condition based on data. We do this via hypothesis testing. In total, there are $|V|\kappa$ hypotheses in one call of `sCentralSample` of the form

$$H_{0,(v,i)} : X_u \perp\!\!\!\perp X_w | X_v \text{ vs. } H_{1,(v,i)} : \text{ not } H_{0,(v,i)}.$$

The index (v, i) indicates the $v \in V$ in the outer loop and the $i \in \{1, \ldots, \kappa\}$ in the inner loop of `sCentralSample`.

We have discussed the background of these conditional independence tests in Section 2.5. For each (v, i), we calculate $\hat{\rho}_{u,w|v}$ by Yule's and Kendall's recursion formula and its Fisher z-transform $\hat{z}_{u,w|v}$. We then reject $H_{0,(v,i)}$ if

$$\sqrt{n-4}|\hat{z}_{u,w|v}| > \Phi^{-1}\left(1 - \frac{\alpha_{v,i}}{2}\right).$$

Here, $\alpha_{v,i} \in (0, 1)$ is some significance level. We discuss proper choices in the following parts.

Algorithm 3: sCentralSample(V)

```
Parameter κ;
ŝ_sample(v) := 0;
for all v ∈ V do
    for i = 1, ..., κ do
        Pick u, w uniformly at random in V \ {v};
        Test H_{0,(v,i)} at some level α_{v,i} ∈ (0, 1);
        if H_{0,(v,i)} is rejected then
            ŝ_sample(v) := ŝ_sample(v) + 1/κ;
        end
    end
end
return arg min_v ŝ_sample(v)
```

Note that each null hypotheses is random, it depends on the randomly sampled $u, w \in V \setminus \{v\}$. To ease the flow of argumentation, we write $\mathcal{H}_{0,v}$ for the hypothetical set of all possible null hypotheses $H_{0,(v,(u,w))}$.[2] With this,

$$\mathcal{H}_{0,v} = \bigcup_{u,w \in V \setminus \{v\}} \{H_{0,(v,(u,w))}\}$$

and

$$|\mathcal{H}_{0,v}| = \big(|V| - 1\big)^2.$$

For fixed data, we store the result of a hypothesis test $H_{0,(v,(u,w))}$ in $R_{0,(v,(u,w))}$, i.e.

$$R_{0,(v,(u,w))} \in \{\text{Reject } H_{0,(v,(u,w))}, \text{ Not Reject } H_{0,(v,(u,w))}\},$$

whereas each null hypothesis is tested with respect to some level $\alpha_{v,(u,w)} \in (0, 1)$. Overall,

$$\mathcal{R}_{0,v} = \bigcup_{u,w \in V \setminus \{v\}} \{R_{0,(v,(u,w))}\}$$

denotes the set of all hypothetical results. We write $N_{1,v}$ for the number of type 1 errors in $\mathcal{R}_{0,v}$, similarly, $N_{2,v}$ stands for type 2 errors. Furthermore, let $n_{1,v}$ denote the number of type 1 errors after running `sCentralSample` for a particular $v \in V$ and testing each null hypotheses at level $\alpha_{v,i}$. Similarly, let $n_{2,v}$ stand for type 2 errors.

[2] The index $(v, (u, w))$ indicates the null hypothesis $X_u \perp\!\!\!\perp X_w | X_v$.

Note that $N_{1,v}$ and $N_{2,v}$ are random quantities in the sense that they depend on data. In the same sense $n_{1,v}$ and $n_{2,v}$ are random. Even more holds. If we fix $N_{1,v}$ and $N_{2,v}$, $n_{1,v}$ and $n_{2,v}$ are still random. Namely,

$$n_{1,v}|N_{1,v} = j_1 \sim \text{Bin}\left(\kappa, \frac{j_2}{\frac{(|V|-1)^2}{2}}\right)$$

and

$$n_{2,v}|N_{2,v} = j_2 \sim \text{Bin}\left(\kappa, \frac{j_2}{\frac{(|V|-1)^2}{2}}\right),$$

for $0 \leq j_1, j_2 \leq (|V|-1)^2/2$. Neglecting computational issues, we can rewrite `sCentralSample`.

Algorithm 4: sCentralSample(V): Rewritten

```
Parameter κ;
ŝ_sample(v) := 0;
for all v ∈ V do
    for all u, w ∈ V \ {v} do
        Test H_{0,(v,(u,w))} at some level α_{v,(u,w)} and store the result in
          R_{0,(v,(u,w))}.
    end
    for i = 1, ..., κ do
        Pick uniformly some H_{0,(v,i)} := H_{0,(v,(u,w))} from H_{0,v};
        if H_{0,(v,i)} is rejected then
            ŝ_sample(v) := ŝ_sample(v) + 1/κ;
        end
    end
end
return arg min_v ŝ_sample(v)
```

This conceptual evolution is helpful when analyzing differences between `sCentral` and `sCentralSample`. Of course, this is not how `sCentralSample` is run in practice, the original version is less complex. But the rewritten version gives us a good framework to think about the original version of `sCentralSample`.

Both `sCentralSample` and `sCentral` return vertices that minimize $\hat{s}_{\text{sample}}(v)$ resp. $\hat{s}(v)$. These vertices can be related to the minimizer of $s(v)$ and finally $c(v)$. In the paper of Lugosi et al. (2021), this relation has already been established

for the non-sample case. To obtain similar results for the sample case, we need that $\hat{s}_{\text{sample}}(v)$ is not too far away from $\hat{s}(v)$, assuming that κ is the same for both algorithms and in each iteration both algorithms work with the same u, w. Under this assumption, we can relate $\hat{s}_{\text{sample}}(v)$, $\hat{s}(v)$ and the number of type 1 and type 2 errors. Each type 1 error with respect to $H_{0,(v,i)}$ wrongly increases $\hat{s}_{\text{sample}}(v)$ by $1/\kappa$. Each type 2 error with respect to $H_{0,(v,i)}$ wrongly decreases $\hat{s}_{\text{sample}}(v)$ by $1/\kappa$. In detail,

$$\hat{s}_{\text{sample}}(v) = \hat{s}(v) + \frac{1}{\kappa}(n_{1,v} - n_{2,v}). \tag{3.1}$$

In the following parts, we look on equation (3.1) from two different angles. Our first point of view is from a multiple testing perspective. We will show that the familywise error rate can be related to results on centrality. For the second point of view, we test each hypothesis in `sCentralSample` at the same significance level, and relate results from error propagation to centrality again.

The multiple testing approach was motivated "a-priori". That means, we had specific error bounds on centrality in mind and incorporated approaches from multiple testing directly into `sCentralSample` to achieve these bounds; prior to running the algorithm. In that sense, the multiple testing approach is also very constructive, as it allows to relate error bounds to arbitrary familywise error rates.

Approach 2 was motivated "a-posteriori". We think that testing each hypothesis at the same level is a natural thing to do and is not primarily motivated by error bounds which one has in mind prior to running the algorithm.[3]

Both approaches should be read rather independently from each other, even though some results are transferable in either direction.

Perspective 1: Multiple testing

We start with the multiple testing approach. In later propositions and proofs, the fact that $\hat{s}_{\text{sample}}(v)$ is not too far away from $\hat{s}(v)$ is expressed by

$$\limsup_{n\to\infty} \text{Prob}\left(\exists v \in V : \ \hat{s}_{\text{sample}}(v) - \hat{s}(v) > \frac{\delta}{2}\right). \tag{3.2}$$

The notion $\text{Prob}(\cdot)$ denotes any probability calculation under the regime $\boldsymbol{X}^{(i)} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ for $i = 1, \ldots, n$ and u, w are drawn uniformly at random in each step of `sCentralSample`. The limes superior is due to the asymptotic nature of the hypotheses tests.

[3]The PC-algorithm is a classical example where this approach is used. See e.g Spirtes et al. (2000) or Kalisch and Bühlmann (2007).

We aim to bound the term in (3.2) either uniformly for all $\delta > 0$ at once or for some fixed $\delta > 0$. A uniform bound can be achieved by deriving an upper bound on

$$\limsup_{n\to\infty} \text{Prob}\Big(\exists v \in V : \ \hat{s}_{\text{sample}}(v) > \hat{s}(v)\Big), \tag{3.3}$$

because

$$\begin{aligned}\limsup_{n\to\infty} \text{Prob}\Big(\exists v \in V : \ \hat{s}_{\text{sample}}(v) - \hat{s}(v) > \frac{\delta}{2}\Big) \\ \leq \limsup_{n\to\infty} \text{Prob}\Big(\exists v \in V : \ \hat{s}_{\text{sample}}(v) > \hat{s}(v)\Big)\end{aligned}$$

for all $\delta > 0$.
Let $I_{0,v}$ contain all the indices $i \in \{1, \ldots, \kappa\}$ such that $H_{0,(v,i)}$ is true. We can upper bound (3.3) by

$$\begin{aligned}&\limsup_{n\to\infty} \text{Prob}\Big(\exists v \in V : \ \hat{s}_{\text{sample}}(v) > \hat{s}(v)\Big) \\ &\quad \leq \limsup_{n\to\infty} \text{Prob}\Big(\exists v \in V, \ i \in I_{0,v} : \ H_{0,(v,i)} \text{ is rejected}\Big) \\ &\quad \leq |V| \max_v \limsup_{n\to\infty} \text{Prob}\Big(\exists i \in I_{0,v} : \ H_{0,(v,i)} \text{ is rejected}\Big) \\ &\quad =: |V| \max_v \limsup_{n\to\infty} \text{FWER}_v. \end{aligned} \tag{3.4}$$

Here, FWER_v denotes the (asymptotic) familywise error rate (FWER) corresponding to the iteration of a particular v.

There are several multiple testing procedures which control the FWER; we refer to Shaffer (1995), Section 4.2 of Hochberg and Tamhane (1987) and Drton and Perlman (2007). One can usually distinguish between procedures based on the marginal distributions of the test statistics and procedures based on the joint distribution of the test statistics. Furthermore, one can distinguish between single-stage and multi-stage procedures.

We will only focus procedures based on the marginal distributions of the test statistics. Procedures based on the joint distribution have the challenge that the joint distribution of the test statistics depend on the uniformly sampled null hypotheses in each iteration. It may even be the case that one null hypothesis is drawn more than once, which leads to degeneracies in the joint distribution. We were not able to make this joint approach work.

3. Lugosi-Truszkowski-Velona-Zwiernik-algorithm

For approaches based on the marginal distributions, we look at the Bonferroni procedure (single-stage) and Holm's stepdown method (multi-stage), originally due to Fisher (1935) and Holm (1979). We start with outlining Bonferroni. For each v there are κ hypotheses tests. If we test each hypothesis at level $\frac{\alpha}{\kappa|V|}$, we achieve that

$$|V| \max_v \limsup_{n\to\infty} \mathrm{FWER}_v \leq |V| \max_v \kappa \frac{\alpha}{\kappa|V|} = \alpha.$$

By inequality (3.4), we obtain $\limsup_{n\to\infty} \mathrm{Prob}(\exists v \in V : \ \hat{s}_{\mathrm{sample}}(v) > \hat{s}(v)) \leq \alpha$. The advantage of Bonferroni is that it neatly integrates with the structure of `sCentralSample`, as no comparison between different hypotheses is involved. Therefore, we can simply test each hypothesis at level $\frac{\alpha}{\kappa|V|}$.

Algorithm 5: sCentralSampleBonferroni(V)

```
Parameter κ;
ŝ_sample(v) := 0;
for all v ∈ V do
    for i = 1, ..., κ do
        Pick u, w uniformly at random in V \ {v};
        Test H_{0,(v,i)} at level α/(κ|V|);
        if H_{0,(v,i)} is rejected then
            ŝ_sample(v) := ŝ_sample(v) + 1/κ;
        end
    end
end
return arg min_v ŝ_sample(v)
```

For the stepdown method of Holm, we firstly calculate all p-values $\hat{p}_{v,i}$ for a fixed $v \in V$. Then, we order the p-values and find the smallest $k_v \in \{1, \ldots, \kappa\}$ such that $\hat{p}_{(k_v)} > \alpha/(|V|(\kappa + 1 - k_v))$. Having done that, we reject all hypotheses corresponding to the p-values $\hat{p}_{(1)}, \ldots, \hat{p}_{(k_v-1)}$. Including this procedure in `sCentralSample` requires some additional iterations in comparison to the Bonferroni approach. But on the other hand, we achieve higher overall power.

Algorithm 6: sCentralSampleHolm(V)

Parameter κ;
$\hat{s}_{\text{sample}}(v) := 0$;
for *all* $v \in V$ **do**
  for $i = 1, \ldots, \kappa$ **do**
    Pick u, w uniformly at random in $V \setminus \{v\}$;
    Calculate p-values $\hat{p}_{v,i}$ corresponding to null hypotheses $H_{0,(v,i)}$;
  end
  Order p-values $\hat{p}_{v,i}$ and find smallest $k_v \in [\kappa]$ such that $\hat{p}_{(k_v)} > \frac{\alpha/|V|}{\kappa+1-k_v}$;
  Reject all $H_{0,(v,i)}$ corresponding to the p-values $\hat{p}_{(1)}, \ldots, \hat{p}_{(k_v-1)}$;
  for $j = 1, \ldots, \kappa$ **do**
    if $H_{0,(v,i)}$ *is rejected* **then**
      $\hat{s}_{\text{sample}}(v) := \hat{s}_{\text{sample}}(v) + \frac{1}{\kappa}$;
    end
  end
end
return $\arg\min_v \hat{s}_{sample}(v)$

A bound on the probability in (3.2) can also be derived for a fixed $\delta > 0$. This is then a non-uniform bound. To derive this bound, we turn to the gFWER, the generalization of the FWER. The gFWER(k) is the probability of at least k type 1 errors, where k is to be specified. Starting again from (3.2), we have

$$\begin{aligned}
&\limsup_{n\to\infty} \text{Prob}\left(\exists v \in V : \ \hat{s}_{\text{sample}}(v) - \hat{s}(v) > \frac{\delta}{2}\right) \\
&\qquad \leq \limsup_{n\to\infty} \text{Prob}\left(\exists v \in V, n_{1,v} > \frac{\delta\kappa}{2}\right) \\
&\qquad \leq |V| \max_v \limsup_{n\to\infty} \text{Prob}\left(n_{1,v} > \frac{\delta\kappa}{2}\right) \\
&\qquad = |V| \max_v \limsup_{n\to\infty} \text{Prob}\left(n_{1,v} \geq \left\lfloor \frac{\delta\kappa}{2} \right\rfloor + 1\right) \\
&\qquad =: |V| \max_v \limsup_{n\to\infty} \text{gFWER}_v\left(\left\lfloor \frac{\delta\kappa}{2} \right\rfloor + 1\right).
\end{aligned}$$

Lehmann and Romano (2005) have extended the usual Bonferroni and Holm procedure to achieve control over the gFWER (see also Dudoit et al. (2004) for a discussion). To generalize the Bonferroni approach, we can simply include $\lfloor \delta\kappa/2 \rfloor + 1$

as a factor and test each hypothesis at level

$$\left(\left\lfloor \frac{\delta\kappa}{2} \right\rfloor + 1\right)\frac{\alpha}{\kappa|V|}$$

to achieve $\limsup_{n\to\infty} \text{Prob}(\exists v \in V : \ \hat{s}_{\text{sample}}(v) - \hat{s}(v) > \frac{\delta}{2}) \leq \alpha$. We can improve on this by using a generalization of Holm's method. For that, let

$$\alpha_j := \begin{cases} \frac{\lfloor \frac{\delta\kappa}{2} \rfloor + 1}{\kappa} \frac{\alpha}{|V|} & \text{if } j \leq \left(\lfloor \frac{\delta\kappa}{2} \rfloor + 1\right) \\ \frac{\lfloor \frac{\delta\kappa}{2} \rfloor + 1}{\kappa + \lfloor \frac{\delta\kappa}{2} \rfloor + 1 - j} \frac{\alpha}{|V|} & \text{if } j > \left(\lfloor \frac{\delta\kappa}{2} \rfloor + 1\right) \end{cases}.$$

Note that we have suppressed the dependence of α_j on κ and δ in the notation. Now, let k_v be the smallest integer such that

$$\hat{p}_{(k_v)} > \alpha_{k_v}.$$

Then, we reject all $H_{0,(v,i)}$ corresponding to the p-values $\hat{p}_{(1)}, \ldots, \hat{p}_{(k_v-1)}$.

In this thesis we will not focus too much on the gFWER algorithms. First of all, it requires to actively set the parameter δ, which has only some interpretation in a soon-to-follow proposition. Secondly, the gFWER algorithms are the same as the FWER algorithms for small enough δ. We view these extensions rather as technical tidbits.

From now on when we talk about `sCentralSample`, we talk about the general version neglecting the precise multiple testing procedure.

For `sCentral`, Lugosi et al. (2021) show that the vertex $\hat{v}$ returned from `sCentral`, is "central enough". The result is even true for general undirected graphs and therefore, we will state the result in its more general form.

Proposition 3.2. *Let $G = (V, E)$ be an undirected graph and let $\mathbf{\Sigma} \in \mathcal{M}(G)$. The time and query complexity of computing $\hat{v} =$ `sCentral`(V) are $\mathcal{O}(|V|\kappa)$. Moreover, for any $\delta > 0$,*

$$\text{Prob}\Big(s(\hat{v}) \geq s(v^o) + 2\delta\Big) \leq 2|V| \exp\big(-2\delta^2\kappa\big).$$

Here, Prob$(\cdot)$ *stands for any probability calculation under the regime that u, w are sampled uniformly at random at each iteration of* `sCentral`.

This proposition states that $\hat{v}$ is not too bad in comparison to v^o in terms of s-centrality. We can derive similar results for the sample versions.

Proposition 3.3. *The time and query complexity of* `sCentralSampleBonferroni` *(or its gFWER generalization) are*

$$\mathcal{O}(|V|\kappa).$$

The query complexity of `sCentralSampleHolm` *(or its gFWER generalization) is*

$$\mathcal{O}(|V|\kappa),$$

whereas the time complexity is

$$\mathcal{O}(|V|\kappa \ln(\kappa)).$$

Proof. We calculate $\hat{z}_{u,w|v}$ using Yule and Kendall's recursion formula (see equation (2.3)). To do this, we need to query $\hat{\Sigma}_{uw}, \hat{\Sigma}_{uv}, \hat{\Sigma}_{vw}, \hat{\Sigma}_{uu}, \hat{\Sigma}_{ww}, \hat{\Sigma}_{vv}$, so at most 6 entries. Therefore, for both Bonferroni and Holm, the query complexity is $\mathcal{O}(|V|\kappa)$. In `sCentralSampleBonferroni` the time complexity of a single hypothesis test does not depend on $|V|$ or κ. There are at most $|V|\kappa$ hypotheses tests, thus the time complexity is $\mathcal{O}(|V|\kappa)$.

In `sCentralSampleHolm`, κ p-values need to be sorted for each $v \in V$. Sorting algorithms like mergesort or heapsort take $\mathcal{O}(\kappa \ln(\kappa))$ time to do this (c.f. Cormen et al. (2009), Part II). Finding the smallest $k_v \in [\kappa]$ such that $\hat{p}_{(k_v)} > \alpha/(|V|(\kappa+1-k_v))$, has time complexity $\mathcal{O}(\kappa)$. Therefore, for a fixed $v \in V$, the time complexity is $\mathcal{O}(\kappa \ln(\kappa))$. Thus, the overall time complexity is $\mathcal{O}(|V|\kappa \ln(\kappa))$.

The proof for gFWER generalizations is very similar, we omit it. □

The probabilistic part of Proposition 3.2 has a sample analog as well.

Proposition 3.4. *Let $G = (V, E)$ be a graph and let $\mathbf{\Sigma} \in \mathcal{M}(G)$. Let $\boldsymbol{X} \sim \mathcal{N}(0, \mathbf{\Sigma})$ and let $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$ be i.i.d. random vectors with the same distribution as $\boldsymbol{X}$ and let*

$$\overline{\boldsymbol{X}} := \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{X}^{(i)}.$$

Let $\hat{\mathbf{\Sigma}}$ be the estimator of $\mathbf{\Sigma}$ given by

$$\hat{\mathbf{\Sigma}} := \frac{1}{n-1} \sum_{i=1}^{n} \boldsymbol{X}^{(i)} \boldsymbol{X}^{(i)^T}$$

and let $\tilde{v} =$ `sCentralSample`(V). *Assume that each hypothesis in* `sCentralSample` *(e.g. by the Bonferroni or Holm version) is tested such that*

$$|V| \max_v \limsup_{n\to\infty} \mathrm{FWER}_v \leq \alpha.$$

Then, for any $\delta > 0$,

$$\limsup_{n\to\infty} \mathrm{Prob}\Big(s(\tilde{v}) \geq s(v^o) + 2\delta \Big) \leq \alpha + 2|V| \exp\Big(-\frac{\delta^2}{2}\kappa \Big).$$

Here, $\mathrm{Prob}(\cdot)$ *stands for any probability calculation under the regime* $\boldsymbol{X}^{(i)} \overset{i.i.d.}{\sim} \mathcal{N}(0, \boldsymbol{\Sigma})$ *for all* $i = 1, \ldots, n$ *and that* u, w *are sampled uniformly at random in each iteration of* `sCentralSample`.

Remark 3.5. The bound on `sCentralSample` is bigger than that on `sCentral`. We have an additional term α, and even the exponential term $\exp(-\delta^2/(2\kappa))$ decreases slower in δ resp. κ.

Proof. We have

$$\begin{aligned}
&\mathrm{Prob}\Big(\max_v |\hat{s}_{\mathrm{sample}}(v) - s(v)| \geq \delta \Big) \\
&\quad \leq \mathrm{Prob}\Big(\max_v \big\{ |\hat{s}_{\mathrm{sample}}(v) - \hat{s}(v)| + |\hat{s}(v) - s(v)| \big\} \geq \delta \Big) \\
&\quad \leq \mathrm{Prob}\Big(\max_v \big\{ |\hat{s}_{\mathrm{sample}}(v) - \hat{s}(v)| \big\} + \max_v \big\{ |\hat{s}(v) - s(v)| \big\} \geq \delta \Big) \\
&\quad \leq \mathrm{Prob}\Big(\max_v |\hat{s}_{\mathrm{sample}}(v) - \hat{s}(v)| \geq \frac{\delta}{2} \text{ or } \max_v |\hat{s}(v) - s(v)| \geq \frac{\delta}{2} \Big) \\
&\quad \leq \mathrm{Prob}\Big(\max_v |\hat{s}_{\mathrm{sample}}(v) - \hat{s}(v)| \geq \frac{\delta}{2} \Big) + \mathrm{Prob}\Big(\max_v |\hat{s}(v) - s(v)| \geq \frac{\delta}{2} \Big) \\
&\quad \leq \mathrm{Prob}\Big(\max_v \big\{ \hat{s}_{\mathrm{sample}}(v) - \hat{s}(v) \big\} \geq \frac{\delta}{2} \text{ or } \max_v \big\{ \hat{s}(v) - \hat{s}_{\mathrm{sample}}(v) \big\} \geq \frac{\delta}{2} \Big) \\
&\qquad + \mathrm{Prob}\Big(\max_v |\hat{s}(v) - s(v)| \geq \frac{\delta}{2} \Big) \\
&\quad \leq \mathrm{Prob}\Big(\max_v \big\{ \hat{s}_{\mathrm{sample}}(v) - \hat{s}(v) \big\} \geq \frac{\delta}{2} \Big) \\
&\qquad + \mathrm{Prob}\Big(\max_v \big\{ \hat{s}(v) - \hat{s}_{\mathrm{sample}}(v) \big\} \geq \frac{\delta}{2} \Big) \\
&\qquad + \mathrm{Prob}\Big(\max_v |\hat{s}(v) - s(v)| \geq \frac{\delta}{2} \Big).
\end{aligned}$$

Due to subadditivity, applying the limes superior on both sides yields

$$\begin{aligned}
&\limsup_{n\to\infty} \mathrm{Prob}\Big(\max_v |\hat{s}_{\mathrm{sample}}(v) - s(v)| \geq \delta \Big) \\
&\qquad \leq \limsup_{n\to\infty} \mathrm{Prob}\Big(\max_v \big\{ \hat{s}_{\mathrm{sample}}(v) - \hat{s}(v) \big\} \geq \frac{\delta}{2} \Big) \\
&\qquad\quad + \limsup_{n\to\infty} \mathrm{Prob}\Big(\max_v \big\{ \hat{s}(v) - \hat{s}_{\mathrm{sample}}(v) \big\} \geq \frac{\delta}{2} \Big) \\
&\qquad\quad + \limsup_{n\to\infty} \mathrm{Prob}\Big(\max_v |\hat{s}(v) - s(v)| \geq \frac{\delta}{2} \Big).
\end{aligned}$$

We need to upper-bound each of the tree terms.
Let us start with $\limsup_{n\to\infty} \mathrm{Prob}(\max_v |\hat{s}(v) - s(v)| \geq \frac{\delta}{2})$. Note that this term does not depend on n, so we drop the limes superior. Now, we can use an argument already used by Lugosi et al. (2021) for the proof of Proposition 3.2. Recall that for every $v \in V$, $\kappa\hat{s}(v)$ is a binomial random variable with mean $\kappa s(v)$. Hence, by Hoeffding's inequality, originally due to Hoeffding (1963), and the union bound,

$$\mathrm{Prob}\Big(\max_v |\hat{s}(v) - s(v)| \geq \frac{\delta}{2} \Big) \leq 2|V| \exp\Big(-\frac{\delta^2}{2}\kappa \Big).$$

To bound $\limsup_{n\to\infty} \mathrm{Prob}(\max_v \{\hat{s}_{\mathrm{sample}}(v) - \hat{s}(v)\} \geq \frac{\delta}{2})$, note that

$$\begin{aligned}
&\limsup_{n\to\infty} \mathrm{Prob}\Big(\max_v \big\{ \hat{s}_{\mathrm{sample}}(v) - \hat{s}(v) \big\} \geq \frac{\delta}{2} \Big) \\
&\qquad = \limsup_{n\to\infty} \mathrm{Prob}\Big(\exists v \in V :\ \hat{s}_{\mathrm{sample}}(v) - \hat{s}(v) \geq \frac{\delta}{2} \Big) \\
&\qquad \leq \limsup_{n\to\infty} \mathrm{Prob}\Big(\exists v \in V :\ \hat{s}_{\mathrm{sample}}(v) - \hat{s}(v) > 0 \Big) \\
&\qquad = \limsup_{n\to\infty} \mathrm{Prob}\Big(\exists v \in V :\ \hat{s}_{\mathrm{sample}}(v) > \hat{s}(v) \Big) \\
&\qquad \leq \alpha
\end{aligned}$$

by assumption on the multiple testing procedure. Lastly, to upper bound $\mathrm{Prob}(\max_v \{\hat{s}(v) -$

$\hat{s}_{\text{sample}}(v)\} \geq \frac{\delta}{2}$), note that

$$\begin{aligned}
&\limsup_{n\to\infty} \text{Prob}\left(\max_v \{\hat{s}(v) - \hat{s}_{\text{sample}}(v)\} \geq \frac{\delta}{2} \right) \\
&\quad = \limsup_{n\to\infty} \text{Prob}\left(\exists v \in V : \ \hat{s}(v) - \hat{s}_{\text{sample}}(v) \geq \frac{\delta}{2} \right) \\
&\quad = \limsup_{n\to\infty} \text{Prob}\left(\exists v \in V : \ \frac{1}{\kappa}(n_{2,v} - n_{1,v}) \geq \frac{\delta}{2} \right) \\
&\quad \leq \limsup_{n\to\infty} \text{Prob}\left(\exists v \in V : \ n_{2,v} \geq \frac{\kappa\delta}{2} \right) \\
&\quad \leq |V| \max_v \limsup_{n\to\infty} \text{Prob}\left(n_{2,v} \geq \frac{\kappa\delta}{2} \right).
\end{aligned}$$

For an arbitrary fixed $v \in V$, $N_{2,v} = 0 \Rightarrow n_{2,v} = 0$. Thus,

$$\limsup_{n\to\infty} \text{Prob}\left(n_{2,v} \geq \frac{\kappa\delta}{2} \right) \leq \limsup_{n\to\infty} \text{Prob}\left(n_{2,v} > 0 \right) \leq \limsup_{n\to\infty} \text{Prob}\left(N_{2,v} > 0 \right).$$

We want to show that $\limsup_{n\to\infty} \text{Prob}(N_{2,v} > 0) = 0$. To do this, we will first of all show that $\hat{\mathbf{\Sigma}}$ converges entrywise to $\mathbf{\Sigma}$ almost surely. The proof of this fact is a standard argument using the Strong Law of Large Numbers (SLLN), see e.g. Klenke (2020), Section 5.3. Note that for $v, u \in V$, both $X_v^{(1)}, \ldots, X_v^{(n)}$ and $X_u^{(1)}, \ldots, X_u^{(n)}$ are i.i.d. respectively. Hence, the products $X_v^{(1)}X_u^{(1)}, \ldots, X_v^{(n)}X_u^{(n)}$ are i.i.d. as well. Furthermore,

$$\mathbb{E}\big[\big|X_v^{(1)}X_u^{(1)}\big|\big] = \mathbb{E}\big[\big|X_v^{(1)}\big|\big|X_u^{(1)}\big|\big] \overset{\text{Cauchy-Schwarz}}{\leq} \sqrt{\mathbb{E}\big[{X_v^{(1)}}^2\big]}\sqrt{\mathbb{E}\big[{X_u^{(1)}}^2\big]} < \infty,$$

as each $X_v^{(i)}$ and each $X_u^{(i)}$ is univariate Gaussian. Therefore, the assumptions of the SLLN are satisfied. Thus, for each $v, u \in V$,

$$\frac{1}{n}\sum_{i=1}^{n} X_v^{(i)}X_u^{(i)} \xrightarrow{\text{a.s.}} \mathbb{E}\Big[X_v^{(i)}X_u^{(i)}\Big].$$

Also, $\frac{n}{n-1} \longrightarrow 1$, so by the Continuous Mapping Theorem (see van der Vaart (1998), Theorem 2.3),

$$\hat{\Sigma}_{u,v} = \frac{1}{n-1}\sum_{i=1}^{n} X_v^{(i)}X_u^{(i)} = \frac{n}{n-1}\left(\frac{1}{n}\sum_{i=1}^{n} X_v^{(i)}X_u^{(i)}\right) \xrightarrow{\text{a.s.}} \Sigma_{u,v}.$$

Using the recursion formula of Kendall and Yule, the fact, that Fisher's z-transform is continuous and the Continuous Mapping Theorem again, shows that for each $u, w, v \in V$ with $u \neq v \neq w$,

$$\hat{z}_{u,w|v} \xrightarrow{\text{a.s.}} z_{u,w|v}.$$

Recall that we do not reject the null hypothesis, i.e. that $\rho_{u,w|v} = 0$, if

$$|\hat{z}_{u,w|v}| \leq \frac{1}{\sqrt{n-4}} \Phi^{-1}\left(1 - \frac{\alpha_{v,(u,w)}}{2}\right).$$

for $\alpha_{v,(u,w)} \in (0,1)$ constant in n. If $z_{u,w|v} \neq 0$ (equivalent to $\rho_{u,w|v} \neq 0$), then almost surely

$$\begin{aligned}
\lim_{n\to\infty} |\hat{z}_{u,w|v}| &\overset{\text{Cont. Map. Theo.}}{=} |\lim_{n\to\infty} \hat{z}_{u,w|v}| \\
&= |z_{u,w|v}| \\
&> 0 \\
&= \lim_{n\to\infty} \frac{1}{\sqrt{n-4}} \Phi^{-1}\left(1 - \frac{\alpha_{v,(u,w)}}{2}\right).
\end{aligned}$$

Therefore asymptotically, we almost surely always reject if the null hypothesis is false. Thus, as almost sure convergence implies convergence in probability, $\limsup_{n\to\infty} \text{Prob}(N_{2,v} > 0) = 0$.

We finish by noting that

$$\begin{aligned}
&\limsup_{n\to\infty} \text{Prob}\Big(s(\tilde{v}) \geq s(v^o) + 2\delta \Big) \\
&\quad = \limsup_{n\to\infty} \text{Prob}\Big(s(\tilde{v}) - \hat{s}_{\text{sample}}(\tilde{v}) + \hat{s}_{\text{sample}}(\tilde{v}) - s(v^o) \geq 2\delta \Big) \\
&\quad \leq \limsup_{n\to\infty} \text{Prob}\Big(s(\tilde{v}) - \hat{s}_{\text{sample}}(\tilde{v}) + \hat{s}_{\text{sample}}(v^o) - s(v^o) \geq 2\delta \Big) \\
&\quad \leq \limsup_{n\to\infty} \text{Prob}\Big(\max_v |\hat{s}_{\text{sample}}(v) - s(v)| \geq \delta \Big) \\
&\quad \leq \alpha + 2|V| \exp\Big(-\frac{\delta^2}{2}\kappa \Big).
\end{aligned}$$

□

Remark 3.6. We actually do not need the assumption that $\boldsymbol{X}$ is centered. The Strong Law of Large numbers in the proof is still applicable in the non-centered case and the statement of the theorem would be true as well.

Remark 3.7. Note that there is a hidden trade-off with respect to α. Reducing α increases the probability of a type 2 error for finite n. Said differently, $\text{Prob}(n_{2,v} > 0)$ resp. $\text{Prob}(N_{2,v} > 0)$ converges slower to 0 for smaller α. We give an approximate bound on $\text{Prob}(N_{2,v} > 0)$ to study the dependence on α and n in more detail. For

this, we say $\hat{z}_{u,w|v} \sim \mathcal{N}(z_{u,w|v}, 1/(n-4))$ for large enough n. Of course, this is only approximate. We will later discuss exact arguments as well, however, these do not work for arbitrary levels of α.

Let $I_{0,v}$ denote the set of true hypotheses for a fixed $v \in V$. Then,

$$\begin{aligned}
&\mathrm{Prob}\Big(N_{2,v} > 0\Big) \\
&\qquad = \mathrm{Prob}\Big(N_{2,v} \geq 1\Big) \\
&\qquad = \mathrm{Prob}\Big(\exists u, w \in V \setminus \{v\} : \ H_{0,(v,(u,w))} \text{ is false but not rejected } \Big) \\
&\qquad \leq |I_{0,v}^C| \cdot \max_{u,w \in V\setminus\{v\} \text{ and } H_{0,(v,(u,w))} \text{ is false}} \mathrm{Prob}\Big(H_{0,(v,(u,w))} \text{ is not rejected } \Big) \\
&\qquad \leq \frac{\big(|V|-1\big)^2}{2} \max_{u,w \in V\setminus\{v\} \text{ and } H_{0,(v,(u,w))} \text{ is false}} \mathrm{Prob}\Big(H_{0,(v,(u,w))} \text{ is not rejected } \Big).
\end{aligned}$$

Here, we adopt the convention that

$$\max_{u,w \in V\setminus\{v\} \text{ and } H_{0,(v,(u,w))} \text{ is false}} \mathrm{Prob}\Big(H_{0,(v,(u,w))} \text{ is rejected } \Big) = 0,$$

if there is no $u, w \in V \setminus \{v\}$ such that $H_{0,(v,(u,w))}$ is false.

Now, suppose there is at least one false hypothesis. Then, there are some v, u, w such that $H_{0,(v,(u,w))}$ is false. In particular, we have, writing $q_{v,(u,w)} := \Phi^{-1}(1 -$

$\alpha_{v,(u,w)}/2)$,

$$\begin{aligned}
&\text{Prob}\Big(H_{0,(v,(u,w))} \text{ is not rejected } \Big| z_{u,w|v\neq 0}\Big) \\
&\quad = \text{Prob}\Big(\hat{z}_{u,w|v} \leq \frac{q_{v,(u,w)}}{\sqrt{n-4}}\Big| z_{u,w|v\neq 0}\Big) \\
&\quad \overset{\text{for large enough } n}{=} \int_{-\frac{q_{v,(u,w)}}{\sqrt{n-4}}}^{\frac{q_{v,(u,w)}}{\sqrt{n-4}}} \phi_{0,\frac{1}{n-4}}(x)dx \\
&\quad = \int_{-\frac{q_{v,(u,w)}}{\sqrt{n-4}}}^{\frac{q_{v,(u,w)}}{\sqrt{n-4}}} \frac{\sqrt{n-4}}{\sqrt{2\pi}} \exp\Big(-\frac{(n-4)(x-z_{u,w|v})^2}{2}\Big)dx \\
&\quad = \int_{-\frac{q_{v,(u,w)}}{\sqrt{n-4}}-z_{u,w|v}}^{\frac{q_{v,(u,w)}}{\sqrt{n-4}}-z_{u,w|v}} \frac{\sqrt{n-4}}{\sqrt{2\pi}} \exp\Big(-\frac{(n-4)x^2}{2}\Big)dx \\
&\quad = \int_{-\frac{q_{v,(u,w)}}{\sqrt{n-4}}-z_{u,w|v}}^{\frac{q_{v,(u,w)}}{\sqrt{n-4}}-z_{u,w|v}} \frac{\sqrt{n-4}}{\sqrt{2\pi}} \exp\Big(-\frac{\big(x\sqrt{n-4}\big)^2}{2}\Big)dx \\
&\quad = \int_{-q_{v,(u,w)}-z_{u,w|v}\sqrt{n-4}}^{q_{v,(u,w)}-z_{u,w|v}\sqrt{n-4}} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{x^2}{2}\Big)dx.
\end{aligned}$$

If $z_{u,w|v} > 0$, then

$$\begin{aligned}
&\int_{-q_{v,(u,w)}-z_{u,w|v}\sqrt{n-4}}^{q_{v,(u,w)}-z_{u,w|v}\sqrt{n-4}} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{x^2}{2}\Big)dx \\
&\quad \leq \int_{-\infty}^{q_{v,(u,w)}-z_{u,w|v}\sqrt{n-4}} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{x^2}{2}\Big)dx \\
&\quad \overset{\text{by symmetry}}{=} \int_{-q_{v,(u,w)}+z_{u,w|v}\sqrt{n-4}}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{x^2}{2}\Big)dx \\
&\quad \overset{\text{for } n \text{ large enough}}{\leq} \exp\Big(-\frac{\big(-q_{v,(u,w)}+z_{u,w|v}\sqrt{n-4}\big)^2}{2}\Big) \\
&\quad = \exp\Big(-\frac{\big(-q_{v,(u,w)}+|z_{u,w|v}|\sqrt{n-4}\big)^2}{2}\Big).
\end{aligned}$$

In the last inequality we used the Chernoff bound (c.f. Hoeffding (1963)) and the fact that $-q_{v,(u,w)} + z_{u,w|v}\sqrt{n-4} \geq 0$ for n large enough.

We proceed similarly for the case $z_{u,w|v} < 0$, i.e.

$$\begin{aligned}
&\int_{-q_{v,(u,w)}-z_{u,w|v}\sqrt{n-4}}^{q_{v,(u,w)}-z_{u,w|v}\sqrt{n-4}} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{x^2}{2}\Big)dx \\
&\qquad \leq \int_{-q_{v,(u,w)}-z_{u,w|v}\sqrt{n-4}}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{x^2}{2}\Big)dx \\
&\qquad \overset{\text{for } n \text{ large enough}}{\leq} \exp\Bigg(-\frac{\big(-q_{v,(u,w)}-z_{u,w|v}\sqrt{n-4}\big)^2}{2}\Bigg) \\
&\qquad = \exp\Bigg(-\frac{\big(-q_{v,(u,w)}+|z_{u,w|v}|\sqrt{n-4}\big)^2}{2}\Bigg).
\end{aligned}$$

Again, we used that $-q_{v,(u,w)} - z_{u,w|v}\sqrt{n-4} \geq 0$ for n large enough. So in both cases , i.e. $z_{u,w|v} \neq 0$ (and assuming n is overall large enough) we have

$$\begin{aligned}
\int_{-q_{v,(u,w)}-z_{u,w|v}\sqrt{n-4}}^{q_{v,(u,w)}-z_{u,w|v}\sqrt{n-4}} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{x^2}{2}\Big)dx &\leq \exp\Bigg(-\frac{\big(-q_{v,(u,w)}+|z_{u,w|v}|\sqrt{n-4}\big)^2}{2}\Bigg) \\
&\leq \exp\Bigg(-\frac{\big(-q_{min}+z_{min}\sqrt{n-4}\big)^2}{2}\Bigg),
\end{aligned}$$

whereas $z_{min} := \min_{v\in V} \min_{u,w\in V\setminus\{v\},\, z_{u,w|v}\neq 0} |z_{u,w|v}|$ and
$q_{min} := \min_{v\in V} \min_{u,w\in V\setminus\{v\},\, z_{u,w|v}\neq 0} q_{v,(u,w)}$.
Therefore,

$$\text{Prob}\Big(N_{2,v} > 0\Big) \leq \frac{(|V|-1)^2}{2} \exp\Bigg(-\frac{\big(-q_{min}+z_{min}\sqrt{n-4}\big)^2}{2}\Bigg)$$

approximately. At this point, we observe that q_{min} is inversely related to α in our testing procedures. Decreasing α increases q_{min}. Hence the bound is bigger for smaller α.[4]
Furthermore, we see the importance of the true partial correlations $\rho_{u,w|v}$. The larger the lower bound on the nonzero absolute partial correlations and hence by monotonicity on Fisher's z-transform, the smaller the upper bound.

With this and for large enough n, the statement of Proposition 3.4 reads ap-

[4] As long as q_{min} is still less than $z_{min}\sqrt{n-4}$, because if that is not the case, the bound does not hold anymore.

proximately as follows: For any $\delta > 0$,

$$\begin{aligned}&\mathrm{Prob}\Big(s(\tilde{v}) \geq s(v^o) + 2\delta\Big)\\ &\qquad \leq \alpha + 2|V| \exp\Big(-\frac{\delta^2}{2}\kappa\Big) + \frac{(|V|-1)^2}{2}\exp\Big(-\frac{\big(-q_{min} + z_{min}\sqrt{n-4}\big)^2}{2}\Big).\end{aligned}$$

Remark 3.8. One may try to tighten the bound in remark 3.7 further. One approach is to directly derive a bound on

$$\mathrm{Prob}\Big(n_{2,v} \geq \frac{\kappa\delta}{2}\Big).$$

We have

$$\begin{aligned}\mathrm{Prob}\Big(n_{2,v} \geq \frac{\kappa\delta}{2}\Big) &= \sum_{j=1}^{\frac{(|V|-1)^2}{2}} \mathrm{Prob}\Big(n_{2,v} \geq \frac{\kappa\delta}{2}\Big| N_{2,v} = j\Big)\,\mathrm{Prob}\Big(N_{2,v} = j\Big)\\ &= \sum_{j=1}^{\frac{(|V|-1)^2}{2}} \sum_{l=0}^{\kappa} \binom{\kappa}{l} \Big(\frac{j}{\frac{(|V|-1)^2}{2}}\Big)^l \Big(1 - \frac{j}{\frac{(|V|-1)^2}{2}}\Big)^{\kappa-l} \mathrm{Prob}\Big(N_{2,v} = j\Big).\end{aligned}$$

However, continuing from here is not trivial. We have not further pursued it.

Remark 3.9. The result is true for the Bonferroni and Holm procedure. But as Holm's procedure leads to less type 2 errors, $\mathrm{Prob}(n_{2,v} > 0)$ resp. $\mathrm{Prob}(N_{2,v} > 0)$ is potentially lower.

Remark 3.10. There is also a hidden tradeoff with respect to κ. Increasing κ decreases the exponential bound on the one hand, but on the other hand, it also increases the total number of hypotheses tests and hence forces us to reduce the significance level for an individual hypothesis, which reduces power. Recall that in the Bonferroni version each hypothesis is tested at level $\alpha/(\kappa V)$.

A weaker version of Proposition 3.4 is also true, if we only control (3.2) for some fixed $\delta > 0$.

Proposition 3.11. *Start with the same assumptions as in Proposition 3.4. But this time, assume that each hypothesis in* `sCentralSample` *(e.g. by the gFWER-generalizations of Bonferroni or Holm) is tested such that*

$$|V| \max_v \limsup_{n\to\infty} \mathrm{gFWER}_v\Big(\Big\lfloor \frac{\delta\kappa}{2} \Big\rfloor + 1\Big) \leq \alpha.$$

for some $\delta > 0$. Then, for that particular δ,

$$\limsup_{n\to\infty} \mathrm{Prob}\bigg(s(\tilde{v}) \geq s(v^o) + 2\delta\bigg) \leq \alpha + 2|V| \exp\bigg(-\frac{\delta^2}{2}\kappa\bigg).$$

Here, $\mathrm{Prob}(\cdot)$ *stands for any probability calculation under the regime* $\boldsymbol{X}^{(i)} \overset{i.i.d.}{\sim} \mathcal{N}(0, \boldsymbol{\Sigma})$ *for all* $i = 1, \ldots, n$ *and that* u, w *are sampled uniformly at random in each iteration of* `sCentralSample`.

Proof. The proof is very similar to the one of Proposition 3.4. We omit it. □

To show that our procedure finds a good cut vertex with high probability, we need the next proposition. A sample version is not needed.

Proposition 3.12. *For any graph $G = (V, E)$, it holds that if $s(v) < s(v^o) + 2\delta$, then*

$$c(v) < \sqrt{s(v^o) + 2\delta} \leq \sqrt{c(v^*) + 2\delta}.$$

In particular, if $G = (V, E)$ is a tree with $|V| \geq 4$, and $\delta < \frac{1}{6}$, then $c(v) < 1$.

Combining Proposition 3.2 and Proposition 3.12 yields the following result.

Proposition 3.13. *If $G = (V, E)$ is a tree, $|V| \geq 4$, and $\hat{v} =$* `sCentral`(V)*, then*

$$\mathrm{Prob}\bigg(c(\hat{v}) > \sqrt{\frac{11}{12}}\bigg) \leq 2|V| \exp\bigg(-\frac{\kappa}{32}\bigg).$$

Here, $\mathrm{Prob}(\cdot)$ *stands for any probability calculation under the regime that u, w are sampled uniformly at random at each iteration of* `sCentral`.

There is an analogous result for `sCentralSample`.

Proposition 3.14. *Fix $\delta = \frac{1}{8}$. With the same assumptions as in Proposition 3.4 or 3.11 and $|V| \geq 4$, we have*

$$\limsup_{n\to\infty} \mathrm{Prob}\bigg(c(\tilde{v}) > \sqrt{\frac{11}{12}}\bigg) \leq \alpha + 2|V| \exp\bigg(-\frac{\kappa}{128}\bigg).$$

Here, $\mathrm{Prob}(\cdot)$ *stands for any probability calculation under the regime* $\boldsymbol{X}^{(i)} \overset{i.i.d.}{\sim} \mathcal{N}(0, \boldsymbol{\Sigma})$ *for all* $i = 1, \ldots, n$ *and that* u, w *are sampled uniformly at random in each iteration of* `sCentralSample`.

Proof. Fix $\delta = \frac{1}{8}$.
By Proposition 3.12,

$$\begin{aligned}\limsup_{n\to\infty} \text{Prob}\Big(s(\tilde{v}) < s(v^o) + 2\delta)\Big) &\leq \limsup_{n\to\infty} \text{Prob}\Big(c(\tilde{v}) \leq \sqrt{c(v^*) + 2\delta}\Big) \\ &= \limsup_{n\to\infty} \text{Prob}\Big(c(\tilde{v}) \leq \sqrt{\frac{11}{12}}\Big).\end{aligned}$$

Therefore,

$$\begin{aligned}\limsup_{n\to\infty} \text{Prob}\Big(c(\tilde{v}) \leq \sqrt{\frac{11}{12}}\Big) &= \limsup_{n\to\infty} \text{Prob}\Big(c(\tilde{v}) \leq \sqrt{c(v^*) + 2\delta}\Big) \\ &\leq \limsup_{n\to\infty} \text{Prob}\Big(s(\tilde{v}) < s(v^o) + 2\delta\Big) \\ &\overset{\text{proposition 3.4 or 3.11}}{\leq} \alpha + 2|V| \exp\Big(-\frac{\delta^2}{2}\kappa\Big) \\ &= \alpha + 2|V| \exp\Big(-\frac{\kappa}{128}\Big).\end{aligned}$$

□

Perspective 2: Testing each hypothesis at the same significance level α

Here we shine light on another way to think about choosing significance levels. Instead of deriving a multiple testing procedure, we simply test each hypothesis at some fixed level $\alpha \in (0, 1)$. Even though α still controls the probability of a type 1 error for a single hypothesis, it looses its overall interpretability and is rather seen as a simple tuning parameter.

We base our analysis on Kalisch and Bühlmann (2007), in particular Lemma 1 and Lemma 3 therein. Kalisch and Bühlmann themselves base some of their work on Hotelling (1953).

We introduce the following notation and assumptions. Denote the lower bound on the partial correlations $\rho_{u,w|v}$ with $u \neq v \neq w$ by

$$\rho_{min} := \inf\Big\{\big|\rho_{u,w|v}\big| : u, w \in V \setminus \{v\};\ v \in V \text{ and } \rho_{u,w|v} \neq 0\Big\}.$$

Denote the upper bound by M. For the upper bound we have excluded the case $u = w$, as the corresponding partial correlation is always 1. We assume that $M < 1$. We call that assumption (A4)[5].

[5]We follow Kalisch and Bühlmann (2007) with that notation.

Kalisch and Bühlmann (2007) also introduce other assumptions that are required for high-dimensional studies (see Section 3.1 ibid.). We start with a fixed dimension, however, and here the assumptions reduce to the one given above (and Gaussianity).

We now state a very helpful lemma that is based on Lemma 3 ibid.

Lemma 3.15. *Assume (A4). Then for any* $\gamma > 0$,

$$\sup_{u,w \in V\setminus\{v\};\, v \in V} \text{Prob}\Big(\big|\hat{z}_{u,w|v} - z_{u,w|v}\big| > \gamma\Big) \leq \mathcal{O}(n-1)\bigg[\exp\bigg((n-5)\log\bigg(\frac{4-(\gamma/L)^2}{4+(\gamma/L)^2}\bigg)\bigg) + \exp\Big(-C_2(n-1)\Big)\bigg],$$

for some constant $0 < C_2 < \infty$ *and* $L = 1/(1-(1+M)^2/4)$.

Remark 3.16. The larger M, the larger L, and hence the larger the upper bound.

We want to bound the probability that either a type 1 or type 2 error occurs in one call of `sCentralSample`. We start by bounding the supremal probability of a type 1 error with Lemma 3.15. The lemma even allows us to do this for finite n. We fix a particular choice of α, namely

$$\alpha = \alpha_n = 2\cdot\bigg(1 - \Phi\Big(n^{1/2}\cdot\frac{\rho_{min}}{2}\Big)\bigg). \tag{3.5}$$

This choice does not help when choosing the significance level in practice, because it depends on ρ_{min}, which is usually unknown. Nevertheless, we proceed and obtain

$$\begin{aligned}
&\sup_{u,w \in V\setminus\{v\};\, v \in V;\, z_{u,w|v}=0} \text{Prob}\Big(\sqrt{n-4}\big|\hat{z}_{u,w|v}\big| > \Phi^{-1}\Big(1-\frac{\alpha}{2}\Big)\Big) \\
&= \sup_{u,w \in V\setminus\{v\};\, v \in V;\, z_{u,w|v}=0} \text{Prob}\Big(\sqrt{n-4}\big|\hat{z}_{u,w|v} - z_{u,w|v}\big| > \Phi^{-1}\Big(1-\frac{\alpha}{2}\Big)\Big) \\
&= \sup_{u,w \in V\setminus\{v\};\, v \in V;\, z_{u,w|v}=0} \text{Prob}\bigg(\big|\hat{z}_{u,w|v} - z_{u,w|v}\big| > \Big(\frac{n}{n-4}\Big)^{1/2}\frac{\rho_{min}}{2}\bigg).
\end{aligned}$$

We now apply Lemma 3.15. We also use that $\ln((4-x^2)/(4+x^2)) \sim -x^2/2$

(meaning asymptotically equivalent) as $x \to 0$. Together, we then have

$$\begin{aligned}
&\sup_{u,w \in V\setminus\{v\};\, v \in V;\, z_{u,w|v}=0} \text{Prob}\Big(\big|\hat{z}_{u,w|v} - z_{u,w|v}\big| > \Big(\frac{n}{n-4}\Big)^{1/2}\frac{\rho_{min}}{2}\Big) \\
&\quad \leq \mathcal{O}(n-1)\bigg[\exp\Big(-\frac{n(n-5)}{n-4}\cdot\frac{\rho_{min}^2}{8L^2}\Big) + \exp\Big(-C_3(n-1)\Big)\bigg] \\
&\quad \leq \mathcal{O}(n-1)\bigg[\exp\Big(-\Big((n-1)+\frac{4}{n-4}\Big)\cdot\frac{\rho_{min}^2}{8L^2}\Big) + \exp\Big(-C_3(n-1)\Big)\bigg] \\
&\quad \leq \mathcal{O}(n-1)\exp\Big(-C_4(n-1)\rho_{min}^2\Big) \qquad (3.6)
\end{aligned}$$

for some $0 < C_3, C_4 < \infty$. C_4 needs to be smaller for larger M. In this derivation, we see why a sufficiently large decrease of α in n was necessary to obtain this bound. If we had a constant α, then the exponential term would not decrease in n.

We can study a similar supremal probability for type 2 errors, which is given by

$$\begin{aligned}
&\sup_{u,w \in V\setminus\{v\};\, v \in V;\, z_{u,w|v}\neq 0} \text{Prob}\Big(\sqrt{n-4}\big|\hat{z}_{u,w|v}\big| \leq \Phi^{-1}\Big(1-\frac{\alpha}{2}\Big)\Big) \\
&\quad = \sup_{u,w \in V\setminus\{v\};\, v \in V;\, z_{u,w|v}\neq 0} \text{Prob}\Big(\big|\hat{z}_{u,w|v}\big| \leq \Big(\frac{n}{n-4}\Big)^{1/2}\frac{\rho_{min}}{2}\Big).
\end{aligned}$$

Now, we make use of the following implications.

$$\begin{aligned}
\big|\hat{z}_{u,w|v}\big| &\leq \Big(\frac{n}{n-4}\Big)^{1/2}\frac{\rho_{min}}{2} \\
&\implies -\big|\hat{z}_{u,w|v}\big| \geq -\Big(\frac{n}{n-4}\Big)^{1/2}\frac{\rho_{min}}{2} \\
&\implies \rho_{min} - \big|\hat{z}_{u,w|v}\big| \geq \rho_{min} - \Big(\frac{n}{n-4}\Big)^{1/2}\frac{\rho_{min}}{2} \\
&\overset{\text{see below}}{\implies} \big|z_{u,w|v}\big| - \big|\hat{z}_{u,w|v}\big| \geq \rho_{min}\Big(1-\frac{1}{2}\Big(\frac{n}{n-4}\Big)^{1/2}\Big) \\
&\implies \big|z_{u,w|v} - \hat{z}_{u,w|v}\big| \geq \rho_{min}\Big(1-\frac{1}{2}\Big(\frac{n}{n-4}\Big)^{1/2}\Big) \\
&\iff \big|z_{u,w|v} - \hat{z}_{u,w|v}\big| > \rho_{min}\Big(1-\frac{1}{2}\Big(\frac{n}{n-4}\Big)^{1/2}\Big) \quad \text{a.s.}
\end{aligned}$$

see below: Here we used that the lower bound on the partial correlations transfers to the z-transform, because $|z(x)| \geq |x|$.
This derivation also shows why ρ_{min} is appearing in the definition of α, because this was necessary to make these implications. Now, using these implications and Lemma 3.15 yields

$$\begin{aligned}
&\sup_{u,w\in V\setminus\{v\};\, v\in V;\, z_{u,w|v}\neq 0} \text{Prob}\Big(\big|\hat{z}_{u,w|v}\big| \leq \Big(\frac{n}{n-4}\Big)^{1/2}\frac{\rho_{min}}{2}\Big) \\
&\quad\leq \sup_{u,w\in V\setminus\{v\};\, v\in V;\, z_{u,w|v}\neq 0} \text{Prob}\Big(\big|\hat{z}_{u,w|v} - z_{u,w|v}\big| > \rho_{min}\Big(1-\frac{1}{2}\Big(\frac{n}{n-4}\Big)^{1/2}\Big)\Big) \\
&\quad\leq \mathcal{O}(n-1)\Big[\exp\Big(-(n-5)\frac{\big[\rho_{min}\big(1-\frac{1}{2}\big(\frac{n}{(n-4)}\big)^{1/2}\big)\big]^2}{2L^2}\Big) \\
&\qquad + \exp\Big(-C_5(n-1)\Big)\Big] \\
&\quad= \mathcal{O}(n-1)\Big[\exp\Big(-(n-5)\frac{\rho_{min}^2}{2L^2} + \frac{(n-5)\big(\frac{n}{n-4}\big)^{1/2}\rho_{min}^2}{2L^2} - \frac{(n-5)n}{(n-4)}\frac{\rho_{min}^2}{8L^2}\Big) \\
&\qquad + \exp\Big(-C_5(n-1)\Big)\Big] \\
&\quad\leq \mathcal{O}(n-1)\exp\Big(-C_6(n-1)\rho_{min}^2\Big), \qquad (3.7)
\end{aligned}$$

for some $0 < C_5, C_6 < \infty$. C_6 needs to be smaller for larger M.

We can see that for both type 1 and type 2 errors a linear term in $(n-1)$ and an exponential term in $n-1$ are working in opposite directions. But the exponential term will dominate at some point letting both bounds eventually converge to 0.

We also note that a smaller upper bound on the partial correlations M yields a smaller bound on both type 1 and type 2 errors. On the other hand, a larger lower bound ρ_{min} leads to a smaller bound on the respective errors as well. So the lower and upper bound on the partial correlations work in different directions.

We can attribute the occurence in the final bounds of both M and ρ_{min} to different parts in our previous argumentation. The upper bound M is only due to Lemma 3.15 itself; it plays a role in the deviation of the z-transformed sample partial correlation from the true z-transformed partial correlation. The lower bound ρ_{min}, however, only appears due to our choice of α. But we can also not exclude ρ_{min}, because we needed it to derive a bound on type 2 errors.

We can now bound two terms which appeared in the proof of Proposition 3.4,

this time without using the helping hand of asymptotics. Doing so, we obtain a similar result for the second perspective. We have

$$\begin{aligned}
&\text{Prob}\Big(\exists v \in V : \hat{s}_{\text{sample}}(v) > \hat{s}(v)\Big) \\
&\qquad \leq |V|\kappa \sup_{u,w \in V\setminus\{v\};\, v\in V;\, z_{u,w|v}=0} \text{Prob}\Big(\sqrt{n-4}\big|\hat{z}_{u,w|v}\big| > \Phi^{-1}\Big(1-\frac{\alpha}{2}\Big)\Big) \\
&\qquad \leq |V|\kappa\, \mathcal{O}(n-1)\exp\Big(-C_4(n-1)\rho_{min}^2\Big). \qquad (3.8)
\end{aligned}$$

and

$$\begin{aligned}
&\text{Prob}\Big(\exists v \in V : n_{2,v} > 0\Big) \\
&\qquad \leq |V|\kappa \sup_{u,w \in V\setminus\{v\};\, v\in V;\, z_{u,w|v}\neq 0} \text{Prob}\Big(\sqrt{n-4}|\hat{z}_{u,w|v}| \leq \Phi^{-1}\Big(1-\frac{\alpha}{2}\Big)\Big) \\
&\qquad \leq |V|\kappa\, \mathcal{O}(n-1)\exp\Big(-C_6(n-1)\rho_{min}^2\Big). \qquad (3.9)
\end{aligned}$$

With these inequalities, we have adapted the proof of Proposition 3.4 to this second perspective and have thereby proven the following proposition.

Proposition 3.17. *Let $G = (V, E)$ be a graph and let $\mathbf{\Sigma} \in \mathcal{M}(G)$. Let $\boldsymbol{X} \sim \mathcal{N}(0, \mathbf{\Sigma})$ and let $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$ be i.i.d. random vectors with the same distribution as $\boldsymbol{X}$. Furthermore, assume (A4). Then there exists $\alpha \in (0,1)$ with $\alpha \to 0$ for $n \to \infty$ such that if each hypothesis in* `sCentralSample` *is tested at level α, it holds that*

$$\begin{aligned}
&\text{Prob}\Big(s(\tilde{v}) \geq s(v^o) + 2\delta\Big) \\
&\qquad \leq 2|V|\exp\Big(-\frac{\delta^2}{2}\kappa\Big) + |V|\kappa\, \mathcal{O}(n-1)\Big(\exp\Big(-C_4(n-1)\rho_{min}^2\Big) \\
&\qquad\quad + \exp\Big(-C_6(n-1)\rho_{min}^2\Big)\Big) \\
&\qquad \leq 2|V|\exp\Big(-\frac{\delta^2}{2}\kappa\Big) + |V|\kappa\, \mathcal{O}(n-1)\exp\Big(-C(n-1)\rho_{min}^2\Big), \qquad (3.10)
\end{aligned}$$

whereas the constants C_4 and C_6 are defined in (3.6) *and* (3.7) *respectively and $C := \min\{C_4,\, C_6\}$. The constants C_4, C_6 and C need to be smaller for larger M. The particular value of α is given in equation* (3.5)

By a similar argument as well, we have an analog of Proposition 3.14.

Proposition 3.18. *Besides the assumptions from Proposition 3.17 assume that* $|V| \geq 4$. *Then there exists* $\alpha \in (0,1)$ *with* $\alpha \to 0$ *for* $n \to \infty$ *such that if each hypothesis in* `sCentralSample` *is tested at level* α, *it holds that*

$$\begin{aligned}\text{Prob}\bigg(c(\tilde{v}) > \sqrt{\frac{11}{12}}\bigg) &\leq 2|V| \exp\bigg(-\frac{\kappa}{128}\bigg) \\ &\quad + |V|\kappa\, \mathcal{O}(n-1) \exp\bigg(-C(n-1)\rho_{min}^2\bigg),\end{aligned}$$

where C *is as in Proposition 3.17.*

High-dimensional settings

We can extend the previous reasoning to high-dimensional settings. For that, we make the number of vertices dependent on the sample size n and we pay tribute to this in our notation by writing V_n. We assume that $|V_n| = \mathcal{O}(n^a)$ for some $0 \leq a < \infty$. We also assume that κ_n is not too large meaning that $\kappa_n = \mathcal{O}(|V_n|) = \mathcal{O}(n^a)$. Moreover, we assume that κ is not too small. In particular, we assume that $\kappa_n > 2a\ln(n)/\delta^2$ for some $\delta > 0$. Furthermore, we modify assumption (A4) a little bit. We want to have that the minimal partial correlation is not decreasing too fast for increasing sample size, in particular we assume that $\rho_{min} \geq c_n$, whereas $c_n^{-1} = \mathcal{O}(n^d)$ for some $0 < d < 1/2$. We call all of these assumptions (A-N-HD). Therefore we can further examine the bound in (3.8), i.e.

$$\begin{aligned}&\text{Prob}\bigg(\exists v \in V : \hat{s}_{\text{sample}}(v) > \hat{s}(v)\bigg) \\ &\qquad \leq |V_n|\kappa_n\, \mathcal{O}(n-1) \exp\bigg(-C_4(n-1)\rho_{min}^2\bigg) \\ &\qquad \leq |V_n|\kappa_n\, \mathcal{O}(n-1) \exp\bigg(-C_4(n-1)c_n^2\bigg) \\ &\qquad \leq \mathcal{O}\bigg(n^{2a}(n-1)\exp\bigg(-C_4(n-1)n^{2d}\bigg)\bigg) \\ &\qquad \leq \mathcal{O}\bigg(\exp\bigg(2a\ln(n) + \ln(n-1) - C_4(n-1)n^{2d}\bigg)\bigg) \\ &\qquad = \mathcal{O}\bigg(\exp\bigg(2a\ln(n) + \ln(n-1) - C_4(n^{1-2d} - n^{2d})\bigg)\bigg) \\ &\qquad = o(1),\end{aligned}$$

because for $d < 1/2$, the term n^{1-2d} dominates the exponent. Similarly for the

bound in (3.9), we obtain

$$\begin{aligned}
&\text{Prob}\Big(\exists v \in V : n_{2,v} > 0\Big) \\
&\qquad \leq |V_n|\kappa_n \, \mathcal{O}(n-1)\exp\Big(-C_6(n-1)\rho_{min}^2\Big) \\
&\qquad \leq |V_n|\kappa_n \, \mathcal{O}(n-1)\exp\Big(-C_6(n-1)c_n^2\Big) \\
&\qquad \leq \mathcal{O}\Big(n^{2a}(n-1)\exp\Big(-C_6(n-1)n^{2d}\Big)\Big) \\
&\qquad \leq \mathcal{O}\Big(\exp\Big(2a\ln(n)+\ln(n-1)-C_6(n-1)n^{2d}\Big)\Big) \\
&\qquad = \mathcal{O}\Big(\exp\Big(2a\ln(n)+\ln(n-1)-C_6(n^{1-2d}-n^{2d})\Big)\Big) \\
&\qquad = o(1).
\end{aligned}$$

Here, the constants are as introduced in equations (3.6) and (3.7). Moreover, we have

$$\begin{aligned}
\text{Prob}\Big(\max_v |\hat{s}(v)-s(v)| \geq \frac{\delta}{2}\Big) &\leq 2|V_n|\exp\Big(-\frac{\delta^2}{2}\kappa_n\Big) \\
&\leq O\Big(n^a \exp\Big(-\frac{\delta^2}{2}\kappa_n\Big)\Big) \\
&\leq O\Big(\exp\Big(a\ln(n)-\frac{\delta^2}{2}\kappa_n\Big)\Big) \\
&= o(1).
\end{aligned}$$

The exponent converges to $-\infty$ by definition of κ_n, thus, the upper bound converges to 0. Due to the previous inequalities, we get a high-dimensional version of Proposition 3.17 and 3.18.

Proposition 3.19. *Assume* (A-N-HD). *Then there exists* $\alpha \in (0,1)$ *with* $\alpha \to 0$ *for* $n \to \infty$ *such that if each hypothesis in* `sCentralSample` *is tested at level* α, *it holds that*

$$\text{Prob}\Big(s(\tilde{v}) \geq s(v^o) + 2\delta\Big) = o(1).$$

Here, $o(1)$ *refers to a sequence in* n. *The particular value of* α *is given in equation* (3.5).

By a similar argument as well, we have an analog of Proposition 3.14.

Proposition 3.20. *Assume* (A-N-HD) *with* $\delta = \frac{1}{8}$ *and assume that* $|V| \geq 4$. *Then there exists* $\alpha \in (0,1)$ *with* $\alpha \to 0$ *for* $n \to \infty$ *such that if each hypothesis in* `sCentralSample` *is tested at level* α, *it holds that*

$$\mathrm{Prob}\left(c(\tilde{v}) > \sqrt{\frac{11}{12}}\right) = o(1).$$

Here, $o(1)$ refers to a sequence in n. The particular value of α is given in equation (3.5).

3.1.2. Recovering trees

The algorithms `sCentral` resp. `sCentralSample` are part of broader scheme to efficiently reconstruct trees. As we have seen before, `sCentral` resp. `sCentralSample` return a vertex with high probability that has good centrality properties. This vertex is the starting point for the subroutine `ComponentsTree`.

Algorithm 7: ComponentsTree(V)

```
w := sCentral(V);
N := ∅;
Sort |ρ_uw| for u ∈ V \ {w} in decreasing order and put them in list B;
for every u in the order of B do
    t:= true;
    for all v ∈ N do
        if det(Σ_uw,vw) ≠ 0 then
            V_v := V_v ∪ {u};
            t := false;
        end
    end
    if t = true then
        Ê := Ê ∪ {u, w};
        N := N ∪ {u};
        V_u := {u};
    end
end
return all V_u for u ∈ N
```

`ComponentsTree` first picks a vertex $w = \texttt{sCentral}(V)$. Then, it sorts the absolute values of the pairwise correlations $\rho_{u,w} = \Sigma_{u,w}/\sqrt{\Sigma_{u,u}\Sigma_{w,w}}$, where $u \in V \setminus \{w\}$, in descending order. This ordering is stored in an ordered list B. For each $u \in B$, `ComponentsTree` checks whether w separates any of its known neighbors v, and u. A known neighbor is a node that has been identified as a neighbor of w in previous iterations.

If for all known neighbors v the vertex w separates v and u, then `ComponentsTree` adds u as a new known neighbor of w, adds the edge $\{u, w\}$ to $\hat{E}$ which stores all learned edges in the graph and registers a new connected component for u. On the other hand, if there is a known neighbor v such that w does not separate v and u, then `ComponentsTree` adds u to the connected component of that particular known neighbor v.

The algorithm `ReconstructTree` uses `ComponentsTree` recursively. Let $\mathcal{G}(\mathbf{\Sigma})$ be the graph we want to learn. In the first step, `ComponentsTree` partitions V into connected components. Each returned connected component is again partitioned, until the connected components have only one node. Thereby, each call of `ComponentsTree` learns some part of the overall tree structure and stores it in the global variable $\hat{E}$.

Algorithm 8: ReconstructTree(V)

```
if |V| > 1 then
    V_1, ..., V_m ← ComponentsTree(V);
    for i from 1 to m do
        ReconstructTree(V_i);
    end
end
```

Lugosi et al. (2021) show the following proposition.

Proposition 3.21. *Algorithm 8 is correct, that is, if $\mathcal{G}(\mathbf{\Sigma})$ is a tree T, then* `ReconstructTree`(V) *gives $\hat{E} = E(T)$.*

Remark 3.22. In terms of correctness, it does not matter how central the vertex w, which has been returned from `sCentral`, is. We could also choose some endpoint-vertex. But as the example in figure 3.3 shows, centrality is important to reduce complexity.

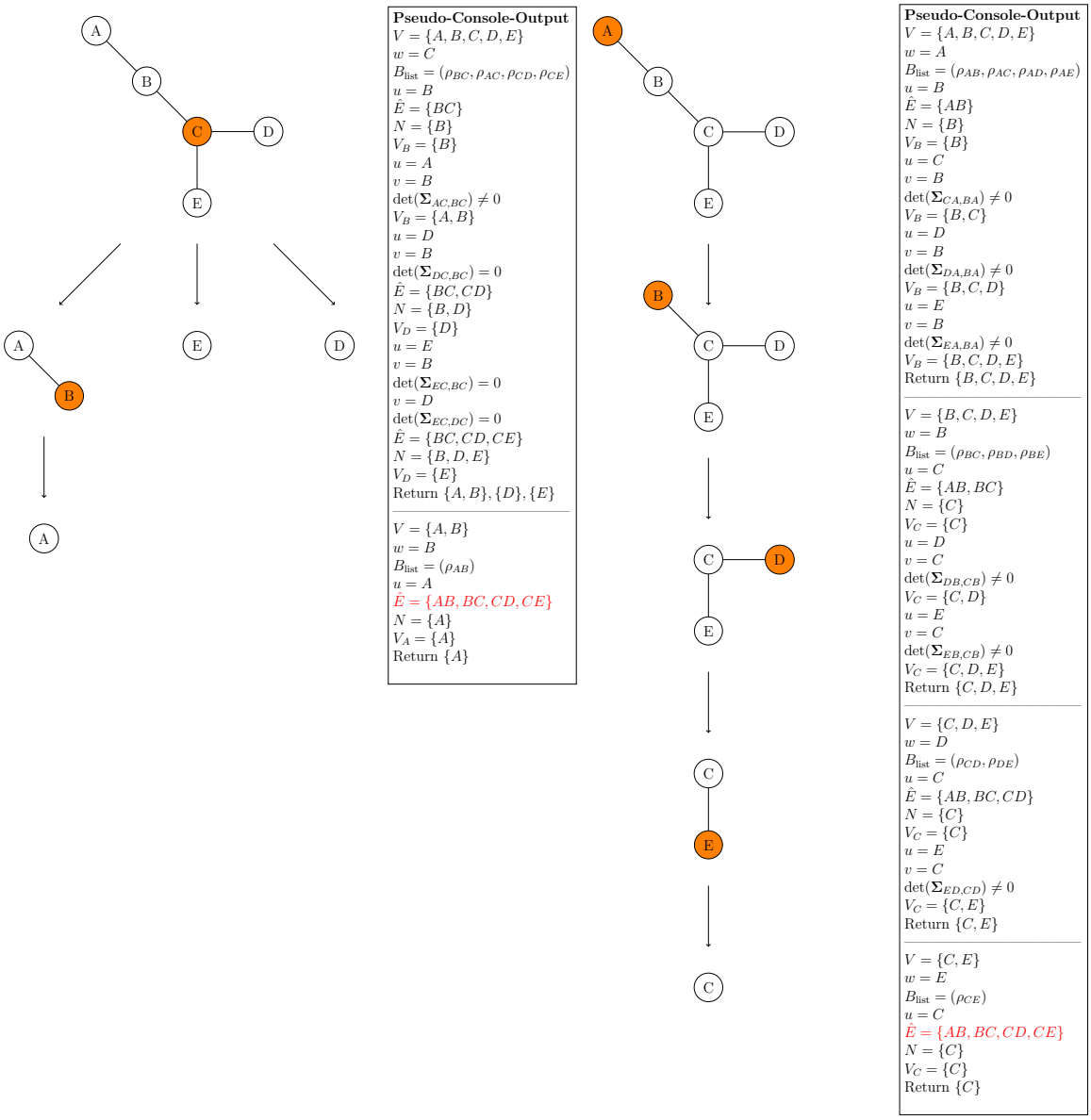


Figure 3.3.: On the left hand side, a centroid is chosen as w in each call of `ComponentsTree`. On the right hand side, a terminal node is chosen. Pseudo-Console-Output illustrates the differences in runtime. In both cases, $\hat{E} = E(T)$, but on the right hand side the algorithm runs longer. In this example, $\rho_{CD} > \rho_{CE}$. With this, the order of each list is specified by Lemma 2.11.

The sample-version of `ComponentsTree` will be called `ComponentsTreeSample`, the sample version of `ReconstructTree` will be called `ReconstructTreeSample`. `ComponentsTreeSample` calls `sCentralSample` with the respective subtree.

To derive a sample version, we need turn the condition $\det(\mathbf{\Sigma}_{uw,vw}) \neq 0$ in `ComponentsTree` into a decision rule based on data. We again do this by hypothesis testing. For that, we test

$$H_{0,(w,u,v)} : X_u \perp\!\!\!\perp X_v | X_w \text{ vs. } H_{1,(w,u,v)} : \text{ not } H_{0,(w,u,v)}.$$

For each (w, u, v), we calculate $\hat{\rho}_{u,v|w}$ by Yule and Kendall's recursion formula and its Fisher z-transform $\hat{z}_{u,v|w}$. We then reject $H_{0,(w,u,v)}$ if

$$\sqrt{n-4}|\hat{z}_{u,v|w}| > \Phi^{-1}\left(1 - \frac{\alpha'_{w,u,v}}{2}\right)$$

for some $\alpha'_{w,u,v} \in (0, 1)$. We discuss a proper choice of $\alpha'_{w,u,v}$ in the following parts. For the remainder of this section, we refer to the hypotheses tests outside `sCentralSample` simply as hypotheses tests, as `sCentralSample` has already been discussed. To indicate to which hypotheses we are referring, we write α to refer to hypotheses inside `sCentralSample` and α' to refer to hypotheses outside `sCentralSample`.

To analyze the correctness of `ComponentsTreeSample`, we compare it with a slightly different version of `ComponentsTree`, where we replace $w = \texttt{sCentral}(V)$ with $w = \texttt{sCentralSample}(V)$. In terms of correctness, this does not change `ComponentsTree`. The computational behavior may worsen of course, but for our analysis of correctness, we do not care about this. With this slight modification, we see that each type of error has severe implications for subsequent parts of `ReconstructTreeSample`. In case of a type 1 error, `ComponentsTreeSample` sets t wrongly to false, wrongly omits some edge from $\hat{E}$, forgets a neighbor, and also comes up with different connected components. Therefore, each subsequent call of `ComponentsTreeSample` works with a wrong subtree. (Type 2 errors have similar effects.)

We again discuss two perspectives on setting meaningful significance levels $\alpha'_{w,u,v}$. The first perspective is again from multiple testing, whereas the second perspective is from testing each hypothesis (of all calls of `ComponentsTreeSample`) at the same significance level α'. The multiple testing approach is again motivated "a-priori". That means, we have a particular error bound for the reconstruction of trees in mind and incorporate multiple testing procedures to achieve that particular bound (at least asymptotically). The second approach is again motivated "a-posteriori". We first run the algorithm and then analyze the probability of errors. This time, this motivational difference has also practical implications. Each call of `ComponentsTreeSample` depends on prior calls of `ComponentsTreeSample`. If a prior call returns wrong components, subsequent calls of `ComponentsTreeSample` are doomed to work with a wrong subtree and hence, output components and learned

edges themselves, which may be correct for the given subtree, but are wrong overall. This again influences subsequent calls negatively and so on. And even within one call of `ComponentsTreeSample`, results of later hypotheses depend on results of prior hypotheses.

Therefore, if we want to achieve a certain error rate "a-priori", meaning we adjust `ReconstructTreeSample` and `ComponentsTreeSample` before starting the initial call of `ReconstructTreeSample`, we need to predict what the algorithm will do later. We do not know what subsequent calls will be doing, which hypotheses they will test, how many hypotheses there will be and how many subsequent calls there will be. Thus, we need to come up with a multiple testing procedure for which we do not know the overall number of hypotheses.

The "a-posteriori" approach is different in that respect. Here, we do not need to think ahead what the algorithm will do later. We just test each hypothesis at some level α' and study the probability of an error after the algorithm has finished. Because we do this after, and not before the algorithm has run, we know how many calls and how many number of hypotheses there have been.

We will focus on these particular interpretations of the two perspectives, others are certainly possible. To be consistent, we use the multiple testing approach for `sCentralSample` with the multiple testing approach for `ReconstructTreeSample`. Similarly, the approach with a constant significance level for each hypothesis of `sCentralSample` is combined with the same approach for `ReconstructTreeSample`. Again, one could combine these algorithms differently, but we think that this is the most natural and consistent way to do it.

Perspective 1: Multiple testing

We start with the multiple testing approach. The trouble in deriving suitable testing procedures is that the null hypotheses of later hypotheses tests depend on the result of prior hypotheses tests. Also the number of tests depends on prior results.

Algorithm 9: ReconstructTreeSample (V)

```
if |V| > 1 then
    V_1, ..., V_m ← ComponentsTreeSample(V);
    for i from 1 to m do
        ReconstructTreeSample(V_i);
    end
end
```

One approach is to bound the overall number of hypotheses tests. For this, let $d := \Delta(\mathcal{G}(\mathbf{\Sigma}))$ be the maximum degree of the underlying concentration graph $\mathcal{G}(\mathbf{\Sigma})$. Note that in each call of `ComponentsTreeSample`(V_i), no matter what the subtree V_i is, $|N_{\text{sample}}| \leq d$. Here, N_{sample} contains all the known neighbors after finishing both loops in `ComponentsTreeSample`(V_i). Thus, the total number of hypotheses tests, which we denote by N_H, can be bounded by $\sum_i (|V_i| - 1)d$. Here, the sum is over all components V_i that have been returned at some point by starting `ReconstructTree`(V).

This bound again needs to be bounded, as we can neither specify how many elements in the sum are occuring nor how big each $|V_i|$ is. For this, note that $|V_i| \leq |V|$, i.e. each subtree is smaller than the initial tree. The procedure `sCentralSample` and `ComponentsTreeSample` are called at most $|V| - 1$-times. Thus, there are at most $|V| - 1$ elements in the sum and we have

$$N_H \leq \sum_i (|V_i| - 1)d \leq d \sum_i (|V| - 1) \leq d(|V| - 1)^2 \leq (|V| - 1)^3.$$

Implementing an overall multi-stage multiple testing procedure such at Holm's stepdown procedure is difficult due to the inherent dependence structure. What we can do, however, is a Bonferroni bound. With this procedure, each hypothesis (outside `sCentralSample`) is tested at level $\alpha'/(|V|-1)^3$, to ensure that the overall asymptotic FWER is less than α'. The bound on N_H does not look very tight. An untight bound is bad for power; we aim for a better approach.

As our attempts to improve the bound by a more complex combinatorial argument were not fruitful, we develop a warm-start procedure. Before starting `ReconstructTreeSample`, we set a starting value of how many hypotheses are likely to be tested (outside `sCentralSample`). Based on this starting value, a Bonferroni correction is applied on each test. While executing the algorithm the actual number of tests is counted. If the actual number is lower than the starting value, `ReconstructTreeSample` has successfully controlled the overall FWER of one call of `ReconstructTreeSample`. If on the other hand the actual value is higher than the starting value, we restart `ReconstructTreeSample`. This time, the value of how many hypotheses are likely to be tested is the prior actual value (plus some overhang). This warm-start procedure will run as long as the actual value is lower than the "guessed" value. Note that each new initial call of `ReconstructTreeSample` increases the query and time complexity.

Algorithm 10: ComponentsTreeSample(V)

w := `sCentralSample`(V);
$N_{\text{sample}} := \emptyset$;
Sort $|\hat{\rho}_{uw}|$ for $u \in V \setminus w$ in decreasing order and put them in list B_{sample};
for *every* u *in the order of* B_{sample} **do**
  t:= true;
  for *all* $v \in N_{sample}$ **do**
    Test $H_{0,(w,u,v)}$ at $\alpha'_{w,u,v}$;
    if $H_{0,(w,u,v)}$ *is rejected* **then**
      $V_{\text{sample},v} := V_{\text{sample},v} \cup \{u\}$;
      $t := false$;
    end
  end
  if $t = true$ **then**
    $\hat{E}_{\text{sample}} := \hat{E}_{\text{sample}} \cup \{u, w\}$;
    $N_{\text{sample}} := N_{\text{sample}} \cup \{u\}$;
    $V_{\text{sample},u} := \{u\}$;
  end
end
return *all* $V_{sample,u}$ *for* $u \in N_{sample}$

Algorithm 11: warmStart(V)

Parameter likely-number-of-tests;
Function $\delta(x)$;
ReconstructTreeSample(V);
while *actual-number-of-tests* > *likely-number-of-tests* **do**
  likely-number-of-tests = actual-number-of-tests
  $+\delta$(actual-number-of-tests);
  ReconstructTreeSample(V);
end

If the familywise error rate of one initial call of `ReconstructTreeSample` is controlled by α', we can show a sample analog of Proposition 3.21.

Proposition 3.23. *Let our testing procedure be such that the overall asymptotic FWER (excluding* `sCentralSample`*) of one initial call of* `ReconstructTreeSample` *is* $\leq \alpha'$*. Then asymptotically in* n*, algorithm 9 is correct with probability of at least* $1 - \alpha'$*. That is, if* $\mathcal{G}(\mathbf{\Sigma})$ *is a tree* T*, then* `ReconstructTreeSample`(V) *gives* $\hat{E}_{sample} = E(T)$ *with probability of at least* $1 - \alpha'$*.*
Here, the probability refers to any calculation under the regime $\boldsymbol{X}^{(i)} \sim \mathcal{N}(0, \mathbf{\Sigma})$

for all $i = 1, \ldots, n$.

Proof. As we have discussed before, for the correctness of `ComponentsTree` it does not matter whether we use $w = \texttt{sCentral}(V)$ or some other $w \in V$. For computational efficiency this will matter of course, but for the analysis of correctness this is not important. So to proof this proposition, we introduce a slightly tweaked version of `ComponentsTree` and `ReconstructTree`. Here, the only difference is that we take $w = \texttt{sCentralSample}(V)$. This small adjustment allows us to compare `ComponentsTreeSample` and `ReconstructTreeSample` more directly with `ComponentsTree` and `ReconstructTree`.

For every $w \in V$, either $w = \texttt{sCentralSample}(V)$ in one of the calls of `ComponentsTreeSample` (call such vertex central), or $\{w\}$ is one of the components $V_1, \ldots, V_m$ which are returned at the lowest recursion level by `ComponentsTreeSample`. Such vertex will be called terminal.

The first call of `ComponentsTreeSample` picks some vertex $w = \texttt{sCentralSample}(V)$ which induces the partition $\mathcal{C}^w = \{V_1, \ldots, V_m\}$ of $V \setminus \{w\}$. For each vertex $u \in V \setminus \{w\}$, `ComponentsTreeSample` compares the sample pairwise correlations $|\hat{\rho}_{u,w}|$ and puts them in descending order in a list B_{sample}. As explained in the proof of Proposition 3.4, $\hat{\rho}_{u,w} \xrightarrow{\text{a.s.}} \rho_{u,w}$. Therefore, the ordering of B_{sample} is almost surely the same as that of B asymptotically.

Let v be a neighbor of w and let u be any other vertex in the same connected component $C \in \mathcal{C}^w$. Then, v separates u and w. As each correlation (between distinct nodes) is strictly less than 1 due to multivariate Gaussianity, $\lim_{n\to\infty} |\hat{\rho}_{v,w}| > \lim_{n\to\infty} |\hat{\rho}_{u,w}|$ almost surely due to Lemma 2.11, which is about the factorization of pairwise correlations over trees. Hence, for any $C \in \mathcal{C}^w$, the vertex v in C, which is a neighbor of w, comes earlier in the order specified in `ComponentsTree` and almost surely in the order specified in the asymptotic version of `ComponentsTreeSample` than any other vertex in C.

What can still go wrong is that v or u are admitted too early to some other connected component corresponding to some $v' \in V \setminus C$ in `ComponentsTreeSample`. In terms of hypotheses, this would mean that $H_{0,(w,u,v')}$ or $H_{0,(w,v,v')}$ is rejected even though these null hypotheses are true. So the wrong admission to some other connected component is a type 1 error, and the overall FWER is controlled by α'.

Now, as v comes first in the order in `ComponentsTree`, it holds that $v \in N$ For all other $u \in C$, it holds that $\det(\mathbf{\Sigma}_{uw,vw}) \neq 0$. Equivalently, it holds that $H_{0,(w,u,v)}$ is false for all $u \in C \setminus \{v\}$. If no type 1 error is occuring overall, it holds that $v \in N_{\text{sample}}$, because, as discussed earlier, v comes first in the order of `ComponentsTreeSample` almost surely. Thus, the hypotheses $H_{0,(w,u,v)}$ will be

tested in `ComponentsTreeSample`. Again, asymptotically, all $H_{0,(w,u,v)}$ are rejected. Hence asymptotically and in case of no overall type 1 errors, the first call of `ComponentsTreeSample`,

(i) adds to $\hat{E}_{\text{sample}}$ the edges between the central vertex w and its neighbours in T,

(ii) assigns all vertices to their connected components in $\mathcal{C}^w$.

Because each $\mathcal{G}[V_{\text{sample},i}]$ is asymptotically a tree in case of no overall type 1 errors, the same argument can be applied to subsequent calls of `ReconstructTreeSample`. That probability is bounded from below by $1 - \alpha'$. Hence, asymptotically and in case of no overall type 1 errors, these two properties hold by induction at any call of `ComponentsTreeSample`. This implies in particular that $\hat{E}_{\text{sample}} \subseteq E(T)$ asymptotically.

The opposite inclusion works as follows. Let $uv \in E(T)$, and u or v is central, i.e. there is some call of `ComponentsTreeSample` where either u or v is returned from `sCentralSample`. In that case, $uv \in \hat{E}_{\text{sample}}$ by (i). On the other hand, if u, v are both terminal, then there is some call of `ComponentsTreeSample` that places them in different sets $V_{\text{sample},i}$. Property (ii) now implies that there is no edge $uv \in E(T)$. □

Remark 3.24. In figure 3.3 we have seen that `sCentral` is important for complexity, not for correctness. For the sample version we do not know whether `sCentralSample` has an effect on correctness. It may be that less or more errors occur when splitting the graph with respect to a central vertex. We will explore that in Chapter 4.

Besides correctness, there is a result on complexity. Here, the probabilistic procedure `sCentral` plays a crucial role to reduce it.

Theorem 3.25. *Suppose $\mathcal{G}(\boldsymbol{\Sigma})$ is a tree $T = (V, E)$ with maximum degree $\Delta(T) \leq d$. Fix $\varepsilon < 1$ and define $\kappa = \left\lceil 32 \ln\left(\frac{2|V|^2}{\varepsilon}\right) \right\rceil$ to be the parameter of algorithm 2. Then, with probability of at least $1 - \varepsilon$, Algorithm 8 requires time and queries of order*

$$\mathcal{O}\left(|V| \ln(|V|) \max\left\{ \ln\left(\frac{|V|}{\varepsilon}\right), d \right\} \right).$$

Here, the probability refers to any calculation under the regime that u, w are sampled uniformly at random in each call of `sCentral`.

A similar result is true for the sample version.

Theorem 3.26. *Suppose $\mathcal{G}(\mathbf{\Sigma})$ is a tree $T = (V, E)$ with maximum degree $\Delta(T) \leq d$. Let our testing procedure be such that the asymptotic FWER of each call of algorithm 3 is $\leq \alpha/|V|$. Fix $\alpha < \varepsilon < 1$ and let $\kappa = \left\lceil 128 \ln(\frac{2|V|^2}{\varepsilon - \alpha}) \right\rceil$ be the parameter of algorithm 3. Then, asymptotically with probability of at least $1 - \varepsilon$, algorithm 9 requires time and queries of order*

$$\mathcal{O}\left(|V| \ln(|V|) \max\left\{ \ln\left(\frac{|V|}{\varepsilon - \alpha} \right), d \right\} \right).$$

if `sCentralSampleBonferroni` *is used. The query complexity is the same if* `sCentralHolm` *is used, the time complexity is*

$$\mathcal{O}\left(|V| \ln(|V|) \max\left\{ \ln\left(\frac{|V|}{\varepsilon - \alpha} \right) \ln\left(\ln\left(\frac{|V|}{\varepsilon - \alpha} \right) \right), d \right\} \right).$$

The probability refers to any calculation under the regime $\boldsymbol{X}^{(i)} \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{\Sigma})$ for all $i = 1, \ldots, n$ and that u, w are sampled uniformly at random in each call and iteration of `sCentralSample`.

Proof. The proof is conceptually the same as the one of Theorem 3.25.
First, we analyze the complexity of one call of `ComponentsTreeSample`(V). By Proposition 3.3, the call of `sCentralSampleBonferroni`(V) takes time and queries of order $\mathcal{O}(|V|\kappa)$, the call of `sCentralSampleHolm`(V) has query complexity $\mathcal{O}(|V|\kappa)$ and time complexity $\mathcal{O}(|V|\kappa \ln(\kappa))$. `ComponentsTreeSample` then queries $|V|$ pairwise sample correlations. Sorting these pairwise correlations by algorithms like mergesort or heapsort (c.f. Cormen et al. (2009), Part II) takes time $\mathcal{O}(|V| \log |V|)$. Partitioning V into sets V_i takes time and queries both of order $\mathcal{O}(d|V|)$, since $|N_{\text{sample}}| \leq d$. For all calls of `ReconstructTreeSample`$(V_i)$ in that new recursion level (i.e. distance from the first call of `ReconstructTreeSample` in the recursion tree), it holds that $V_i \cap V_j = \emptyset$. For each V_i in a recursion level, the time complexity is

$$\mathcal{O}\left(|V_i|\kappa + |V_i| \ln |V_i| + |V_i| d \right)$$

if we use `sCentralSampleBonferroni` and

$$\mathcal{O}\left(|V_i|\kappa \ln(\kappa) + |V_i| \ln |V_i| + |V_i| d \right)$$

if we use `sCentralHolm`. The query complexity in both cases is

$$\mathcal{O}\left(|V_i|\kappa + |V_i| d \right).$$

Furthermore, where the sum is for one recursion level, it holds that

$$\begin{aligned}\sum_i \big(|V_i|\kappa + |V_i|\ln|V_i| + |V_i|d\big) &\leq \sum_i \big(|V_i|\kappa + |V_i|\ln|V| + |V_i|d\big)\\ &= |V|\kappa + |V|\ln|V| + |V|d,\\ \sum_i \big(|V_i|\kappa\ln(\kappa) + |V_i|\ln|V_i| + |V_i|d\big) &\leq |V|\kappa\ln(\kappa) + |V|\ln|V| + |V|d,\\ \sum_i \big(|V_i|\kappa + |V_i|d\big) &\leq |V|\kappa + |V|d\end{aligned}$$

due to disjointness. Hence, in each recursion level, time complexity is of order

$$\mathcal{O}\bigg(|V|\kappa + |V|\ln|V| + |V|d\bigg)$$

if we use `sCentralSampleBonferroni` and

$$\mathcal{O}\bigg(|V|\kappa\ln(\kappa) + |V|\ln|V| + |V|d\bigg)$$

if we use `sCentralSampleHolm`. In both cases the query complexity is of order

$$\mathcal{O}\bigg(|V|\kappa + |V|d\bigg).$$

Assume first that $\tilde{v} = \texttt{sCentralSample}(V)$ satisfies $c(\tilde{v}) \leq \gamma := \sqrt{11/12}$ in each call with $|V| \geq 4$. At the second recursion level (the first level is the initial call of `ReconstructTreeSample`), the size of the maximal subtree is bounded by $|V|c(\tilde{v}_1)$, where $\tilde{v}_1$ is the vertex returned from `sCentralSample`. Inductively, for the N-th level, the size of the maximal subtree is bounded by $|V|c(\tilde{v}_1)c(\tilde{v}_2)\cdots c(\tilde{v}_{N-1})$, where $\tilde{v}_i$ denotes the vertex returned from the respective call of `sCentralSample`. For the maximal recursion depth (abbrev. depth), it holds that

$$\begin{aligned}1 &\leq |V|c(\tilde{v}_1)c(\tilde{v}_2)\cdots c(\tilde{v}_{\text{depth}-2})\\ &\leq |V|\gamma^{\text{depth}-4}c(\tilde{v}_{\text{depth}-3})c(\tilde{v}_{\text{depth}-2})\\ &\leq |V|\gamma^{\text{depth}-4}.\end{aligned}$$

Solving this inequality yields that the recursion depth is at most $\log_{1/\gamma}(|V|) + 4$. Thus, overall, `ReconstructTreeSample` has time complexity

$$\mathcal{O}\bigg(|V|\log_{1/\gamma}(|V|)\big[\kappa + \ln|V| + d\big]\bigg)$$

if `sCentralSampleBonferroni` is used and

$$\mathcal{O}\Big(|V|\log_{1/\gamma}(|V|)\big[\kappa\ln(\kappa)+\ln|V|+d\big]\Big)$$

if `sCentralSampleHolm` is used. Moreover, for both versions the query complexity is

$$\mathcal{O}\Big(|V|\log_{1/\gamma}(|V|)\big[\kappa+d\big]\Big).$$

Since

$$\kappa=\mathcal{O}\Big(\ln\big(|V|/(\varepsilon-\alpha)\big)+\ln(d)\Big)$$

and

$$\log_{1/\gamma}(x)=\frac{\ln(x)}{\ln(1/\gamma)}$$

for all $x>0$, the announced bounds follow.

It remains to show that with probability of at least $1-\varepsilon$ and the given choice of κ, $c(\tilde{v})\leq\gamma$ in each call with $|V|\geq 4$. The probability that $c(\tilde{v})>\gamma$ for a single call of `sCentralSample` is asymptotically at most

$$\frac{\alpha}{|V|}+2|V|\exp\Big(-\frac{\kappa}{128}\Big)$$

by Proposition 3.14. As `ReconstructTreeSample`(V) runs, the procedure `sCentralSample` is called at most $|V|$ times which is the total number of available vertices. From the union bound, the asymptotic probability that in at least one call, $\tilde{v}=$ `sCentralSample`(V) satisfies $c(\tilde{v})>\alpha$, is at most

$$|V|\Big(\frac{\alpha}{|V|}+2|V|\exp\Big(-\frac{\kappa}{128}\Big)\Big)=\alpha+2|V|^2\exp\Big(-\frac{\kappa}{128}\Big).$$

Demanding the latter to be at most ε, we obtain the indicated value for κ and the desired result. □

Remark 3.27. We need the FWER of one call of `sCentral` to be $\frac{\alpha}{|V|}$ to arrive at this complexity bound. In the Bonferroni case this means testing each hypothesis at level $\frac{\alpha}{\kappa|V||V_i|}$, where V_i is the respective subtree.

Remark 3.28. Asymptotically, the version that uses `sCentralSampleHolm` is worse in time complexity. But, for finite n, `sCentralSampleHolm` may return a more central vertex than `sCentralSampleBonferroni` which reduces overall time complexity.

Perspective 2: Testing each hypothesis at the same signifance the same significance level α'

Here we explore the approach of testing each hypothesis (outside `sCentralSample`) at level $\alpha' \in (0,1)$. We then run the algorithm, count how many hypotheses have been tested in each call of `ComponentsTreeSample`, how many calls there have been, and analyze with which probability the correct tree has been returned.

Again, for this analysis we compare `ComponentsTreeSample` to a slightly modified version of `ComponentsTree`, where we run `sCentralSample` instead of `sCentral`. As mentioned above, this does not change the correctness of `ComponentsTree`. With this tweak, we just need to study the probability of type 1 and 2 errors and the probability that the sorting of pairwise correlations is correct under the assumption that all previous steps have been correct.

The calls of `ReconstructTreeSample` (equivalently the calls of `ComponentsTreeSample`) can be represented in a recursion tree. However, we do not need that particular representation. Methodologically, the calls of `ComponentsTreeSample` are executed sequentially, and we label all calls of `ComponentsTreeSample` in short and chronologically by $\texttt{CTS}_1, \ldots, \texttt{CTS}_{n_{\text{Calls}}}$, where n_{Calls} is the total number of calls of `ComponentsTreeSample`. Note that because we study the errors *after* running `ReconstructTreeSample`, we actually know the value of n_{Calls}. The parent of a particular call is the call which output is inputted into the respective call. Note that by our labelling of the calls, the parent call has always a lower index than the child call. The difference in the index is not necessarily 1, but that is also not important for the analysis. It is also not important that some steps may actually run in parallel, because in that case the parent still has a lower index than its child.

We are interested in the probability that the returned tree $\hat{\mathcal{G}}$ is correct, i.e. $\hat{\mathcal{G}} = \mathcal{G}(\mathbf{\Sigma})$. We have

$$\text{Prob}\left(\hat{\mathcal{G}} = \mathcal{G}(\Sigma)\right) \geq \text{Prob}\left(\texttt{CTS}_1, \ldots, \texttt{CTS}_{n_{\text{Calls}}} \text{are correct}\right). \tag{3.11}$$

Here, we say that a call $\texttt{CTS}_i$ is correct, if no type 1 and type 2 errors occur in that particular call and the ordering of the sample partial correlations is correct.

We can factorize the right-hand side in (3.11). We have

$$\begin{aligned}
&\text{Prob}\Big(\texttt{CTS}_1, \ldots, \texttt{CTS}_{n_{\text{Calls}}} \text{ are correct}\Big) \\
&\quad = \text{Prob}\Big(\texttt{CTS}_{n_{\text{Calls}}} \text{is correct} \;\Big|\; \texttt{CTS}_1, \ldots, \texttt{CTS}_{n_{\text{Calls}}-1} \text{ are correct}\Big) \\
&\qquad \cdot \text{Prob}\Big(\texttt{CTS}_1, \ldots, \texttt{CTS}_{n_{\text{Calls}}-1} \text{ are correct}\Big) \\
&\quad = \ldots \\
&\quad = \text{Prob}\Big(\texttt{CTS}_{n_{\text{Calls}}} \text{ is correct} \;\Big|\; \texttt{CTS}_1, \ldots, \texttt{CTS}_{n_{\text{Calls}}-1} \text{ are correct}\Big) \\
&\qquad \cdot \text{Prob}\Big(\texttt{CTS}_{n_{\text{Calls}}-1} \text{ is correct} \;\Big|\; \texttt{CTS}_1, \ldots, \texttt{CTS}_{n_{\text{Calls}}-2} \text{ are correct}\Big) \\
&\qquad \cdots \text{Prob}\Big(\texttt{CTS}_2 \text{ is correct} \;\Big|\; \texttt{CTS}_1 \text{ is correct}\Big) \cdot \text{Prob}\Big(\texttt{CTS}_1 \text{ is correct}\Big)
\end{aligned}$$

We could restrict the conditioned set simply to all ancestor calls (i.e. parent call, parent call of the parent call, etc.), but this is not needed.
We now want to bound the probability that $\texttt{CTS}_i$ is correct under the assumption that all prior calls of $\texttt{CTS}_i$ have been correct, i.e. we want to bound

$$\text{Prob}\Big(\texttt{CTS}_i \text{ is correct} \;\Big|\; \texttt{CTS}_1, \ldots, \texttt{CTS}_{i-1} \text{ are correct}\Big).$$

We have that

$$\begin{aligned}
&\text{Prob}\Big(\texttt{CTS}_i \text{ is correct} \;\Big|\; \texttt{CTS}_1, \ldots, \texttt{CTS}_{i-1} \text{ are correct}\Big) \\
&\quad = \text{Prob}\Big(\text{No type 1 and type 2 errors, ordering of list } B_{\text{sample}} \text{ is correct} \;\Big| \\
&\qquad\qquad \texttt{CTS}_1, \ldots, \texttt{CTS}_{i-1} \text{ are correct}\Big) \\
&\quad = \text{Prob}\Big(\text{No type 1 and type 2 errors} \;\Big| \\
&\qquad\qquad \text{Ordering of list } B_{\text{sample}} \text{ is correct}, \texttt{CTS}_1, \ldots, \texttt{CTS}_{i-1} \text{ are correct}\Big) \\
&\qquad \cdot \text{Prob}\Big(\text{Ordering of list } B_{\text{sample}} \text{ is correct} \;\Big|\; \texttt{CTS}_1, \ldots, \texttt{CTS}_{i-1} \text{ are correct}\Big).
\end{aligned} \tag{3.12}$$

We abbreviate the first factor as $1-\eta_i$ and the second factor as $1-\theta_i$. We can calculate η_i based on equations (3.6) and (3.7). These equations were based on Kalisch and Bühlmann (2007) who themselves used results from Hotelling (1953). Recall that the significance level for each hypothesis was set to

$$\alpha' = \alpha'_n = 2 \cdot \left(1 - \Phi\left(n^{1/2} \cdot \frac{\rho_{min}}{2}\right)\right),$$

to derive theoretical results. Under assumption (A4), i.e. that there is some upper bound $M < 1$ on the partial correlations $\rho_{u,w|v}$ for $u, w \in V \setminus \{v\}$ and $u \neq w$, we have

$$\eta_i \leq N_{H,i}\, \mathcal{O}(n-1) \exp\left(-C_i(n-1)\rho_{min}^2\right), \tag{3.13}$$

whereas $N_{H,i}$ is the number of hypotheses tests (outside `sCentralSample`) in $\texttt{CTS}_i$ and $0 < C_i < \infty$. Here, C_i needs to be smaller for larger M.

To calculate θ_i, we need a new assumption and a new lemma (see Lemma 1 in Kalisch and Bühlmann (2007)). The new assumption is very similar to (A4), we therefore call it (A4′). It is that all pairwise correlations $\rho_{u,w}$ for $u \neq w$ are upper-bounded by some $M' < 1$.

Lemma 3.29. *Assume (A4′) (and Gaussianity). Then, for any* $0 < \gamma \leq 2$,

$$\sup_{u,w \in V} \text{Prob}\left(\left|\hat{\rho}_{u,w} - \rho_{u,w}\right| > \gamma\right) \leq C'(n-2) \exp\left((n-4)\log\left(\frac{4-\gamma^2}{4+\gamma^2}\right)\right),$$

for some constant $0 < C' < \infty$ *which depends on* M' *only.*

Now, define

$$d_{min} := \min_{w \in V} \min_{\substack{u,u' \in V \setminus \{w\};\, u \neq u' \\ u \text{ separates } u' \text{ and } w}} \left||\rho_{u,w}| - |\rho_{u',w}|\right|.$$

It holds that d_{min} is positive because of Lemma 2.11 and no pairwise correlation between distinct vertices has absolute value 1 due to multivariate Gaussianity. Due to this positivity, we can use Lemma 3.29 with $\gamma = d_{min}$. We also note that the ordering of the $|\rho_{u,w}|$ for fixed w is the same as the ordering of the $|\hat{\rho}_{u,w}|$ for fixed w if $||\hat{\rho}_{u,w}| - |\rho_{u,w}|| < d_{min}$ for all $u \in V \setminus \{w\}$. Also recall that $\ln((4-x^2)/(4+x^2)) \sim -x^2/2$ as $x \to 0$. Furthermore, $\{||\hat{\rho}_{u,w}| - |\rho_{u,w}|| = d_{min}\}$ has measure zero. In total we obtain that

$$\theta_i \leq C'(n-2) \exp\left(-(n-4)\frac{d_{min}^2}{2}\right), \tag{3.14}$$

for some constant $0 < C' < \infty$ which depends on M' only.

Combining equations (3.12), (3.13) and (3.14) yields

$$\begin{aligned}
&\text{Prob}\Big(\texttt{CTS}_i \text{ is correct} \;\Big|\; \texttt{CTS}_1, \dots, \texttt{CTS}_{i-1} \text{ are correct}\Big) \\
&\quad \geq \Big[1 - N_{H,i}\, \mathcal{O}(n-1) \exp\Big(- C_i (n-1) \rho_{min}^2 \Big)\Big] \\
&\qquad \cdot \Big[1 - C'(n-2) \exp\Big(- (n-4) \frac{d_{min}^2}{2} \Big)\Big]. \qquad (3.15)
\end{aligned}$$

Even though in this conceptual setting we know the value of $N_{H,i}$, we may still be interested in bounds. For that, note that $N_{H,i} \leq (|V_i| - 1) \cdot d_i \leq (|V| - 1) \cdot d \leq (|V| - 1)^2$, whereas $|V|_i$ is the inputted subtree and d resp. d_i is the maximum degree in the overall graph resp. the subgraph V_i. Moreover, $n_{\text{Calls}} \leq |V| - 1$. In total, we then have

$$\begin{aligned}
\text{Prob}(\hat{\mathcal{G}} = \mathcal{G}(\Sigma)) &\geq \Big[1 - C'(n-2) \exp\Big(- (n-4) \frac{d_{min}^2}{2} \Big)\Big]^{n_{\text{Calls}}} \\
&\quad \cdot \prod_{i=1}^{n_{\text{Calls}}} \Big[1 - N_{H,i}\, \mathcal{O}(n-1) \exp\Big(- C_i (n-1) \rho_{min}^2 \Big)\Big] \\
&\geq \Big[1 - C'(n-2) \exp\Big(- (n-4) \frac{d_{min}^2}{2} \Big)\Big]^{n_{\text{Calls}}} \\
&\quad \cdot \prod_{i=1}^{n_{\text{Calls}}} \Big[1 - \big(|V| - 1\big) \cdot d \cdot \mathcal{O}(n-1) \exp\Big(- C_i (n-1) \rho_{min}^2 \Big)\Big] \\
&\geq \Big[1 - C'(n-2) \exp\Big(- (n-4) \frac{d_{min}^2}{2} \Big)\Big]^{n_{\text{Calls}}} \\
&\quad \cdot \prod_{i=1}^{n_{\text{Calls}}} \Big[1 - \big(|V| - 1\big)^2 \cdot \mathcal{O}(n-1) \exp\Big(- C_i (n-1) \rho_{min}^2 \Big)\Big] \\
&\geq \Big[1 - C'(n-2) \exp\Big(- (n-4) \frac{d_{min}^2}{2} \Big)\Big]^{|V|-1} \\
&\quad \cdot \Big[1 - \big(|V| - 1\big)^2 \cdot \mathcal{O}(n-1) \exp\Big(- C_{min} (n-1) \rho_{min}^2 \Big)\Big]^{|V|-1}, \qquad (3.16)
\end{aligned}$$

whereas $C_{min} = \min_{i \in \{1, \dots, n_{Calls}\}} C_i$. Letting $n \to \infty$ makes the right-hand side go to 1. On the first glance, this seems to contradict the result from Proposition 3.23, which stated that `ReconstructTreeSample` is asymptotically correct with

probability of at least $1-\alpha'$, but it does not. Firstly, $1-\alpha'$ is only a lower bound. And secondly, note that here $\alpha' = \alpha'_n$ depends on n and converges to 0 for $n \to \infty$. In Proposition 3.23, α' was constant in n. If we let α' go to 0 in that particular proposition, we would obtain a similar asymptotic result.

We can also prove an analog of Theorem 3.26.

Theorem 3.30. *Suppose $\mathcal{G}(\mathbf{\Sigma})$ is a tree $T = (V, E)$ with maximum degree $\Delta(T) \leq d$. Let $\alpha = \alpha_n$ be as defined in equation (3.5) and the significance level in algorithm 3 at which each hypothesis is tested. Let C be as in Proposition 3.17 (applied to the whole graph). Furthermore, fix ε at some value strictly bigger than the term $|V|^3 F_{n-1} \exp(-C(n-1)\rho^2_{min}) =: t$, where F_{n-1} is the term of order $\mathcal{O}(n-1)$ from Proposition 3.17. Moreover, let $\kappa = \left\lceil 128 \ln(\frac{2|V|^2}{\varepsilon - t}) \right\rceil$ be the parameter of algorithm 3. Then, asymptotically with probability of at least $1-\varepsilon$, algorithm 9 requires time and queries of order*

$$\mathcal{O}\left(|V| \ln(|V|) \max\left\{ \ln\left(\frac{|V|}{\varepsilon - t} \right), d \right\} \right).$$

Here, C is as in Proposition 3.17. The probability refers to any calculation under the regime $\boldsymbol{X}^{(i)} \overset{i.i.d}{\sim} \mathcal{N}(0, \mathbf{\Sigma})$ for all $i = 1, \ldots, n$ and that u, w are sampled uniformly at random in each call and iteration of `sCentralSample`.

Proof. The proof is very similar to the one of Theorem 3.26. Basically, replace α by t in the proof and use the result of Proposition 3.18 instead of the one from Proposition 3.14 . □

3.1.3. Lower bounds

Theorem 3.25, similarly Theorem 3.26 or Theorem 3.30 show that the query complexity of `ReconstructTree` resp. `ReconstructTreeSample` is less than $\mathcal{O}(|V|^2)$, asymptotically or for large enough n, depending on the setup. One may ask, in which sense these complexities are already optimal, i.e. whether there are other algorithms which return the correct tree but have lower query complexity.

In the following we will show (meaning we will outline the argument given in Lugosi et al. (2021)) that the the non-sample version is indeed optimal (up to logarithmic factors). To prove the result for the non-sample case, we show that there is no algorithm that can reconstruct trees with maximum degree d in less than $\Omega(dn)$ covariance queries. For the sample case we are not able to proof an analog result. In fact, we do think that doing so is impossible and we state some reasons why we think that is the case.

Let $\mathcal{X}$ be the class of $|V| \times |V|$ covariance matrices whose concentration graph is a tree. We denote the induced tree of a covariance matrix $\mathbf{\Sigma} \in \mathcal{X}$ by $T(\mathbf{\Sigma})$. Furthermore, we write $\mathcal{X}_d$ to be the class of covariance matrices whose concentration graph is a tree with maximum degree less or equal than d.

We start with an easier setting, where we do not have an upper bound on the maximum degree. We will show that any algorithm needs to access the covariance oracle $\Omega(n^2)$ times. We formalize this statement as follows. Let $\mathcal{A}_k$ be the class of all randomized adaptive algorithms that query the covariance matrix at most k-times. Here, adaptive means that the algorithms are allowed to dynamically choose their queries. We denote the returned tree of some algorithm $A \in \mathcal{A}_k$ by $\mathcal{T}(A)$. Moreover, we denote the probability that the algorithm A returns the wrong tree for fixed $\mathbf{\Sigma} \in \mathcal{X}$ by

$$P(A, \mathbf{\Sigma}) := \text{Prob}(\mathcal{T}(A) \neq T(\mathbf{\Sigma})). \tag{3.17}$$

Here, the probability is with respect to the randomization of algorithm A. For each particular algorithm, we are interested in the respective worst case of unsuccessful tree reconstruction, i.e. in $\sup_{\mathbf{\Sigma} \in \mathcal{X}} P(A, \mathbf{\Sigma})$. We are then interested in the algorithm with at most k covariance queries that has the best worst case probability of an error. Formally, we are looking at the minimax risk

$$R(\mathcal{A}_k, \mathcal{X}) = \inf_{A \in \mathcal{A}_k} \sup_{\mathbf{\Sigma} \in \mathcal{X}} P(A, \mathbf{\Sigma}).$$

In Proposition 3.21 we have seen that `ReconstructTree` is correct. Theorem 3.25 is a result on the query complexity of `ReconstructTree`. In particular, this result implies that for each $\varepsilon > 0$ there exists a constant $c > 0$ such that

$$R(\mathcal{A}_k, \mathcal{X}) \leq \varepsilon$$

for all $k > c|V| \ln(|V|)(d + \ln(|V|/\varepsilon)$. Lugosi et al. (2021) have shown that this bound is tight up to logarithmic factors. They started with the case $d = n - 1$.

Theorem 3.31. *For all* $k \leq \binom{|V|}{2}$,

$$R(\mathcal{A}_k, \mathcal{X}) \geq \frac{1}{2} - \frac{k}{(|V| - 1)^2}.$$

In particular, $R(\mathcal{A}_k, \mathcal{X}) \geq \frac{1}{2} - o(1)$, *whenever* $k = o(|V|^2)$.

This theorem can be generalized to the class $\mathcal{X}_d$, i.e. the class of covariance matrices with maximum degree bounded by d.

Theorem 3.32. *For all* $k \leq \binom{|V|}{2}$,

$$R(\mathcal{A}_k, \mathcal{X}_d) \geq \frac{1}{2} - \frac{k}{(|V|-1)^2}.$$

In particular, $R(\mathcal{A}_k, \mathcal{X}_d) \geq \frac{1}{2} - o(1)$, *whenever* $k = o(|V|d)$.

These two results state that if the number of allowed queries is too low, then even the best algorithm is wrong with high probability.

We want to extend these results to the sample case. For that, let $\mathcal{T}(A, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)})$ denote the tree that algorithm A returns given that $\boldsymbol{\Sigma}$ is unknown and only data $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$ is at hand. We then write

$$P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) = \text{Prob}(\mathcal{T}(A, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \neq T(\boldsymbol{\Sigma})). \tag{3.18}$$

Here, the probability is with respect to the randomization of the algorithm A and the fact that the observations $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$ are drawn i.i.d. from $\mathcal{N}(0, \boldsymbol{\Sigma})$. We are again interested in the minimax risk

$$R_{\text{sample}}(\mathcal{A}_k, \mathcal{X}) = \inf_{A \in \mathcal{A}_k} \sup_{\substack{\Sigma \in \mathcal{X} \\ \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \boldsymbol{\Sigma})}} P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}).$$

We now obtain very similar theorems as for the non-sample case.

Theorem 3.33. *For all* $k \leq \binom{|V|}{2}$,

$$R_{sample}(\mathcal{A}_k, \mathcal{X}) \geq \frac{1}{2} - \frac{k}{(|V|-1)^2}.$$

In particular, $R_{sample}(\mathcal{A}_k, \mathcal{X}) \geq \frac{1}{2} - o(1)$, *whenever* $k = o(|V|^2)$.

This theorem can again be generalized to the class $\mathcal{X}_d$, i.e. the class of covariance matrices with maximum degree bounded by d.

Theorem 3.34. *For all* $k \leq \binom{|V|}{2}$,

$$R_{sample}(\mathcal{A}_k, \mathcal{X}_d) \geq \frac{1}{2} - \frac{k}{(|V|-1)^2}.$$

In particular, $R_{sample}(\mathcal{A}_k, \mathcal{X}_d) \geq \frac{1}{2} - o(1)$, *whenever* $k = o(|V|d)$.

Remark 3.35. Intuitively, the result of this theorem should be true. Knowing the true covariance matrix $\boldsymbol{\Sigma}$ yields more information then just knowing finitely many observations and thus should make it more probable, that the best algorithm in the non-sample case returns the correct tree more often than the best algorithm in the sample case. Nevertheless, we give a formal proof of Theorem 3.33 below to put this intuitive idea on a solid basis.

Proof. The proof is very similar to the one of Theorem 3.32, which has already been proven by Lugosi et al. (2021). We have only made some small modifications that encorporate the randomization with respect to the observations. Nonetheless, we think that pointing out these minor adjustments is still necessary for a comprehensive presentation.

We define a probability distribution $\mathcal{D}$ on the set $\mathcal{X}$ and write

$$\begin{aligned} R_{\text{sample}}(\mathcal{A}_k, \mathcal{X}) &\geq \inf_{A \in \mathcal{A}_k} \mathbb{E}_{\boldsymbol{\Sigma} \sim \mathcal{D};\, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \boldsymbol{\Sigma})} \big[P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \big] \\ &= \inf_{A \in \mathcal{A}_k} \mathbb{E}_{\boldsymbol{\Sigma} \sim \mathcal{D}} \big[\mathbb{E}_{\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \boldsymbol{\Sigma})} \big[P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \big] \big]. \end{aligned}$$

Now, we explain how a random symmetric matrix $\boldsymbol{\Sigma}$, which is distributed according to $\mathcal{D}$, is generated. We consider a collection of independent random variables. Let B be a Bernoulli random variable with parameter $1/2$ and let $U_1, \ldots, U_{|V|-1}$ be random variables which are independent and uniformly distributed on $[0, 1]$. Let I, J be distinct indices in the set $\{1, \ldots, |V| - 1\}$ which are uniformly drawn over all of these $(|V| - 1)(|V| - 2)$ distinct pairs. We index the $|V|$ columns and rows from 0 to $|V| - 1$ and define $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(B, U_1, \ldots, U_{|V|-1}, I, J)$ as follows.

- $\Sigma_{i,i} = 1$ for all $i = 0, \ldots, |V| - 1$.
- Regardless of B, I, J, we set $\Sigma_{0,i} = U_i$ for all $i = 1, \ldots, |V| - 1$.
- If $B = 0$, then $\Sigma_{i,j} = U_i U_j$ for all $i, j \in \{1, \ldots, |V| - 1\}$ and $i \neq j$. In this case, the concentration graph is a star with vertex 0 as its center (see Lemma 2.11). Thus, $\boldsymbol{\Sigma} \in \mathcal{X}$.
- If $B = 1$, then $\Sigma_{i,j} = U_i U_j$ for all $i, j \in \{1, \ldots, |V| - 1\}$ such that $i \neq j$ and $\{i, j\} \neq \{I, J\}$. We set $\Sigma_{I,J} = \min\{U_I, U_J\} / \max\{U_I, U_J\}$. Again, due to Lemma 2.11, the concentration graph of $\boldsymbol{\Sigma}$ is a tree where the vertex 0 has degree $|V| - 2$ and every vertex $i \notin \{I, J\}$ has degree 1 and is adjacent to vertex 0. The vertices $0, I, J$ form a path such that if $U_I < U_J$, the vertex J is the middle vertex, and if $U_I > U_J$ the vertex I is the middle vertex.

No algorithm, whether it assumes knowledge of $\boldsymbol{\Sigma}$ or some estimate $\hat{\boldsymbol{\Sigma}}$ which is based on $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$, can distinguish between $\boldsymbol{\Sigma}(0, U_1, \ldots, U_{|V|-1}, I, J)$ and $\boldsymbol{\Sigma}(1, U_1, \ldots, U_{|V|-1}, I, J)$ before querying the entry $\Sigma_{I,J}$ or $\hat{\Sigma}_{I,J}$ respectively. All information about $\Sigma_{I,J}$ is contained in the entry $\Sigma_{I,J}$ resp. $\hat{\Sigma}_{I,J}$, no other entry of $\boldsymbol{\Sigma}$ resp. $\hat{\boldsymbol{\Sigma}}$ does so. Therefore, for fixed $B, U_1, \ldots, U_{|V|-1}$, I, J and $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(B, U_1, \ldots, U_{|V|-1}, I, J)$, it holds that

$$P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \geq \frac{1}{2} \mathbb{E}\big[1_{(I,J) \text{ is not queried}} | (I, J) \big].$$

By monotonicity of the conditional expectation,

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{X}^{(1)},\dots,\boldsymbol{X}^{(n)} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0,\boldsymbol{\Sigma})}\big[P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \dots, \boldsymbol{X}^{(n)})\big] \\
&\qquad \geq \mathbb{E}_{\boldsymbol{X}^{(1)},\dots,\boldsymbol{X}^{(n)} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0,\boldsymbol{\Sigma})}\left[\frac{1}{2}\mathbb{E}\big[1_{(I,J) \text{ is not queried}} | (I,J)\big]\right] \\
&\qquad = \frac{1}{2}\mathbb{E}\big[1_{(I,J) \text{ is not queried}} | (I,J)\big],
\end{aligned}$$

where the last equality follows from the fact that our data-generating process is independent of querying entries. Hence, for any algorithm A,

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\Sigma}\sim\mathcal{D};\, \boldsymbol{X}^{(1)},\dots,\boldsymbol{X}^{(n)} \overset{\text{i.i.d}}{\sim} \mathcal{N}(0,\boldsymbol{\Sigma})}\big[P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \dots, \boldsymbol{X}^{(n)})\big] \\
&\qquad \geq \mathbb{E}_{\boldsymbol{\Sigma}\sim\mathcal{D}}\left[\frac{1}{2}\mathbb{E}\big[1_{(I,J) \text{ is not queried}} | (I,J)\big]\right] \\
&\qquad \geq \mathbb{E}_{I,J}\left[\frac{1}{2}\mathbb{E}\big[1_{(I,J) \text{ is not queried}} | (I,J)\big]\right] \\
&\qquad = \frac{1}{2\binom{|V|-1}{2}} \sum_{\{i,j\}\subseteq\{1,\dots,|V|-1\}: i\neq j} \mathbb{E}\big[1_{(i,j) \text{ is not queried}}\big] \\
&\qquad \geq \frac{\binom{|V|-1}{2} - k}{2\binom{|V|-1}{2}}.
\end{aligned}$$

Here, the last inequality follows from the fact that the maximum number of queries is k, thus, the minimum number of non-queries is $\binom{|V|-1}{2} - k$. Cancelling terms yields the result □

We now take a look at the versions of `ReconstructTreeSample`. Recall from Proposition 3.23 that if A is `ReconstructTreeSampleBonferroni` or `ReconstructTreeSampleHolm` with an overall FWER-control of α', then

$$\limsup_{n\to\infty} P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \dots, \boldsymbol{X}^{(n)}) \leq \alpha'. \tag{3.19}$$

For the version where each hypothesis is tested at a constant level α', we have also seen finite sample bounds (for some specific choice of α'), see e.g. (3.16). The problem is that we have not shown that

$$\limsup_{n\to\infty} \sup_{\boldsymbol{\Sigma}\in\mathcal{X}} P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \dots, \boldsymbol{X}^{(n)}) \leq \alpha',$$

i.e. that the error-probability decreases *uniformly* in n. If we had this result, then

$$\begin{aligned}\frac{1}{2} - \frac{k}{(|V|-1)^2} &\leq \limsup_{n\to\infty} R_{\text{sample}}(\mathcal{A}_k, \mathcal{X}) \\ &\leq \limsup_{n\to\infty} \sup_{\boldsymbol{\Sigma}\in\mathcal{X}} P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \\ &\leq \alpha'. \end{aligned} \tag{3.20}$$

This would have yielded a sample analog of Theorem 3.33 , i.e. asymptotically, any algorithm that wants to achieve an asymptotic correctness of at least $1-\alpha'$ would need at least

$$\frac{(|V|-1)^2}{2}(1-\alpha')$$

queries. However, we do not think that (3.20) is true. We think that for fixed finite n and for each $0 < c < 1$ we can find a bad-enough matrix $\boldsymbol{\Sigma}$ such that

$$c < P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}).$$

In particular this then implies that

$$\sup_{\boldsymbol{\Sigma}\in\mathcal{X}} P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) = 1$$

for all $n \in \mathbb{N}$ and thus,

$$\limsup_{n\to\infty} \sup_{\boldsymbol{\Sigma}\in\mathcal{X}} P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) = 1.$$

Also note that we could prove deduce (3.20) from (3.19), if

$$\limsup_{n\to\infty} \sup_{\boldsymbol{\Sigma}\in\mathcal{X}} P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \leq \sup_{\boldsymbol{\Sigma}\in\mathcal{X}} \limsup_{n\to\infty} P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}). \tag{3.21}$$

First of all observe that the reverse inequality of (3.21) always holds, i.e.

$$\limsup_{n\to\infty} \sup_{\boldsymbol{\Sigma}\in\mathcal{X}} P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \geq \sup_{\boldsymbol{\Sigma}\in\mathcal{X}} \limsup_{n\to\infty} P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}). \tag{3.22}$$

To see this, fix some arbitrary $n \in \mathbb{N}$. Then, for all $\boldsymbol{\Sigma} \in \mathcal{X}$,

$$\sup_{\boldsymbol{\Sigma'}\in\mathcal{X}} P_{\text{sample}}(A, \boldsymbol{\Sigma'}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \geq P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}).$$

As n was arbitrary, it holds for all $\mathbf{\Sigma} \in \mathcal{X}$ that

$$\limsup_{n\to\infty} \sup_{\mathbf{\Sigma}'\in\mathcal{X}} P_{\text{sample}}(A, \mathbf{\Sigma}', \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \geq \limsup_{n\to\infty} P_{\text{sample}}(A, \mathbf{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}).$$

As this result holds for all $\mathbf{\Sigma}$, we can apply the supremum on the right hand side of the inequality and obtain (3.22). However, proving the reverse of (3.22), i.e. (3.21), leads to a problem: Note that for all $n \in \mathbb{N}$ we can find $\mathbf{\Sigma}^*(n) \in \mathcal{X}$ such that

$$\sup_{\mathbf{\Sigma}\in\mathcal{X}} P_{\text{sample}}(A, \mathbf{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \leq P_{\text{sample}}(A, \mathbf{\Sigma}^*(n), \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) + \frac{1}{n}$$

by the definition of the supremum. As the limes superior is subadditive, we then obtain

$$\limsup_{n\to\infty} \sup_{\mathbf{\Sigma}\in\mathcal{X}} P_{\text{sample}}(A, \mathbf{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \leq \limsup_{n\to\infty} P_{\text{sample}}(A, \mathbf{\Sigma}^*(n), \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}).$$

Now, if we would have

$$\limsup_{n\to\infty} P_{\text{sample}}(A, \mathbf{\Sigma}^*(n), \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}) \leq \sup_{\mathbf{\Sigma}\in\mathcal{X}} \limsup_{n\to\infty} P_{\text{sample}}(A, \mathbf{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}),$$

then we would have indeed proven (3.20). But it is not clear why this has to hold, because the sequence $\mathbf{\Sigma}^*(n)$ itself depends on n. If the sequence $\mathbf{\Sigma}^*(n)$ yields a sequence of error probabilities converging to 1, this does not hold. And we think, one can find such a sequence, as we will outline in the following.

Recall the upper bound from (3.16) on $P_{\text{sample}}(A, \mathbf{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)})$. At least the upper bound can be made arbitrarily big, by simultaneously decreasing both d_{min} and ρ_{min}. To see that this simultaneous decrease is possible, consider a Gaussian random vector $\boldsymbol{X}$ with the concentration graph given in figure 3.4.

Figure 3.4.: Example to simultaneously decrease d_{min} and ρ_{min}

Withouth loss of generality we assume that all pairwise correlations are positive. For some arbitrary $\varepsilon > 0$, we can choose $\rho_{2,3} = 1 - \varepsilon$ and fix $\rho_{1,2}$. By Lemma 2.11, $\rho_{1,3} = \rho_{1,2}\rho_{2,3} = \rho_{1,2}(1 - \varepsilon)$. Therefore,

$$d_{min} \leq \rho_{1,2} - \rho_{1,3} = \rho_{1,2}(1 - (1 - \varepsilon)) = \rho_{1,2}\varepsilon.$$

As ε can be arbitrarily small, d_{min} can be arbitrarily small. At the same time, let $\rho_{4,5} = \varepsilon'$ for some $\varepsilon' > 0$ and fix $\rho_{5,6}$. Also choose some $\rho_{1,4} > 0$; this free pairwise

correlation ensures that Lemma 2.11 is true. Now, note that $\rho_{4,5|6} \neq 0$ for the given graph. However,

$$\left|\frac{\varepsilon'\sqrt{1-\rho_{5,6}^2}}{\sqrt{1-\varepsilon'\rho_{5,6}}}\right|$$

can be arbitrarily small for $\varepsilon' \to 0$, and therefore,

$$\begin{aligned}
\left|\frac{\varepsilon'\sqrt{1-\rho_{5,6}^2}}{\sqrt{1-\varepsilon'\rho_{5,6}}}\right| &= \left|\frac{\rho_{4,5}\sqrt{1-\rho_{5,6}^2}}{\sqrt{1-\rho_{4,5}\rho_{5,6}}}\right| \\
&= \left|\frac{\rho_{4,5}(1-\rho_{5,6}^2)}{\sqrt{1-\rho_{4,5}\rho_{5,6}}\sqrt{1-\rho_{5,6}^2}}\right| \\
&\overset{\text{Lemma 2.11}}{=} \left|\frac{\rho_{4,5}-\rho_{4,6}\rho_{5,6}}{\sqrt{1-\rho_{4,6}^2}\sqrt{1-\rho_{5,6}^2}}\right| \\
&= \left|\rho_{4,5|6}\right|
\end{aligned}$$

can be arbitrarily small, which implies that ρ_{min} can be arbitrarily small simultaneously to d_{min} being arbitrary small. The constants in that error bound also do not decrease, therefore, the upper bound on $P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)})$ can be arbitrarily large for fixed n.

An arbitrary large upper bound is of course not implying that the upper-bounded quantity can be arbitrary large as well; but we think that the quantities ρ_{min} and d_{min} are directly influencing $P_{\text{sample}}(A, \boldsymbol{\Sigma}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)})$ and not only through an upper bound.

3.2. Algorithm for binary distributions

Lugosi et al. (2021) briefly mention the extension of their results to discrete data. We will restrict our attention to binary data and build the chapter analogously to Gaussians.

3.2.1. Querying setup

In the Gaussian case, the query complexity was based on querying entries $\Sigma_{u,w}$ (or $\hat{\Sigma}_{u,w}$) from the (sample) covariance matrix. For that, we imagined a covariance oracle that, given some indices u, w, returns the corresponding entry $\Sigma_{u,w}$ (or $\hat{\Sigma}_{u,w}$) from the (sample) covariance matrix.

For the discrete case, we need to clarify what oracle we use. We decided not to base the query complexity on querying entries in the covariance matrix, but querying three-way tables from the set of all three-way tables for a given binary random vector $\boldsymbol{X}$. Note that there are $\binom{|V|}{3}$ unique three-way tables. We do not query entries in the (sample) covariance matrix, because the statistic we use for the sample version, the conditional G^2-statistic, cannot, at least to our knowledge, be calculated by just using the entries of the (sample) covariance matrix. This is really a self-imposed restriction; Loh and Wainwright (2013) have shown that the concentration matrix encodes conditional independence patterns similarly as for Gaussians for almost all binary distributions. Nevertheless, the G^2-statistic is a popular choice for testing these types of independence, see Section 2.7. Moreover, we are able to obtain similar theoretic bounds on the query complexity as for Gaussians, in fact, we have never used that the maximum number of queries for Gaussians is $|V|(|V|+1)/2$; we have just bounded the number of iterations. Therefore, we think that discussing the G^2-statistic and doing simulations with it (see Chapter 4) is still worthwile.

To calculate the G^2-statistic we need to know the total number $n_{u,w,v}(i_u, i_w, i_v)$ of observations for each cell (i_u, i_w, i_v) in the three-way table and marginal counts that can be derived from that three-way table. Therefore, in the sample case, we can always calculate $G^2(\hat{p}_{u,w,v}; \hat{p}_v)$ for given distinct u, w, v. In the non-sample case we query the three-way table that contains the true cell probabilities. Note that by knowing these true cell probabilities, we can calculate $\Sigma_{u,v}, \Sigma_{u,w}, \Sigma_{v,w}, \Sigma_{u,u}, \Sigma_{w,w}$, and $\Sigma_{v,v}$. We illustrate this with $\Sigma_{u,w}$. Given the cell probabilities $p_{u,w}(i_u, i_w)$ for

all cells (i_u, i_w), we have in the binary case that

$$\begin{aligned}
\Sigma_{u,w} &= \mathbb{E}[X_u X_w] - \mathbb{E}[X_u]\mathbb{E}[X_w] \\
&= \text{Prob}(X_u = 0, X_w = 0) \cdot 0 \cdot 0 \\
&\quad + \text{Prob}(X_u = 0, X_w = 1) \cdot 0 \cdot 1 \\
&\quad + \text{Prob}(X_u = 1, X_w = 0) \cdot 1 \cdot 0 \\
&\quad + \text{Prob}(X_u = 1, X_w = 1) \cdot 1 \cdot 1 \\
&\quad - \text{Prob}(X_u = 0) \cdot 0 \cdot \text{Prob}(X_w = 0) \cdot 0 \\
&\quad - \text{Prob}(X_u = 0) \cdot 0 \cdot \text{Prob}(X_w = 1) \cdot 1 \\
&\quad - \text{Prob}(X_u = 1) \cdot 1 \cdot \text{Prob}(X_w = 0) \cdot 0 \\
&\quad - \text{Prob}(X_u = 1) \cdot 1 \cdot \text{Prob}(X_w = 1) \cdot 1 \\
&= p_{u,w}(1,1) - p_u(1)p_w(1).
\end{aligned}$$

Note that $p_{u,w}(1,1)$, $p_u(1)$ and $p_w(1)$ can be calculated out of any non-sample three-way table which has u and v as its columns. Therefore, we can calculate $\Sigma_{u,v}, \Sigma_{u,w}, \Sigma_{v,w}, \Sigma_{u,u}, \Sigma_{w,w}$, and $\Sigma_{v,v}$ based on a three-way table if and only if the respective three-way table has columns u, w, v.

3.2.2. Finding central vertices

We need to derive binary versions of `sCentral` and `sCentralSample`, which we call `sCentralBinary` and `sCentralBinarySample`. As discussed in Zwiernik (2019) or Lugosi et al. (2021), it is also true for binary distributions that $\rho_{u,w|v} = 0$ (or equivalenty $\det(\mathbf{\Sigma}_{uv,vw}) = 0$ or $\rho_{u,w} = \rho_{w,v}\rho_{w,u}$) implies that X_u and X_w are independent given X_v. Therefore, `sCentralBinary` has the same form as `sCentral`.

Algorithm 12: sCentralBinary(V)

```
Parameter κ;
ŝ(v) := 0;
for all v ∈ V do
    for i = 1, ..., κ do
        Pick u, w uniformly at random in V \ {v};
        if det(Σ_{uv,vw}) ≠ 0 then
            ŝ(v) := ŝ(v) + 1/κ;
        end
    end
end
return arg min_v ŝ(v)
```

For the sample version, we again need to turn the condition that $X_u \perp\!\!\!\perp X_w | X_v$ into a decision based on data. We do this by testing the hypotheses

$$H_{0,(v,i)} : X_u \perp\!\!\!\perp X_w | X_v \text{ vs. } H_{1,(v,i)} : \text{ not } H_{0,(v,i)}.$$

Again, the index (v, i) indicates the $v \in V$ in the outer loop and the $i \in \{1, \ldots, \kappa\}$ in the inner loop of `sCentralBinarySample`.

We have seen in Section 2.7 that the G^2-statistic for testing conditional independence is a usual choice. Therefore, we reject $H_{0,(v,i)}$ if $G^2(\hat{p}_{u,w,v}; \hat{p}_v) > (\chi^2)^{-1}_{1-\alpha_{v,i},2}$. We can choose the significance level similarly as in the Gaussian case, either by multiple testing methods or by setting the level to some constant value $\alpha \in (0, 1)$.

Algorithm 13: sCentralBinarySample(V)

```
Parameter κ;
ŝ_sample(v) := 0;
for all v ∈ V do
    for i = 1, ..., κ do
        Pick u, w uniformly at random in V \ {v};
        Test H_{0,(v,i)} at some level α_{v,i} ∈ (0, 1);
        if H_{0,(v,i)} is rejected then
            ŝ_sample(v) := ŝ_sample(v) + 1/κ;
        end
    end
end
return arg min_v ŝ_sample(v)
```

Perspective 1: Multiple Testing

We define `sCentralBinarySampleBonferroni` and `sCentralBinarySampleHolm` similarly as in the Gaussian case; the results can be obtained in a similar manner.

Proposition 3.36. *Let $G = (V, E)$ be an undirected graph and $p \in \mathcal{M}_+(G)$. The time and query complexity of computing $\hat{v} =$ `sCentralBinary`(V) are $\mathcal{O}(|V|\kappa)$. Moreover, for any $\delta > 0$,*

$$\text{Prob}\Big(s(\hat{v}) \geq s(v^o) + 2\delta \Big) \leq 2|V| \exp\big(-2\delta^2 \kappa \big).$$

Here, Prob$(\cdot)$ *stands for any probability calculation under the regime that u, w are sampled uniformly at random at each iteration of* `sCentralBinary`.

Proof. The proof is exactly the same as for Gaussians: `sCentral` and `sCentralBinary` are very similar algorithms (if not to say the same up to the notion of querying). Furthermore and as discussed earlier, both the Gaussian case and the binary case share the decomposition of pairwise correlations over trees, see Lemma 2.13 (thus checking $\det(\mathbf{\Sigma}_{uv,vw}) = 0$ is in both cases equivalent to whether $X_u \perp\!\!\!\perp X_w | X_v$). Hence, the probabilistic part of the proposition is true for `sCentralBinary` as well. As there are at most $|V|\kappa$ iterations, there are at most $|V|\kappa$ queries of three-way tables. Thus, the query part of the proposition is also true. □

We similarly obtain a result for the time and query complexity for the sample version of `sCentralBinary`.

Proposition 3.37. *The time and query complexity of* `sCentralBinarySampleBonferroni` *(or its gFWER generalization) are*

$$\mathcal{O}(|V|\kappa).$$

The query complexity of `sCentralBinarySampleHolm` *(or its gFWER generalization) is*

$$\mathcal{O}(|V|\kappa),$$

whereas the time complexity is

$$\mathcal{O}(|V|\kappa \ln(\kappa)).$$

Proof. Clearly, the time complexity is the same as for Gaussians (in the sense that they have the same order), because the algorithms in both cases are basically the same. Our query complexity in the binary case is given by the number of queried three-way tables. Clearly, the number of queries is bounded by the total number of iterations times some constant, therefore the query complexity in both cases is $\mathcal{O}(|V|\kappa)$. □

We continue with the analogy of results.

Proposition 3.38. *Let $G = (V, E)$ be an undirected graph and $p \in \mathcal{M}_+(G)$. Let $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$ be i.i.d. random vectors with the same distribution as $\boldsymbol{X}$ and let*

$$\overline{X} := \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{X}^{(i)}.$$

Let $\tilde{v} =$ `sCentralBinarySample`(V). *Assume that each hypothesis in* `sCentralBinarySample` *(e.g. by the Bonferroni or Holm version) is tested such that*

$$|V| \max_v \limsup_{n\to\infty} \mathrm{FWER}_v \leq \alpha.$$

Then, for any $\delta > 0$,

$$\limsup_{n\to\infty} \mathrm{Prob}\Big(s(\tilde{v}) \geq s(v^o) + 2\delta \Big) \leq \alpha + 2|V| \exp\Big(-\frac{\delta^2}{2}\kappa \Big).$$

Here, the probability refers to the randomization within `sCentralBinarySample` *and the i.i.d. sampling scheme for the observations* $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$.

Proof. Recall the proof of the analog proposition for Gaussians (Proposition 3.4). We needed to bound three terms. The first term was

$$\limsup_{n\to\infty} \mathrm{Prob}\Big(\max_v |\hat{s}(v) - s(v)| \geq \frac{\delta}{2} \Big).$$

Similarly as for Gaussians, we can still apply Hoeffding's inequality (c.f. Hoeffding (1963)) to bound that term, because the distributional assumption referred to the uniform sampling scheme in each iteration and not the distribution of the underlying random vector $\boldsymbol{X}$.

The second term was

$$\limsup_{n\to\infty} \mathrm{Prob}\Big(\max_v \big\{ \hat{s}_{\mathrm{sample}}(v) - \hat{s}(v) \big\} \geq \frac{\delta}{2} \Big).$$

This term is as for Gaussians less or equal than α by the setup of our multiple testing procedure.

The third term was

$$\limsup_{n\to\infty} \mathrm{Prob}(N_{2,v} > 0) = 0,$$

where $N_{2,v}$ was the total number of hypothetical type 2 errors when testing all possible null hypotheses that might be drawn in the iteration corresponding to a particular node $v \in V$. Now, note that for all (i_u, i_w, i_v) it holds by the Strong Law of Large Numbers that $N_{u,w,v}(i_u, i_w, i_v)/n \to p_{u,w,v}(i_u, i_w, i_v)$ almost surely. Similarly, $N_{u,v}(i_u, i_v)/n \to p_{u,v}(i_u, i_v)$, $N_{w,v}(i_w, i_v)/n \to p_{w,v}(i_w, i_v)$ and $N_v(i_v)/n \to p_v(i_v)$ almost surely. Therefore, by the Continuous Mapping Theorem,

$$\ln\left(\frac{N_{u,w,v}(i_u, i_w, i_v) N_v(i_v)}{N_{u,v}(i_u, i_v) N_{w,v}(i_w, i_v)} \right) \longrightarrow \ln\left(\frac{p_{u,w,v}(i_u, i_w, i_v)}{p_{u,v}(i_u, i_v) p_{w,v}(i_w, i_v)} \right)$$

almost surely. Thus, by the Continuous Mapping Theorem again,

$$\sum_{i_v \in \mathcal{I}_v} \sum_{(i_u, i_w) \in \mathcal{I}_{u,w}} N_{u,v}(i_u, i_v)/n \ln\left(\frac{N_{u,w,v}(i_u, i_w, i_v)n}{N_{u,v}(i_u, i_v)N_{w,v}(i_w, i_v)}\right)$$
$$\longrightarrow \sum_{i_v \in \mathcal{I}_v} \sum_{(i_u, i_w) \in \mathcal{I}_{u,w}} p_{u,w,v}(i_u, i_w, i_v) \ln\left(\frac{p_{u,w,v}(i_u, i_w, i_v)}{p_{u,v}(i_u, i_v)p_{w,v}(i_w, i_v)}\right)$$

almost surely. This limit is related to the mutual information (a fact that we will discuss in more detail later in this section). For now, it is enough to know that if $X_u \not\perp\!\!\!\perp X_w | X_v$, then

$$\sum_{i_v \in \mathcal{I}_v} \sum_{(i_u, i_w) \in \mathcal{I}_{u,w}} p_{u,w,v}(i_u, i_w, i_v) \ln\left(\frac{p_{u,w,v}(i_u, i_w, i_v)}{p_{u,v}(i_u, i_v)p_{w,v}(i_w, i_v)}\right) > 0,$$

see e.g. Section 2.3 of Cover and Thomas (2006). This implies that

$$G^2(\hat{p}_{u,w,v}; \hat{p}_v) = 2 \sum_{v \in \mathcal{I}_v} \sum_{(i_u, i_w) \in \mathcal{I}_{u,w}} N_{u,w,v}(i_u, i_w, i_v) \ln\left(\frac{N_{u,w,v}(i_u, i_w, i_v)N_v(i_v)}{N_{u,v}(i_u, i_v)N_{w,v}(i_w, i_v)}\right) \to \infty$$

almost surely if $X_u \not\perp\!\!\!\perp X_w | X_v$. Thus, if $X_u \not\perp\!\!\!\perp X_w | X_v$, we asymptotically almost surely reject. As almost sure convergence implies convergence in probability, hence $\limsup_{n\to\infty} \text{Prob}(N_{2,v} > 0) = 0$. □

Similarly, we obtain the result for a multiple testing procedure based on the gFWER. Here, the gFWER-versions of `sCentralBinary` again follow the same steps as for Gaussians.

Proposition 3.39. *Assume the same assumptions as in Proposition 3.38. But this time, assume that each hypothesis in* `sCentralBinarySample` *(e.g. by the gFWER-generalizations of Bonferroni or Holm) is tested such that*

$$|V| \max_v \limsup_{n\to\infty} \text{gFWER}_v\left(\left\lfloor \frac{\delta\kappa}{2} \right\rfloor + 1\right) \leq \alpha.$$

for some $\delta > 0$. Then, for that particular δ,

$$\limsup_{n\to\infty} \text{Prob}\left(s(\tilde{v}) \geq s(v^o) + 2\delta\right) \leq \alpha + 2|V| \exp\left(-\frac{\delta^2}{2}\kappa\right).$$

Here, the probability refers to the randomization within `sCentralBinarySample` *and the i.i.d. sampling scheme for the observations $(\boldsymbol{X}^{(i)})_{i=1}^n$.*

Proof. The proof is very similar to the one of Proposition 3.38. We omit it. □

Now, by combining Proposition 3.38 and Proposition 3.12, we obtain an analog of Proposition 3.13.

Proposition 3.40. *If $G = (V, E)$ is a tree, $|V| \geq 4$, and $\hat{v} =$ `sCentralBinary`(V), then*

$$\text{Prob}\bigg(c(\hat{v}) > \sqrt{\frac{11}{12}}\bigg) \leq 2|V| \exp\bigg(-\frac{\kappa}{32}\bigg).$$

Here, Prob$(\cdot)$ *stands for any probability calculation under the regime that u, w are sampled uniformly at random at each iteration of* `sCentralBinary`*.*

Proof. The proof is exactly the same as for Gaussians, see Proposition 3.13. □

Similarly, for the sample case.

Proposition 3.41. *Fix $\delta = \frac{1}{8}$. With the same assumptions as in Proposition 3.38 or 3.39 and $|V| \geq 4$, we have*

$$\limsup_{n\to\infty} \text{Prob}\bigg(c(\tilde{v}) > \sqrt{\frac{11}{12}}\bigg) \leq \alpha + 2|V| \exp\bigg(-\frac{\kappa}{128}\bigg).$$

Here the probability refers to the randomization within `sCentralSample` *and the i.i.d. sampling scheme for the observations $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$.*

Proof. The proof is exactly the same as for Gaussians, see Proposition 3.14. □

Perspective 2: Testing each hypothesis at the same significance level α)

In the Gaussian case, we used a lemma from Kalisch and Bühlmann (2007) (see Lemma 3.15), which were based on Hotelling (1953). These results are only applicable to Gaussians and not the binary or discrete case. Therefore, we need to derive a similar lemma for the G^2-statistic first.

There are several approaches in the literature to obtain concentration inequalities for related quantities of the G^2-statistic. Early examples are Hoeffding (1965), Hoeffding (1967) or Kallenberg (1985), who use Sanov's Theorem (Cover and Thomas (2006), Theorem 11.4.1) to obtain large-deviation results for multinomial distributions, in particular, the likelihood ratio, which we saw is up to constants the G^2-statistic. However, these approaches are not so flexible, especially for deriving bounds on the probability of a type 2 error. Newer approaches for obtaining concentration bounds can be found in Antos and Kontoyiannis (2001) or Agrawal (2020). We use the results from Antos and Kontoyiannis (2001), as this bound is well suited to obtain a lemma similarly to Lemma 3 of Kalisch and Bühlmann (2007) in the Gaussian case.

We start by deriving these bounds for unconditional independence tests, i.e. $X_u \perp\!\!\!\perp X_w$, and then move to conditional independence tests of the form $X_u \perp\!\!\!\perp X_w | X_v$. We formulate these results for general discrete variables, to underline the fact that the reasoning works there as well. We then go back to the binary case in which we were originally interested in.

Let $\boldsymbol{X}$ be a discrete random vector with distribution $\{p(\boldsymbol{i}); \boldsymbol{i} \in \mathcal{I}\}$ for some state space $\mathcal{I}$ and let there be n i.i.d observations $\boldsymbol{X}^{(i)}$ with the same distribution as $\boldsymbol{X}$. Let the empirical distribution of p be given by

$$\hat{p}(\boldsymbol{i}) := \frac{N(\boldsymbol{i})}{n}$$

for all $\boldsymbol{i} \in \mathcal{I}$. Let the entropy of some discrete probability measure q on $\mathcal{I}$ be defined by

$$H(q) := -\sum_{\boldsymbol{i} \in \mathcal{I}} q(\boldsymbol{i}) \ln(q(\boldsymbol{i})).$$

For some p, we also call $H(p)$ the population entropy and $H(\hat{p})$ the sample entropy. Note that $H(\hat{p})$ is the plug-in and maximum likelihood estimator of $H(p)$. However, it does not necessarily hold that $H(\hat{p})$ is an unbiased estimator of $H(p)$ (see e.g. Antos and Kontoyiannis (2001)). ibid. furthermore show the following result.[6]

Lemma 3.42. *For all $n \in \mathbb{N}$ and $\varepsilon > 0$,*

$$\text{Prob}\left(\left| H(\hat{p}) - \mathbb{E}\big[H(\hat{p})\big] \right| > \varepsilon \right) \leq 2 \exp\left(\frac{-n\varepsilon^2}{2 \ln^2(n)} \right).$$

The expectation is with respect to p.

Remark 3.43. The number of components of $\boldsymbol{X}$ and the precise form of p do not matter directly for the upper bound (however, they may matter indirectly via a possible choice ε). Besides that, the bound does not seem very tight; we only have $n/\ln^2(n)$ in the exponent.

We later also need to bound the bias $\mathbb{E}[H(\hat{p})] - \hat{p}$. An interesting argument is given in Paninski (2003), Proposition 1 and the part before that. It holds that the empirical entropy (with respect to the true entropy) is negatively biased. That means

$$\mathbb{E}\big[H(\hat{p})\big] - \hat{p} \leq 0.$$

[6] In the original paper, Antos and Kontoyiannis use $\log_2$ instead of the natural logarithm. But as $\log_2(x) = \ln(x)/\ln(2)$, all following results hold for the natural logarithm as well.

To see this upper bound, and a corresponding lower bound, ibid. suggests the following expansion, which can be easily verified by some algebra.[7] It holds that

$$H(\hat{p}) = H(p) + \sum_{\boldsymbol{i}\in\mathcal{I}} \big(p(\boldsymbol{i}) - \hat{p}(\boldsymbol{i})\big) \ln(p(\boldsymbol{i})) - D_{KL}(\hat{p}; p), \tag{3.23}$$

whereas $D_{KL}(\hat{p}; p)$ is the Kullback-Leibler divergence between $\hat{p}$ and p. It is given by

$$D_{KL}(\hat{p}; p) := \sum_{\boldsymbol{i}\in\mathcal{I}} \hat{p}(\boldsymbol{i}) \ln\left(\frac{\hat{p}(\boldsymbol{i})}{p(\boldsymbol{i})}\right).$$

The term $\sum_{\boldsymbol{i}\in\mathcal{I}}(p(\boldsymbol{i}) - \hat{p}(\boldsymbol{i})) \ln(p(\boldsymbol{i}))$ in equation (3.23) has mean 0 (with respect to p), because $\mathbb{E}[\hat{p}(\boldsymbol{i})] = p(\boldsymbol{i})$ for all $\boldsymbol{i} \in \mathcal{I}$. Therefore,

$$\mathbb{E}\big[H(\hat{p})\big] - H(p) = -\mathbb{E}\big[D_{KL}(\hat{p}; p)\big].$$

A well-known fact is that $D_{KL}(\hat{p}; p) \geq 0$ and equality holds if and only if $\hat{p}(i) = p(i)\ \forall i \in \mathcal{I}$ (see e.g. Cover and Thomas (2006), Section 2.3). This shows that

$$\mathbb{E}[H(\hat{p})] - H(p) \leq 0.$$

Gibbs and Su (2002) explore several relationships between "metrics"[8] of probability measures. In particular, for the Kullback-Leibler divergence one can obtain that

$$0 \leq D_{KL}(\hat{p}; p) \leq \ln(1 + \chi^2(\hat{p}; p)), \tag{3.24}$$

whereas

$$\chi^2(\hat{p}; p) = \sum_{i\in\mathcal{I}} \frac{(\hat{p}(i) - p(i))^2}{p(i)^2}$$

is the Pearson chi-square functional.[9] It holds that

$$\mathbb{E}\big[\chi^2(\hat{p}; p)\big] = \frac{|\mathrm{supp}(p)| - 1}{n} \leq \frac{|\mathcal{I}| - 1}{n} \tag{3.25}$$

[7] There are some deeper connections to the theory of empirical processes and the Frechet derivative for distributions. In particular, $\sum_{i\in\mathcal{I}}\big(p(\boldsymbol{i}) - \hat{p}(\boldsymbol{i})\big) \ln(p(\boldsymbol{i})) = -dH\big(p; \hat{p} - p\big)$, where $dH\big(p; \hat{p} - p\big)$ denotes the Frechet derivative of H with respect to p in the direction of $\hat{p} - p$. We do not need this level of understanding for our argumentation, hence we will not cover it and refer the reader to Paninski (2003) and Serfling (1980).

[8] We put this term in quotation marks, as the Kullback-Leiber divergence is not a metric in the strict sense. It does not satisfy symmetry and the triangle inequality, see Gibbs and Su (2002).

[9] We have now used the letter χ to indicate the chi-square statistic, the chi-square distribution with its corresponding quantiles, and now to indicate the functional. It should be clear from the context and the precise notation of subscript, superscript and subsequent brackets to which of these notions we refer to.

for all p. The support of p, denoted by $\operatorname{supp}(p)$, contains all cells with nonzero probability. Under our usual assumptions $\operatorname{supp}(p) = \mathcal{I}$. Now, combining the (3.24) and (3.25) together with Jensen's inequality for the logarithm yields

$$\begin{aligned} -\ln\left(1 + \frac{|\mathcal{I}| - 1}{n}\right) &= -\ln\big(1 + \chi^2(\hat{p}; p)\big) \\ &\leq -\mathbb{E}\Big[\ln\big(1 + \chi^2(\hat{p}; p)\big)\Big] \\ &\leq -\mathbb{E}\Big[D_{KL}(\hat{p}; p)\Big] \\ &= \mathbb{E}\Big[H(\hat{p})\Big] - H(p). \end{aligned}$$

Combining all prior steps then yields the following lemma (Proposition 1 in Paninski (2003)).

Lemma 3.44. *It holds that*

$$-\ln\left(1 + \frac{|\mathcal{I}| - 1}{n}\right) \leq \mathbb{E}\big[H(\hat{p})\big] - \hat{p} \leq 0.$$

Remark 3.45. All prior results can of course be applied to subvectors $\boldsymbol{X_A}$ of $\boldsymbol{X}$ with their respective marginals p_A for some $A \subseteq V$.

Now, for $u \neq w$, let $\{p_{u,w}(i_u, i_w) : (i_u, i_w) \in \mathcal{I}_{u,w}\}$ denote the marginal distribution of (X_u, X_w) and let $\{p_u(i_u) : i_u \in \mathcal{I}_u\}$ resp. $\{p_w(i_w) : i_w \in \mathcal{I}_w\}$ denote the marginal distribution of X_u resp. X_w. The mutual information between X_u and X_w is defined by

$$I(p_{u,w}) := \sum_{(i_u, i_w) \in \mathcal{I}_{u,w}} p_{u,w}(i_u, i_w) \ln\left(\frac{p_{u,w}(i_u, i_w)}{p_u(i_u) p_w(i_w)}\right). \tag{3.26}$$

Note that $I(p_{u,w}) \geq 0$ and equality holds if and only if $X_u \perp\!\!\!\perp X_w$. Furthermore, we use the convention that $0 \ln(0/0) = 0$ and $0 \ln(0/\text{positive term}) = 0$ (c.f. Cover and Thomas (2006), Section 2.3). Furthermore (see e.g. ibid., Section 2.4),

$$I(p_{u,w}) = H(p_u) + H(p_w) - H(p_{u,w}). \tag{3.27}$$

The empirical information for some $p_{u,w}$ is then given by $I(\hat{p}_{u,w})$. By Lemma 3.44 and equation (3.27), we can derive the following lemma.

Lemma 3.46. *Let $u \neq w$. Then, for all $n \in \mathbb{N}$ and $\varepsilon > 0$,*

$$\operatorname{Prob}\Big(\big|I(\hat{p}_{u,w}) - \mathbb{E}\big[I(\hat{p}_{u,w})\big]\big| > \varepsilon\Big) \leq 6\exp\bigg(\frac{-n\varepsilon^2}{18\ln^2(n)}\bigg).$$

Proof. We have

$$\begin{aligned}
&\operatorname{Prob}\Big(\big|I(\hat{p}_{u,w}) - \mathbb{E}\big[I(\hat{p}_{u,w})\big]\big| > \varepsilon\Big) \\
&\quad= \operatorname{Prob}\Big(\big|H(\hat{p}_u) + H(\hat{p}_w) - H(\hat{p}_{u,w}) - \mathbb{E}\big[H(\hat{p}_u) + H(\hat{p}_w) - H(\hat{p}_{u,w})\big]\big| > \varepsilon\Big) \\
&\quad= \operatorname{Prob}\Big(\big|H(\hat{p}_u) - \mathbb{E}\big[H(\hat{p}_u)\big] + H(\hat{p}_w) - \mathbb{E}\big[H(\hat{p}_w)\big] \\
&\qquad\qquad -\big(H(\hat{p}_{u,w}) - \mathbb{E}\big[H(\hat{p}_{u,w})\big]\big)\big| > \varepsilon\Big) \\
&\quad\leq \operatorname{Prob}\Big(\big|H(\hat{p}_u) - \mathbb{E}\big[H(\hat{p}_u)\big]\big| + \big|H(\hat{p}_w) - \mathbb{E}\big[H(\hat{p}_w)\big]\big| \\
&\qquad\qquad +\big|H(\hat{p}_{u,w}) - \mathbb{E}\big[H(\hat{p}_{u,w})\big]\big| > \varepsilon\Big) \\
&\quad\leq \operatorname{Prob}\Big(\big|H(\hat{p}_u) - \mathbb{E}\big[H(\hat{p}_u)\big]\big| > \frac{\varepsilon}{3} \text{ or } \big|H(\hat{p}_w) - \mathbb{E}\big[H(\hat{p}_w)\big]\big| > \frac{\varepsilon}{3} \\
&\qquad\qquad \text{or } \big|H(\hat{p}_{u,w}) - \mathbb{E}\big[H(\hat{p}_{u,w})\big]\big| > \frac{\varepsilon}{3}\Big) \\
&\overset{\text{union bound}}{\leq} \operatorname{Prob}\Big(\big|H(\hat{p}_u) - \mathbb{E}\big[H(\hat{p}_u)\big]\big| > \frac{\varepsilon}{3}\Big) \\
&\qquad + \operatorname{Prob}\Big(\big|H(\hat{p}_w) - \mathbb{E}\big[H(\hat{p}_w)\big]\big| > \frac{\varepsilon}{3}\Big) \\
&\qquad + \operatorname{Prob}\Big(\big|H(\hat{p}_{u,w}) - \mathbb{E}\big[H(\hat{p}_{u,w})\big]\big| > \frac{\varepsilon}{3}\Big) \\
&\overset{\text{Lemma 3.42}}{\leq} 3\cdot 2\exp\bigg(\frac{-n(\varepsilon/3)^2}{2\ln^2(n)}\bigg) \\
&\quad= 6\exp\bigg(\frac{-n\varepsilon^2}{18\ln^2(n)}\bigg).
\end{aligned}$$

□

So far, we have derived a bound on $|I(\hat{p}_{u,w}) - \mathbb{E}[I(\hat{p}_{u,w})]|$, however, we want to have a bound on $|I(\hat{p}_{u,w}) - I(p_{u,w})|$. Unfortunately, the empirical mutual information is not necessarily an unbiased estimator of the mutual information, as was the

case for the entropy. Therefore, we need to work a little bit on that. The lemma and the corresponding proof are given in the following.

Lemma 3.47. *Let $u \neq w$. Then, for all $n \in \mathbb{N}$ and $\varepsilon > 6\ln(1 + (|\mathcal{I}_{u,w}| - 1)/n)$,*

$$\text{Prob}\bigg(|I(\hat{p}_{u,w}) - I(p_{u,w})| > \varepsilon\bigg) \leq 6\exp\bigg(\frac{-n\varepsilon^2}{72\ln^2(n)}\bigg).$$

Proof. We have

$$\begin{aligned}
&\text{Prob}\bigg(\big|I(\hat{p}_{u,w}) - I(p_{u,w})\big| > \varepsilon\bigg) \\
&\quad\leq \text{Prob}\bigg(\big|I(\hat{p}_{u,w}) - \mathbb{E}\big[I(\hat{p}_{u,w})\big] + \mathbb{E}\big[I(\hat{p}_{u,w})\big] - I(p_{u,w})\big| > \varepsilon\bigg) \\
&\quad\leq \text{Prob}\bigg(\big|I(\hat{p}_{u,w}) - \mathbb{E}\big[I(\hat{p}_{u,w})\big]\big| + \big|\mathbb{E}\big[I(\hat{p}_{u,w})\big] - I(p_{u,w})\big| > \varepsilon\bigg) \\
&\quad\leq \text{Prob}\bigg(\big|I(\hat{p}_{u,w}) - \mathbb{E}\big[I(\hat{p}_{u,w})\big]\big| + \big|\mathbb{E}\big[I(\hat{p}_{u,w})\big] - I(p_{u,w})\big| > \varepsilon\bigg) \\
&\quad\leq \text{Prob}\bigg(\big|I(\hat{p}_{u,w}) - \mathbb{E}\big[I(\hat{p}_{u,w})\big]\big| \text{ or } \big|\mathbb{E}\big[I(\hat{p}_{u,w})\big] - I(p_{u,w})\big| > \frac{\varepsilon}{2}\bigg) \\
&\quad\overset{\text{union bound}}{\leq} \text{Prob}\bigg(\big|I(\hat{p}_{u,w}) - \mathbb{E}\big[I(\hat{p}_{u,w})\big]\big| > \frac{\varepsilon}{2}\bigg) \\
&\qquad + \text{Prob}\bigg(\big|\mathbb{E}\big[I(\hat{p}_{u,w})\big] - I(p_{u,w})\big| > \frac{\varepsilon}{2}\bigg) \\
&\quad\leq 6\exp\bigg(\frac{-n(\varepsilon/2)^2}{18\ln^2(n)}\bigg) + \text{Prob}\bigg(\big|\mathbb{E}\big[I(\hat{p}_{u,w})\big] - I(p_{u,w})\big| > \frac{\varepsilon}{2}\bigg) \\
&\quad= 6\exp\bigg(\frac{-n\varepsilon^2}{72\ln^2(n)}\bigg) + \text{Prob}\bigg(\big|\mathbb{E}\big[I(\hat{p}_{u,w})\big] - I(p_{u,w})\big| > \frac{\varepsilon}{2}\bigg). \qquad (3.28)
\end{aligned}$$

Here, we have treated the bias $\mathbb{E}[I(\hat{p}_{u,w})] - I(p_{u,w})$ in some sense as a random variable. But note that the bias is not random, the corresponding probability is either 0 or 1. Instead of the union bound, we could have simply subtracted the absolute value of the bias from both sides of the inequality inside the probability-expression. This is a valid approach, but obfuscates the exponential-term with the bias, which we do not prefer.

We now apply Lemma 3.44 several times to obtain a bound for the bias of the mutual information, i.e.

$$
\begin{aligned}
&\Big|\mathbb{E}[I(\hat{p}_{u,w})] - I(p_{u,w})\Big| \\
&\quad= \Big|\mathbb{E}\big[H(\hat{p}_u) + H(\hat{p}_w) - H(\hat{p}_{u,w})\big] - \big(H(p_u) + H(p_w) - H(p_{u,w})\big)\Big| \\
&\quad= \Big|\mathbb{E}\big[H(\hat{p}_u)\big] - H(p_u) + \mathbb{E}\big[H(\hat{p}_w)\big] - H(p_w) - \big(\mathbb{E}\big[H(\hat{p}_{u,w})\big] - H(p_{u,w})\big)\Big| \\
&\quad\leq \Big|\mathbb{E}\big[H(\hat{p}_u)\big] - H(p_u)\Big| + \Big|\mathbb{E}\big[H(\hat{p}_w)\big] - H(p_w)\Big| + \Big|\mathbb{E}[H(\hat{p}_{u,w})] - H(p_{u,w})\Big| \\
&\quad\leq \ln\left(1 + \frac{|\mathcal{I}_u| - 1}{n}\right) + \ln\left(1 + \frac{|\mathcal{I}_w| - 1}{n}\right) + \ln\left(1 + \frac{|\mathcal{I}_{u,w}| - 1}{n}\right) \\
&\quad\leq 3\ln\left(1 + \frac{|\mathcal{I}_{u,w}| - 1}{n}\right).
\end{aligned}
$$

Therefore, for all $\varepsilon > 6\ln\left(1 + \frac{|\mathcal{I}_{u,w}|-1}{n}\right)$,

$$\mathrm{Prob}\left(\big|\mathbb{E}\big[I(\hat{p}_{u,w})\big] - I(p_{u,w})\big| > \frac{\varepsilon}{2}\right) = 0.$$

Combining this with equation 3.28 yields what we wanted to show, i.e. for all $\varepsilon > 6\ln\left(1 + \frac{|\mathcal{I}_{u,w}|-1}{n}\right)$ it holds that

$$\mathrm{Prob}\left(\big|I(\hat{p}_{u,w}) - I(p_{u,w})\big| > \varepsilon\right) \leq 6\exp\left(\frac{-n\varepsilon^2}{72\ln^2(n)}\right).$$

□

We want to use the previous lemma to obtain similar bounds for the G^2-statistic. Note and recall that

$$
\begin{aligned}
G^2(\hat{p}_{u,w}) &= 2\sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} N_{u,w}(i_u,i_w)\ln\left(\frac{N_{u,w}(i_u,i_w)n}{N_u(i_u)N_w(i_w)}\right) \\
&= 2n\sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} \hat{p}_{u,w}(i_u,i_w)\ln\left(\frac{\hat{p}_{u,w}(i_u,i_w)}{\hat{p}_u(i_u)\hat{p}_w(i_w)}\right) \\
&= 2nI(\hat{p}_{u,w}),
\end{aligned}
$$

whereas $I(\hat{p}_{u,w})$ was the empirical mutual information between X_u and X_w. Similarly, we have

$$G^2(p_{u,w}) = 2nI(p_{u,w}).$$

We can now apply Lemma 3.47 and obtain the following lemma.

Lemma 3.48. *Let $u \neq w$. Then, for all $n \in \mathbb{N}$ and $\varepsilon > 12n\ln(1 + (|\mathcal{I}_{u,w}| - 1)/n)$,*

$$\mathrm{Prob}\Big(\big|G^2(\hat{p}_{u,w}) - G^2(p_{u,w})\big| > \varepsilon\Big) \leq 6\exp\bigg(\frac{-\varepsilon^2}{288n\ln^2(n)}\bigg).$$

Proof. We have

$$\begin{aligned}\mathrm{Prob}\Big(\big|G^2(\hat{p}_{u,w}) - G^2(p_{u,w})\big| > \varepsilon\Big) &= \mathrm{Prob}\Big(\big|I(\hat{p}_{u,w}) - I(\hat{p}_{u,w})\big| > \frac{\varepsilon}{2n}\Big) \\ &\overset{\text{Lemma 3.47}}{\leq} 6\exp\bigg(-n\frac{(\varepsilon/2n)^2}{72\ln^2(n)}\bigg) \\ &= 6\exp\bigg(\frac{-\varepsilon^2}{288n\ln^2(n)}\bigg).\end{aligned}$$

□

Remark 3.49. If we vary n, Lemma 3.48 is only useful if ε increases fast enough in n.

Remark 3.50. Also note that $\lim_{n\to\infty} 12n\ln(1 + (|\mathcal{I}_{u,w}| - 1)/n) = 12(|I_{u,w}| - 1)$ (we will show a similar fact later in a different context). This particular lower bound on allowed ε is not a later problem for the choices of ε we look at.

We now transfer the results from the unconditional G^2-statistic to the conditional G^2-statistic.

Lemma 3.51. *Let u, w, v be distinct. Then, for all $n \in \mathbb{N}$ and $\varepsilon > 24|\mathcal{I}_v|n\ln(1 + (|\mathcal{I}_{u,w}| - 1)/n)$,*

$$\begin{aligned}&\mathrm{Prob}\Big(\big|G^2(\hat{p}_{u,w,v}; \hat{p}_v) - G^2(p_{u,w,v}; p_v)\big| > \varepsilon\Big) \\ &\qquad \leq 6|\mathcal{I}_v|\exp\bigg(\frac{-\varepsilon^2}{1152|\mathcal{I}_v|^2 n\ln^2(n)}\bigg) \\ &\qquad + 2|\mathcal{I}_v|\exp\bigg(\frac{-\varepsilon^2}{8n|\mathcal{I}_v|^2(\max_{i_v\in\mathcal{I}_v}\{I(p_{u,w|X_v=i_v})\})^2}\bigg).\end{aligned}$$

Proof. Recall that

$$\begin{aligned}
G^2(\hat{p}_{u,w,v};\hat{p}_v) &= 2n \sum_{i_v\in\mathcal{I}_v} \sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} \hat{p}_{u,w,v}(i_u,i_w,i_v)\ln\left(\frac{\hat{p}_{u,w,v}(i_u,i_w,i_v)\hat{p}_v(i_v)}{\hat{p}_{u,v}(i_u,i_v)\hat{p}_{w,v}(i_w,i_v)}\right)\\
&= 2n \sum_{i_v\in\mathcal{I}_v} \hat{p}_v(i_v)\\
&\qquad \sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} \hat{p}_{u,w|X_v=i_v}(i_u,i_w,i_v)\ln\left(\frac{\hat{p}_{u,w|X_v=i_v}(i_u,i_w|i_v)}{\hat{p}_{u|X_v=i_v}(i_u|i_v)\hat{p}_{w|X_v=i_v}(i_w|i_v)}\right)\\
&= \sum_{i_v\in\mathcal{I}_v} \hat{p}_v(i_v) G^2(\hat{p}_{u,w|X_v=i_v}).
\end{aligned}$$

Now, for each fixed value of v, we can apply Lemma 3.48 on $G^2(\hat{p}_{u,w|X_v=i_v})$. We can see this as looking at the (scaled by $2n$) mutual information between X_u and X_w if they have joint distribution $p_{u,w|X_v=i_v}$ with marginal distributions $p_{u|X_v=i_v}$ and $p_{w|X_v=i_v}$. We obtain that for all $n\in\mathbb{N}$ and $\varepsilon > 12n\ln(1+(|\mathcal{I}_{u,w}|-1)/n)$

$$\mathrm{Prob}\left(\left|G^2(\hat{p}_{u,w|X_v=i_v}) - G^2(p_{u,w|X_v=i_v})\right| > \varepsilon\right) \le 6\exp\left(\frac{-\varepsilon^2}{288n\ln^2(n)}\right). \tag{3.29}$$

To conclude for $G^2(\hat{p}_{u,w,v};\hat{p}_v)$ note that

$$\begin{aligned}
&\mathrm{Prob}\left(\left|G^2(\hat{p}_{u,w,v};\hat{p}_v) - G^2(p_{u,w,v};p_v)\right| > \varepsilon\right)\\
&\quad= \mathrm{Prob}\Bigg(\Big|\sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(\hat{p}_{u,w|X_v=i_v}) - \sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(p_{u,w|X_v=i_v})\\
&\qquad + \sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(p_{u,w|X_v=i_v}) - \sum_{i_v\in\mathcal{I}_v} p_v(i_v)G^2(p_{u,w|X_v=i_v})\Big| > \varepsilon\Bigg)\\
&\quad\le \mathrm{Prob}\Bigg(\Big|\sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(\hat{p}_{u,w|X_v=i_v}) - \sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(p_{u,w|X_v=i_v})\Big|\\
&\qquad + \Big|\sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(p_{u,w|X_v=i_v}) - \sum_{i_v\in\mathcal{I}_v} p_v(i_v)G^2(p_{u,w|X_v=i_v})\Big| > \varepsilon\Bigg)\\
&\quad\le \mathrm{Prob}\Bigg(\Big|\sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(\hat{p}_{u,w|X_v=i_v}) - \sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(p_{u,w|X_v=i_v})\Big| > \frac{\varepsilon}{2}\\
&\qquad \text{or } \Big|\sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(p_{u,w|X_v=i_v}) - \sum_{i_v\in\mathcal{I}_v} p_v(i_v)G^2(p_{u,w|X_v=i_v})\Big| > \frac{\varepsilon}{2}\Bigg)\\
&\quad\overset{\text{union bound}}{\le} \mathrm{Prob}\Bigg(\Big|\sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(\hat{p}_{u,w|X_v=i_v}) - \sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(p_{u,w|X_v=i_v})\Big| > \frac{\varepsilon}{2}\Bigg)
\end{aligned}$$

$$+\operatorname{Prob}\Big(\Big|\sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(p_{u,w|X_v=i_v})-\sum_{i_v\in\mathcal{I}_v}p_v(i_v)G^2(p_{u,w|X_v=i_v})\Big|>\frac{\varepsilon}{2}\Big). \tag{3.30}$$

We bound each of the two terms in the upper bound separately. For the first term we have

$$\begin{aligned}
&\operatorname{Prob}\Big(\Big|\sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(\hat{p}_{u,w|X_v=i_v})-\sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(p_{u,w|X_v=i_v})\Big|>\frac{\varepsilon}{2}\Big)\\
&\quad\le\operatorname{Prob}\Big(\sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)\Big|G^2(\hat{p}_{u,w|X_v=i_v})-G^2(p_{u,w|X_v=i_v})\Big|>\frac{\varepsilon}{2}\Big)\\
&\quad\le\operatorname{Prob}\Big(\bigcup_{i_v\in\mathcal{I}_v}\Big\{\hat{p}_v(i_v)\Big|G^2(\hat{p}_{u,w|X_v=i_v})-G^2(p_{u,w|X_v=i_v})\Big|>\frac{\varepsilon}{2|\mathcal{I}_v|}\Big\}\Big)\\
&\quad\overset{\text{union bound}}{\le}\sum_{i_v\in\mathcal{I}_v}\operatorname{Prob}\Big(\hat{p}_v(i_v)\Big|G^2(\hat{p}_{u,w|X_v=i_v})-G^2(p_{u,w|X_v=i_v})\Big|>\frac{\varepsilon}{2|\mathcal{I}_v|}\Big)\\
&\quad\le\sum_{i_v\in\mathcal{I}_v}\operatorname{Prob}\Big(\Big|G^2(\hat{p}_{u,w|X_v=i_v})-G^2(p_{u,w|X_v=i_v})\Big|>\frac{\varepsilon}{2|\mathcal{I}_v|}\Big)\\
&\quad\overset{(3.29)}{\le}\sum_{i_v\in\mathcal{I}_v}6\exp\Big(\frac{-(\varepsilon/(2|\mathcal{I}_v|))^2}{288n\ln^2(n)}\Big)\\
&\quad=6|\mathcal{I}_v|\exp\Big(\frac{-\varepsilon^2}{1152|\mathcal{I}_v|^2n\ln^2(n)}\Big)
\end{aligned}$$

for all $n\in\mathbb{N}$ and $\varepsilon>24|\mathcal{I}_v|n\ln(1+(|\mathcal{I}_{u,w}|-1)/n)$. For the second term we have

$$\begin{aligned}
&\operatorname{Prob}\Big(\Big|\sum_{i_v\in\mathcal{I}_v}\hat{p}_v(i_v)G^2(p_{u,w|X_v=i_v})-\sum_{i_v\in\mathcal{I}_v}p_v(i_v)G^2(p_{u,w|X_v=i_v})\Big|>\frac{\varepsilon}{2}\Big)\\
&\quad\le\operatorname{Prob}\Big(\sum_{i_v\in\mathcal{I}_v}G^2(p_{u,w|X_v=i_v})\big|\hat{p}_v(i_v)-p_v(i_v)\big|>\frac{\varepsilon}{2}\Big)\\
&\quad\le\operatorname{Prob}\Big(|\mathcal{I}_v|\max_{i_v\in\mathcal{I}_v}\{G^2(p_{u,w|X_v=i_v})\}\big|\hat{p}_v(i_v)-p_v(i_v)\big|>\frac{\varepsilon}{2}\Big)\\
&\quad\le\operatorname{Prob}\Big(\max_{i_v\in\mathcal{I}_v}\{G^2(p_{u,w|X_v=i_v})\}\big|\hat{p}_v(i_v)-p_v(i_v)\big|>\frac{\varepsilon}{2|\mathcal{I}_v|}\Big)\\
&\quad\le\operatorname{Prob}\Big(\bigcup_{i_v\in\mathcal{I}_v}\Big\{G^2(p_{u,w|X_v=i_v})\big|\hat{p}_v(i_v)-p_v(i_v)\big|\Big\}>\frac{\varepsilon}{2|\mathcal{I}_v|}\Big)\\
&\quad\overset{\text{union bound}}{\le}\sum_{i_v\in\mathcal{I}_v}\operatorname{Prob}\Big(G^2(p_{u,w|X_v=i_v})\big|\hat{p}_v(i_v)-p_v(i_v)\big|>\frac{\varepsilon}{2|\mathcal{I}_v|}\Big)
\end{aligned}$$

$$
\begin{aligned}
&\leq \sum_{i_v \in \mathcal{I}_v} \text{Prob}\left(\left| \hat{p}_v(i_v) - p_v(i_v) \right| > \frac{\varepsilon}{2|\mathcal{I}_v| G^2(p_{u,w|X_v=i_v})} \right) \\
&\overset{(*)}{\leq} \sum_{i_v \in \mathcal{I}_v} 2 \exp\left[-2n \left(\frac{\varepsilon}{2|\mathcal{I}_v| G^2(p_{u,w|X_v=i_v})} \right)^2 \right] \\
&= \sum_{i_v \in \mathcal{I}_v} 2 \exp\left(-n \frac{\varepsilon^2}{2|\mathcal{I}_v|^2 (G^2(p_{u,w|X_v=i_v}))^2} \right) \\
&= \sum_{i_v \in \mathcal{I}_v} 2 \exp\left(\frac{-\varepsilon^2}{8n|\mathcal{I}_v|^2 (I(p_{u,w|X_v=i_v}))^2} \right) \\
&\leq 2|\mathcal{I}_v| \exp\left(\frac{-\varepsilon^2}{8n|\mathcal{I}_v|^2 (\max_{i_v \in \mathcal{I}_v} \{I(p_{u,w|X_v=i_v})\})^2} \right)
\end{aligned}
$$

for all $n \in \mathbb{N}$ and $\varepsilon > 0$. For inequality (*) we have used Hoeffding's inequality (see e.g. Hoeffding (1963)).[10] Plugging both bounds back into equation (3.30) yields that for all $n \in \mathbb{N}$ and $\varepsilon > 24|\mathcal{I}_v| n \ln(1 + (|\mathcal{I}_{u,w}| - 1)/n)$

$$
\begin{aligned}
&\text{Prob}\left(\left| G^2(\hat{p}_{u,w,v}; \hat{p}_v) - G^2(p_{u,w,v}; p_v) \right| > \varepsilon \right) \\
&\qquad \leq 6|\mathcal{I}_v| \exp\left(\frac{-\varepsilon^2}{1152|\mathcal{I}_v|^2 n \ln^2(n)} \right) \\
&\qquad\qquad + 2|\mathcal{I}_v| \exp\left(\frac{-\varepsilon^2}{8n|\mathcal{I}_v|^2 (\max_{i_v \in \mathcal{I}_v} \{I(p_{u,w|X_v=i_v})\})^2} \right).
\end{aligned}
$$

□

We have finally arrived at a lemma that is the discrete analog to Lemma 3.15 (Lemma 3 in Kalisch and Bühlmann (2007)) for the Gaussian case. We are now able to proceed similarly as for the Gaussian case. With this result we can bound the probability of a type 1 error and the probability of a type 2 error. At this point, we restrict to the binary case again. Doing so is not strictly necessary, but this makes the notation especially for the degrees of freedom a little bit lighter. For the general discrete case, one needs to replace 2 degrees of freedom for the chi-square distribution by $(|d_u| - 1)(|d_w| - 1)d_v$-degrees of freedom.

[10] Sanov's Theorem would have also worked (see e.g. Cover and Thomas (2006), Theorem 11.4.1). However, the choice we have made is more user-friendly for our later argumentation.

We define

$$G^2_{min} := \min_{v\in V;\, u,w\in V\setminus\{v\};\, X_u \not\perp\!\!\!\perp X_w | X_v} G^2(p_{u,w,v}; p_v),$$
$$G^2_{max} := \max_{v\in V;\, u,w\in V\setminus\{v\};\, X_u \not\perp\!\!\!\perp X_w | X_v} G^2(p_{u,w,v}; p_v).$$

Let

$$I(p_{u,w,v}; p_v) := \sum_{i_v\in\mathcal{I}_v} \sum_{(i_u,i_w)\in\mathcal{I}_{u,w}} p_{u,w,v}(i_u, i_w, i_v) \ln\left(\frac{p_{u,w,v}(i_u, i_w, i_v)p_v(n, i_v)}{p_{u,v}(i_u, i_v)p_{w,v}(i_w, i_v)}\right).$$

This term is also known as the conditional mutual information X_u and X_w given X_v. Clearly, $2nI(p_{u,w,v}; p_v) = G^2(p_{u,w,v}; p_v)$. Furthermore, we define

$$I_{min} := \min_{v\in V;\, u,w\in V\setminus\{v\};\, X_u \not\perp\!\!\!\perp X_w | X_v} I(p_{u,w,v}; p_v),$$
$$I_{max} := \max_{v\in V;\, u,w\in V\setminus\{v\};\, X_u \not\perp\!\!\!\perp X_w | X_v} I(p_{u,w,v}; p_v).$$

By this, $I_{min} > 0$ and

$$G^2_{min} = 2nI_{min} > 0.$$

It holds that $I(p_{u,w,v}; p_v) = 0$ if and only if $X_u \perp\!\!\!\perp X_w | X_v$ (see Cover and Thomas (2006), Section 2.6). Thus loosely speaking, I_{min} measures how well non-independencies can be distinguished from independencies. We start by looking at a fixed vertex size $|V|$ in n and assume that I_{min} and I_{max} are also constant in n. We look at particular significance levels of the form

$$\alpha := \alpha_{n,k} := 1 - \chi^2_2(n^k I_{min}), \tag{3.31}$$

whereas χ^2_2 denotes the chi-square distribution function with 2 degrees of freedom and

$$k := k(n) \in \left(\max\left\{\frac{1}{2}, \frac{\ln(\frac{48}{I_{min}} \ln(1 + \frac{3}{n}))}{\ln(n)} + 1\right\}, \frac{\ln(1 - \frac{48}{I_{min}} \ln(1 + \frac{3}{n}))}{\ln(n)} + 1\right). \tag{3.32}$$

For all given fixed choices of I_{min}, the term $\ln((48/I_{min}) \ln(1 + 3/n))/(\ln(n)) + 1$ converges to 0 for $n \to \infty$ from above, the term $\ln(1-(48/I_{min}) \ln(1+3/n))/ \ln(n)+ 1$ converges to 1 from below. This can be seen by applying the rule of L'Hospital. First of all note that

$$\begin{aligned}\lim_{n\to\infty} n \ln\left(1 + \frac{3}{n}\right) &= \lim_{n\to\infty} \frac{\ln(1 + \frac{3}{n})}{1/n} \\ &= \lim_{n\to\infty} \frac{\frac{1}{1+\frac{3}{n}}(-3)n^{-2}}{-n^{-2}} \\ &= 3.\end{aligned}$$

We then have

$$\begin{aligned}\lim_{n\to\infty} \frac{\ln(\frac{48}{I_{min}} \ln(1+\frac{3}{n}))}{\ln(n)} &= \lim_{n\to\infty} \frac{\frac{-3}{(1+\frac{3}{n})n^2 \ln(1+\frac{3}{n})}}{\frac{1}{n}} \\ &= \lim_{n\to\infty} \frac{-3}{(1+\frac{3}{n})n \ln(1+\frac{3}{n})} \\ &= -1.\end{aligned}$$

Moreover,

$$\lim_{n\to\infty} \frac{\ln(1-\frac{48}{I_{min}} \ln(1+\frac{3}{n}))}{\ln(n)} = 0.$$

Thus, the given interval in (3.32) is never empty for large enough n; the interval also gets larger for increasing n due to monotonicity, thus, we could also fix some allowed $k_0 := k(n_0)$ for some $n_0 \in \mathbb{N}$ and use k_0 for all $n \geq n_0$. We also note that, as $k(n) \in (1/2, 1)$, $\alpha \to 0$ for $n \to \infty$. We point out that this construction of α is very similar to the construction in the Gaussian case. For the binary case, we have the chi-square distribution instead of the normal distribution and I_{min} instead of ρ_{min}, see Kalisch and Bühlmann (2007) or equation (3.5) in this thesis.

Now, note that $X_u \perp\!\!\!\perp X_w | X_v$ if and only if $G^2(p_{u,w,v}; p_v) = 0$. Recall that we reject $X_u \perp\!\!\!\perp X_w | X_v$ at level $\alpha \in (0,1)$ if $|G^2(\hat{p}_{u,w,v}; \hat{p}_v)| = G^2(\hat{p}_{u,w,v}; \hat{p}_v) > (\chi^2)^{-1}_{1-\alpha,2}$, whereas $(\chi^2)^{-1}_{1-\alpha,2}$ is the $1-\alpha$-quantile of a chi-square distribution with

2 degrees of freedom. Therefore, the supremal probability of a type 1 error is

$$\begin{aligned}
&\sup_{v\in V;\, u,w\in V\setminus\{v\};\, X_u \perp\!\!\!\perp X_w|X_v} \mathrm{Prob}\left(G^2(\hat{p}_{u,w,v};\hat{p}_v) > (\chi^2)^{-1}_{1-\alpha,2}\right) \\
&\quad= \sup_{v\in V;\, u,w\in V\setminus\{v\};\, X_u \perp\!\!\!\perp X_w|X_v} \mathrm{Prob}\left(\left|G^2(\hat{p}_{u,w,v};\hat{p}_v) - G^2(p_{u,w,v};p_v)\right| > (\chi^2)^{-1}_{1-\alpha,2}\right) \\
&\quad= \sup_{v\in V;\, u,w\in V\setminus\{v\};\, X_u \perp\!\!\!\perp X_w|X_v} \mathrm{Prob}\left(\left|G^2(\hat{p}_{u,w,v};\hat{p}_v) - G^2(p_{u,w,v};p_v)\right| > n^k I_{min}\right) \\
&\quad\overset{\text{Lemma 3.51}}{\leq} \sup_{v\in V;\, u,w\in V\setminus\{v\};\, X_u \perp\!\!\!\perp X_w|X_v} \left[12\exp\left(\frac{-(n^k I_{min})^2}{4608 n \ln^2(n)}\right)\right. \\
&\qquad\left. + 4\exp\left(\frac{-(n^k I_{min})^2}{32n(\max_{i_v\in\{0,1\}}\{I(p_{u,w|X'_v=i_v})\})^2}\right)\right] \\
&\quad= 12\exp\left(\frac{-n^{2k-1} I_{min}^2}{4608\ln^2(n)}\right) \\
&\qquad + 4\exp\left(\frac{-n^{2k-1} I_{min}^2}{32(\max_{v\in V;\, u,w\in V\setminus\{v\};\, X_u \not\perp\!\!\!\perp X_w|X_v} \max_{i_v\in\{0,1\}}\{I(p_{u,w|X_v=i_v})\})^2}\right) \\
&\quad\leq 12\exp\left(\frac{-n^{2k-1} I_{min}^2}{4608\ln^2(n)}\right) \\
&\qquad + 4\exp\left(\frac{-n^{2k-1} I_{min}^2}{32 I_{max}^2}\right). \qquad (3.33)
\end{aligned}$$

This bound converges to 0 for all $k > 1/2$, because then

$$\frac{n^{2k-1}}{\ln^2(n)} \to \infty$$

and

$$n^{2k-1} \to \infty$$

for $n \to \infty$. However, note that Lemma 3.51 is only applicable for $\varepsilon = n^k I_{min} > 48n\ln(1+3/n)$. If we choose $k = k(n) > \ln((48/I_{min})\ln(1+3/n))/\ln(n) + 1$, the assumption is satisfied for all $n \in \mathbb{N}_{>1}$. Therefore, the bound in (3.33) converges to 0 if we choose $k = k(n) > \max\{1/2, \ln((48/I_{min})\ln(1+3/n))/\ln(n)\} + 1$.

To obtain a bound on the supremal probability of a type 2 error note that we reject the null hypothesis $X_u \perp\!\!\!\perp X_w|X_v$ at level α if $|G^2(\hat{p}_{u,w,v};\hat{p}_v)| = G^2(\hat{p}_{u,w,v};\hat{p}_v) \leq (\chi^2)^{-1}_{1-\alpha,2}$. Thus, the supremal probability of a type 2 error is given by

$$\begin{aligned}
&\sup_{v\in V;\, u,w\in V\setminus\{v\};\, X_u \not\perp\!\!\!\perp X_w|X_v} \mathrm{Prob}(G^2(\hat{p}_{u,w,v};\hat{p}_v) \leq (\chi^2)^{-1}_{1-\alpha,2}) \\
&\quad= \sup_{v\in V;\, u,w\in V\setminus\{v\};\, X_u \not\perp\!\!\!\perp X_w|X_v} \mathrm{Prob}(G^2(\hat{p}_{u,w,v};\hat{p}_v) \leq n^k I_{min}).
\end{aligned}$$

We now make use of the following implications for all $(u, w, v) \in \{v' \in V;\ u', w' \in V \setminus \{v'\} :\ X_{u'} \not\perp\!\!\!\perp X_{w'} | X_{v'} \text{ and } u' \neq w'\}$. It holds that

$$\begin{aligned}
G^2(\hat{p}_{u,w,v}; \hat{p}_v) &\leq n^k I_{min} \\
&\implies -G^2(\hat{p}_{u,w,v}; \hat{p}_v) \geq -n^k I_{min} \\
&\implies G^2_{min} - G^2(\hat{p}_{u,w,v}; \hat{p}_v) \geq G^2_{min} - n^k I_{min} \\
&\implies G^2_{min} - G^2(\hat{p}_{u,w,v}; \hat{p}_v) \geq I_{min} n(2 - n^{k-1}) \\
&\overset{X_u \not\perp\!\!\!\perp X_w | X_v}{\implies} G^2(p_{u,w,v}; p_v) - G^2(\hat{p}_{u,w,v}; \hat{p}_v) \geq I_{min} n(2 - n^{k-1}) \\
&\implies G^2(p_{u,w,v}; p_v) - G^2(\hat{p}_{u,w,v}; \hat{p}_v) > I_{min} n(1 - n^{k-1}) \\
&\implies \left| G^2(p_{u,w,v}; p_v) - G^2(\hat{p}_{u,w,v}; \hat{p}_v) \right| > I_{min} n(1 - n^{k-1}).
\end{aligned}$$

Note that $I_{min} n(1 - n^{k-1}) > 0$ for all $n \in \mathbb{N}$ if and only if $k < 1$. Furthermore, the assumption of Lemma 3.51, i.e. $\varepsilon = I_{min} n(1 - n^{k-1}) > 48n \ln(1 + 3/n)$, is satisfied for all $k < 1$ and large enough n, or said differently, for a fixed n and all $k = k(n) < \ln(1 - (48/I_{min}) \ln(1 + 3/n))/\ln(n) + 1$. Therefore,

$$\begin{aligned}
&\sup_{v \in V;\ u,w \in V \setminus \{v\};\ X_u \not\perp\!\!\!\perp X_w | X_v} \text{Prob}\left(G^2(\hat{p}_{u,w,v}; \hat{p}_v) \leq (\chi^2)^{-1}_{1-\alpha,2} \right) \\
&\leq \sup_{v \in V;\ u,w \in V \setminus \{v\};\ X_u \not\perp\!\!\!\perp X_w | X_v} \text{Prob}\left(\left| G^2(\hat{p}_{u,w,v}; \hat{p}_v) - G^2(p_{u,w,v}; p_v) \right| > I_{min} n(1 - n^{k-1}) \right) \\
&\overset{\text{Lemma 3.51}}{\leq} \sup_{v \in V;\ u,w \in V \setminus \{v\};\ X_u \not\perp\!\!\!\perp X_w | X_v} \left[12 \exp\left(\frac{-(I_{min} n(1 - n^{k-1}))^2}{4608 n \ln^2(n)} \right) \right. \\
&\qquad \left. + 4 \exp\left(\frac{-(I_{min} n(1 - n^{k-1}))^2}{32n(\max_{i_v \in \{0,1\}} \{I(p_{u,w|X_v = i_v})\})^2} \right) \right] \\
&= 12 \exp\left(\frac{-I^2_{min} n(1 - n^{k-1})^2}{4608 \ln^2(n)} \right) \\
&\qquad + 4 \exp\left(\frac{-I^2_{min} n(1 - n^{k-1})^2}{32(\max_{v \in V;\ u,w \in V \setminus \{v\};\ X_u \not\perp\!\!\!\perp X_w | X_v} \max_{i_v \in \{0,1\}} \{I(p_{u,w|X_v = i_v})\})^2} \right) \\
&\leq 12 \exp\left(\frac{-I^2_{min} n(1 - n^{k-1})^2}{4608 \ln^2(n)} \right) \\
&\qquad + 4 \exp\left(\frac{-I^2_{min} n(1 - n^{k-1})^2}{32 I^2_{max}} \right) \qquad (3.34)
\end{aligned}$$

Both terms converge to 0 (for $n \to \infty$) for all $0 < k < 1$, because for these choices of k,

$$\frac{n(1 - n^{k-1})^2}{\ln(n)^2} \to \infty$$

and

$$n(1-n^{k-1})^2 \to \infty$$

for $n \to \infty$.

For the procedure `sCentralBinarySample` this then implies (for the values of k given in (3.32)).

$$\begin{aligned}\text{Prob}\Big(\exists v \in V : \hat{s}_{\text{sample}}(v) > \hat{s}(v)\Big) \\ \leq 12|V|\kappa \ \exp\left(\frac{-n^{2k-1}I_{min}^2}{4608\ln^2(n)}\right) \\ + 4|V|\kappa \ \exp\left(\frac{-n^{2k-1}I_{min}^2}{32I_{max}^2}\right).\end{aligned} \tag{3.35}$$

and

$$\begin{aligned}\text{Prob}\Big(\exists v \in V : n_{2,v} > 0\Big) \\ \leq 12|V|\kappa \ \exp\left(\frac{-I_{min}^2 n(1-n^{k-1})^2}{4608\ln^2(n)}\right) \\ + 4|V|\kappa \ \exp\left(\frac{-I_{min}^2 n(1-n^{k-1})^2}{32I_{max}^2}\right).\end{aligned} \tag{3.36}$$

Therefore, we obtain centrality results for `sCentralBinarySample`, similar to Proposition 3.17 in the Gaussian case.

Proposition 3.52. *Let $G = (V, E)$ be an undirected graph and $p \in \mathcal{M}_+(G)$. Then, there exists $\alpha \in (0, 1)$ with $\alpha \to 0$ for $n \to \infty$ such that if each hypothesis in* `sCentralBinarySample` *is tested at level α, it holds that*

$$\begin{aligned}\text{Prob}\Big(s(\tilde{v}) \geq s(v^o) + 2\delta\Big) \leq 2|V|\exp\left(-\frac{\delta^2}{2}\kappa\right) \\ + 12|V|\kappa\left[\exp\left(\frac{-n^{2k-1}I_{min}^2}{4608\ln^2(n)}\right)\right. \\ \left. + \exp\left(\frac{-I_{min}^2 n(1-n^{k-1})^2}{4608\ln^2(n)}\right)\right] \\ + 4|V|\kappa\left[\exp\left(\frac{-n^{2k-1}I_{min}^2}{32I_{max}^2}\right)\right. \\ \left. + \exp\left(\frac{-I_{min}^2 n(1-n^{k-1})^2}{32I_{max}^2}\right)\right].\end{aligned}$$

Particular choices of α and k are discussed in (3.31) *and in* (3.32) *respectively.*

We then also have an analog of Proposition 3.18 by the same proof.

Proposition 3.53. *Assume the assumptions from Proposition* (3.52)*. Furthermore, assume that* $|V| \geq 4$*. Then, there exists* $\alpha \in (0,1)$ *with* $\alpha \to 0$ *for* $n \to \infty$ *such that if each hypothesis in* `sCentralSample` *is tested at level* α*, it holds that*

$$\begin{aligned}\text{Prob}\left(c(\tilde{v}) > \sqrt{\frac{11}{12}}\right) \leq 2|V|\exp\left(-\frac{\kappa}{128}\right) \\ + 12|V|\kappa\left[\exp\left(\frac{-n^{2k-1}I_{min}^2}{4608\ln^2(n)}\right) + \exp\left(\frac{-I_{min}^2 n(1-n^{k-1})^2}{4608\ln^2(n)}\right)\right] \\ + 4|V|\kappa\left[\exp\left(\frac{-n^{2k-1}I_{min}^2}{32I_{max}^2}\right) + \exp\left(\frac{-I_{min}^2 n(1-n^{k-1})^2}{32I_{max}^2}\right)\right].\end{aligned}$$

Remark 3.54. Here, we see that a larger I_{min} leads to lower upper bounds. Loosely speaking, I_{min} measures how well conditional non-independencies can be distinguished from conditional independencies. That result is similar to the Gaussian case, where ρ_{min} takes the role of I_{min}. Therefore, in both distribution settings, we need a strong distinction between conditional independencies and conditional non-independencies to achieve faster convergence rate with respect to the sample size.

High-dimensional settings

We also imagine the case where $|V| = |V_n| = \mathcal{O}(n^a)$ for some $0 \leq a < \infty$ and that $\kappa = \kappa_n = \mathcal{O}(|V_n|) = \mathcal{O}(n^a)$. Moreover, we assume that $\kappa_n > 2a\ln(n)/\delta^2$ for some $\delta > 0$. Note that so far, this is the same setup as for Gaussians.
Furthermore, we assume that $1/I_{min} = \mathcal{O}(n^l)$ (equivalently $I_{min} = \Omega(n^{-l})$) with $0 < l \leq \frac{1}{2} - c$ for some arbitrary $c > 0$ and $1/I_{max} = \Omega(I_{min})$. This says that I_{min} is not decreasing too fast while I_{max} is not increasing too fast in n.

Recall that in the Gaussian setting the analog was $1/\rho_{min} = \mathcal{O}(n^l)$, here for $0 < l < 1/2$. We also had that the maximal partial correlation was bounded from above by some constant M. This is even a slightly stronger than the requirement that $1/I_{max} = \Omega(I_{min})$ in the binary case.

We define the value of α similarly to the one in (3.31), this time k needs to be an element in a slightly different set.

$$k := k(n) \in \left(\max\left\{\frac{1}{2} + l, \frac{\ln(\frac{48}{I_{min}}\ln(1+\frac{3}{n}))}{\ln(n)} + 1\right\}, \frac{\ln(1-\frac{48}{I_{min}}\ln(1+\frac{3}{n}))}{\ln(n)} + 1\right). \tag{3.37}$$

This bound behaves very similar to the one for non-high-dimensional settings (see equation 3.32). For some $C > 0$ we have that

$$\begin{aligned}
\lim_{n\to\infty} \frac{\ln(\frac{48}{I_{min}} \ln(1+\frac{3}{n}))}{\ln(n)} &= \lim_{n\to\infty} \frac{\ln(48\ln(1+\frac{3}{n})) + \ln(\frac{1}{I_{min}})}{\ln(n)} \\
&\leq \lim_{n\to\infty} \frac{\ln(48\ln(1+\frac{3}{n})) + \ln(Cn^l)}{\ln(n)} \\
&= \lim_{n\to\infty} \frac{\ln(48\ln(1+\frac{3}{n})) + l\ln(Cn)}{\ln(n)} \\
&= -1 + l.
\end{aligned}$$

To bound the upper term, we first note that by the rule of L'Hospital,

$$\begin{aligned}
\lim_{n\to\infty} 48n^l \ln(1+\frac{3}{n}) &= \lim_{n\to\infty} \frac{48\ln(1+\frac{3}{n})}{\frac{1}{n^l}} \\
&= \lim_{n\to\infty} \frac{48\ln(1+\frac{3}{n})}{\frac{1}{n^l}} \\
&= \lim_{n\to\infty} \frac{\frac{-144}{n^2+3n}}{\frac{-l}{n^{l+1}}} \\
&= \lim_{n\to\infty} \frac{-144n^l}{-l(n+3)} \\
&= 0 \qquad (3.38)
\end{aligned}$$

as $0 < l \leq 1/2 - c < 1$.

Now as $1/I_{min} = \mathcal{O}(n^l)$, there is a constant $C > 0$ such that $1/I_{min} \leq Cn^l$. Equation (3.38) implies that $48Cn^l \ln(1+3/n))$ and hence $(48/I_{min})\ln(1+3/n))$ get arbitrarely close to 0. Thus, $1-48Cn^l \ln(1+3/n))$ and hence $1-(48/I_{min})\ln(1+3/n))$ are positive for large enough n. We can now use the fact that $\ln(1-x)$ is monotonically decreasing for all $x > 0$, which yields that

$$\frac{\ln(1-\frac{48}{I_{min}}\ln(1+\frac{3}{n}))}{\ln(n)} \geq \frac{\ln(1-48Cn^l\ln(1+\frac{3}{n}))}{\ln(n)}.$$

Moreover, equation (3.38) also implies that

$$\lim_{n\to\infty} \frac{\ln(1-48Cn^l\ln(1+\frac{3}{n}))}{\ln(n)} = 0.$$

Thus,

$$\lim_{n\to\infty} \frac{\ln(1-\frac{48}{I_{min}}\ln(1+\frac{3}{n}))}{\ln(n)} \geq 0,$$

which implies that the upper bound in (3.37) is asymptotically bounded from below by 1.

Therefore, for large enough n we can find a suitable k. All together, we abbreviate the previous assumptions for the high-dimensional binary case by (A-HD-B).

We now revisit (3.35) and (3.36). These are true in the high-dimensional setting as well; the arguments prior are the same. Therefore, we obtain that

$$\begin{aligned}
&\text{Prob}\Big(\exists v \in V : \hat{s}_{\text{sample}}(v) > \hat{s}(v)\Big) \\
&\qquad \leq 12|V_n|\kappa \ \exp\left(\frac{-n^{2k-1}I_{min}^2}{4608\ln^2(n)}\right) \\
&\qquad\quad + 4|V_n|\kappa \ \exp\left(\frac{-n^{2k-1}I_{min}^2}{32I_{max}^2}\right). \\
&\qquad \leq \mathcal{O}\left(n^{2a}\exp\left(\frac{-n^{2k-1}n^{-2l}}{4608\ln^2(n)}\right)\right) \\
&\qquad\quad + \mathcal{O}\left(n^{2a}\exp\left(\frac{-n^{2k-1}}{32}\right)\right) \\
&\qquad \leq \mathcal{O}\left(\exp\left(2a\ln(n) - \frac{n^{2k-1}n^{-2l}}{4608\ln^2(n)}\right)\right) \\
&\qquad\quad + \mathcal{O}(\exp\left(2a\ln(n) - \frac{n^{2k-1}}{32}\right)\Big).
\end{aligned}$$

By observing that $0 < 2(k-l) - 1 < 1$ for all allowed choices of l and k, we have

$$2a\ln(n) - \frac{n^{2k-1}n^{-2l}}{4608\ln^2(n)} = \frac{9216a\ln(n)^3 - n^{2(k-l)-1}}{4608\ln(n)^2} \to -\infty$$

for $n \to \infty$ (albeit rather slowly). Furthermore, as $k > 1/2$,

$$2a\ln(n) - \frac{n^{2k-1}}{32} \to -\infty.$$

Thus,

$$\text{Prob}\Big(\exists v \in V : \hat{s}_{\text{sample}}(v) > \hat{s}(v)\Big) = o(1).$$

Moreover,

$$\begin{aligned}
&\mathrm{Prob}\Big(\exists v \in V : n_{2,v} > 0\Big) \\
&\qquad \leq 12|V_n|\kappa \ \exp\Big(\frac{-I_{min}^2 n(1-n^{k-1})^2}{4608\ln^2(n)}\Big) \\
&\qquad\quad + 4|V_n|\kappa \ \exp\Big(\frac{-I_{min}^2 n(1-n^{k-1})^2}{32 I_{max}^2}\Big) \\
&\qquad \leq \mathcal{O}\Big(n^{2a}\exp\Big(-\frac{n^{-2l}n(1-n^{k-1})^2}{4608\ln^2(n)}\Big)\Big) \\
&\qquad\quad + \mathcal{O}\Big(n^{2a}\exp\Big(\ln(n) - \frac{n(1-n^{k-1})^2}{32}\Big)\Big) \\
&\qquad \leq \mathcal{O}\Big(\exp\Big(2a\ln(n) - \frac{n^{-2l}n(1-n^{k-1})^2}{4608\ln^2(n)}\Big)\Big) \\
&\qquad\quad + \mathcal{O}\Big(\exp\Big(2a\ln(n) - \frac{n(1-n^{k-1})^2}{32}\Big)\Big).
\end{aligned}$$

Again, by observing that $0 < -2l + 1$ and $k < 1$, we have that

$$2a\ln(n) - \frac{n^{-2l}n(1-n^{k-1})^2}{4608\ln^2(n)} = \frac{9216a\ln(n)^3 - n^{1-2l}(1-n^{k-1})^2}{4608\ln(n)^2} \to -\infty$$

for $n \to \infty$ (albeit rather slowly). Furthermore, because $k < 1$,

$$2a\ln(n) - \frac{n(1-n^{k-1})^2}{32} \to -\infty.$$

Thus,

$$\mathrm{Prob}\Big(\exists v \in V : n_{2,v} > 0\Big) = o(1).$$

Finally, we have

$$\begin{aligned}
\mathrm{Prob}\Big(\max_v |\hat{s}(v) - s(v)| \geq \frac{\delta}{2}\Big) &\leq 2|V_n|\exp\Big(-\frac{\delta^2}{2}\kappa_n\Big) \\
&\leq O\Big(n^a \exp\Big(-\frac{\delta^2}{2}\kappa_n\Big)\Big) \\
&\leq O\Big(\exp\Big(a\ln(n) - \frac{\delta^2}{2}\kappa_n\Big)\Big) \\
&= o(1),
\end{aligned}$$

by the assumed lower bound on κ_n. Therefore, we obtain our binary high-dimensional analog of Proposition 3.52.

Proposition 3.55. *Assume the the high-dimensional assumptions for the binary case (A-B-HD). Then, there exists* $\alpha \in (0,1)$ *with* $\alpha \to 0$ *for* $n \to \infty$ *such that if each hypothesis in* `sCentralBinarySample` *is tested at level* α*, it holds that*

$$\mathrm{Prob}\bigg(s(\tilde{v}) \geq s(v^o) + 2\delta\bigg) = o(1).$$

Particular choices of α *and* k *are discussed in* (3.31) *and in* (3.37)

Then, there is also the binary high-dimensional analog of Proposition 3.53. Again, the proof idea is the same, only the bounds differ.

Proposition 3.56. *Assume the same assumptions as in Proposition 3.55 with* $\delta = 1/8.$. *Furthermore, assume that* $|V| \geq 4$*. Then there exists* $\alpha \in (0,1)$ *with* $\alpha \to 0$ *for* $n \to \infty$ *such that if each hypothesis in* `sCentralSample` *is tested at level* α*, it holds that*

$$\mathrm{Prob}\bigg(c(\tilde{v}) > \sqrt{\frac{11}{12}}\bigg) = o(1).$$

3.2.3. Recovering trees

As explained in the previous section, the Gaussian case and binary case share a convenient factorization, see Lemma 2.11 and Lemma 2.13. Therefore, defining the algorithms `ComponentsTreeBinary` and `ReconstructTreeBinary`) is straightforward.

Algorithm 14: ComponentsTreeBinary(V)

```
w := sCentral(V);
N := ∅;
Sort |ρ_uw| for u ∈ V \ {w} in decreasing order and put them in list B;
for every u in the order of B do
    t:= true;
    for all v ∈ N do
        if det(Σ_{uw,vw}) ≠ 0 then
            V_v := V_v ∪ {u};
            t := false;
        end
    end
    if t = true then
        Ê := Ê ∪ {u, w};
        N := N ∪ {u};
        V_u := {u};
    end
end
return all V_u for u ∈ N
```

Algorithm 15: ReconstructTreeBinary(V)

```
if |V| > 1 then
    V_1, ..., V_m ← ComponentsTreeBinary(V);
    for i from 1 to m do
        ReconstructTreeBinary(V_i);
    end
end
```

We can derive results for `ReconstructTreeBinary`, which are very similar to results in the Gaussian case.

Proposition 3.57. *Algorithm 15 is correct, that is, if $\mathcal{G}(p)$ is a tree T, then* `ReconstructTreeBinary`(V) *gives $\hat{E} = E(T)$.*

Proof. As there is an exact analog of Lemma 2.11 for binary random vectors, namely Lemma 2.13, the proof of this Proposition is the same as the proof for Gaussians, see Proposition 3.21. □

We also obtain an analog proposition for complexity.

Theorem 3.58. *Suppose $\mathcal{G}(p)$ is a tree $T = (V, E)$ with maximum degree $\Delta(T) \leq d$. Fix $\varepsilon < 1$ and define $\kappa = \left\lceil 32 \ln\left(\frac{2|V|^2}{\varepsilon}\right) \right\rceil$ to be the parameter of algorithm 12.*

Then, with probability of at least $1-\varepsilon$*, algorithm 15 requires time and queries of order*

$$\mathcal{O}\left(|V|\ln(|V|)\max\left\{\ln\left(\frac{|V|}{\varepsilon}\right), d\right\}\right).$$

Here, the probability refers to any calculation under the regime that u, w *are sampled uniformly at random in each call of* `sCentralBinary`*.*

Proof. First of all note that all prior results that were needed for the proof in the Gaussian case are analogously true for the binary case.

Now, observe, that there are two reasons for querying, firstly, for obtaining pairwise correlations in the beginning of `ComponentsTreeBinary`, and secondly, for testing conditional independence. To calculate the pairwise correlations, we need to query $\mathcal{O}(|V|)$ three-way tables; that is the same order as in the Gaussian case. Furthermore, we have already seen in the proof of Proposition 3.37 and Section 3.2.1 that we need one particular three-way table for testing one particular conditional independence; in the Gaussian case we need at most 4 entries of the covariance matrix, therefore, the query complexity due to testing conditional independence has the same order in both the Gaussian and the binary case. Therefore, the proof of Theorem 3.25 is valid in the binary case as well. □

Similarly as for Gaussians, we look at deriving suitable testing procedures from the perspective of multiple testing and testing each hypothesis at a constant significance level $\alpha \in (0,1)$. The theory is then very similar.

Perspective 1: Multiple Testing

The algorithms `ReconstructTreeBinarySample` and `ComponentsTreeBinarySample` are basically the same as for Gaussians. This time, we test the hypotheses

$$H_{0,(w,u,v)} : X_u \perp\!\!\!\perp X_v | X_w \text{ vs. } H_{1,(w,u,v)} : \text{ not } H_{0,(w,u,v)}.$$

by the conditional G^2-statistic, introduced in Section 2.7.

Algorithm 16: ReconstructTreeBinarySample (V)

```
if |V| > 1 then
    V_1, ..., V_m ← ComponentsTreeBinarySample(V);
    for i from 1 to m do
        ReconstructTreeBinarySample(V_i);
    end
end
```

Algorithm 17: ComponentsTreeBinarySample(V)

```
w := sCentralSample(V);
N_sample := ∅;
Sort |ρ̂_uw| for u ∈ V \ w in decreasing order and put them in list B_sample;
for every u in the order of B_sample do
    t:= true;
    for all v ∈ N_sample do
        Test H_{0,(w,u,v)} at α'_{w,u,v};
        if H_{0,(w,u,v)} is rejected then
            V_{sample,v} := V_{sample,v} ∪ {u};
            t := false;
        end
    end
    if t = true then
        Ê_sample := Ê_sample ∪ {u, w};
        N_sample := N_sample ∪ {u};
        V_{sample,u} := {u};
    end
end
return all V_{sample,u} for u ∈ N_sample
```

We again use the same warm-start procedure with a Bonferroni bound to obtain an asymptotic control over all type 1 errors of all calls of `ReconstructTreeBinarySample`.

Proposition 3.59. *Let our testing procedure be such that the overall asymptotic FWER (excluding* `sCentralBinarySample`*) of one initial call of* `ReconstructTreeBinarySample` *is* $\leq \alpha'$*. Then asymptotically in* n*, algorithm 16 is correct with probability of at least* $1 - \alpha'$*. That is, if* $\mathcal{G}(p)$ *is a tree* T*, then* `ReconstructTreeBinarySample`(V) *gives* $\hat{E}_{sample} = E(T)$*.*
Here, the probability refers to drawing i.i.d. observations from the underlying binary random vector.

Proof. We have shown in the proof of Proposition 3.38 that the G^2-statistic makes (almost surely) no type 2 errors asymptotically. Furthermore, we have a binary analog of Lemma 2.11, namely Lemma 2.13. We also assume no linear dependence among the components of $\boldsymbol{X}$. Therefore, the proof is exactly the same as for Gaussians. □

Theorem 3.60. *Suppose* $\mathcal{G}(p)$ *is a tree* $T = (V, E)$ *with maximum degree* $\Delta(T) \leq d$*. Let our testing procedure be such that the asymptotic FWER of each call of*

algorithm 13 is $\leq \alpha/|V|$. *Fix* $\alpha < \varepsilon < 1$ *and let* $\kappa = \left\lceil 128 \ln(\frac{2|V|^2}{\varepsilon-\alpha}) \right\rceil$ *be the parameter of algorithm 13. Then, asymptotically with probability of at least* $1-\varepsilon$, *algorithm 16 requires time and queries of order*

$$\mathcal{O}\left(|V| \ln(|V|) \max\left\{ \ln\left(\frac{|V|}{\varepsilon - \alpha} \right), d \right\} \right).$$

if `sCentralSampleBinaryBonferroni` *is used. The query complexity is the same if* `sCentralBinaryHolm` *is used, the time complexity is*

$$\mathcal{O}\left(|V| \ln(|V|) \max\left\{ \ln\left(\frac{|V|}{\varepsilon - \alpha} \right) \ln\left(\ln\left(\frac{|V|}{\varepsilon - \alpha} \right) \right), d \right\} \right).$$

Here, the probability refers to drawing i.i.d. observations from the underlying binary random vector and the randomization of the algorithm. .

Proof. As explained in proof of Theorem 3.58: First of all note that all prior results that were needed for the proof in the Gaussian case are analogously true for the binary case.

Now, observe, that there are two reasons for querying, firstly, for obtaining pairwise sample correlations in the beginning of `ComponentsTreeBinary`, and secondly, for testing conditional independence. To calculate the pairwise sample correlations, we need to query $\mathcal{O}(|V|)$ three-way tables; that is the same order as in the Gaussian case. Furthermore, we have already seen in the proof of Proposition 3.37 and Section 3.2.1 that we need one particular three-way table for testing one particular conditional independence; in the Gaussian case we need at most 6 entries of the covariance matrix, therefore, the query complexity due to testing conditional independence has the same order in both the Gaussian and the binary case. Therefore, the proof of Theorem 3.25 is valid in the binary case as well. □

Perspective 2: Testing each hypothesis at the same significance level α'

Recall, that we use

$$\hat{\rho}_{u,w} = \frac{\sum_{i=1}^n (X_u^{(i)} - \overline{X}_u)(X_w^{(i)} - \overline{X}_w)}{\sqrt{\sum_{i=1}^n (X^{(i)} - \overline{X}_u)^2} \sqrt{\sum_{i=1}^n (X^{(i)} - \overline{X}_w)^2}},$$

given i.i.d. observations $(\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)})$ that have the same distribution as the underlying binary random vector $\boldsymbol{X}$ as an estimator of $\rho_{u,w}$. Here, $\overline{X} = \frac{1}{n} \sum_{i=1}^n \boldsymbol{X}^{(i)}$.

As for Gaussians, we need to give a probabilistic bound on $|\hat{\rho}_{u,w} - \rho_{u,w}| > \gamma$ for some $\gamma > 0$, so we need an analog of Lemma 3.29. The literature on these types

of bounds for binary (or discrete) random vectors is not very well developed in comparison to Gaussians. In Section 2.7 we have seen that $\sqrt{n}(\hat{\rho}_{u,w} - \rho_{u,w}) \longrightarrow \mathcal{N}(0, \eta^2)$ in distribution, where the precise value of η depends on the cross-moments of X_u and X_w and is given in Ferguson (1996), Theorem 8, Section 8. Thus, we expect some form of decay. However, more elaborate results as in Hotelling (1953) or Kalisch and Bühlmann (2007), preferably tight, seem not to be there.

Nevertheless, there is some hope. Several authors have studied the moments of $\hat{\rho}_{u,w}$ for general distributions, see for e.g. Cook (1951a), Cook (1951b) or Subrahmaniam and Gajjar (1980). The formulas for these moments depend on the joint cumulants of X_u and X_w. Due to their length we will only state them conceptually, for the complete expression we refer to the just given sources. Given that the cumulants for binary random vectors are finite, we obtain that

$$\mathbb{E}\big[\hat{\rho}_{u,w}^j\big] = \rho_{u,w}^j + \mathcal{O}(n^{-1}). \tag{3.39}$$

The precise constants depend on the joint cumulants of X_u and X_w. Now (3.39) allows us to obtain an upper bound that is decreasing in n, albeit not exponentially.

Lemma 3.61. *For any $\gamma > 0$ and large enough n,*

$$\sup_{u,w \in V} \mathrm{Prob}\Big(\big|\hat{\rho}_{u,w} - \rho_{u,w}\big| > \gamma\Big) \leq \frac{\mathcal{O}(n^{-1})}{\gamma^2}.$$

Proof. Fix some arbitrary $u, w \in V$. Then, we have

$$\begin{aligned}
&\mathrm{Prob}\Big(\big|\hat{\rho}_{u,w} - \rho_{u,w}\big| > \gamma\Big) \\
&\quad = \mathrm{Prob}\Big(\big|\hat{\rho}_{u,w} - \mathbb{E}[\hat{\rho}_{u,w}] + \mathbb{E}[\hat{\rho}_{u,w}] - \rho_{u,w}\big| > \gamma\Big) \\
&\quad \leq \mathrm{Prob}\Big(\big|\hat{\rho}_{u,w} - \mathbb{E}[\hat{\rho}_{u,w}]\big| + \big|\mathbb{E}[\hat{\rho}_{u,w}] - \rho_{u,w}\big| > \gamma\Big) \\
&\quad \leq \mathrm{Prob}\Big(\big|\hat{\rho}_{u,w} - \mathbb{E}[\hat{\rho}_{u,w}]\big| > \frac{\gamma}{2} \text{ or } \big|\mathbb{E}[\hat{\rho}_{u,w}] - \rho_{u,w}\big| > \frac{\gamma}{2}\Big) \\
&\quad \overset{\text{union bound}}{\leq} \mathrm{Prob}\Big(\big|\hat{\rho}_{u,w} - \mathbb{E}[\hat{\rho}_{u,w}]\big| > \frac{\gamma}{2}\Big) + \mathrm{Prob}\Big(\big|\mathbb{E}[\hat{\rho}_{u,w}] - \rho_{u,w}\big| > \frac{\gamma}{2}\Big).
\end{aligned}$$

We need to bound both terms, we start with $\mathrm{Prob}(|\hat{\rho}_{u,w} - \mathbb{E}[\hat{\rho}_{u,w}]| > \gamma/2)$. As the moments of $\hat{\rho}_{u,w}$ are finite for each n, because $\mathbb{E}[\hat{\rho}_{u,w}^j] \leq \mathbb{E}[1] = 1$ for all $j \in \mathbb{N}$, we

can apply Chebyshev's inequality (Klenke (2020), Theorem 5.11) and obtain

$$\begin{aligned}
&\text{Prob}\bigg(\big|\hat{\rho}_{u,w} - \mathbb{E}[\hat{\rho}_{u,w}]\big| > \frac{\gamma}{2}\bigg) \\
&\qquad \leq 4\frac{\text{Var}\big[\hat{\rho}_{u,w}\big]}{\gamma^2} \\
&\qquad = 4\frac{\mathbb{E}\big[\hat{\rho}_{u,w}^2\big] - \big(\mathbb{E}\big[\hat{\rho}_{u,w}\big]\big)^2}{\gamma^2} \\
&\qquad = 4\frac{\rho_{u,w}^2 + \mathcal{O}(n^{-1}) - \big(\rho_{u,w} + \mathcal{O}(n^{-1})\big)^2}{\gamma^2} \\
&\qquad = \frac{\mathcal{O}(n^{-1})}{\gamma^2}.
\end{aligned}$$

Based on our preparations, the second term is rather easy to bound. Note that this probability is either 0 or 1, as the bias is not a random quantity anymore. We use (3.39) and obtain that

$$\mathbb{E}[\hat{\rho}_{u,w}] - \rho_{u,w} = \mathcal{O}(n^{-1}).$$

Therefore, for n large enough[11],

$$\text{Prob}\bigg(\big|\mathbb{E}[\hat{\rho}_{u,w}] - \rho_{u,w}\big| > \frac{\gamma}{2}\bigg) = 0.$$

□

Similarly as for Gaussians, let

$$d_{min} = \min_{w \in V} \min_{\substack{u,u' \in V \setminus \{w\};\, u \neq u' \\ u \text{ separates } u' \text{ and } w}} \big||\rho_{u,w}| - |\rho_{u',w}|\big|.$$

As we assume no linear dependence between the components of $\boldsymbol{X}$, we obtain a result that analog to Gaussians: Let the significance level α' for testing each hypothesis in `ReconstructTreeBinarySample` be as given in 3.31. Let p_E denote the supremal probability of an error (type 1 or type 2) in these conditional independence tests. We have shown in Section 3.2.2 that

$$\begin{aligned}
p_E := 12\bigg[\exp\bigg(\frac{-n^{2k-1}I_{min}^2}{4608\ln^2(n)}\bigg) + \exp\bigg(\frac{-I_{min}^2 n(1-n^{k-1})^2}{4608\ln^2(n)}\bigg)\bigg] \\
+ 4\bigg[\exp\bigg(\frac{-n^{2k-1}I_{min}^2}{32I_{max}^2}\bigg) + \exp\bigg(\frac{-I_{min}^2 n(1-n^{k-1})^2}{32I_{max}^2}\bigg)\bigg].
\end{aligned}$$

[11] For details on "how large" we need to know the precise constants. For that, see Cook (1951a), Cook (1951b) and Subrahmaniam and Gajjar (1980).

Then, for large enough but still finite n, the procedure `ReconstructTreeBinarySample` returns the correct tree with probability

$$\begin{aligned}
\text{Prob}(\hat{\mathcal{G}} = \mathcal{G}(p)) &\geq \left[1 - \frac{\mathcal{O}(n^{-1})}{d_{min}^2}\right]^{n_{\text{Calls}}} \cdot \prod_{i=1}^{n_{\text{Calls}}} \left[1 - N_{H,i}\, p_E\right] \\
&\geq \left[1 - \frac{\mathcal{O}(n^{-1})}{d_{min}^2}\right]^{n_{\text{Calls}}} \cdot \left[1 - \big(|V| - 1\big) \cdot d \cdot p_E)\right]^{n_{\text{Calls}}} \\
&\geq \left[1 - \frac{\mathcal{O}(n^{-1})}{d_{min}^2}\right]^{n_{\text{Calls}}} \cdot \left[1 - \big(|V| - 1\big)^2 \cdot p_E)\right]^{n_{\text{Calls}}} \\
&\geq \left[1 - \frac{\mathcal{O}(n^{-1})}{d_{min}^2}\right]^{|V|-1} \cdot \left[1 - \big(|V| - 1\big)^2 \cdot p_E)\right]^{|V|-1}.
\end{aligned}$$

Letting $n \to \infty$ makes the right hand side converge to 1. We can also prove an analog of Theorem 3.30 for binary data.

Theorem 3.62. *Suppose $\mathcal{G}(p)$ is a tree $T = (V, E)$ with maximum degree $\Delta(T) \leq d$. Let $\alpha = \alpha_n$ be as defined in 3.31 be the significance level in algorithm 13 at which each hypothesis is tested. Furthermore, fix $\varepsilon > |V|^3 p_E =: t$ and let $\kappa = \left\lceil 128 \ln(\frac{2|V|^2}{\varepsilon - t}) \right\rceil$ be the parameter of algorithm 13. Then, asymptotically with probability of at least $1 - \varepsilon$, algorithm 16 requires time and queries of order*

$$\mathcal{O}\left(|V| \ln(|V|) \max\left\{ \ln\left(\frac{|V|}{\varepsilon - t}\right), d \right\}\right).$$

Here, the probability refers to drawing i.i.d. observations from the underlying binary random vector and the randomization of the algorithm.

Proof. The proof is very similar to the one of Theorem 3.30, because we have done the same preparation. □

4. Simulations

4.1. The general setup

We start this Section by giving some remarks on the underlying technical preconditions. The simulations have been executed on 6 servers which we have rented from Amazon Web Services. Each instance was a t2.2xlarge with 32GB RAM and 8 threads on an Intel-Xeon-core. We set up each server using the templates of Aslett (2020).

We have implemented all mentioned algorithms and (later to be introduced) benchmark algorithms in R (R Core Team (2021)); our particular R-version was 4.0.2.[1] We note that there are already implementations for some of the algorithms available, but we wanted to use the same-level programming language to make the results comparable. We give appropriate references for the R-packages in the appendix.

In the Gaussian case, `ReconstructTree` and `ReconstructTreeSample` are useful when size prohibits storing or estimating the entire (sample) covariance matrix. In our theoretical discussions for Gaussians, we have imagined a (sample) covariance oracle that takes two indices as input and returns the corresponding entry of the (sample) covariance matrix as output. We have implemented that particular querying setup. We calculate the covariance matrix beforehand and each algorithm then queries from an already calculated matrix.

But there are other possibilities to implement querying and to define what constitutes a query in practice. Querying from an oracle may not be the most practically relevant implementation. We may as well be required to read in data first and calculate each entry of the covariance matrix and also that can be done in several ways. Each querying implementation introduces a specific time cost associated to one query. Querying directly from an oracle has a lower time cost than querying from some external system and calculating each entry first. Thus, the overall time complexity depends on the implementation and how costly it is in terms of time to do an extra query. In our theoretical discussion, we have separated time and query complexity, but in practice these two are intimately related. Algorithms with a

[1] The code can be found under the URL `https://github.com/TomHochsprung/Master-Thesis`

high query complexity are expected to have a rather low time complexity when a query does not cost much time, but a rather high time complexity when a query is very expensive in terms of time.

For most of the simulations, we imagine querying from an oracle, so the setup when a query does not cost much. Nevertheless, we want to make some comparisons of different querying setups. We also consider to read in data from a csv-file componentwise using the fread-function from the datatable-package (Dowle and Srinivasan (2021)). We either read in all observations at once or more slowly, observation by observation and then calculate the respective entry of the sample covariance matrix in an online-fashion using the Welford algorithm, see e.g Knuth (1998), Welford (1962) or Chan et al. (1983).

We also do some simulations for the binary case; even though they are not as elaborate as for Gaussians. We similarly imagine an oracle case, where the respective three-way tables with the corresponding G^2-statistic have already been calculated.

In this thesis, we have derived several sample versions of `ReconstructTreeSample` using two different paradigms. For the simulations, we focus on the version where each hypothesis is tested at the same significance level α resp. α'. We use several values of κ; we use abbreviations to indicate which algorithm and value of κ has been used, e.g. `RTS_10` indicates the algorithm `ReconstructTreeSample` with no multiple testing adjustment and a value of $\kappa = 10$.
We proceed in the same manner for the binary case. The algorithm `ReconstructTreeBinarySample` is abbreviated by `RTBS`.
We also do some simulation for `sCentralSample`, we will abbreviate that algorithm by `sCS`.

4.2. Evaluation metrics

We are interested in correctness, the time complexity and the query complexity of `ReconstructTreeSample` and the soon to be introduced benchmark algorithms. We measure time complexity by measuring the run time. We measure query complexity by measuring the fraction of entries in the (upper half) of the sample covariance matrix that have been queried in the Gaussian case or the number of unique queried three-way tables for the binary case.

We use two different measures of correctness:

- 0-1 correctness[2]: Either the returned tree is correct (1) or it is not (0).

[2]This term is not standard in the literature; we have made it up.

- Structural Hamming Distance (SHD) as discussed by Kalisch and Bühlmann (2007) or Tsamardinos et al. (2006): In the case of undirected graphs, the SHD is the number of wrongly included edges plus the number of falsely omitted edges.

We remark that there are other measures of correctness. Tsamardinos et al. (2006) also discuss the Kullback-Leibler divergence, which indirectly measures wrong edges. However, there are two issues with that measure. Firstly, wrong edges are only indirectly penalized. Secondly, one usually uses or needs to use an empirical version instead of the actual Kullback-Leiber divergence, because a non-empirical version requires probabilistic inferences to be made for the networks in comparison; each may take an exponential amount of time with respect to the number of variables.

4.3. Benchmark algorithms

4.3.1. Chow-Liu algorithm

Several algorithms for reconstructing concentration graphs exist, we refer to chapter 3 of Drton and Maathuis (2017) for a recent exposition. We have chosen the Chow-Liu algorithm as one of our benchmark algorithms. The Chow-Liu algorithm is originally due to Chow and Liu (1968) and reconstructs the structure of distributions that factorize with respect to a tree.

We give a brief outline of the Chow-Liu algorithm, following Drton and Maathuis (2017). We focus on the discrete case, which was originally considered by Chow and Liu and then give some comments on the Gaussian case. The Chow-Liu algorithm uses a helpful factorization property: Let $\boldsymbol{X}$ be a random vector such that its distribution factorizes with respect to a tree, i.e

$$p(\boldsymbol{i}) = \prod_{vw \in E} \frac{p_{vw}(i_v, i_w)}{p_v(i_v) p_w(i_w)} \prod_{v \in V} p_v(i_v). \tag{4.1}$$

Given some i.i.d. sample $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(n)}$, the goal is to find a tree $T = (V, E)$ that maximizes the maximum log-likelihood

$$\hat{L}(T) := \sum_{\boldsymbol{i} \in \mathcal{I}} N(\boldsymbol{i}) \log(\hat{p}_T(\boldsymbol{i})).$$

Here, $\hat{p}_T(\boldsymbol{i})$ denotes the maximum likelihood of the joint probability $p(\boldsymbol{i})$ in the graphical model given by tree $T = (V, E)$.

Recall that $\hat{p}_{v,w}(i_v, i_w)$ denotes the relative frequency of seeing the pair (X_v, X_w) in state (i_v, i_w) and $\hat{p}_v(i_v)$ denotes the relative frequency of seeing X_v in state i_v. The maximum likelihood $\hat{p}_T(\boldsymbol{i})$ can be obtained by inserting $\hat{p}_{v,w}$ and $\hat{p}_v$ into equation (4.1). This yields

$$\frac{1}{n}\hat{L}(T) = \sum_{vw \in E} I(\hat{p}_{v,w}) + \text{constant},$$

whereas $I(\hat{p}_{v,w})$ is the empirical mutual information (see also equation (3.26)). As the mutual information is nonnegative (see e.g. Cover and Thomas (2006), Section 2.3), the maximum likelihood tree is a maximum spanning tree for the complete graph, where the edge weights are $I(\hat{p}_{v,w})$.
This problem can be solved by Kruskal's algorithm, originally due to Kruskal Jr. (1956). One adds the next largest edge from the not already chosen edges and checks whether it creates a cycle. One then repeats this step as many times as possible.

One can similarly apply the Chow-Liu algorithm to the Gaussian case. Writing f for the joint density, the empirical mutual information is given by

$$I(\hat{f}_{v,w}) = -\frac{1}{2}\log(1 - \hat{\rho}^2_{v,w}),$$

where $\hat{\rho}_{v,w}$ is the empirical correlation between X_v and X_w. By monotonicity, these edge weights can be replaced by $|\hat{\rho}_{vw}|$, which we do.

We want to leave some words on implementation. We have implemented Kruskal's algorithm as presented in Section 23.2 of Cormen et al. (2009). For that, the data structure for disjoint sets from Section 21.3 of ibid. was used. This implementation of Kruskal's algorithm has time complexity $\mathcal{O}(|V|^2 \ln(|V|))$, see Section 23.2 of ibid.

In the binary case, we query two-way tables. Note that there are $|V|(|V|-1)/2$ unique two-way tables.
Depending on the querying setup, we may need to add the time to calculate the entries of the sample covariance matrix resp. the two-way tables to the execution time of the Chow-Liu algorithm. Theoretically, calculating these entries takes time $\mathcal{O}(|V|^2)$, so the order of the time complexity does not change. But in practice, this increases the execution time.

Deriving the query complexity is easier. The Chow-Liu algorithm needs to query each unique entry resp. two-way table, thus there are $(|V|+1)V/2$ queries.

4.3.2. Randomly guessing trees

Randomly guessing trees has zero query complexity. The correctness of that procedure will be analyzed in the following.

We assume that we guess trees by drawing them uniformly at random out of all possible labelled trees with $|V|$ vertices. There are $|V|^{|V|-2}$ labelled trees with $|V|$ vertices, this number is also known as the Caley number (see e.g. Wu and Chao (2004)).[3]

Let the underlying true graph be T and let $\hat{T}$ be the randomly guessed tree. Then,

$$\text{Prob}(T = \hat{T}) = |V|^{-|V|+2}$$

This number is already very small for a moderate amount of vertices; for example, if $|V| = 10$, then $\text{Prob}(T = \hat{T}) = 10^{-8}$. Thus in practice, we never expect to guess the correct tree.

We can also look at the expected SHD. Let $\text{SHD}(T, \hat{T})$ denote the SHD between T and $\hat{T}$. We can try to calculate the expected SHD. First of all note that $\text{SHD}(T, \hat{T})$ is always even, because both T and $\hat{T}$ are trees and thus, one wrongly omitted edge corresponds to one wrongly included edge. Furthermore, note that $\text{SHD}(T, \hat{T}) = 2(|V| - 1)$ if and only if all edges of $\hat{T}$ are wrong.

We now have

$$\mathbb{E}[\text{SHD}(T, \hat{T})] \geq \text{Prob}(E(\hat{T}) \text{ contains no edge of } E(T)) \cdot 2(|V| - 1)$$

Note that the number of possible edges that are not part of $E(T)$ is $|V|^2 - (|V| - 1)$. This term grows quadratically in $|V|$. The number of edges which are part of T is $|V| - 1$, and this term only grows linearly in $|V|$. Therefore,

$$\begin{aligned}&\text{Prob}(E(\hat{T}) \text{ contains no edge of } E(T))\\ &\quad >> \text{Prob}(E(\hat{T}) \text{ contains at least one edge of } E(T))\end{aligned}$$

for large enough $|V|$. This implies that

$$\text{Prob}(E(\hat{T}) \text{ contains no edge of } E(T)) \to 1$$

for $|V| \longrightarrow \infty$. Thus,

$$\mathbb{E}[\text{SHD}(T, \hat{T})] \longrightarrow 2(|V| - 1)$$

[3] Prüfer sequences are a way to do draw labelled trees uniformly at random, we introcude that procedure in the subsequent section.

for $|V| \longrightarrow \infty$. Therefore for large enough $|V|$, the randomly guessed tree shares no edge with the correct tree with high probability. Therefore, if `ReconstructTreeSample` has an SHD significantly better than $\approx 2(|V| - 1)$ for large enough trees (we state some simulated values later), it is better than randomly guessing trees.

4.4. Generating data

4.4.1. Generating tree structures

To do simulations, we need to have an underlying tree structure that is the concentration graph of a Gaussian or binary random vector and that we wish to reconstruct. We imagine *fixed tree structures* and *random tree structures.*

We take a look at two particular *fixed tree structures*, the Markov chain and the star.

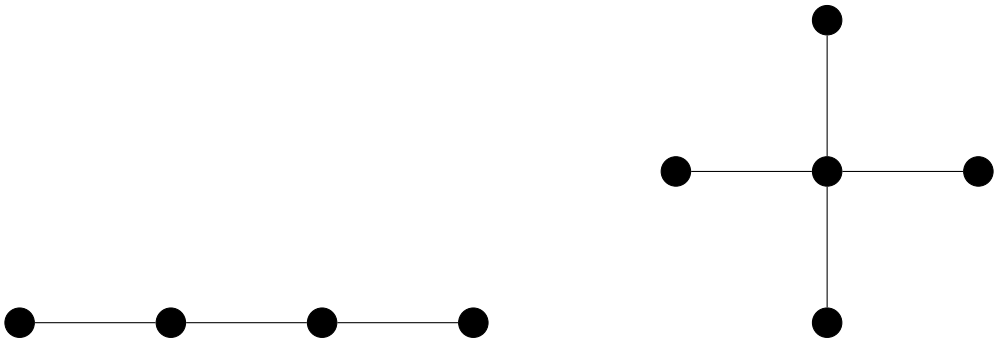


Figure 4.1.: Left: An example of a Markov chain. Right: An example of a star.

For the Chow-Liu algorithm, Tan et al. (2010) and Tan et al. (2011) have shown that the Markov chain and the star are the extremal tree structures (under some assumptions) for Gaussians and binary distributions. That means that the number of errors in the Chow-Liu algorithm decreases the fastest for the Markov chain and the slowest for the star.

Lemma 2.11 resp. Lemma 2.13 show that the pairwise correlation decomposes over a tree. Intuitively, due to this factorization, trees with a larger diameter have a faster decay of pairwise correlations between non-adjacent vertices than trees with a smaller diameter. Therefore for trees with larger diameters, the estimates of the respective partial correlations can be further away from the true partial correlation such that the ordering of the estimates is still correct. Note that the

tree with the largest diameter is the Markov chain, and the tree with the smallest diameter is the star.

We do not know from a theoretic perspective which tree structure is better in terms of correctness for `ReconstructTreeSample`. On the one hand, in the beginning of `ComponentsTreeSample` a similar ordering of absolute pairwise correlations as in the Chow-Liu algorithm is happening. We therefore expect some similar behavior. However, how the conditional independence tests behave with respect to the tree structure is not so clear. It is similarly difficult to make predictions about the runtime. In Theorem 3.26 we have seen that a larger maximum degree leads to a higher time and query complexity. Note that for a fixed size $|V|$ the star has the largest maximum degree. However, that is only an asymptotic result. Similarly interesting, and that is harder to assess, is whether `ReconstructTreeSample` does more errors for stars or chains for finite n. Also note that if `sCentralSample` chooses the centroid of a star in `ComponentsTreeSample`, then each component of the remaining graph has only one vertex, thus there is only one call of `ComponentsTreeSample`. However, if every other vertex is picked, then the maximal remaining subgraph has size $|V| - 1$. Thus, the performance of `sCentralSample` is in particular important for trees with a higher maximum degree, so in particular for the star.

We generate *random trees* using Prüfer sequences.[4] For that, we generate a random sequence of length $|V| - 2$ with values in $\{1, \ldots, |V|\}$ by drawing each component from a uniform distribution with support $1, \ldots, |V|$. There is no restriction on how often certain values can occur in that sequence.
The set of all these sequences is isomorphic to the set of all labelled trees with $|V|$ vertices. This isomorphism and the algorithm to map such a sequence to a particular tree is originally due to Prüfer (1918). We have implemented the algorithm that decodes such a sequence into a tree as given in Wu and Chao (2004).

[4]This is a standard approach. For example, the C-version of the igraph-package contains the function *igraph_tree_game* which does the same, see Csardi and Nepusz (2006).

Algorithm 18: Prüfer Decoding

$P :=$ input Prüfer sequence $P = (\xi_1, \xi_2, \ldots, \xi_{|V|-2})$;
$S := V$;
for *i = 1 to* $|V| - 2$ **do**
    v := the smallest element of the set S that does not occur in P;
    Connect vertex v to vertex ξ_i;
    Remove v from the set S;
    Remove the element ξ_i from the sequence P;
end
Connect the remaining two vertices in V.

By this setup, we draw labelled trees uniformly at random out of all possible labelled trees for a given V.

Our chosen strategy to obtain *random tree structures* is not the only one. For example, we could have used regular trees, i.e. trees with a prespecified number of children. Or we could have set up a repository of trees and then draw from that repository (see e.g Tsamardinos et al. (2006) for such an approach).

4.4.2. Generating covariance matrices for a given tree structure

Generating covariance matrices $\boldsymbol{\Sigma}$ such that the concentration graph $\mathcal{G}(\boldsymbol{\Sigma})$ is a particular tree, is a nontrivial problem for nontrivial cases. We have to ensure that both certain entries of $\boldsymbol{K}$ are zero respectively non-zero and that $\boldsymbol{K}$ (or equivalently $\boldsymbol{\Sigma}$) is positive definite. What is helpful to us, however, is that each positive definite matrix $\boldsymbol{\Sigma}$ can be identified with a Gaussian random vector $\boldsymbol{X}$ such that $\boldsymbol{\Sigma}$ is the covariance matrix of $\boldsymbol{X}$. Indeed, each symmetric positive definite matrix can be diagonalized by the Spectral Theorem (see e.g. Horn and Johnson (2013), Theorem 2.5.6), with only real eigenvalues on the diagonal. That is, there is an orthonormal matrix $\boldsymbol{U}$ and a diagonal matrix $\boldsymbol{D}$ with the eigenvalues on the diagonal, such that

$$\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T.$$

Clearly, there is Gaussian random vector $\boldsymbol{Y}$ such that $\boldsymbol{Y} \sim \mathcal{N}(0, \boldsymbol{D})$, just take independent univariate normal random variables with $\boldsymbol{Y}_i \sim \mathcal{N}(0, \boldsymbol{D}_{ii})$. By a basic transformation theorem (see e.g. Anderson (2003), Theorem 2.4.5), we then know that

$$\boldsymbol{X} := \boldsymbol{U}\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}).$$

Therefore, we just need to come up with positive definite matrices that satisfy the respective constraints on $\boldsymbol{K}$. Then we know that the inverse $\boldsymbol{\Sigma}$ is also positive definite and there is a Gaussian vector $\boldsymbol{X}$ with that particular covariance matrix.

Now, suppose that we are given some adjacency matrix corresponding to a tree. The adjacaceny matrix of a graph encodes all edges. The i, j-th entry is 1 if and only if there is an edge between i and j. There are several approaches to come up with a covariance matrix $\boldsymbol{\Sigma}$ that fits the particular covariance matrix. We refer to Córdoba et al. (2020) for an overview. We look at two approaches, they are based on

- strictly diagonally dominant matrices and
- the Cholesky decomposition.

The approach using strictly diagonally dominant matrices is based on making matrices positive definite by sufficiently increasing the diagonal. In particular, we firstly replace each 1 in the adjacency matrix with a value drawn from a uniform distribution between 0.1 and 1 and with probability 1/2 change the sign of that value. This ensures that the zero-constraints on $\boldsymbol{K}$ are satisfied.

However, the resulting matrix from the previous procedure, call it $\boldsymbol{A} = [a_{i,j}]$, is not necessarily positive definite. Let the i-th deleted row sum of $\boldsymbol{A}$ be defined by $\sum_{j \neq i} a_{i,j}$. Now, we increase each diagonal element $a_{i,i}$ by the respective i-th deleted row sum and add $\varepsilon = 1$ on each diagonal entry. We let the resulting matrix be our concentration matrix $\boldsymbol{K}$. By the previous steps, $\boldsymbol{K}$ is invertible and all eigenvalues are strictly positive and bounded from below by ε. To see this, let λ be an eigenvalue of some matrix $\boldsymbol{M}$ with corresponding eigenvector $\boldsymbol{\omega}$. Then $\boldsymbol{M}\boldsymbol{\omega} = \lambda\boldsymbol{\omega}$. Adding ε on the diagonal yields, $(\boldsymbol{M} + \varepsilon\boldsymbol{I})\boldsymbol{\omega} = \boldsymbol{M}\boldsymbol{\omega} + \varepsilon\boldsymbol{\omega} = \lambda\boldsymbol{\omega} + \varepsilon\boldsymbol{\omega} = (\lambda + \varepsilon)\boldsymbol{\omega}$. If the respective diagonal elements of $\boldsymbol{M}$ are only the respective absolute deleted row sums of $\boldsymbol{A}$ plus the diagonal element a_{ii}, all eigenvalues of $\boldsymbol{M}$ are nonnegative by Geršgorins Theorem, see e.g. Geršgorin (1931),Varga (2004) or Fang and O'Leary (2008). Thus, adding ε on each diagonal element m_{ii} yields that all eigenvalues of $\boldsymbol{K}$ are lower bounded by ε. Matrices such that each diagonal element is strictly bigger than the remaining absolute deleted row-sum are called strictly diagonally dominant.

The second approach is based on a Cholesky decomposition. For this approach, we refer to Córdoba et al. (2020), Roverato (2000) , Wermuth (1980) and Paulsen et al. (1989). For a given undirected graph, we direct edges in a particular way. We choose a root node and make all undirected edges directed such that these directed edges flow out from the root node. That means that there are no colliders in the directed graph, i.e. two arrowheads pointing at each other at the same vertex. For this directed graph, we choose a labelling of the directed edges such that the directed graph is topologically sorted, i.e. all edges point from vertices with a lower order to one with a higher order. In particular, the root node has the

lowest order. For more general graphs, one could choose a vertex labelling that fulfills the so-called perfect elimination ordering. For a definition of that see e.g. Paulsen et al. (1989) or Roverato (2000).

Now, the adjacency matrix of that directed graph is an upper triangular matrix. In that adjacency matrix, we replace each 1 randomly with a value between 0.1 and 1, and then change the sign with probability 1/2. Then, we add 1 on the diagonal. Suppose that we call the resulting matrix $\boldsymbol{U}$. Then, $\boldsymbol{K} := \boldsymbol{U}\boldsymbol{U}^T$ is a valid concentration matrix for the original undirected graph.[5]

There are other approaches for generating matrices corresponding to an undirected graph. One approach is sampling from a matrix distribution whose support is the set of all symmetric positive matrices that satisfy the correponding undirected graph structure. A typical distribution for this are the hyper Wishart distributions, see Lauritzen and Dawid (1993) and Letac and Hélène Massam (2007). Further approaches are discussed in Córdoba et al. (2020).

4.4.3. Generating binary random vectors

We also need to explain how we generate data from the distribution of a binary random vector whose conditional independence graph is some given tree T. The idea builds on the generating process for Gaussians.

First of all, we generate matrices by the Cholesky-approach outlined in Section 4.4.2 and we randomly generate success probabilities $p_1(1), \ldots, p_{|V|}(1)$. For Gaussians, we have discussed that one can always find a random vector $\boldsymbol{Z}$ such that the particular matrix is the covariance matrix of $\boldsymbol{Z}$. However, for binary distributions, this is not necessarily the case, for counterexamples we refer to Macke et al. (2009). Nevertheless, suppose for a moment that the respective matrix is indeed the covariance matrix $\boldsymbol{\Sigma}$ of some binary random vector $\boldsymbol{X}$ with the respective success probabilities $p_1(1), \ldots, p_{|V|}(1)$. Let $p = p(\boldsymbol{i})_{\boldsymbol{i}\in\mathcal{I}}$ be a distribution of $\boldsymbol{X}$.[6] Let $\mathcal{G}(p)$ denote the conditional independence graph and suppose that $\mathcal{G}(p) \neq T$ with positive Lebesgue measure. Then, there are two nodes u, v such that either $uv \notin E(\mathcal{G}(p))$, but $uv \in E(T)$, or $uv \notin E(T)$, but $uv \in E(\mathcal{G}(p))$.

[5] Sometimes, this is stated in reverse order. That means that edges point from vertices with a higher order to vertices with a lower order. Then, one has the same result where one replaces $\boldsymbol{U}$ with a lower triangular matrix. These two versions are equivalent, one can see this by introducing the permutation matrix with ones on the antidiagonal. Then, multiplying the lower triangular matrix from both sides by this permutation matrix yields $\boldsymbol{U}$.

[6] Here, we are a little bit sloppy with the notation, because we use p for both the success probabilities and the distribution of $\boldsymbol{X}$. Strictly speaking, we should have said, let $q = q(\boldsymbol{i})_{\boldsymbol{i}\in\mathcal{I}}$ be a distribution of $\boldsymbol{X}$ such that $q_1(1) = p_1(1), \ldots, q_{|V|}(1) = p_{|V|}(1)$.

We start with the first case, i.e. $uv \notin E(\mathcal{G}(p))$, but $uv \in E(T)$. By construction of $\boldsymbol{K} = \boldsymbol{\Sigma}^{-1}$, the entry $K_{u,v} \neq 0$, because $uv \in E(T)$. However, $p \in \mathcal{M}_+(\mathcal{G}(p))$ by definition and hence by Lemma 2.12, it holds that $K_{u,v} = 0$ because $uv \notin E(\mathcal{G}(p))$. This is a contradiction.

The second case is similar. If $uv \in E(\mathcal{G}(p))$, but $uv \notin E(T)$, then by construction of $\boldsymbol{K}$, it holds that $K_{u,v} = 0$ because $uv \notin E(T)$. However, $p \in \mathcal{M}_+(\mathcal{G}(p))$ by definition and hence by Lemma 2.12, it holds that $K_{u,v} \neq 0$ up to Lebesgue null sets because $uv \notin E(\mathcal{G}(p))$. This is a contradiction up to Lebesgue null sets.

Therefore, $\mathcal{G}(p) \neq T$ holds only with Lebesgue measure zero. Also note that each distribution p with the right success probabilities and the right covariance matrix works.[7] Thus, we only need to have a method that allows us to generate data from a binary random vector with given covariance matrix and success probabilities and have some validation procedure if the matrix and the success probabilities are indeed corresponding to a binary random vector $\boldsymbol{X}$.

We use an approach discussed in ibid. and Leisch et al. (1998) to generate binary random vectors for a given covariance matrix and given success probabilities. For that, we generate data $\boldsymbol{Z}^{(1)}, \dots, \boldsymbol{Z}^{(n)}$ from $\mathcal{N}(\boldsymbol{\gamma}, \boldsymbol{\Lambda})$ with $\gamma \in \mathbb{R}^{|V|}$ and $\boldsymbol{\Lambda} \in \mathbb{R}^{|V| \times |V|}$, and then define data $\boldsymbol{X}^{(1)}, \dots, \boldsymbol{X}^{(n)}$ by truncating the respective component of $\boldsymbol{Z}^{(i)}$, i.e. $\boldsymbol{X}_j^{(i)} = 1$ if $\boldsymbol{Z}_j^{(i)} > 0$ and $\boldsymbol{X}_j^{(i)} = 0$ if $\boldsymbol{Z}_j^{(i)} \leq 0$ for all $j \in \{1, \dots, |V|\}$ and $i \in \{1, \dots, n\}$. Clearly, each $\boldsymbol{X}^{(i)}$ is binary. If we want that each $\boldsymbol{X}^{(i)}$ has been drawn from a binary distribution with particular covariance matrix $\boldsymbol{\Sigma}$ and particular success probabilities $p_1(1), \dots, p_{|V|}(1)$, $\boldsymbol{\gamma}$ and $\boldsymbol{\Lambda}$ need to take particular values. Assume without loss of generality that $\Lambda_{i,i} = 1$ for all $i \in \{1, \dots, |V|\}$. Then,

$$\begin{aligned} p_i(1) &= \Phi(\gamma_i) \\ \Sigma_{i,i} &= \Phi(p_i(1))\Phi(-p_i(1)) \\ \Sigma_{i,j} &= \Psi(\gamma_i, \gamma_j, \Lambda_{i,j}) \end{aligned}$$

for $i \neq j$. Here, $\Psi(x, y, \lambda) = \Phi_2(x, y, \lambda) - \Phi(x)\Phi(y)$ and Φ_2 is the cumulative distribution function of a bivariate Gaussian with mean $\mathbf{0}$ correlation λ.
Now, the resulting matrix $\boldsymbol{\Lambda}$ is not necessarily positive definite; however, if it is, $\boldsymbol{\Sigma}$ is a valid covariance matrix for $\boldsymbol{X}$.

The R-package bindata from ibid. (also see Leisch et al. (2021)) implements this above transformation and allows to simulate a binary random vector for given covariance matrix and success probabilities; it throws an error, if the matrix is

[7] We are not sure, whether the success probabilities and the covariance matrix uniquely specify the distribution p of $\boldsymbol{X}$. However, uniqueness is also not too important.

not valid. In practice, it is surprisingly difficult to calibrate the parameters such that $\mathbf{\Lambda}$ is positive definite. In that sense, this method is rather restrictive. In our particular implementation, we randomly choose each $p_1(1), \ldots, p_{|V|}(1)$ from a uniform distribution between 0.4 and 0.6; which is also a rather restrictive setting; however, for more unrestrictive settings we had trouble generating valid covariance matrices. We generate random matrices using the Cholesky-method; however now, we add 2 on the diagonal instead of 1, as this (in our experience) also results in $\mathbf{\Lambda}$ being positive definite more often.

4.5. Simulation results

4.5.1. Results for Gaussians

Before we look at reconstructing trees, we restrict ourselves to the subprocedure `sCentralSample`, as this enhances our overall understanding of `ReconstructTreeSample`. For that, we look at the example where the underlying tree is a chain, a star or random. We vary n, κ and fix $\alpha = 5 \cdot 10^{-4}$. We do 20 replications; if the underlying tree is random, we choose a new random tree for each replicate. We plot the mean false positive rate (FPR), the mean false negative (FNR), the mean centrality of the returned vertices; here, we average all centralities if multiple vertices have been returned by `sCentralSample`, the mean fraction of times when solely centroids have been returned, the relative query complexity, i.e. the number of queries divided by the number of all unique entries in the covariance matrix, and the execution time in seconds. Each quantity is plotted with 1 standard error; also recall, that a larger value of $c(v)$ indicates a less central vertex.

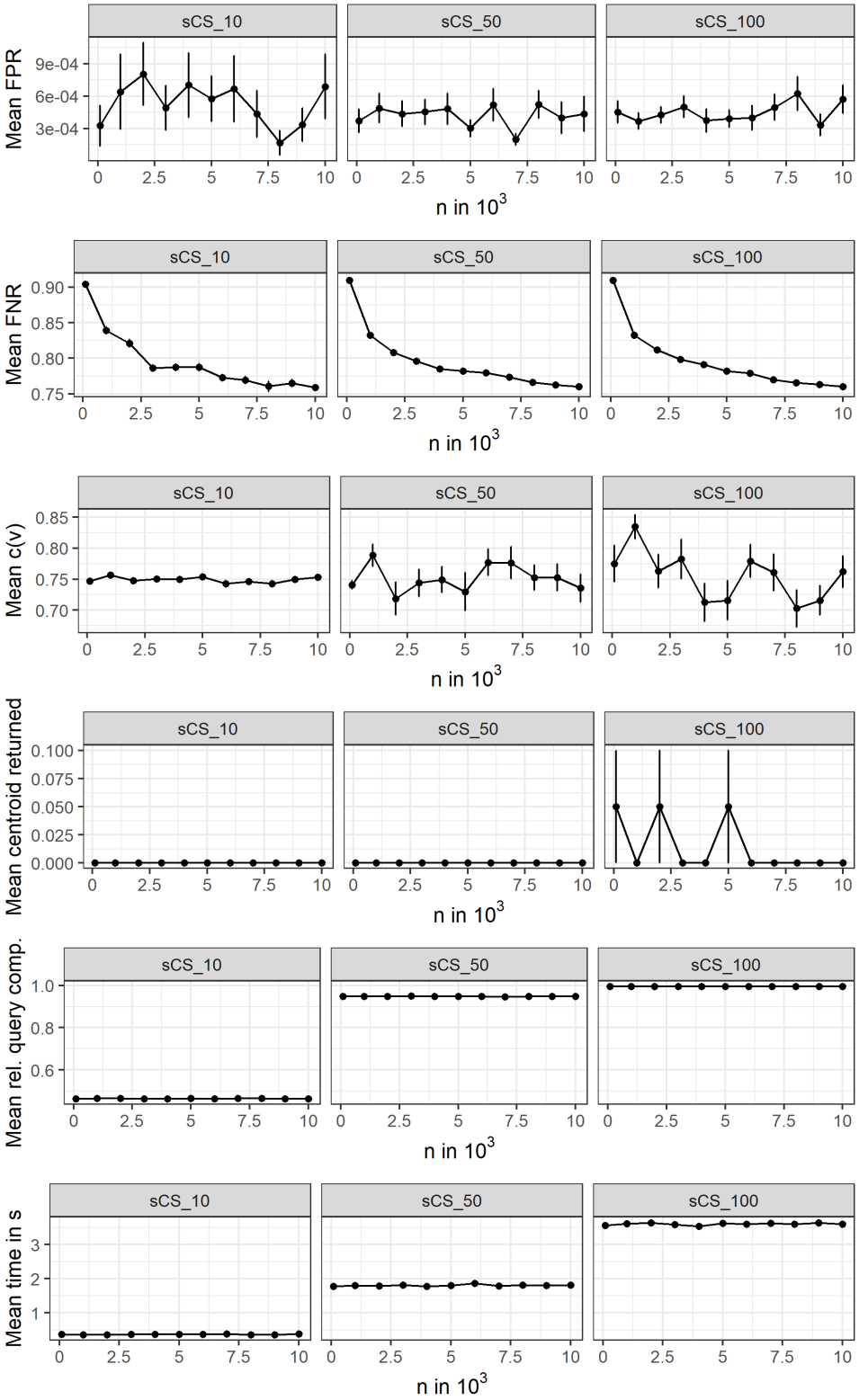


Figure 4.2.: `sCentralSample` for chains with $|V| = 100$ and $\alpha = 5 \cdot 10^{-4}$ using the Cholesky method.

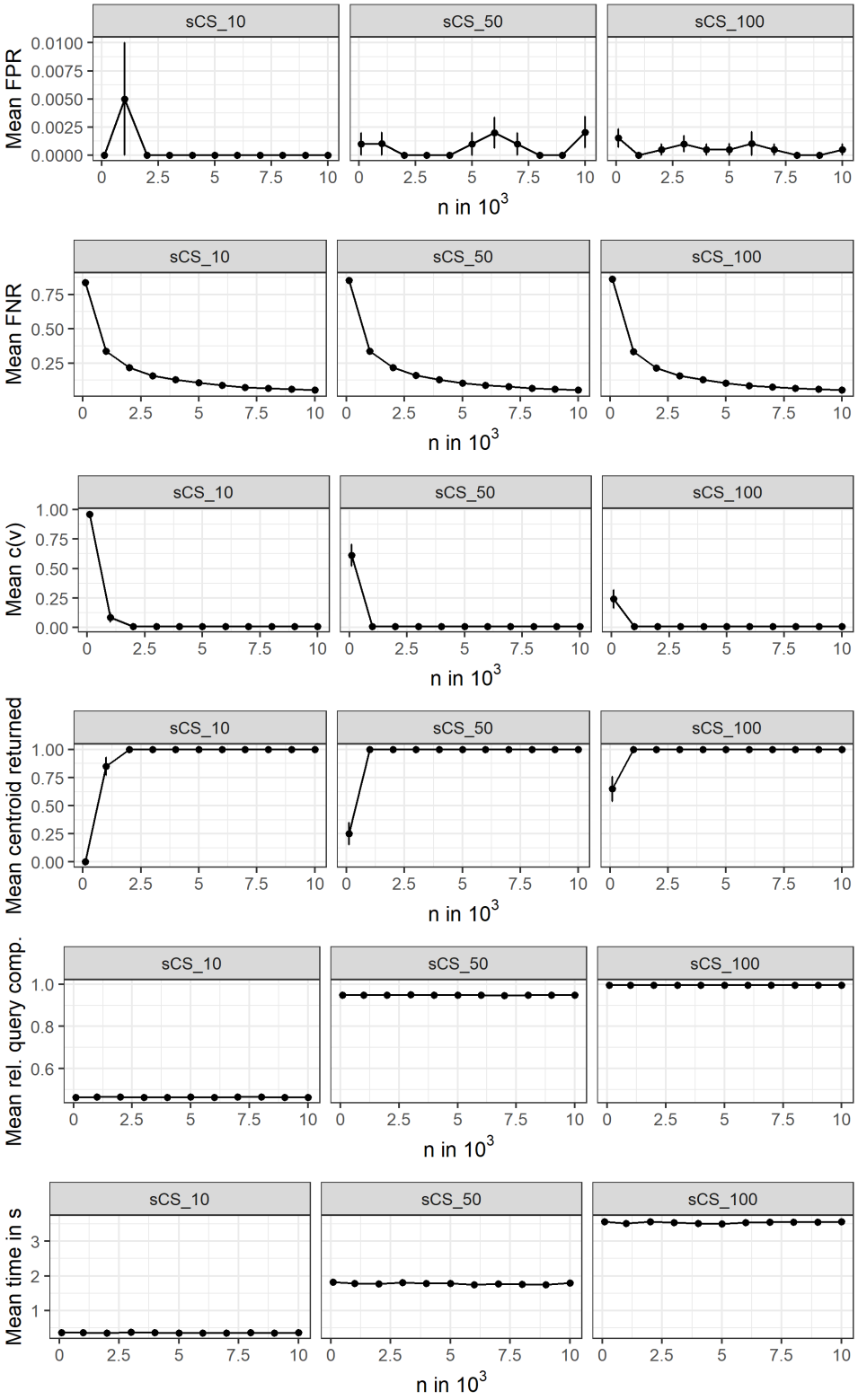


Figure 4.3.: `sCentralSample` for stars with $|V| = 100$ and $\alpha = 5 \cdot 10^{-4}$ using the Cholesky method.

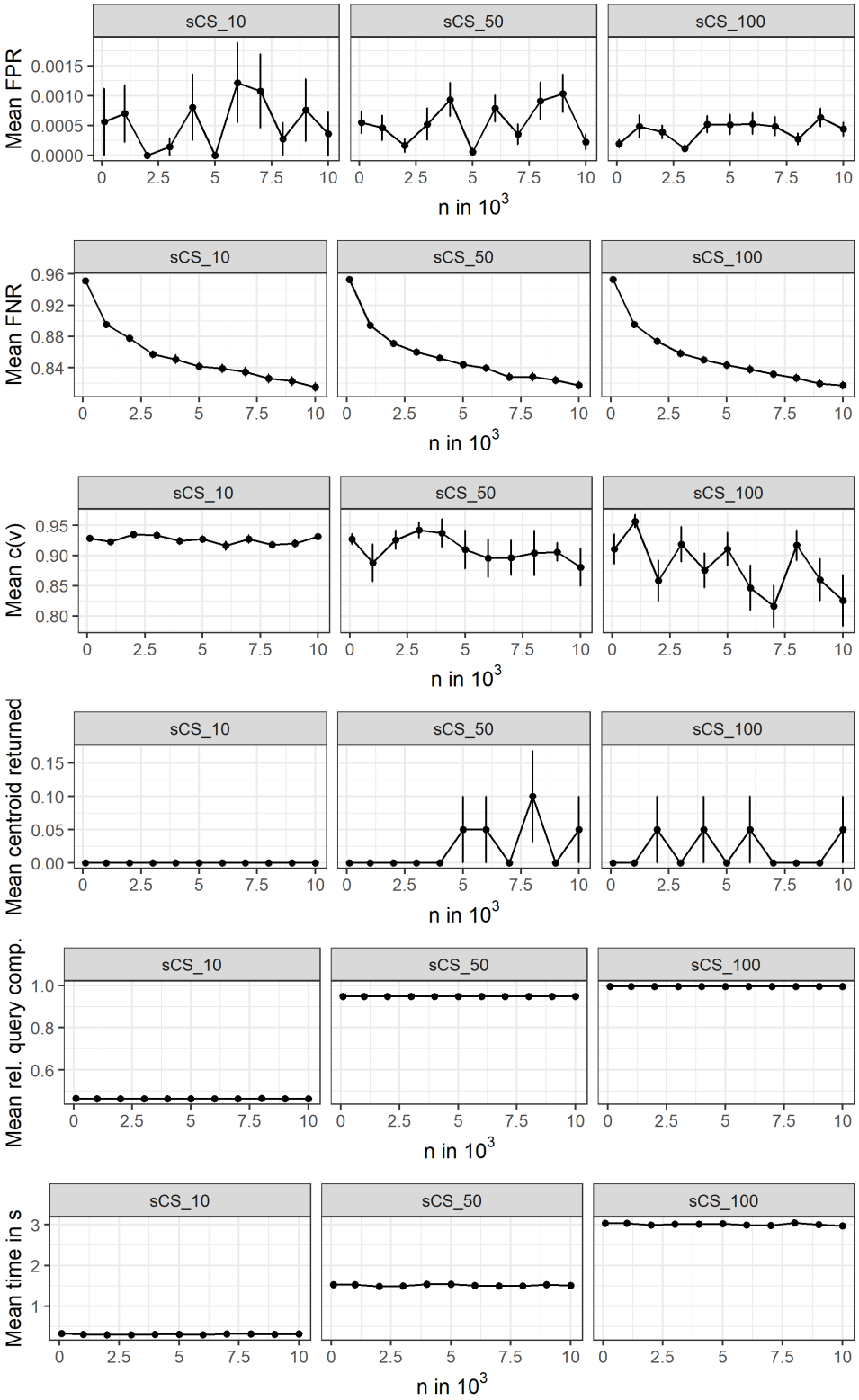


Figure 4.4.: `sCentralSample` for random trees with $|V| = 100$ and $\alpha = 5 \cdot 10^{-4}$ using the Cholesky method.

For each underlying tree, the execution time of `sCentralSample` scales linearly with κ, as predicted by theory. Moreover, a higher value of κ corresponds to a higher query complexity; both time and query complexity are constant in n.

We also observe that the false positive rate does not depend on κ and n; however, the false negative rate decreases for increasing sample size. The false negative rate also seems to be independent of κ, which is also not very surprising. The decrease in the false negative rate directly translates to a lower value of $c(v)$, i.e. more central vertices. This decrease is especially strong for the star. For the star, the maximal remaining subgraph for each non-centroid has size $|V| - 1$, the maximal remaining subgraph of the centroid has size 1. Thus, if a centroid is returned more often by `sCentralSample`, the mean centrality drops very fast.

We continue the analysis on reconstructing trees by giving some results on guessing uniformly trees at random. Note that guessing trees at random does not depend on the distribution of the underlying random vector $\boldsymbol{X}$. We base the random guessing of trees on 1000 replications; the brackets show 1 standard error.

$\lvert V\rvert$	Mean SHD	Mean SHD $/(\lvert V\rvert - 1)$	Mean 0-1 correctness
10	14.5 (0.07)	1.611 (0.008)	0 (0)
20	34.16 (0.08)	1.798 (0.004)	0 (0)
50	93.98 (0.08)	1.918 (0.002)	0 (0)
100	193.98 (0.09)	1.9594 (0.0009)	0 (0)
1000	1994.04 (0.09)	1.99603 (0.00009)	0 (0)

Table 4.1.: The correctness for randomly guessing chains for different sizes of $|V|$.

$\lvert V\rvert$	Mean SHD	Mean SHD $/(\lvert V\rvert - 1)$	Mean 0-1 correctness
10	14.41 (0.05)	1.601 (0.006)	0 (0)
20	34.17 (0.06)	1.798 (0.003)	0 (0)
50	94.05 (0.06)	1.919 (0.001)	0 (0)
100	194.05 (0.06)	1.9601 (0.0006)	0 (0)
1000	1994.03 (0.06)	1.99602 (0.00006)	0 (0)

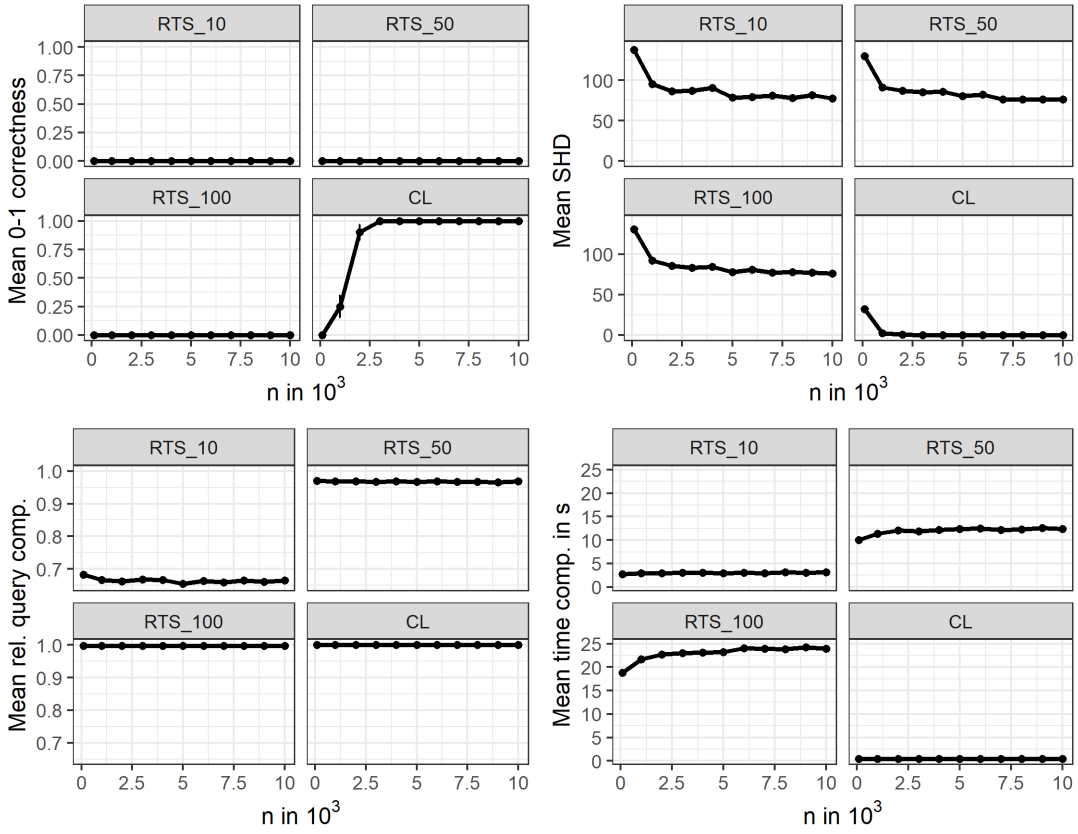
Table 4.2.: The correctness for randomly guessing stars for different sizes of $|V|$.

$\|V\|$	Mean SHD	Mean SHD $/(\|V\|-1)$	Mean 0-1 correctness
10	14.49 (0.07)	1.61 (0.007)	0 (0)
20	34.14 (0.08)	1.797 (0.004)	0 (0)
50	94.05 (0.08)	1.919 (0.002)	0 (0)
100	194.21 (0.08)	1.9617 (0.0008)	0 (0)
1000	1993.89 (0.09)	1.99588 (0.00009)	0 (0)

Table 4.3.: The correctness for randomly guessing trees for different sizes of $|V|$.

In each case, especially for a larger number of vertices, the randomly guessed tree is nearly completely wrong.

We now move to `ReconstructTreeSample` itself by starting with at a fixed number of vertices $|V|$. We vary n for the chain, the star and random tree structures and different combinations of κ, α and α'. The respective combinations are given in the figure description. For each value of n, we have generated 20 random covariance matrices by both the method using strictly diagonally dominant matrices and the Cholesky method. For the star and the chain we have fixed the underlying tree structure, for random trees, we chose a new tree structure for each replicate. The plots show the respective mean with 1 standard error. In the tables we have rounded to the minimum of 2 and the number of significant digits (due to space limitations). The abbreviation "CL" indicates Chow-Liu.



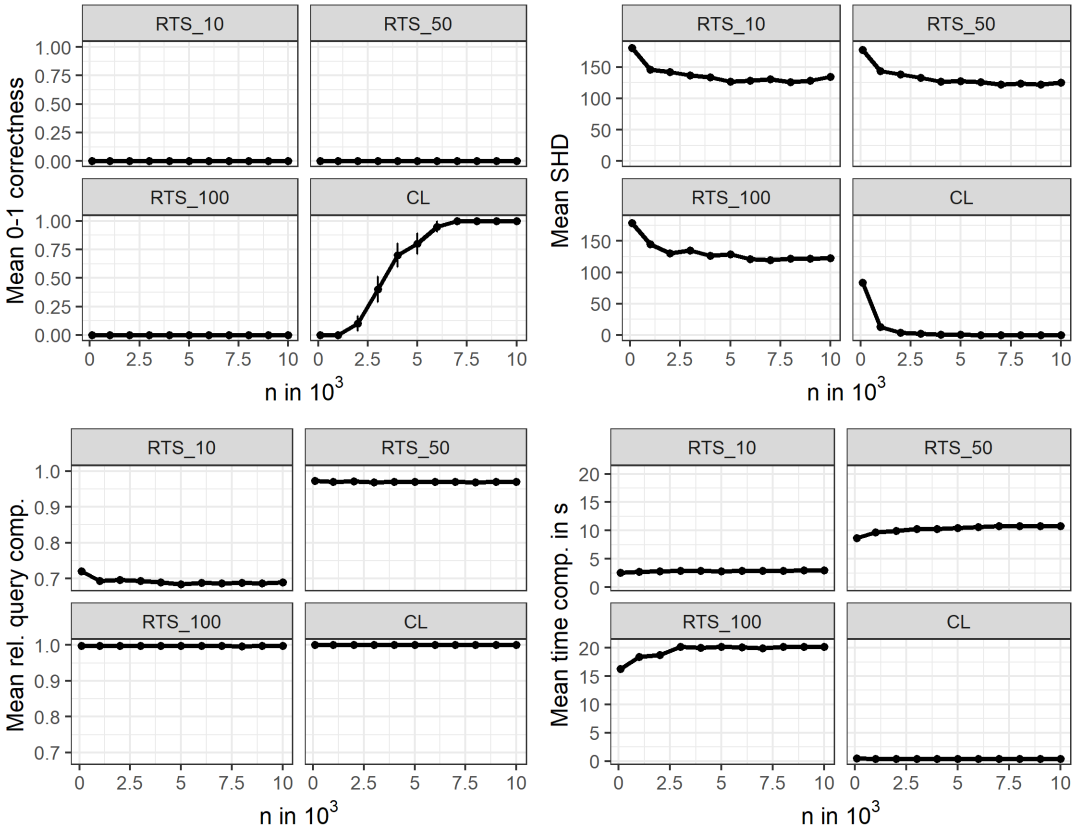
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0.2 (0.1)
2000	0 (0)	0 (0)	0 (0)	0.9 (0.07)
3000	0 (0)	0 (0)	0 (0)	1 (0)
4000	0 (0)	0 (0)	0 (0)	1 (0)
5000	0 (0)	0 (0)	0 (0)	1 (0)
6000	0 (0)	0 (0)	0 (0)	1 (0)
7000	0 (0)	0 (0)	0 (0)	1 (0)
8000	0 (0)	0 (0)	0 (0)	1 (0)
9000	0 (0)	0 (0)	0 (0)	1 (0)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
138 (3)	130 (3)	131 (3)	32 (2)
95 (3)	91 (3)	92 (2)	2.1 (0.4)
86 (2)	87 (3)	86 (4)	0.2 (0.1)
87 (3)	85 (2)	83 (3)	0 (0)
91 (2)	86 (3)	84 (3)	0 (0)
79 (3)	80 (3)	78 (2)	0 (0)
80 (2)	82 (2)	81 (2)	0 (0)
81 (3)	76 (2)	77 (2)	0 (0)
78 (2)	76 (2)	78 (2)	0 (0)
82 (2)	76 (2)	77 (3)	0 (0)
77 (2)	76 (2)	76 (2)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.68 (0)	0.97 (0)	1 (0)	1 (0)
1000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
2000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
3000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
4000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
5000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
6000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
7000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
10000	0.67 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.7 (0.05)	10 (0.1)	18.7 (0.3)	0.49 (0.01)
2.92 (0.05)	11.3 (0.2)	21.7 (0.2)	0.42 (0.01)
2.97 (0.04)	12 (0.2)	22.7 (0.4)	0.42 (0.01)
3.05 (0.05)	11.9 (0.2)	23 (0.4)	0.44 (0.01)
3.07 (0.05)	12.2 (0.2)	23.1 (0.5)	0.42 (0.01)
2.92 (0.05)	12.4 (0.2)	23.2 (0.5)	0.44 (0.01)
3.06 (0.04)	12.5 (0.2)	24 (0.2)	0.44 (0.01)
2.99 (0.04)	12.2 (0.2)	23.9 (0.3)	0.44 (0.01)
3.13 (0.05)	12.3 (0.1)	23.8 (0.4)	0.45 (0.01)
3.06 (0.06)	12.5 (0.2)	24.2 (0.5)	0.46 (0.01)
3.15 (0.04)	12.4 (0.2)	23.9 (0.3)	0.45 (0.01)

Figure 4.5.: Chain with $|V| = 100$ and $\alpha = \alpha' = 0.05$ using the Cholesky-method.



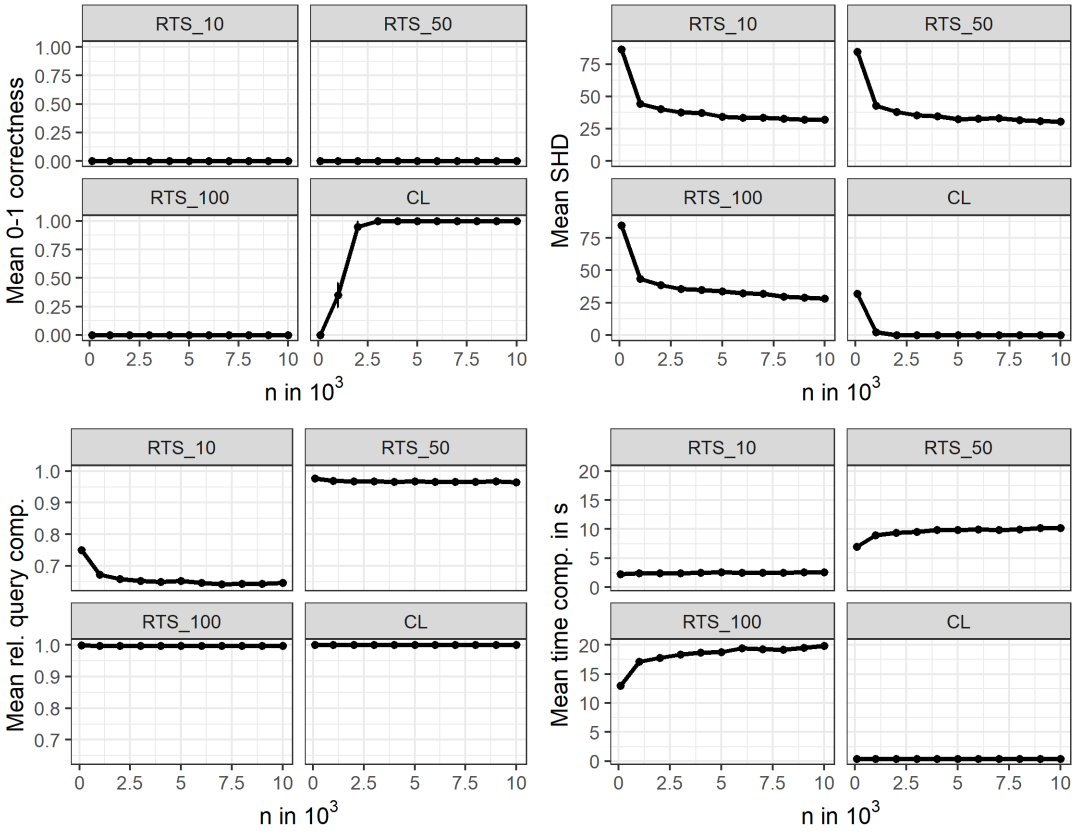
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0 (0)
2000	0 (0)	0 (0)	0 (0)	0.1 (0.07)
3000	0 (0)	0 (0)	0 (0)	0.4 (0.1)
4000	0 (0)	0 (0)	0 (0)	0.7 (0.1)
5000	0 (0)	0 (0)	0 (0)	0.8 (0.09)
6000	0 (0)	0 (0)	0 (0)	0.95 (0.05)
7000	0 (0)	0 (0)	0 (0)	1 (0)
8000	0 (0)	0 (0)	0 (0)	1 (0)
9000	0 (0)	0 (0)	0 (0)	1 (0)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
180 (2)	177 (2)	178 (2)	84 (2)
146 (3)	144 (3)	145 (3)	12.5 (0.9)
142 (2)	139 (3)	130 (3)	3.6 (0.5)
137 (3)	133 (2)	135 (2)	1.9 (0.4)
134 (2)	127 (3)	126 (3)	0.7 (0.3)
127 (2)	128 (2)	128 (3)	0.4 (0.2)
128 (3)	126 (2)	122 (3)	0.1 (0.1)
130 (3)	122 (3)	120 (2)	0 (0)
126 (3)	124 (3)	122 (2)	0 (0)
128 (2)	122 (2)	122 (2)	0 (0)
134 (3)	125 (3)	122 (3)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.72 (0)	0.97 (0)	1 (0)	1 (0)
1000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
2000	0.7 (0)	0.97 (0)	1 (0)	1 (0)
3000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
4000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
5000	0.68 (0)	0.97 (0)	1 (0)	1 (0)
6000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
7000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
10000	0.69 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.56 (0.03)	8.6 (0.1)	16.3 (0.2)	0.5 (0.01)
2.76 (0.03)	9.7 (0.1)	18.4 (0.3)	0.39 (0.01)
2.8 (0.03)	10 (0.1)	18.7 (0.2)	0.38 (0)
2.9 (0.07)	10.3 (0.1)	20.2 (0.3)	0.39 (0.01)
2.88 (0.05)	10.3 (0.1)	20 (0.3)	0.39 (0.01)
2.83 (0.03)	10.5 (0.1)	20.2 (0.3)	0.38 (0)
2.88 (0.04)	10.6 (0.1)	20.1 (0.2)	0.38 (0)
2.9 (0.04)	10.8 (0.2)	19.9 (0.3)	0.38 (0)
2.93 (0.04)	10.8 (0.1)	20.2 (0.2)	0.37 (0)
2.95 (0.05)	10.8 (0.1)	20.2 (0.2)	0.38 (0)
2.96 (0.05)	10.8 (0.2)	20.1 (0.2)	0.38 (0.01)

Figure 4.6.: Chain with $|V| = 100$ and $\alpha = \alpha' = 0.05$ using strictly diagonally dominant matrices.



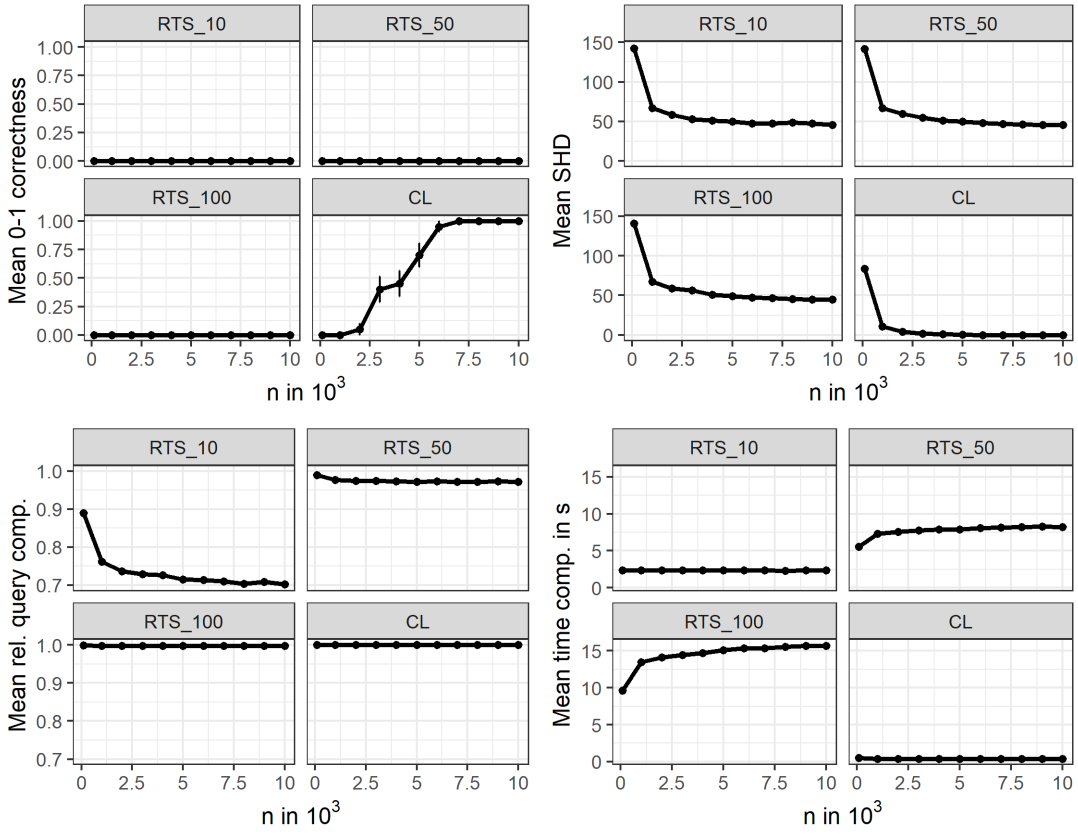
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0.3 (0.1)
2000	0 (0)	0 (0)	0 (0)	0.95 (0.05)
3000	0 (0)	0 (0)	0 (0)	1 (0)
4000	0 (0)	0 (0)	0 (0)	1 (0)
5000	0 (0)	0 (0)	0 (0)	1 (0)
6000	0 (0)	0 (0)	0 (0)	1 (0)
7000	0 (0)	0 (0)	0 (0)	1 (0)
8000	0 (0)	0 (0)	0 (0)	1 (0)
9000	0 (0)	0 (0)	0 (0)	1 (0)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
86 (2)	85 (1)	85 (1)	32 (2)
44 (1)	43 (1)	43 (1)	2 (0.4)
40 (1)	38 (1)	38 (1)	0.1 (0.1)
37.6 (0.9)	35.5 (0.9)	36 (1)	0 (0)
37.1 (0.9)	35 (1)	34.9 (0.9)	0 (0)
34 (1)	32 (1)	34 (1)	0 (0)
33.6 (0.9)	33 (1)	32 (1)	0 (0)
34 (1)	33 (0.9)	32 (1)	0 (0)
33 (0.7)	32 (1)	30 (1)	0 (0)
32 (1)	31 (1)	29 (1)	0 (0)
31.9 (0.9)	30.4 (0.7)	28.3 (0.9)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.75 (0.01)	0.98 (0)	1 (0)	1 (0)
1000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
2000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
3000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
4000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
5000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
6000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
7000	0.64 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.64 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.64 (0)	0.97 (0)	1 (0)	1 (0)
10000	0.65 (0)	0.96 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.22 (0.02)	7.01 (0.07)	13 (0.2)	0.41 (0.01)
2.41 (0.03)	9 (0.1)	17.1 (0.2)	0.38 (0.01)
2.44 (0.03)	9.4 (0.09)	17.8 (0.2)	0.38 (0)
2.44 (0.03)	9.6 (0.1)	18.4 (0.2)	0.38 (0.01)
2.53 (0.04)	9.9 (0.1)	18.7 (0.2)	0.38 (0)
2.57 (0.03)	9.9 (0.08)	18.8 (0.2)	0.4 (0.01)
2.53 (0.03)	10 (0.1)	19.5 (0.3)	0.38 (0)
2.54 (0.03)	9.9 (0.1)	19.3 (0.2)	0.39 (0.01)
2.52 (0.02)	9.9 (0.1)	19.2 (0.2)	0.38 (0)
2.56 (0.03)	10.2 (0.1)	19.5 (0.2)	0.39 (0.01)
2.61 (0.02)	10.2 (0.1)	19.9 (0.1)	0.39 (0.01)

Figure 4.7.: Chain with $|V| = 100$ and $\alpha = \alpha' = 5 \cdot 10^{-4}$ using the Cholesky-method.



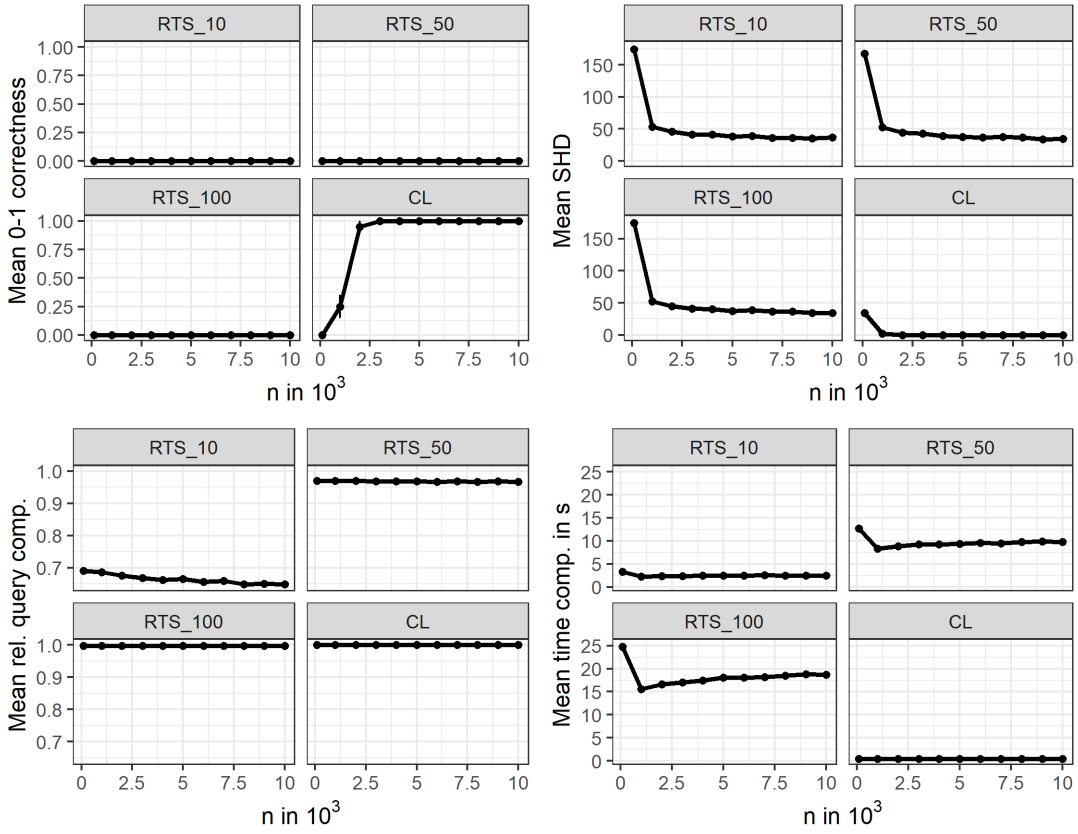
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0 (0)
2000	0 (0)	0 (0)	0 (0)	0.05 (0.05)
3000	0 (0)	0 (0)	0 (0)	0.4 (0.1)
4000	0 (0)	0 (0)	0 (0)	0.4 (0.1)
5000	0 (0)	0 (0)	0 (0)	0.7 (0.1)
6000	0 (0)	0 (0)	0 (0)	0.95 (0.05)
7000	0 (0)	0 (0)	0 (0)	1 (0)
8000	0 (0)	0 (0)	0 (0)	1 (0)
9000	0 (0)	0 (0)	0 (0)	1 (0)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
142 (2)	142 (2)	141 (1)	83 (2)
67 (1)	67 (1)	67 (1)	11 (1)
58 (1)	59.7 (0.8)	58.8 (0.8)	4.1 (0.5)
53 (1)	55 (1)	57 (1)	1.9 (0.4)
51 (1)	51.3 (0.7)	50.8 (0.7)	1.3 (0.3)
50.2 (0.7)	50 (1)	49 (1)	0.6 (0.2)
47.8 (0.9)	48.4 (0.9)	47 (1)	0.1 (0.1)
47.4 (0.7)	46.6 (0.8)	46 (1)	0 (0)
48.6 (0.9)	46.6 (0.8)	45.5 (0.8)	0 (0)
47 (1)	45.5 (0.9)	45.1 (0.8)	0 (0)
45.5 (0.8)	45.5 (0.9)	44.8 (0.8)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.89 (0)	0.99 (0)	1 (0)	1 (0)
1000	0.76 (0)	0.98 (0)	1 (0)	1 (0)
2000	0.74 (0)	0.97 (0)	1 (0)	1 (0)
3000	0.73 (0)	0.97 (0)	1 (0)	1 (0)
4000	0.73 (0)	0.97 (0)	1 (0)	1 (0)
5000	0.72 (0)	0.97 (0)	1 (0)	1 (0)
6000	0.71 (0)	0.97 (0)	1 (0)	1 (0)
7000	0.71 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.7 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.71 (0)	0.97 (0)	1 (0)	1 (0)
10000	0.7 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.34 (0.02)	5.55 (0.05)	9.64 (0.05)	0.48 (0.01)
2.35 (0.02)	7.31 (0.06)	13.5 (0.1)	0.39 (0.01)
2.34 (0.02)	7.55 (0.03)	14.12 (0.09)	0.38 (0.01)
2.35 (0.02)	7.78 (0.06)	14.4 (0.1)	0.38 (0.01)
2.37 (0.03)	7.87 (0.05)	14.7 (0.09)	0.37 (0)
2.35 (0.02)	7.91 (0.05)	15.09 (0.08)	0.36 (0)
2.37 (0.02)	8.12 (0.05)	15.3 (0.1)	0.38 (0.01)
2.37 (0.02)	8.16 (0.08)	15.4 (0.1)	0.37 (0)
2.32 (0.02)	8.2 (0.05)	15.54 (0.09)	0.37 (0)
2.37 (0.02)	8.28 (0.06)	15.7 (0.1)	0.38 (0.01)
2.38 (0.02)	8.23 (0.05)	15.7 (0.1)	0.38 (0.01)

Figure 4.8.: Chain with $|V| = 100$ and $\alpha = \alpha' = 5 \cdot 10^{-4}$ using strictly diagonally dominant matrices.



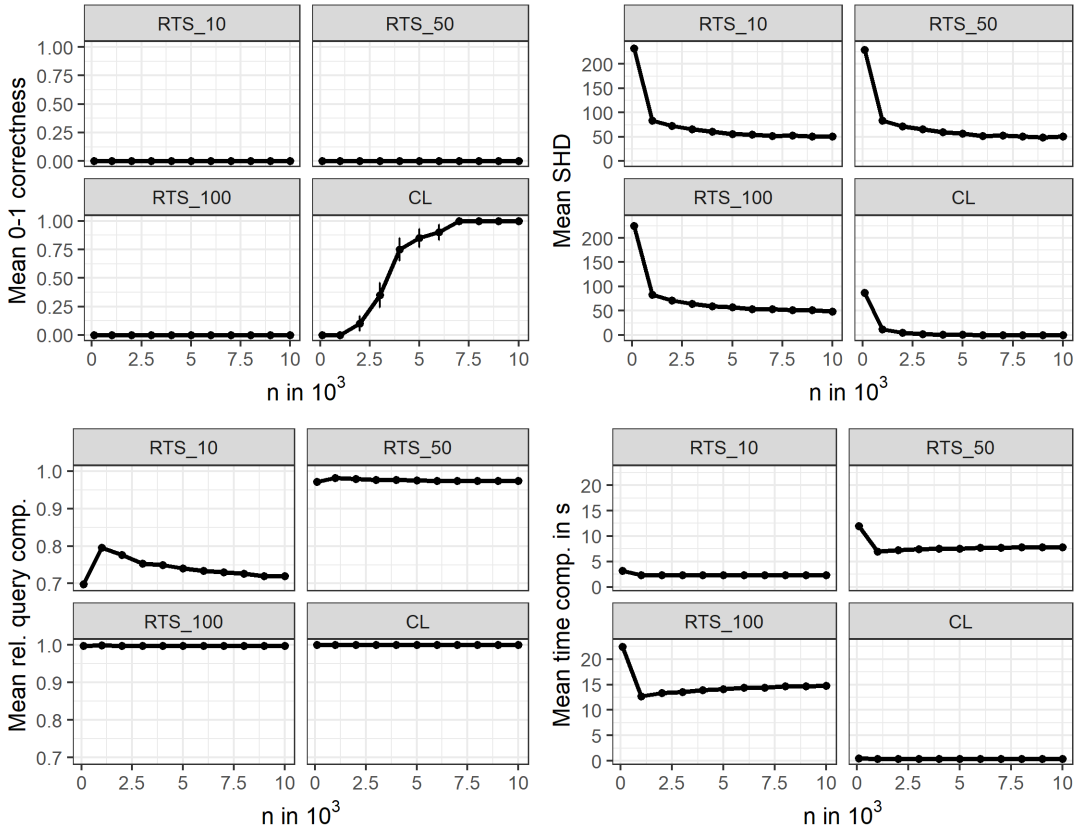
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0.2 (0.1)
2000	0 (0)	0 (0)	0 (0)	0.95 (0.05)
3000	0 (0)	0 (0)	0 (0)	1 (0)
4000	0 (0)	0 (0)	0 (0)	1 (0)
5000	0 (0)	0 (0)	0 (0)	1 (0)
6000	0 (0)	0 (0)	0 (0)	1 (0)
7000	0 (0)	0 (0)	0 (0)	1 (0)
8000	0 (0)	0 (0)	0 (0)	1 (0)
9000	0 (0)	0 (0)	0 (0)	1 (0)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
174 (4)	167 (3)	175 (3)	34 (2)
53 (1)	53 (1)	52 (1)	2.2 (0.4)
46.1 (0.9)	44 (0.9)	45 (0.9)	0.1 (0.1)
42 (1)	43 (1)	41 (0.8)	0 (0)
41 (1)	39.1 (0.9)	40 (1)	0 (0)
38 (1)	38 (0.9)	37.6 (0.7)	0 (0)
39 (1)	37 (1)	39 (0.7)	0 (0)
36.1 (0.8)	37.5 (0.8)	36.4 (0.9)	0 (0)
36 (1)	37 (1)	36.6 (0.9)	0 (0)
35.6 (0.8)	34 (1)	35 (1)	0 (0)
37 (1)	34.3 (0.9)	34.5 (0.8)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.69 (0)	0.97 (0)	1 (0)	1 (0)
1000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
2000	0.68 (0)	0.97 (0)	1 (0)	1 (0)
3000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
4000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
5000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
6000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
7000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
10000	0.65 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
3.32 (0.08)	12.7 (0.3)	24.7 (0.4)	0.47 (0.01)
2.3 (0.02)	8.27 (0.09)	15.6 (0.1)	0.41 (0)
2.38 (0.02)	8.8 (0.1)	16.7 (0.2)	0.4 (0)
2.4 (0.03)	9.2 (0.1)	17 (0.2)	0.42 (0.01)
2.42 (0.03)	9.3 (0.1)	17.4 (0.2)	0.43 (0.01)
2.51 (0.03)	9.3 (0.07)	18.1 (0.2)	0.42 (0.01)
2.46 (0.02)	9.5 (0.1)	18.1 (0.1)	0.43 (0.01)
2.53 (0.03)	9.44 (0.09)	18.2 (0.2)	0.42 (0.01)
2.48 (0.03)	9.7 (0.1)	18.5 (0.2)	0.44 (0.01)
2.46 (0.03)	9.8 (0.1)	18.8 (0.2)	0.45 (0.01)
2.5 (0.02)	9.7 (0.1)	18.7 (0.1)	0.45 (0.01)

Figure 4.9.: Chain with $|V| = 100$ and $\alpha = \alpha' = \max(2 \cdot (1 - \Phi(1.64 \cdot (n/|V|)^{0.5})), 2 \cdot \sqrt{\text{Machine Precision}})$ using the Cholesky-method.



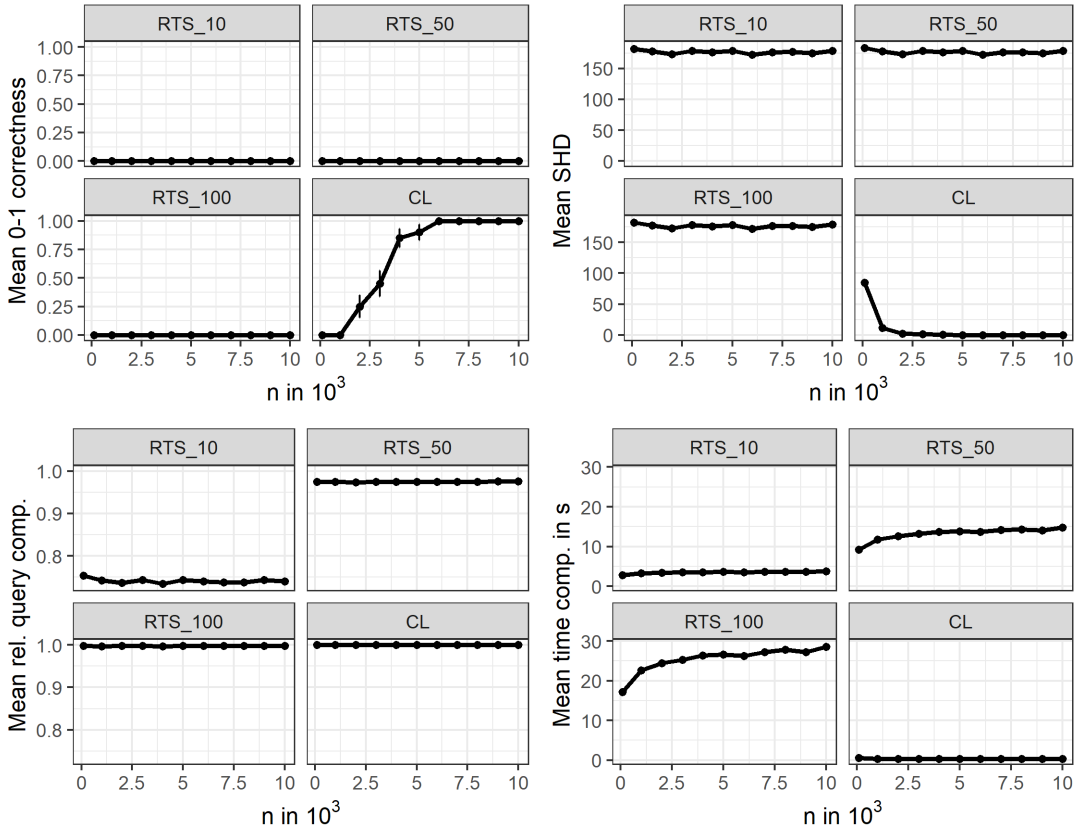
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0 (0)
2000	0 (0)	0 (0)	0 (0)	0.1 (0.07)
3000	0 (0)	0 (0)	0 (0)	0.3 (0.1)
4000	0 (0)	0 (0)	0 (0)	0.8 (0.1)
5000	0 (0)	0 (0)	0 (0)	0.85 (0.08)
6000	0 (0)	0 (0)	0 (0)	0.9 (0.07)
7000	0 (0)	0 (0)	0 (0)	1 (0)
8000	0 (0)	0 (0)	0 (0)	1 (0)
9000	0 (0)	0 (0)	0 (0)	1 (0)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
232 (3)	229 (3)	225 (3)	87 (2)
84 (1)	83 (1)	83 (1)	12.1 (0.8)
73 (1)	71.2 (0.7)	71 (1)	4.8 (0.6)
66 (1)	65 (1)	64 (1)	1.9 (0.4)
61 (1)	59 (1)	59 (1)	0.5 (0.2)
56 (0.8)	57 (0.7)	57 (1)	0.4 (0.2)
54.9 (0.8)	52.1 (0.9)	53.1 (0.8)	0.2 (0.1)
52 (0.8)	52.6 (0.9)	53.1 (0.8)	0 (0)
53 (0.7)	50.9 (0.8)	51 (1)	0 (0)
51 (0.7)	49 (1)	51.1 (0.6)	0 (0)
50.3 (0.6)	51 (1)	48.5 (0.8)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.7 (0)	0.97 (0)	1 (0)	1 (0)
1000	0.8 (0)	0.98 (0)	1 (0)	1 (0)
2000	0.78 (0)	0.98 (0)	1 (0)	1 (0)
3000	0.75 (0)	0.98 (0)	1 (0)	1 (0)
4000	0.75 (0)	0.98 (0)	1 (0)	1 (0)
5000	0.74 (0)	0.98 (0)	1 (0)	1 (0)
6000	0.73 (0)	0.97 (0)	1 (0)	1 (0)
7000	0.73 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.73 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.72 (0)	0.97 (0)	1 (0)	1 (0)
10000	0.72 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
3.22 (0.06)	12 (0.2)	22.4 (0.4)	0.52 (0.02)
2.34 (0.02)	6.94 (0.06)	12.7 (0.1)	0.41 (0.01)
2.36 (0.03)	7.24 (0.07)	13.4 (0.2)	0.39 (0.01)
2.31 (0.02)	7.47 (0.07)	13.5 (0.1)	0.39 (0)
2.32 (0.02)	7.55 (0.07)	14 (0.1)	0.4 (0.01)
2.34 (0.02)	7.58 (0.07)	14.1 (0.1)	0.39 (0.01)
2.35 (0.02)	7.73 (0.08)	14.35 (0.08)	0.39 (0)
2.31 (0.02)	7.68 (0.05)	14.4 (0.1)	0.39 (0)
2.3 (0.02)	7.78 (0.05)	14.7 (0.1)	0.39 (0.01)
2.3 (0.02)	7.86 (0.05)	14.7 (0.1)	0.41 (0.01)
2.32 (0.03)	7.77 (0.07)	14.8 (0.1)	0.42 (0.01)

Figure 4.10.: Chain with $|V| = 100$ and $\alpha = \alpha' = \max(2 \cdot (1 - \Phi(1.64 \cdot (n/|V|)^{0.5})), 2 \cdot \sqrt{\text{Machine Precision}})$ using strictly diagonally dominant matrices.



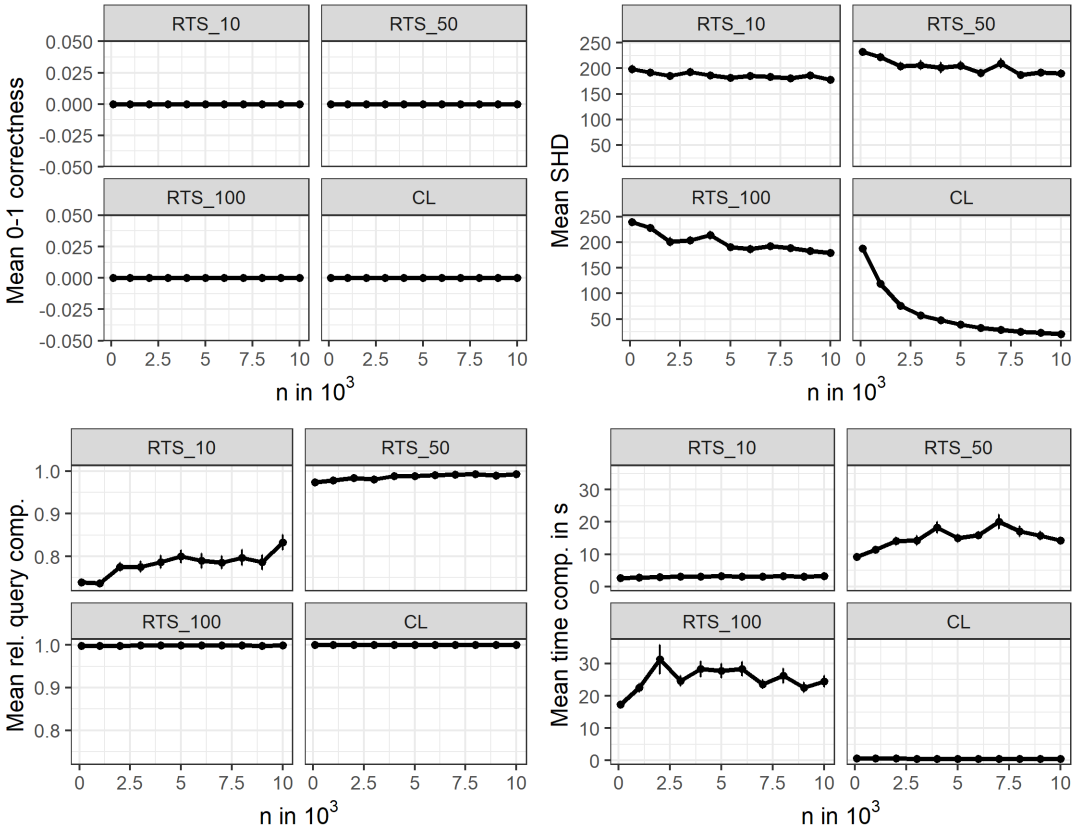
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0 (0)
2000	0 (0)	0 (0)	0 (0)	0.2 (0.1)
3000	0 (0)	0 (0)	0 (0)	0.4 (0.1)
4000	0 (0)	0 (0)	0 (0)	0.85 (0.08)
5000	0 (0)	0 (0)	0 (0)	0.9 (0.07)
6000	0 (0)	0 (0)	0 (0)	1 (0)
7000	0 (0)	0 (0)	0 (0)	1 (0)
8000	0 (0)	0 (0)	0 (0)	1 (0)
9000	0 (0)	0 (0)	0 (0)	1 (0)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
182 (2)	183 (2)	182 (2)	84 (2)
178 (2)	178 (2)	178 (2)	11.7 (0.8)
173 (2)	173 (2)	173 (2)	2.4 (0.4)
179 (2)	179 (2)	178 (2)	1.3 (0.3)
176 (2)	176 (2)	176 (2)	0.4 (0.2)
178 (2)	178 (2)	178 (2)	0.2 (0.1)
172 (2)	172 (2)	172 (2)	0 (0)
176 (2)	176 (2)	176 (2)	0 (0)
177 (2)	177 (2)	177 (2)	0 (0)
175 (2)	175 (2)	175 (2)	0 (0)
179 (2)	179 (2)	179 (2)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.75 (0)	0.98 (0)	1 (0)	1 (0)
1000	0.74 (0)	0.97 (0)	1 (0)	1 (0)
2000	0.74 (0)	0.97 (0)	1 (0)	1 (0)
3000	0.74 (0)	0.98 (0)	1 (0)	1 (0)
4000	0.73 (0)	0.97 (0)	1 (0)	1 (0)
5000	0.74 (0)	0.97 (0)	1 (0)	1 (0)
6000	0.74 (0)	0.98 (0)	1 (0)	1 (0)
7000	0.74 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.74 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.74 (0)	0.98 (0)	1 (0)	1 (0)
10000	0.74 (0)	0.98 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.78 (0.04)	9.2 (0.1)	17.2 (0.2)	0.53 (0.01)
3.25 (0.05)	11.8 (0.2)	22.6 (0.4)	0.32 (0)
3.36 (0.04)	12.7 (0.2)	24.5 (0.4)	0.3 (0)
3.47 (0.05)	13.2 (0.3)	25.2 (0.5)	0.3 (0)
3.58 (0.07)	13.6 (0.3)	26.4 (0.6)	0.3 (0)
3.61 (0.05)	13.9 (0.2)	26.6 (0.4)	0.31 (0.01)
3.57 (0.06)	13.7 (0.3)	26.3 (0.6)	0.3 (0)
3.65 (0.07)	14.1 (0.3)	27.3 (0.6)	0.3 (0)
3.71 (0.06)	14.4 (0.3)	27.8 (0.5)	0.3 (0)
3.67 (0.05)	14.1 (0.2)	27.2 (0.5)	0.3 (0)
3.79 (0.05)	14.8 (0.2)	28.6 (0.5)	0.3 (0)

Figure 4.11.: Star with $|V| = 100$ and $\alpha = \alpha' = 0.05$ using the Cholesky-method.



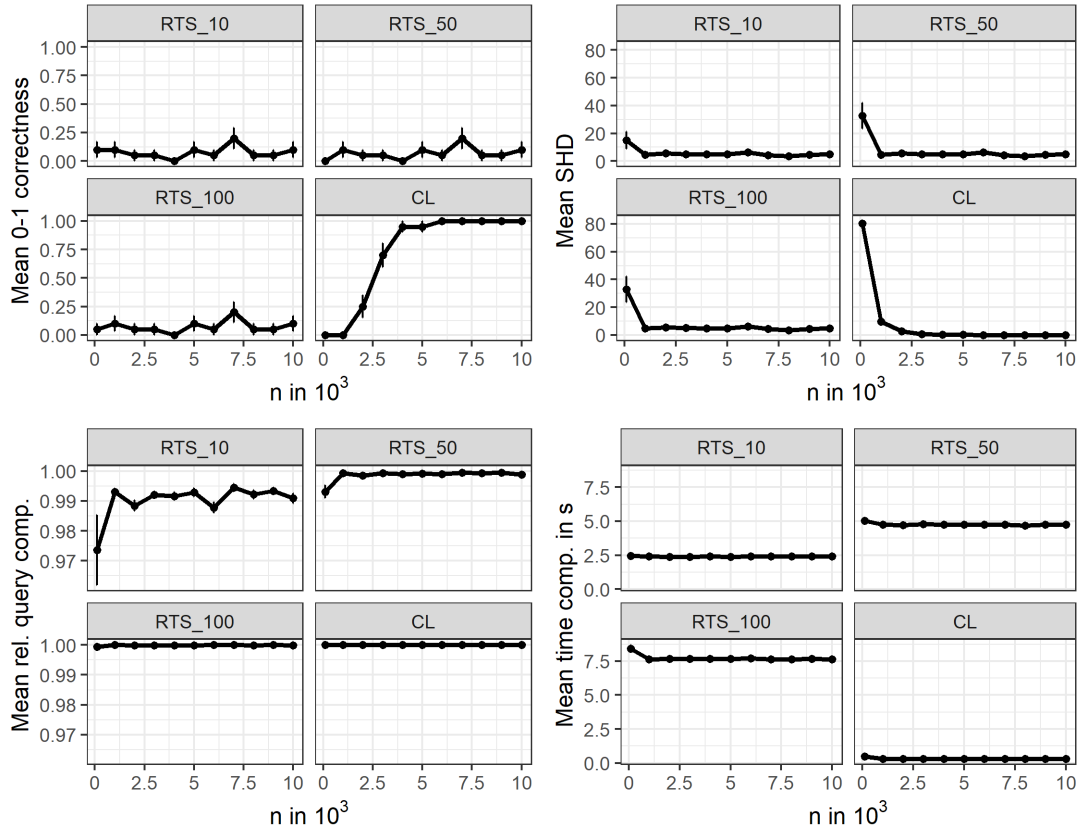
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0 (0)
2000	0 (0)	0 (0)	0 (0)	0 (0)
3000	0 (0)	0 (0)	0 (0)	0 (0)
4000	0 (0)	0 (0)	0 (0)	0 (0)
5000	0 (0)	0 (0)	0 (0)	0 (0)
6000	0 (0)	0 (0)	0 (0)	0 (0)
7000	0 (0)	0 (0)	0 (0)	0 (0)
8000	0 (0)	0 (0)	0 (0)	0 (0)
9000	0 (0)	0 (0)	0 (0)	0 (0)
10000	0 (0)	0 (0)	0 (0)	0 (0)

RTS_10	RTS_50	RTS_100	CL
199 (7)	232 (4)	239 (2)	187 (1)
192 (5)	222 (6)	228 (6)	119 (1)
185 (5)	204 (8)	201 (9)	76 (2)
193 (6)	206 (9)	204 (7)	58 (1)
186 (3)	200 (10)	214 (8)	48 (1)
181 (2)	205 (8)	191 (5)	39 (2)
185 (6)	191 (7)	186 (7)	33 (1)
184 (3)	210 (9)	192 (4)	29 (1)
181 (3)	187 (6)	188 (4)	25 (1)
187 (4)	192 (6)	183 (3)	23 (1)
177 (2)	190 (7)	179 (2)	20 (1)

n	RTS_10	RTS_50	RTS_100	CL
100	0.74 (0)	0.97 (0)	1 (0)	1 (0)
1000	0.74 (0)	0.98 (0)	1 (0)	1 (0)
2000	0.78 (0.01)	0.98 (0)	1 (0)	1 (0)
3000	0.78 (0.01)	0.98 (0)	1 (0)	1 (0)
4000	0.79 (0.01)	0.99 (0)	1 (0)	1 (0)
5000	0.8 (0.02)	0.99 (0)	1 (0)	1 (0)
6000	0.79 (0.02)	0.99 (0)	1 (0)	1 (0)
7000	0.79 (0.02)	0.99 (0)	1 (0)	1 (0)
8000	0.8 (0.02)	0.99 (0)	1 (0)	1 (0)
9000	0.79 (0.02)	0.99 (0)	1 (0)	1 (0)
10000	0.83 (0.02)	0.99 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.7 (0.04)	9.1 (0.1)	17.3 (0.2)	0.58 (0.01)
2.76 (0.04)	11.5 (0.4)	22 (1)	0.61 (0.01)
2.96 (0.07)	14 (1)	31 (4)	0.58 (0.01)
3.1 (0.2)	14 (1)	25 (2)	0.54 (0.01)
3.1 (0.1)	18 (2)	28 (2)	0.53 (0.01)
3.2 (0.2)	15 (1)	28 (2)	0.51 (0.01)
3.1 (0.1)	16 (1)	28 (2)	0.51 (0.02)
3.1 (0.1)	20 (2)	24 (1)	0.47 (0.01)
3.2 (0.2)	17 (2)	26 (2)	0.5 (0.02)
3.1 (0.1)	16 (1)	23 (2)	0.48 (0.02)
3.3 (0.2)	14.2 (0.9)	24 (2)	0.46 (0.01)

Figure 4.12.: Star with $|V| = 100$ and $\alpha = \alpha' = 0.05$ using strictly diagonally dominant matrices.



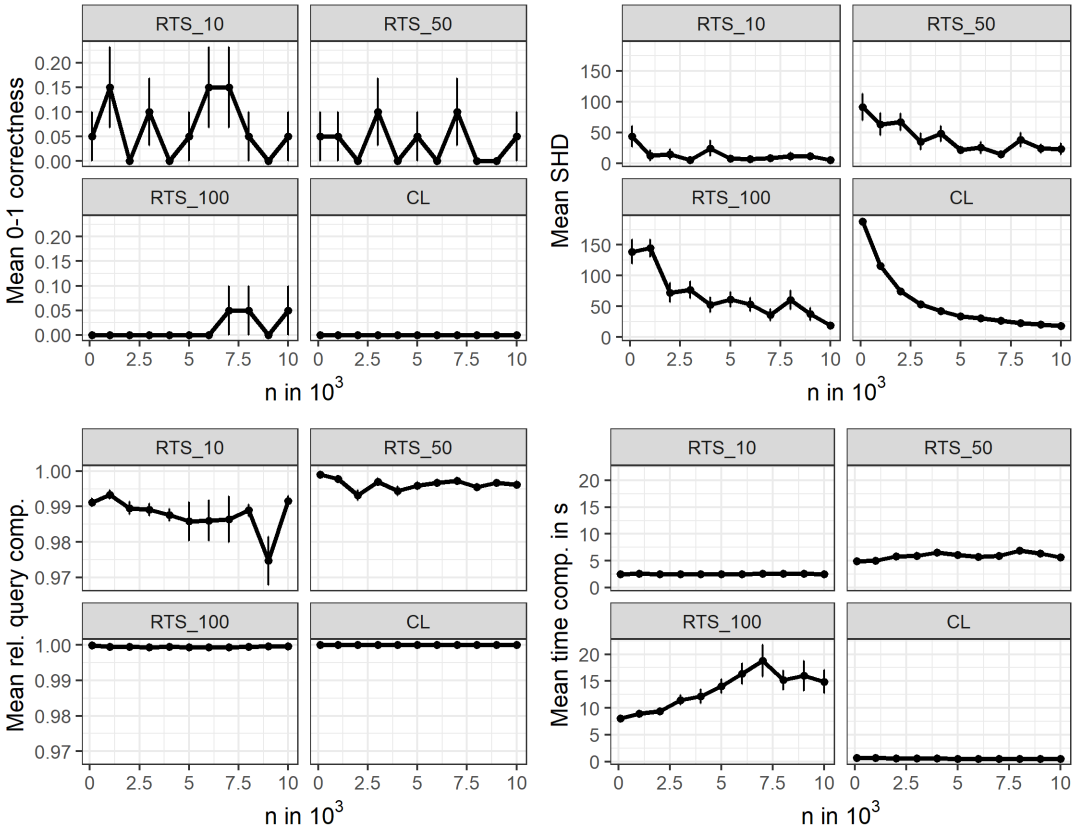
n	RTS_10	RTS_50	RTS_100	CL
100	0.1 (0.07)	0 (0)	0.05 (0.05)	0 (0)
1000	0.1 (0.07)	0.1 (0.07)	0.1 (0.07)	0 (0)
2000	0.05 (0.05)	0.05 (0.05)	0.05 (0.05)	0.2 (0.1)
3000	0.05 (0.05)	0.05 (0.05)	0.05 (0.05)	0.7 (0.1)
4000	0 (0)	0 (0)	0 (0)	0.95 (0.05)
5000	0.1 (0.07)	0.1 (0.07)	0.1 (0.07)	0.95 (0.05)
6000	0.05 (0.05)	0.05 (0.05)	0.05 (0.05)	1 (0)
7000	0.2 (0.09)	0.2 (0.09)	0.2 (0.09)	1 (0)
8000	0.05 (0.05)	0.05 (0.05)	0.05 (0.05)	1 (0)
9000	0.05 (0.05)	0.05 (0.05)	0.05 (0.05)	1 (0)
10000	0.1 (0.07)	0.1 (0.07)	0.1 (0.07)	1 (0)

RTS_10	RTS_50	RTS_100	CL
15 (6)	33 (9)	33 (9)	80 (2)
4.7 (0.7)	4.7 (0.7)	4.7 (0.7)	10 (1)
5.4 (0.6)	5.4 (0.6)	5.4 (0.6)	2.7 (0.5)
5 (0.5)	5 (0.5)	5 (0.5)	0.7 (0.3)
4.8 (0.6)	4.8 (0.6)	4.8 (0.6)	0.1 (0.1)
4.8 (0.8)	4.8 (0.8)	4.8 (0.8)	0.1 (0.1)
6.3 (0.7)	6.3 (0.7)	6.3 (0.7)	0 (0)
4.3 (0.8)	4.3 (0.8)	4.3 (0.8)	0 (0)
3.5 (0.5)	3.5 (0.5)	3.5 (0.5)	0 (0)
4.5 (0.5)	4.5 (0.5)	4.5 (0.5)	0 (0)
4.8 (0.8)	4.8 (0.8)	4.8 (0.8)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.97 (0.01)	0.99 (0)	1 (0)	1 (0)
1000	0.99 (0)	1 (0)	1 (0)	1 (0)
2000	0.99 (0)	1 (0)	1 (0)	1 (0)
3000	0.99 (0)	1 (0)	1 (0)	1 (0)
4000	0.99 (0)	1 (0)	1 (0)	1 (0)
5000	0.99 (0)	1 (0)	1 (0)	1 (0)
6000	0.99 (0)	1 (0)	1 (0)	1 (0)
7000	0.99 (0)	1 (0)	1 (0)	1 (0)
8000	0.99 (0)	1 (0)	1 (0)	1 (0)
9000	0.99 (0)	1 (0)	1 (0)	1 (0)
10000	0.99 (0)	1 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.45 (0.04)	5 (0.1)	8.4 (0.3)	0.5 (0.01)
2.41 (0.01)	4.75 (0.02)	7.63 (0.04)	0.32 (0)
2.39 (0.01)	4.73 (0.02)	7.68 (0.04)	0.32 (0.01)
2.38 (0.01)	4.78 (0.02)	7.67 (0.04)	0.3 (0)
2.42 (0.01)	4.75 (0.02)	7.68 (0.03)	0.3 (0)
2.39 (0.01)	4.74 (0.02)	7.68 (0.04)	0.3 (0)
2.4 (0.01)	4.76 (0.03)	7.72 (0.04)	0.31 (0)
2.42 (0.02)	4.74 (0.03)	7.63 (0.05)	0.3 (0)
2.4 (0.02)	4.7 (0.02)	7.63 (0.03)	0.3 (0)
2.42 (0.01)	4.75 (0.02)	7.66 (0.02)	0.3 (0)
2.42 (0.01)	4.75 (0.02)	7.64 (0.04)	0.31 (0)

Figure 4.13.: Star with $|V| = 100$ and $\alpha = \alpha' = 5 \cdot 10^{-4}$ using the Cholesky-method.



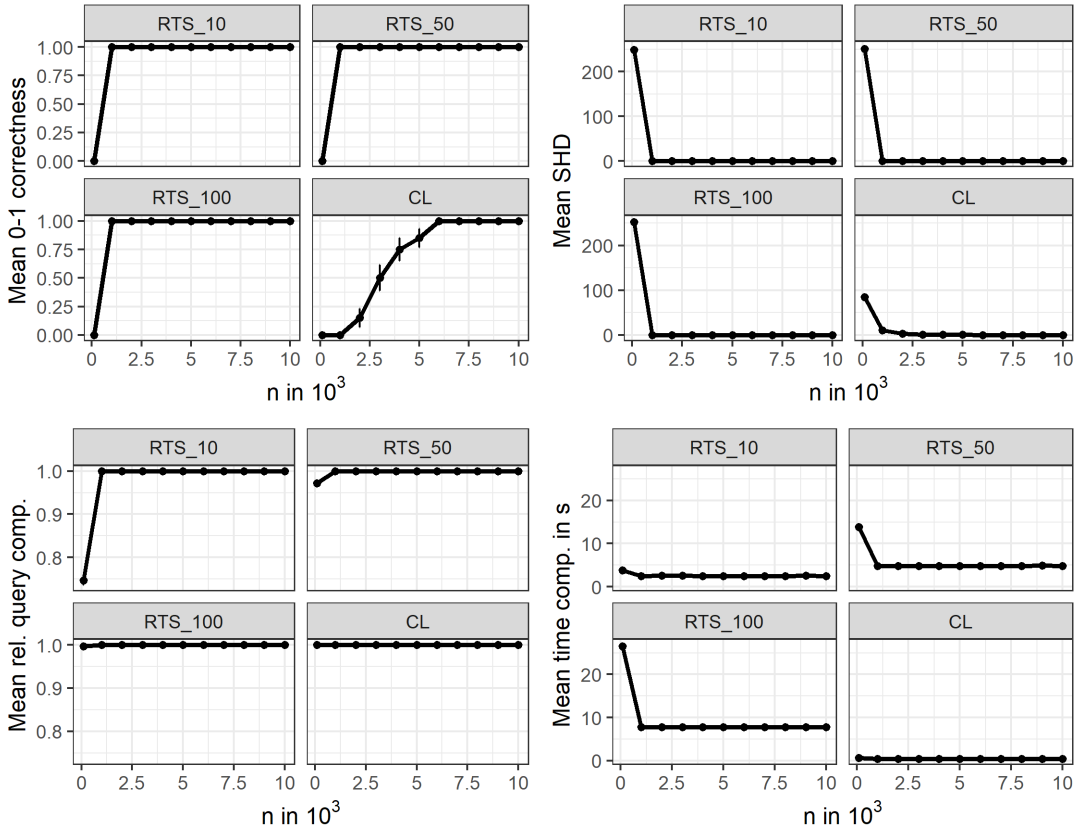
n	RTS_10	RTS_50	RTS_100	CL
100	0.05 (0.05)	0.05 (0.05)	0 (0)	0 (0)
1000	0.15 (0.08)	0.05 (0.05)	0 (0)	0 (0)
2000	0 (0)	0 (0)	0 (0)	0 (0)
3000	0.1 (0.07)	0.1 (0.07)	0 (0)	0 (0)
4000	0 (0)	0 (0)	0 (0)	0 (0)
5000	0.05 (0.05)	0.05 (0.05)	0 (0)	0 (0)
6000	0.15 (0.08)	0 (0)	0 (0)	0 (0)
7000	0.15 (0.08)	0.1 (0.07)	0.05 (0.05)	0 (0)
8000	0.05 (0.05)	0 (0)	0.05 (0.05)	0 (0)
9000	0 (0)	0 (0)	0 (0)	0 (0)
10000	0.05 (0.05)	0.05 (0.05)	0.05 (0.05)	0 (0)

RTS_10	RTS_50	RTS_100	CL
40 (20)	90 (20)	140 (20)	188 (1)
13 (9)	60 (20)	140 (10)	116 (2)
14 (9)	70 (10)	70 (20)	74 (2)
5.6 (0.8)	40 (10)	80 (10)	53 (2)
20 (10)	50 (10)	50 (10)	43 (2)
7 (3)	22 (6)	60 (10)	34 (1)
7 (3)	30 (10)	50 (10)	31 (1)
8 (3)	14 (4)	36 (9)	27 (1)
11 (6)	40 (10)	60 (20)	23 (1)
12 (3)	24 (8)	40 (10)	20 (1)
5.1 (0.7)	23 (9)	19 (4)	18 (1)

n	RTS_10	RTS_50	RTS_100	CL
100	0.99 (0)	1 (0)	1 (0)	1 (0)
1000	0.99 (0)	1 (0)	1 (0)	1 (0)
2000	0.99 (0)	0.99 (0)	1 (0)	1 (0)
3000	0.99 (0)	1 (0)	1 (0)	1 (0)
4000	0.99 (0)	0.99 (0)	1 (0)	1 (0)
5000	0.99 (0.01)	1 (0)	1 (0)	1 (0)
6000	0.99 (0.01)	1 (0)	1 (0)	1 (0)
7000	0.99 (0.01)	1 (0)	1 (0)	1 (0)
8000	0.99 (0)	1 (0)	1 (0)	1 (0)
9000	0.97 (0.01)	1 (0)	1 (0)	1 (0)
10000	0.99 (0)	1 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.53 (0.02)	4.96 (0.04)	8.06 (0.04)	0.63 (0.02)
2.56 (0.02)	5.05 (0.06)	8.9 (0.2)	0.64 (0.02)
2.53 (0.02)	5.8 (0.3)	9.3 (0.5)	0.59 (0.02)
2.52 (0.02)	5.9 (0.6)	11.5 (0.9)	0.58 (0.01)
2.54 (0.02)	6.5 (0.6)	12 (1)	0.55 (0.01)
2.47 (0.01)	6.1 (0.4)	14 (1)	0.51 (0.02)
2.53 (0.02)	5.7 (0.4)	16 (2)	0.54 (0.02)
2.55 (0.02)	5.9 (0.4)	19 (3)	0.5 (0.02)
2.56 (0.04)	6.9 (0.5)	15 (2)	0.51 (0.02)
2.6 (0.03)	6.3 (0.4)	16 (3)	0.5 (0.01)
2.53 (0.02)	5.7 (0.2)	15 (2)	0.49 (0.02)

Figure 4.14.: Star with $|V| = 100$ and $\alpha = \alpha' = 5 \cdot 10^{-4}$ using strictly diagonally dominant matrices.



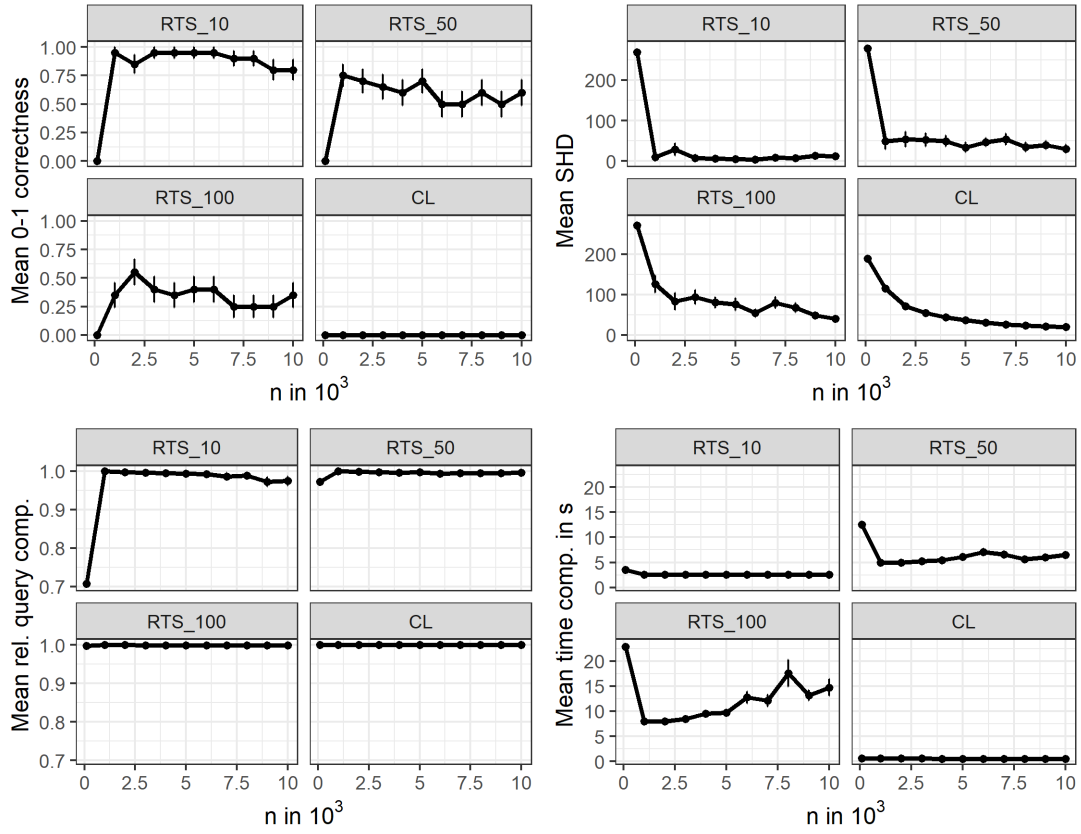
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	1 (0)	1 (0)	1 (0)	0 (0)
2000	1 (0)	1 (0)	1 (0)	0.15 (0.08)
3000	1 (0)	1 (0)	1 (0)	0.5 (0.1)
4000	1 (0)	1 (0)	1 (0)	0.8 (0.1)
5000	1 (0)	1 (0)	1 (0)	0.85 (0.08)
6000	1 (0)	1 (0)	1 (0)	1 (0)
7000	1 (0)	1 (0)	1 (0)	1 (0)
8000	1 (0)	1 (0)	1 (0)	1 (0)
9000	1 (0)	1 (0)	1 (0)	1 (0)
10000	1 (0)	1 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
249 (3)	251 (2)	253 (2)	84 (1)
0 (0)	0 (0)	0 (0)	11 (1)
0 (0)	0 (0)	0 (0)	3.1 (0.5)
0 (0)	0 (0)	0 (0)	1.3 (0.3)
0 (0)	0 (0)	0 (0)	0.6 (0.3)
0 (0)	0 (0)	0 (0)	0.3 (0.2)
0 (0)	0 (0)	0 (0)	0 (0)
0 (0)	0 (0)	0 (0)	0 (0)
0 (0)	0 (0)	0 (0)	0 (0)
0 (0)	0 (0)	0 (0)	0 (0)
0 (0)	0 (0)	0 (0)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.75 (0.01)	0.97 (0)	1 (0)	1 (0)
1000	1 (0)	1 (0)	1 (0)	1 (0)
2000	1 (0)	1 (0)	1 (0)	1 (0)
3000	1 (0)	1 (0)	1 (0)	1 (0)
4000	1 (0)	1 (0)	1 (0)	1 (0)
5000	1 (0)	1 (0)	1 (0)	1 (0)
6000	1 (0)	1 (0)	1 (0)	1 (0)
7000	1 (0)	1 (0)	1 (0)	1 (0)
8000	1 (0)	1 (0)	1 (0)	1 (0)
9000	1 (0)	1 (0)	1 (0)	1 (0)
10000	1 (0)	1 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
3.79 (0.08)	13.9 (0.2)	26.6 (0.3)	0.59 (0.01)
2.51 (0.02)	4.85 (0.02)	7.74 (0.02)	0.39 (0.01)
2.54 (0.02)	4.83 (0.02)	7.76 (0.03)	0.4 (0.01)
2.53 (0.02)	4.86 (0.02)	7.77 (0.03)	0.38 (0.01)
2.51 (0.01)	4.85 (0.02)	7.77 (0.02)	0.4 (0.01)
2.51 (0.01)	4.84 (0.02)	7.8 (0.02)	0.4 (0.01)
2.52 (0.01)	4.84 (0.02)	7.76 (0.02)	0.39 (0.01)
2.52 (0.02)	4.86 (0.03)	7.75 (0.02)	0.41 (0.01)
2.5 (0.01)	4.86 (0.02)	7.77 (0.03)	0.41 (0.01)
2.52 (0.01)	4.87 (0.02)	7.81 (0.03)	0.41 (0.01)
2.5 (0.01)	4.84 (0.02)	7.74 (0.02)	0.41 (0.01)

Figure 4.15.: Star with $|V| = 100$ and $\alpha = \alpha' = \max(2 \cdot (1 - \Phi(1.64 \cdot (n/|V|)^{0.5})), 2 \cdot \sqrt{\text{Machine Precision}})$ using the Cholesky-method.



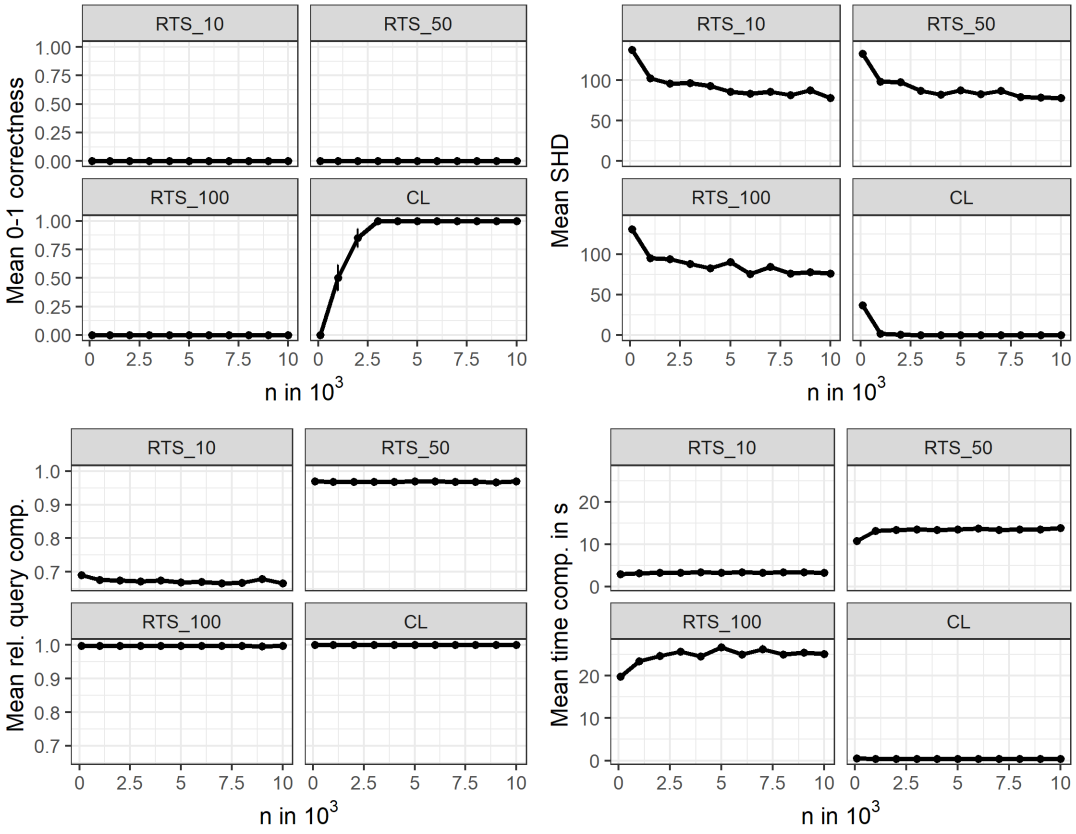
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0.95 (0.05)	0.8 (0.1)	0.3 (0.1)	0 (0)
2000	0.85 (0.08)	0.7 (0.1)	0.6 (0.1)	0 (0)
3000	0.95 (0.05)	0.7 (0.1)	0.4 (0.1)	0 (0)
4000	0.95 (0.05)	0.6 (0.1)	0.3 (0.1)	0 (0)
5000	0.95 (0.05)	0.7 (0.1)	0.4 (0.1)	0 (0)
6000	0.95 (0.05)	0.5 (0.1)	0.4 (0.1)	0 (0)
7000	0.9 (0.07)	0.5 (0.1)	0.2 (0.1)	0 (0)
8000	0.9 (0.07)	0.6 (0.1)	0.2 (0.1)	0 (0)
9000	0.8 (0.09)	0.5 (0.1)	0.2 (0.1)	0 (0)
10000	0.8 (0.09)	0.6 (0.1)	0.3 (0.1)	0 (0)

RTS_10	RTS_50	RTS_100	CL
269 (4)	278 (4)	270 (3)	188.2 (0.8)
10 (10)	50 (20)	130 (20)	115 (2)
30 (20)	50 (20)	80 (20)	71 (2)
8 (8)	50 (20)	90 (20)	54 (1)
6 (6)	50 (10)	80 (10)	44 (2)
6 (6)	30 (10)	80 (20)	37 (2)
4 (4)	50 (10)	50 (10)	30 (2)
9 (6)	50 (10)	80 (10)	26 (1)
7 (5)	30 (10)	70 (10)	24 (2)
14 (6)	40 (10)	48 (6)	21 (1)
13 (6)	30 (10)	40 (7)	20 (1)

n	RTS_10	RTS_50	RTS_100	CL
100	0.71 (0)	0.97 (0)	1 (0)	1 (0)
1000	1 (0)	1 (0)	1 (0)	1 (0)
2000	1 (0)	1 (0)	1 (0)	1 (0)
3000	1 (0)	1 (0)	1 (0)	1 (0)
4000	0.99 (0.01)	1 (0)	1 (0)	1 (0)
5000	0.99 (0.01)	1 (0)	1 (0)	1 (0)
6000	0.99 (0.01)	0.99 (0)	1 (0)	1 (0)
7000	0.99 (0.01)	0.99 (0)	1 (0)	1 (0)
8000	0.99 (0.01)	1 (0)	1 (0)	1 (0)
9000	0.97 (0.01)	0.99 (0)	1 (0)	1 (0)
10000	0.97 (0.01)	1 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
3.54 (0.08)	12.5 (0.2)	22.8 (0.4)	0.62 (0.02)
2.55 (0.02)	4.94 (0.03)	7.99 (0.04)	0.61 (0.02)
2.55 (0.02)	4.91 (0.04)	8.01 (0.05)	0.6 (0.02)
2.56 (0.02)	5.3 (0.1)	8.5 (0.1)	0.57 (0.02)
2.59 (0.03)	5.4 (0.2)	9.5 (0.4)	0.53 (0.02)
2.53 (0.02)	6.1 (0.5)	9.7 (0.5)	0.54 (0.02)
2.56 (0.02)	7.1 (0.6)	13 (1)	0.5 (0.01)
2.58 (0.03)	6.5 (0.6)	12 (1)	0.5 (0.01)
2.51 (0.02)	5.7 (0.3)	18 (3)	0.49 (0.01)
2.57 (0.03)	6 (0.3)	13 (1)	0.49 (0.01)
2.56 (0.02)	6.5 (0.5)	15 (2)	0.49 (0.01)

Figure 4.16.: Star with $|V| = 100$ and $\alpha = \alpha' = \max(2 \cdot (1 - \Phi(1.64 \cdot (n/|V|)^{0.5})), 2 \cdot \sqrt{\text{Machine Precision}})$ using strictly diagonally dominant matrices.



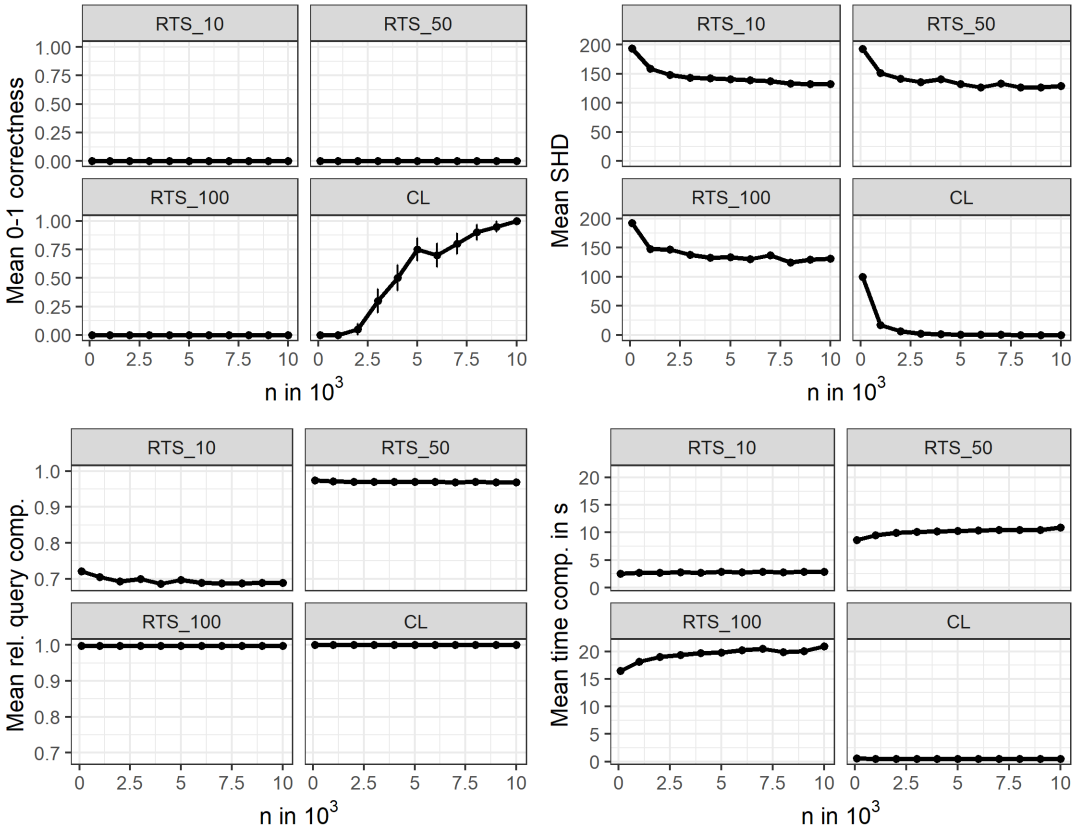
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0.5 (0.1)
2000	0 (0)	0 (0)	0 (0)	0.85 (0.08)
3000	0 (0)	0 (0)	0 (0)	1 (0)
4000	0 (0)	0 (0)	0 (0)	1 (0)
5000	0 (0)	0 (0)	0 (0)	1 (0)
6000	0 (0)	0 (0)	0 (0)	1 (0)
7000	0 (0)	0 (0)	0 (0)	1 (0)
8000	0 (0)	0 (0)	0 (0)	1 (0)
9000	0 (0)	0 (0)	0 (0)	1 (0)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
138 (3)	133 (3)	131 (3)	37 (2)
103 (3)	99 (3)	95 (3)	1.4 (0.4)
96 (3)	98 (3)	94 (4)	0.4 (0.2)
97 (3)	87 (3)	88 (3)	0 (0)
93 (3)	82 (4)	83 (3)	0 (0)
86 (3)	88 (3)	91 (3)	0 (0)
83 (3)	83 (3)	76 (3)	0 (0)
86 (3)	87 (3)	84 (3)	0 (0)
82 (3)	79 (3)	76 (3)	0 (0)
88 (3)	79 (3)	78 (3)	0 (0)
78 (4)	78 (3)	76 (3)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.69 (0)	0.97 (0)	1 (0)	1 (0)
1000	0.68 (0)	0.97 (0)	1 (0)	1 (0)
2000	0.67 (0.01)	0.97 (0)	1 (0)	1 (0)
3000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
4000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
5000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
6000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
7000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.68 (0)	0.97 (0)	1 (0)	1 (0)
10000	0.67 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.91 (0.05)	10.8 (0.2)	19.8 (0.3)	0.47 (0.01)
3.23 (0.08)	13.2 (0.4)	23.4 (0.7)	0.38 (0.01)
3.3 (0.1)	13.3 (0.3)	24.7 (0.8)	0.37 (0.01)
3.33 (0.08)	13.5 (0.4)	25.7 (0.5)	0.38 (0.01)
3.4 (0.1)	13.3 (0.3)	24.5 (0.4)	0.37 (0.01)
3.32 (0.08)	13.5 (0.3)	26.7 (0.5)	0.37 (0.01)
3.4 (0.1)	13.7 (0.4)	25 (0.5)	0.38 (0.01)
3.3 (0.1)	13.4 (0.4)	26.3 (0.8)	0.38 (0.01)
3.4 (0.1)	13.5 (0.4)	25 (0.5)	0.38 (0)
3.45 (0.09)	13.5 (0.3)	25.4 (0.6)	0.39 (0.01)
3.33 (0.09)	13.9 (0.4)	25.1 (0.7)	0.38 (0)

Figure 4.17.: Random trees with $|V| = 100$ and $\alpha = \alpha' = 0.05$ using the Cholesky-method.



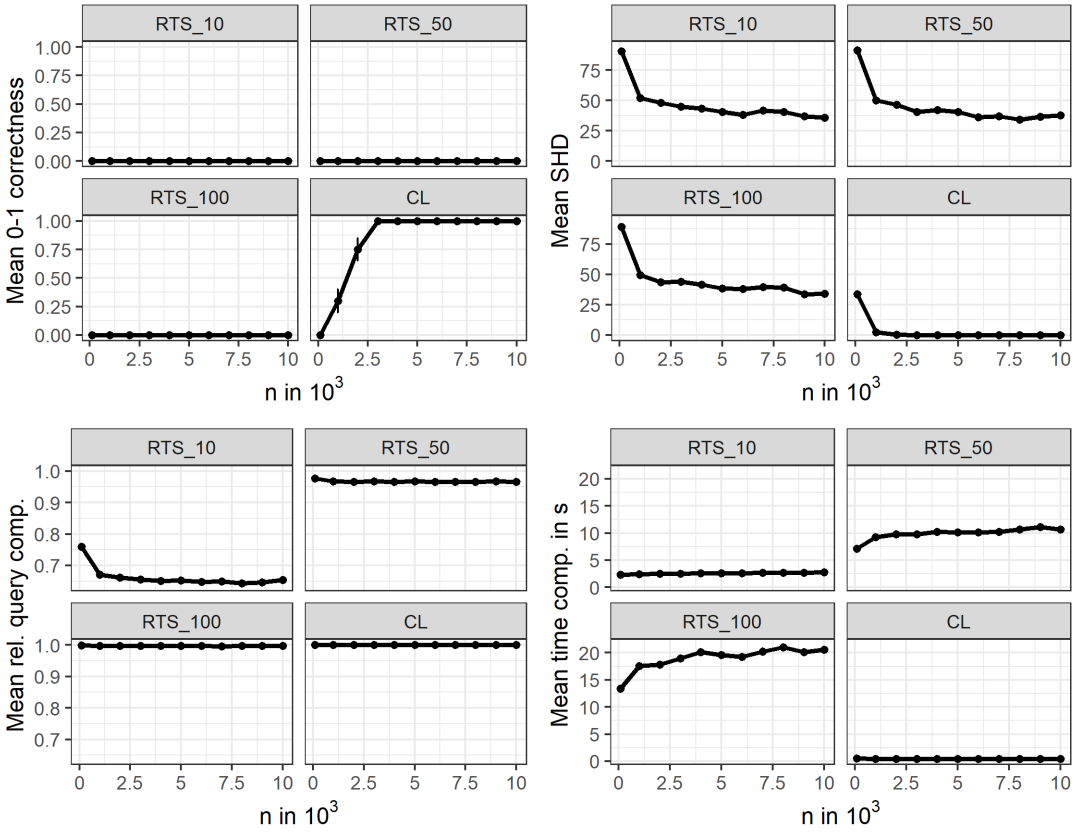
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0 (0)
2000	0 (0)	0 (0)	0 (0)	0.05 (0.05)
3000	0 (0)	0 (0)	0 (0)	0.3 (0.1)
4000	0 (0)	0 (0)	0 (0)	0.5 (0.1)
5000	0 (0)	0 (0)	0 (0)	0.8 (0.1)
6000	0 (0)	0 (0)	0 (0)	0.7 (0.1)
7000	0 (0)	0 (0)	0 (0)	0.8 (0.09)
8000	0 (0)	0 (0)	0 (0)	0.9 (0.07)
9000	0 (0)	0 (0)	0 (0)	0.95 (0.05)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
193 (3)	192 (2)	192 (3)	99 (2)
159 (2)	151 (2)	148 (2)	17 (1)
148 (4)	141 (2)	147 (3)	6.3 (0.8)
143 (3)	136 (3)	138 (3)	2.7 (0.5)
142 (3)	141 (3)	132 (2)	1.4 (0.4)
141 (3)	132 (2)	134 (2)	0.7 (0.3)
139 (3)	127 (3)	130 (4)	0.7 (0.3)
137 (2)	134 (3)	136 (3)	0.4 (0.2)
133 (2)	127 (2)	124 (2)	0.2 (0.1)
132 (3)	126 (3)	129 (2)	0.1 (0.1)
132 (3)	129 (3)	131 (3)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.72 (0)	0.97 (0)	1 (0)	1 (0)
1000	0.71 (0)	0.97 (0)	1 (0)	1 (0)
2000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
3000	0.7 (0)	0.97 (0)	1 (0)	1 (0)
4000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
5000	0.7 (0)	0.97 (0)	1 (0)	1 (0)
6000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
7000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.69 (0)	0.97 (0)	1 (0)	1 (0)
10000	0.69 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.56 (0.03)	8.7 (0.1)	16.5 (0.2)	0.59 (0.02)
2.75 (0.05)	9.6 (0.2)	18.2 (0.3)	0.5 (0.01)
2.71 (0.04)	9.9 (0.2)	19 (0.3)	0.48 (0.02)
2.82 (0.06)	10.1 (0.1)	19.3 (0.2)	0.48 (0.01)
2.72 (0.05)	10.2 (0.1)	19.7 (0.3)	0.48 (0.01)
2.88 (0.05)	10.3 (0.2)	19.8 (0.3)	0.49 (0.01)
2.81 (0.05)	10.4 (0.1)	20.2 (0.3)	0.52 (0.02)
2.87 (0.07)	10.5 (0.1)	20.5 (0.3)	0.49 (0.01)
2.76 (0.02)	10.5 (0.2)	19.9 (0.2)	0.49 (0.01)
2.84 (0.04)	10.5 (0.1)	20.1 (0.2)	0.5 (0.02)
2.88 (0.06)	11 (0.2)	20.9 (0.3)	0.51 (0.02)

Figure 4.18.: Random trees with $|V| = 100$ and $\alpha = \alpha' = 0.05$ using strictly diagonally dominant matrices.



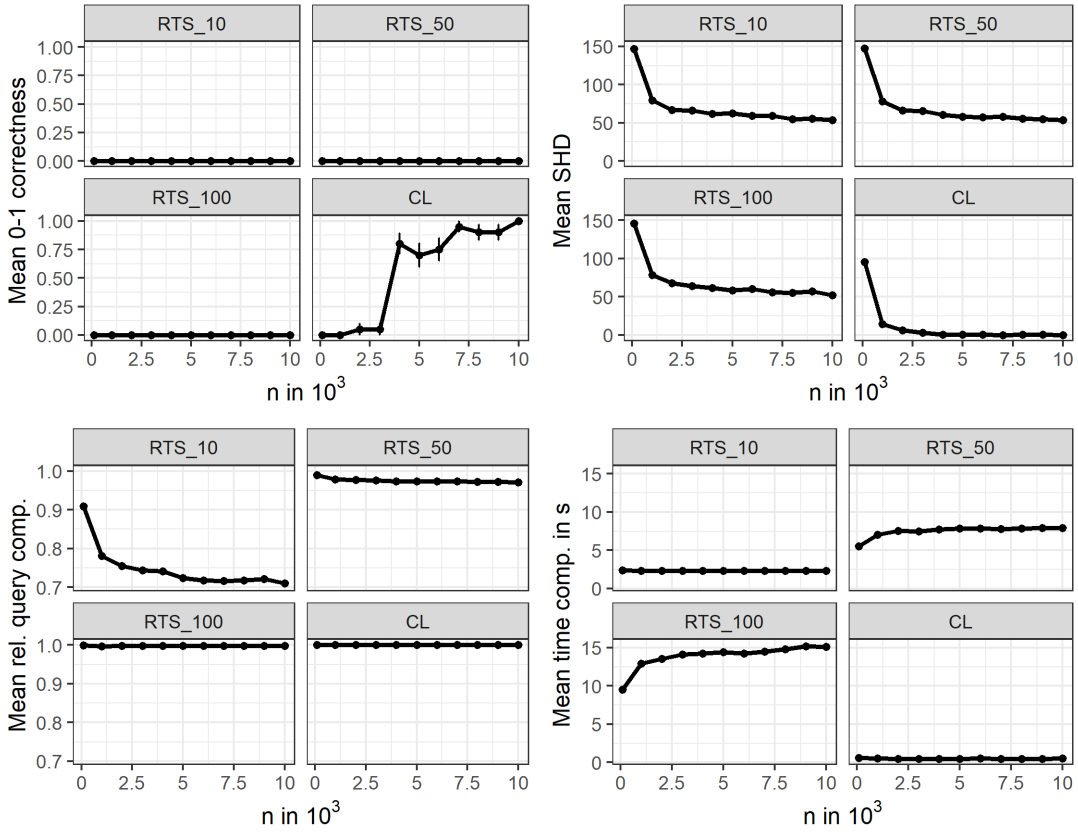
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0.3 (0.1)
2000	0 (0)	0 (0)	0 (0)	0.8 (0.1)
3000	0 (0)	0 (0)	0 (0)	1 (0)
4000	0 (0)	0 (0)	0 (0)	1 (0)
5000	0 (0)	0 (0)	0 (0)	1 (0)
6000	0 (0)	0 (0)	0 (0)	1 (0)
7000	0 (0)	0 (0)	0 (0)	1 (0)
8000	0 (0)	0 (0)	0 (0)	1 (0)
9000	0 (0)	0 (0)	0 (0)	1 (0)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
90 (2)	91 (3)	89 (2)	34 (2)
52 (2)	50 (2)	49 (2)	2.4 (0.5)
48 (2)	46 (2)	44 (2)	0.5 (0.2)
45 (2)	40 (2)	44 (2)	0 (0)
43 (1)	42 (2)	42 (2)	0 (0)
41 (2)	41 (2)	38 (2)	0 (0)
38 (2)	36 (2)	38 (2)	0 (0)
42 (2)	37 (1)	39 (2)	0 (0)
40 (2)	34 (2)	39 (1)	0 (0)
37 (2)	37 (2)	34 (1)	0 (0)
36 (2)	38 (2)	34 (2)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.76 (0.01)	0.98 (0)	1 (0)	1 (0)
1000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
2000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
3000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
4000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
5000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
6000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
7000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.64 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.65 (0)	0.97 (0)	1 (0)	1 (0)
10000	0.65 (0.01)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.31 (0.03)	7.1 (0.07)	13.4 (0.2)	0.49 (0.01)
2.43 (0.04)	9.2 (0.2)	17.5 (0.3)	0.42 (0.01)
2.51 (0.04)	9.8 (0.2)	17.8 (0.3)	0.42 (0.01)
2.49 (0.06)	9.8 (0.2)	19 (0.4)	0.44 (0.01)
2.56 (0.04)	10.3 (0.3)	20.1 (0.5)	0.41 (0.01)
2.64 (0.06)	10.2 (0.2)	19.6 (0.4)	0.45 (0.01)
2.62 (0.06)	10.2 (0.2)	19.2 (0.3)	0.45 (0.01)
2.7 (0.1)	10.2 (0.2)	20.2 (0.6)	0.44 (0.01)
2.66 (0.07)	10.7 (0.3)	21 (0.5)	0.44 (0.01)
2.68 (0.06)	11.1 (0.3)	20.1 (0.4)	0.46 (0.01)
2.8 (0.1)	10.6 (0.3)	20.6 (0.4)	0.46 (0.01)

Figure 4.19.: Random trees with $|V| = 100$ and $\alpha = \alpha' = 5 \cdot 10^{-4}$ using the Cholesky-method.



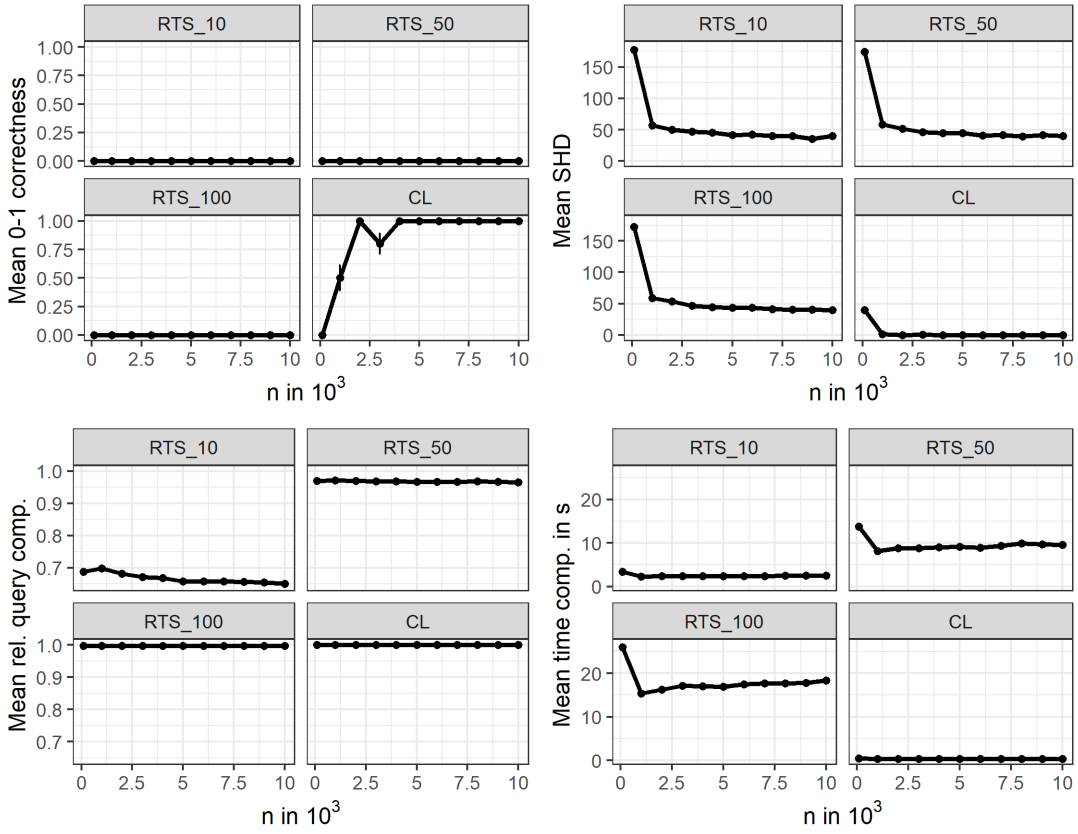
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0 (0)
2000	0 (0)	0 (0)	0 (0)	0.05 (0.05)
3000	0 (0)	0 (0)	0 (0)	0.05 (0.05)
4000	0 (0)	0 (0)	0 (0)	0.8 (0.09)
5000	0 (0)	0 (0)	0 (0)	0.7 (0.1)
6000	0 (0)	0 (0)	0 (0)	0.8 (0.1)
7000	0 (0)	0 (0)	0 (0)	0.95 (0.05)
8000	0 (0)	0 (0)	0 (0)	0.9 (0.07)
9000	0 (0)	0 (0)	0 (0)	0.9 (0.07)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
147 (2)	147 (2)	146 (2)	95 (1)
80 (2)	78 (2)	78 (1)	14.2 (0.7)
67 (2)	66 (1)	68 (1)	6 (0.9)
66 (2)	66 (1)	64 (2)	2.9 (0.3)
62 (1)	61 (1)	61 (1)	0.4 (0.2)
62 (1)	58 (1)	58 (2)	0.7 (0.3)
59 (2)	58 (2)	60 (1)	0.5 (0.2)
59 (2)	58 (1)	56 (2)	0.1 (0.1)
55 (2)	55 (2)	55 (1)	0.2 (0.1)
56 (2)	55 (1)	57 (0.9)	0.2 (0.1)
54 (2)	54 (1)	52 (2)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.91 (0)	0.99 (0)	1 (0)	1 (0)
1000	0.78 (0)	0.98 (0)	1 (0)	1 (0)
2000	0.76 (0)	0.98 (0)	1 (0)	1 (0)
3000	0.74 (0)	0.98 (0)	1 (0)	1 (0)
4000	0.74 (0)	0.97 (0)	1 (0)	1 (0)
5000	0.72 (0)	0.97 (0)	1 (0)	1 (0)
6000	0.72 (0)	0.97 (0)	1 (0)	1 (0)
7000	0.72 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.72 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.72 (0)	0.97 (0)	1 (0)	1 (0)
10000	0.71 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.36 (0.03)	5.54 (0.06)	9.5 (0.1)	0.58 (0.01)
2.34 (0.03)	7.03 (0.05)	12.9 (0.1)	0.5 (0.01)
2.31 (0.02)	7.51 (0.07)	13.5 (0.1)	0.46 (0.01)
2.29 (0.02)	7.45 (0.07)	14.1 (0.2)	0.48 (0.02)
2.34 (0.03)	7.69 (0.08)	14.2 (0.1)	0.47 (0.02)
2.32 (0.03)	7.83 (0.09)	14.4 (0.2)	0.48 (0.01)
2.29 (0.02)	7.88 (0.06)	14.3 (0.1)	0.5 (0.02)
2.33 (0.03)	7.78 (0.07)	14.5 (0.1)	0.48 (0.01)
2.32 (0.02)	7.83 (0.07)	14.8 (0.1)	0.49 (0.02)
2.34 (0.03)	7.9 (0.1)	15.2 (0.1)	0.48 (0.01)
2.29 (0.03)	7.9 (0.08)	15.1 (0.2)	0.5 (0.01)

Figure 4.20.: Random trees with $|V| = 100$ and $\alpha = \alpha' = 5 \cdot 10^{-4}$ using strictly diagonally dominant matrices.



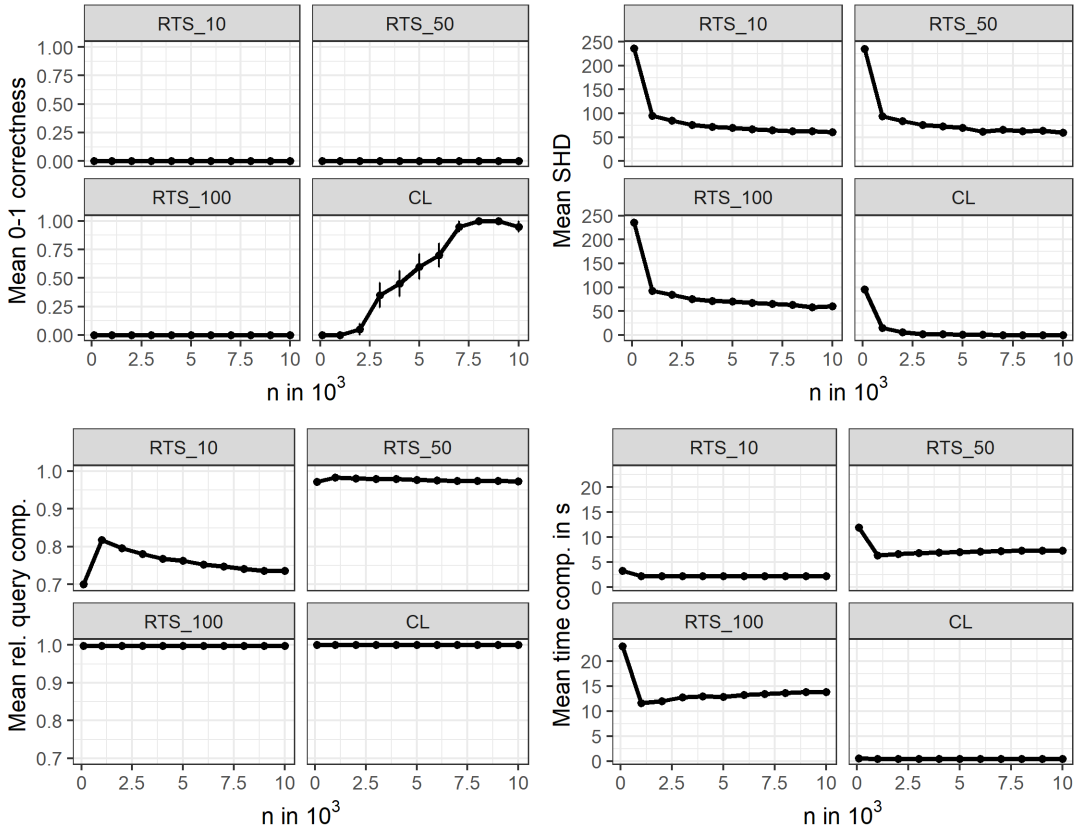
n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0.5 (0.1)
2000	0 (0)	0 (0)	0 (0)	1 (0)
3000	0 (0)	0 (0)	0 (0)	0.8 (0.09)
4000	0 (0)	0 (0)	0 (0)	1 (0)
5000	0 (0)	0 (0)	0 (0)	1 (0)
6000	0 (0)	0 (0)	0 (0)	1 (0)
7000	0 (0)	0 (0)	0 (0)	1 (0)
8000	0 (0)	0 (0)	0 (0)	1 (0)
9000	0 (0)	0 (0)	0 (0)	1 (0)
10000	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
178 (4)	175 (3)	172 (4)	40 (2)
57.3 (0.9)	58 (1)	59 (2)	1.5 (0.4)
50 (2)	52 (1)	54 (2)	0 (0)
47 (2)	46 (2)	47 (1)	0.4 (0.2)
46 (2)	44 (2)	44 (2)	0 (0)
41 (3)	45 (1)	43 (2)	0 (0)
43 (2)	41 (1)	44 (2)	0 (0)
40 (2)	42 (1)	41 (2)	0 (0)
40 (2)	39 (2)	40 (2)	0 (0)
36 (2)	41 (2)	40 (2)	0 (0)
40 (2)	40 (2)	39 (2)	0 (0)

n	RTS_10	RTS_50	RTS_100	CL
100	0.69 (0)	0.97 (0)	1 (0)	1 (0)
1000	0.7 (0.01)	0.97 (0)	1 (0)	1 (0)
2000	0.68 (0)	0.97 (0)	1 (0)	1 (0)
3000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
4000	0.67 (0)	0.97 (0)	1 (0)	1 (0)
5000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
6000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
7000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.66 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.66 (0.01)	0.97 (0)	1 (0)	1 (0)
10000	0.65 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
3.45 (0.07)	13.8 (0.3)	26 (0.5)	0.45 (0.01)
2.3 (0.03)	8.1 (0.1)	15.4 (0.4)	0.36 (0.01)
2.36 (0.03)	8.8 (0.1)	16.2 (0.2)	0.35 (0)
2.41 (0.04)	8.8 (0.2)	17.2 (0.5)	0.35 (0)
2.43 (0.04)	9 (0.2)	17 (0.2)	0.35 (0)
2.37 (0.03)	9.1 (0.1)	16.9 (0.2)	0.34 (0)
2.43 (0.04)	8.9 (0.1)	17.5 (0.3)	0.34 (0)
2.46 (0.04)	9.3 (0.2)	17.7 (0.4)	0.35 (0)
2.49 (0.04)	9.9 (0.2)	17.7 (0.3)	0.35 (0)
2.51 (0.06)	9.7 (0.3)	17.8 (0.3)	0.34 (0)
2.56 (0.06)	9.5 (0.1)	18.4 (0.3)	0.35 (0.01)

Figure 4.21.: Random trees with $|V| = 100$ and $\alpha = \alpha' = \max(2 \cdot (1 - \Phi(1.64 \cdot (n/|V|)^{0.5})), 2 \cdot \sqrt{\text{Machine Precision}})$ using the Cholesky-method.



n	RTS_10	RTS_50	RTS_100	CL
100	0 (0)	0 (0)	0 (0)	0 (0)
1000	0 (0)	0 (0)	0 (0)	0 (0)
2000	0 (0)	0 (0)	0 (0)	0.05 (0.05)
3000	0 (0)	0 (0)	0 (0)	0.3 (0.1)
4000	0 (0)	0 (0)	0 (0)	0.4 (0.1)
5000	0 (0)	0 (0)	0 (0)	0.6 (0.1)
6000	0 (0)	0 (0)	0 (0)	0.7 (0.1)
7000	0 (0)	0 (0)	0 (0)	0.95 (0.05)
8000	0 (0)	0 (0)	0 (0)	1 (0)
9000	0 (0)	0 (0)	0 (0)	1 (0)
10000	0 (0)	0 (0)	0 (0)	0.95 (0.05)

RTS_10	RTS_50	RTS_100	CL
236 (2)	236 (3)	236 (3)	96 (2)
95 (1)	94 (1)	92 (1)	14 (1)
85 (1)	84 (2)	85 (1)	5.9 (0.6)
76 (1)	76 (1)	76 (1)	2.2 (0.4)
72 (1)	73 (2)	72 (1)	1.7 (0.5)
70 (1)	70 (1)	70 (1)	0.9 (0.3)
67 (1)	62 (1)	67 (2)	0.7 (0.3)
65 (1)	66 (1)	65 (1)	0.1 (0.1)
63 (1)	62 (1)	64 (2)	0 (0)
62 (1)	64 (1)	59 (2)	0 (0)
60 (1)	60 (1)	60 (1)	0.1 (0.1)

n	RTS_10	RTS_50	RTS_100	CL
100	0.7 (0)	0.97 (0)	1 (0)	1 (0)
1000	0.82 (0)	0.98 (0)	1 (0)	1 (0)
2000	0.8 (0)	0.98 (0)	1 (0)	1 (0)
3000	0.78 (0)	0.98 (0)	1 (0)	1 (0)
4000	0.77 (0)	0.98 (0)	1 (0)	1 (0)
5000	0.76 (0)	0.98 (0)	1 (0)	1 (0)
6000	0.75 (0)	0.98 (0)	1 (0)	1 (0)
7000	0.75 (0)	0.97 (0)	1 (0)	1 (0)
8000	0.74 (0)	0.97 (0)	1 (0)	1 (0)
9000	0.74 (0)	0.97 (0)	1 (0)	1 (0)
10000	0.74 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
3.3 (0.05)	12 (0.2)	22.9 (0.3)	0.56 (0.01)
2.27 (0.03)	6.41 (0.05)	11.64 (0.09)	0.46 (0.01)
2.26 (0.02)	6.67 (0.07)	11.96 (0.09)	0.45 (0.01)
2.28 (0.02)	6.83 (0.07)	12.7 (0.1)	0.48 (0.02)
2.26 (0.02)	6.95 (0.07)	13 (0.1)	0.45 (0.01)
2.24 (0.02)	7.04 (0.06)	12.9 (0.1)	0.47 (0.01)
2.24 (0.02)	7.14 (0.05)	13.3 (0.1)	0.49 (0.02)
2.27 (0.02)	7.2 (0.06)	13.4 (0.1)	0.47 (0.01)
2.24 (0.03)	7.3 (0.06)	13.63 (0.06)	0.48 (0.01)
2.24 (0.02)	7.31 (0.08)	13.8 (0.1)	0.49 (0.01)
2.23 (0.02)	7.35 (0.07)	13.8 (0.1)	0.48 (0.01)

Figure 4.22.: Random trees with $|V| = 100$ and $\alpha = \alpha' = \max(2 \cdot (1 - \Phi(1.64 \cdot (n/|V|)^{0.5})), 2 \cdot \sqrt{\text{Machine Precision}})$ using strictly diagonally dominant matrices.

We observe that the Chow-Liu algorithm performs best in terms of correctness (compared to itself) if the true graph is a chain and worst if the underlying graph is a star; for random trees the performance of the Chow-Liu algorithm is between these two extremes. This observation empirically confirms Tan et al. (2010). For the chain, the Chow-Liu algorithm starts to return the correct tree for a sample size of $2000 - 3000$, for the star, a sample size of at least $5000 - 6000$ is needed. Also note that the Chow-Liu algorithm has a relative query complexity of 1, i.e. it needs to query each unique entry.

Regarding `ReconstructTreeSample`, we have looked at three different choices of α and α'; we have always chosen α and α' to be equal as the respective hypotheses tests are of the same nature. Given that we do the simulations from the perspective of testing each hypothesis at the same significance level, this approach is reasonable to us (from a multiple testing perspective, this would not be so reasonable). The first choice was $\alpha = \alpha' = 0.05$, the second choice was $\alpha = \alpha' = 5 \cdot 10^{-4}$ and for the third choice we decreased α and α' for increasing n; the formula was $\alpha = \alpha' = \max(2 \cdot (1 - \Phi(1.64 \cdot (n/|V|)^{0.5})), 2 \cdot \sqrt{\text{Machine Precision}})$.[8] This formula is motivated by the theoretical choice from equation (3.5); however, instead of using $n^{0.5}\rho_{min}/2$, which is only of theoretical use, we use $1.64 \cdot (n/|V|)^{0.5}$. This is a term that increases in n; the constant 1.64 is due to the fact that we wanted to have $\alpha = \alpha' \approx 0.1$ for $n = |V|$.

For each choice of α, we observe an improvement of correctness for increasing sample size. For $\alpha = \alpha' = 0.05$, the returned trees are the least correct for chains, stars and random trees; for $\alpha = \alpha' = 5 \cdot 10^{-4}$, the correctness has significantly improved, as can be seen by looking at the respective SHD plots. For the sample-size-dependent α resp. α', the SHD reaches 0 if the tree is the star and the correct tree is always returned. For that particular case, this happens for even smaller n than for the Chow-Liu algorithm. For the chain and random trees, the SHD does not reaches 0, only levels similarly to the one of $\alpha = \alpha' = 5 \cdot 10^{-4}$.

We observe a similar pattern for the time complexity. The time complexity of `ReconstructTreeSample` is usually lower for values of α and α' for which the returned tree is also more correct. This effect is especially visible for the star. We also observe that larger values of κ are associated with a higher time complexity. Recall that the raison d'être of `sCentralSample` and higher values of κ is to reduce the overall time complexity by doing some extra iterations that overall lead to less iterations. As higher values of κ are clearly associated to higher time complexity in the plots, one may be tempted to say that this did not work out. But if one looks closer, one observes that especially for the star and better values of α and

[8] The square root of the machine precision is used by the R-function *all.equal* to distinguish between different input values. For our simulations, $\sqrt{\text{Machine Precision}} = 1.490116 \cdot 10^{-8}$.

α', the time complexity does not scale linearly in κ anymore, as roughly stated in the proof of Theorem 3.26. From $\kappa = 10$ to $\kappa = 50$ one would expect an increase of approximately factor 5, as is roughly the case in the other simulations, but one only sees an increase of factor 2. For the chain, this effect is not really observable. We note that more returned centroids are especially beneficial in the case of stars. If `sCentralSample` does not return the centroid as a cut vertex, the maximal remaining subgraph has size $|V| - 1$. However, if it chooses the centroid, the maximal remaining subgraph has size 1, which is a huge decrease in terms of complexity. This decrease can also be observed in the plots. We conclude that the star is more optimal in terms of time complexity, especially for larger n `ReconstructTreeSample` seems to be faster for the star.

We observe, that the relative query complexity is only significantly less than 1 for the lowest value of κ, and that only for the chain. We observe some decrease of the query complexity in n, which indicates that more central vertices which have been returned from `sCentralSample` lead to less query complexity.

Another observation is that the Chow-Liu algorithm is orders of magnitudes faster than the fastest version of `ReconstructTreeSample`. We investigate two possible reasons for that. The first one is that the number of vertices is simply not large enough and the second reason is that each extra query does not cost much time.
To investigate the first reason, and also study high-dimensional settings, we do another simulation. We look at the vertex sizes $100, 500, 1000$ and 1500. Moreover, we work with at a particular version of `ReconstructTreeSample`, namely without multiple testing adjustment, $\alpha = \alpha' = 5 \cdot 10^{-4}$ and $\kappa = 16 \ln(|V|)$. Furthermore, we look at different sample sizes; we imagine that $n/|V|$ is either $10, 5, 1, 1/5, 1/10$. For each case, we do 20 replications and state the mean with 1 standard error. Moreover, we normalize both the SHD and the time complexity by the number of vertices. The results are given in the following figures.

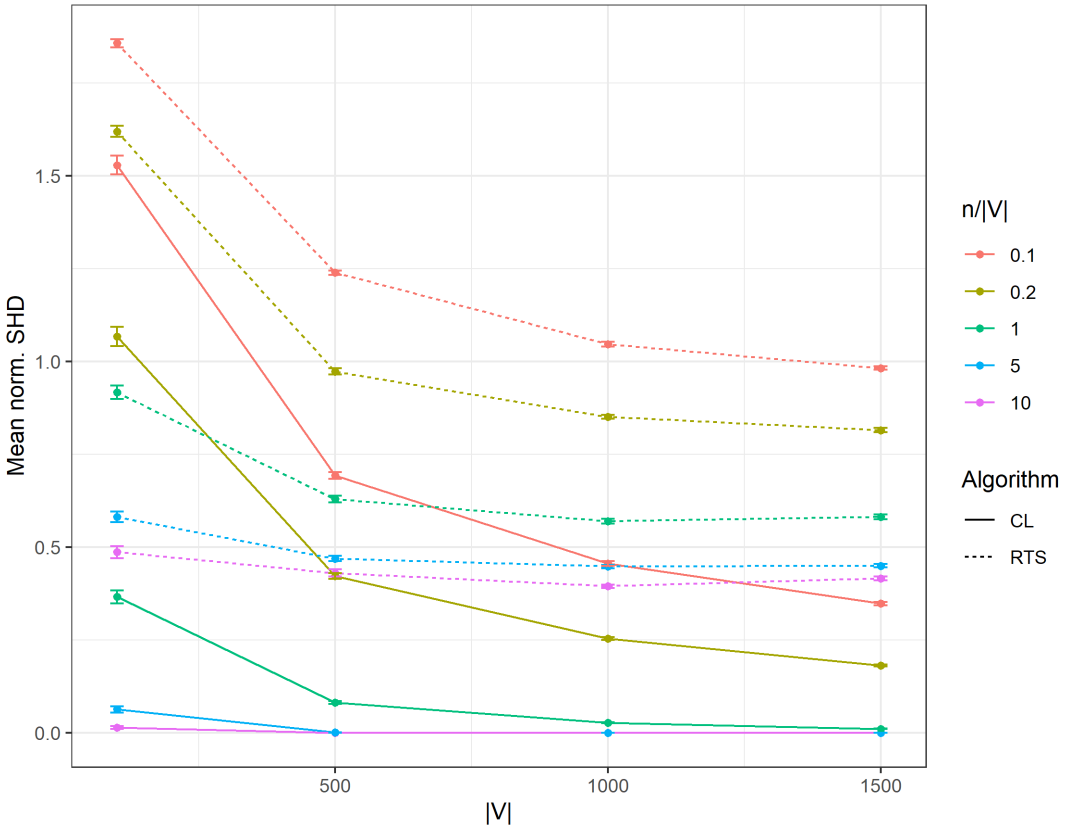


Figure 4.23.: Mean normalized SHD with 1 standard error using the Cholesky method.

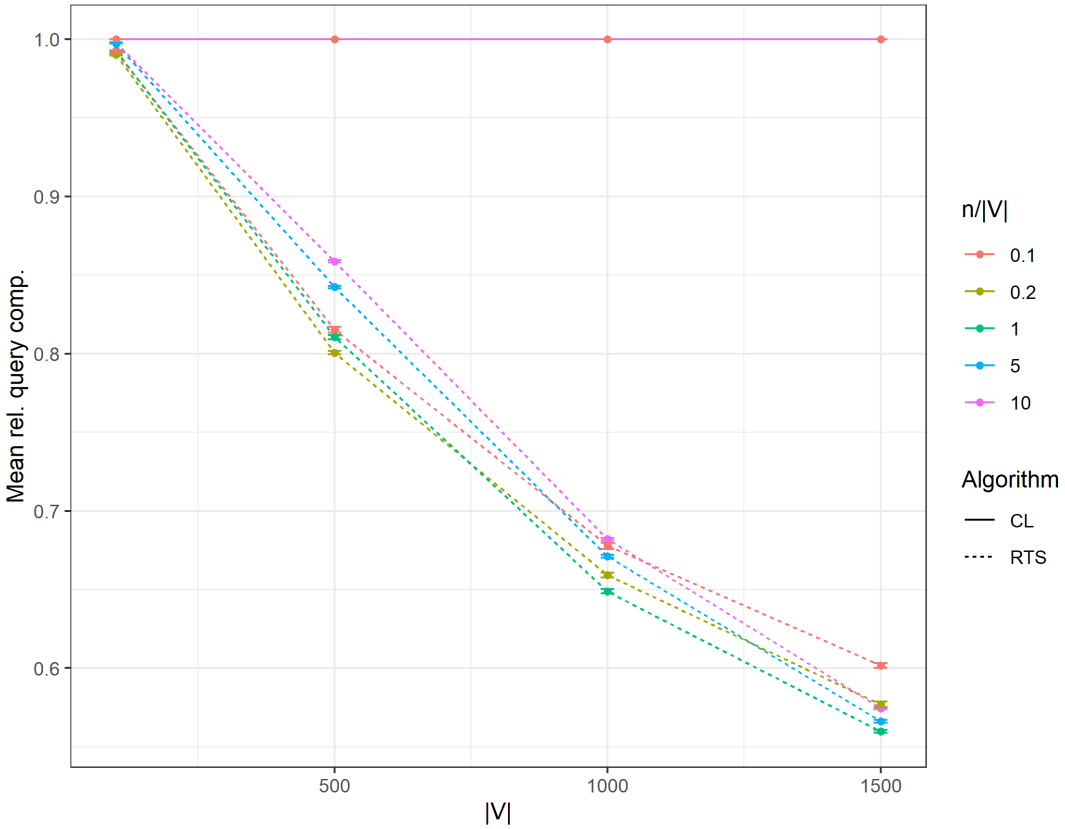


Figure 4.24.: Mean relative query complexity with 1 standard error using the Cholesky method.

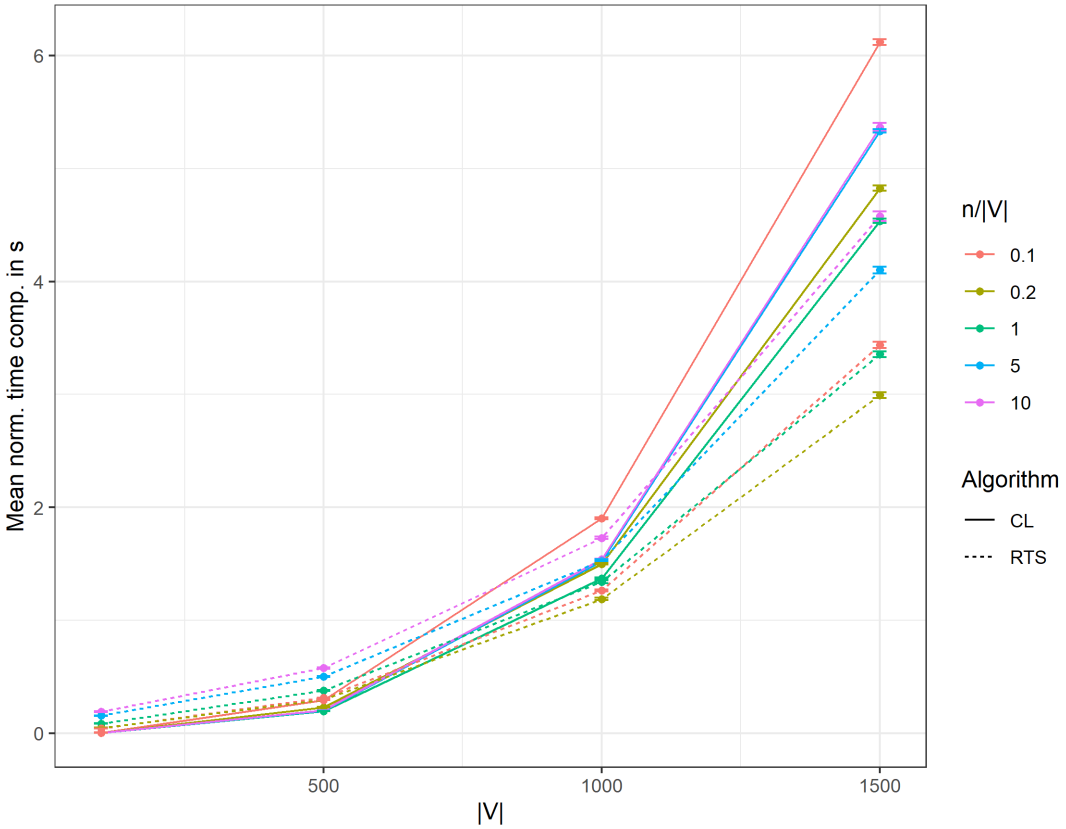


Figure 4.25.: Mean normalized time complexity in seconds with 1 standard error using the Cholesky method.

In this simulation, we can see that `ReconstructTreeSample` is able to reach a level of normalized correctness measured in terms of the normalized SHD, which is significantly better than that of randomly guessing trees and, moreover, which is not increasing for larger $|V|$ (if not to say decreasing), while at the same time, the relative query complexity decreases in $|V|$. Note that the choice of $\kappa = 16 \ln(|V|)$ is motivated by the theoretical value given in Theorem 3.26 and Theorem 3.30.

We also see that the time complexity for the Chow-Liu algorithm is only smaller for small $|V|$, for larger $|V|$, `ReconstructTreeSample` is faster. The relative query complexity of the Chow-Liu algorithm is always 1.

We also observe that these results are true for high-dimensional settings; even for $|V| = 10n$, the correctness of the returned tree is reasonably good, i.e. it is much better than randomly guessing trees. The relative query complexity is only slightly worse in comparison to lower-dimensional settings. We also note that the normalized SHD is usually lower for lower-dimensional settings, but that is not

surprising.

To investigate the second reason for the fact that the Chow-Liu algorithm is orders of magnitude faster (for a small number of vertices), we look at 3 different querying setups; our usual oracle setup, where each entry has been already precalculated (approach 1), and two non-oracle setups, where the respective entries in the covariance matrix need to be calculated when queried. If an entry has already been queried at some point during the execution of `ReconstructTreeSample`, it is stored in the workspace and does need to be calculated again. The calculation also includes reading in data from a csv-file (our mock-database); we consider reading in all observations at once (approach 2) or in an online-fashion using the Welford-algorithm (approach 3). For the Welford-algorithm we refer to Knuth (1998), Welford (1962) and Chan et al. (1983). We fix $|V| = 100$ and plot the mean execution time for reconstructing random trees. We do not focus on the correctness of the three different approaches, because in theory, each approach yields the same result for a queried entry. In practice, however, there may be some issues with numerical stability. We again have 20 replications for each approach and plot the mean with 1 standard error.

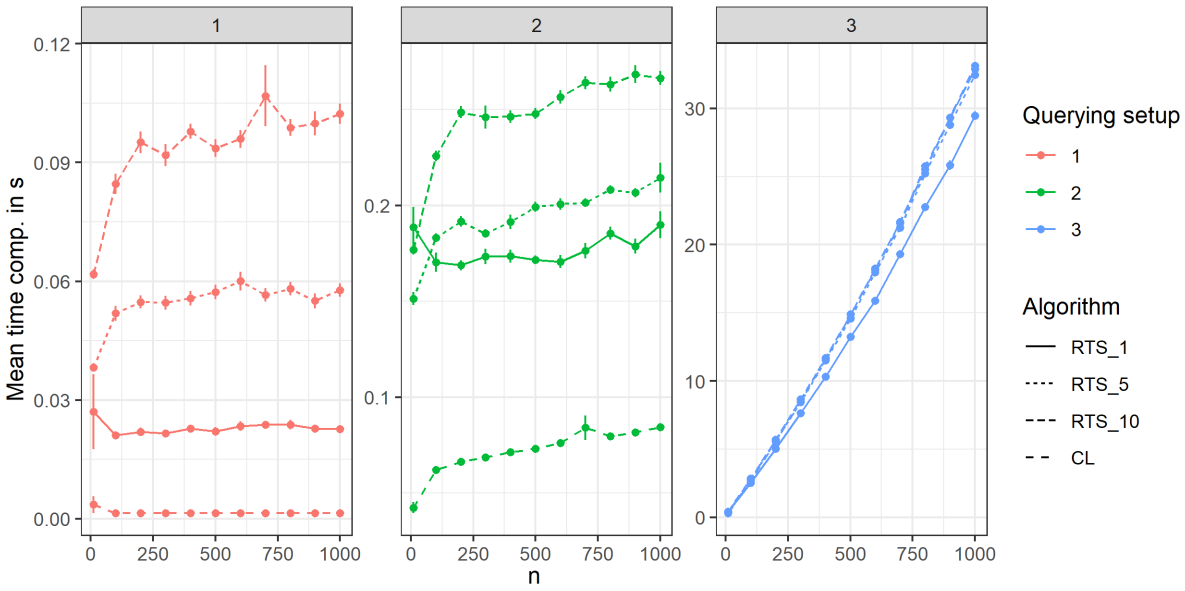
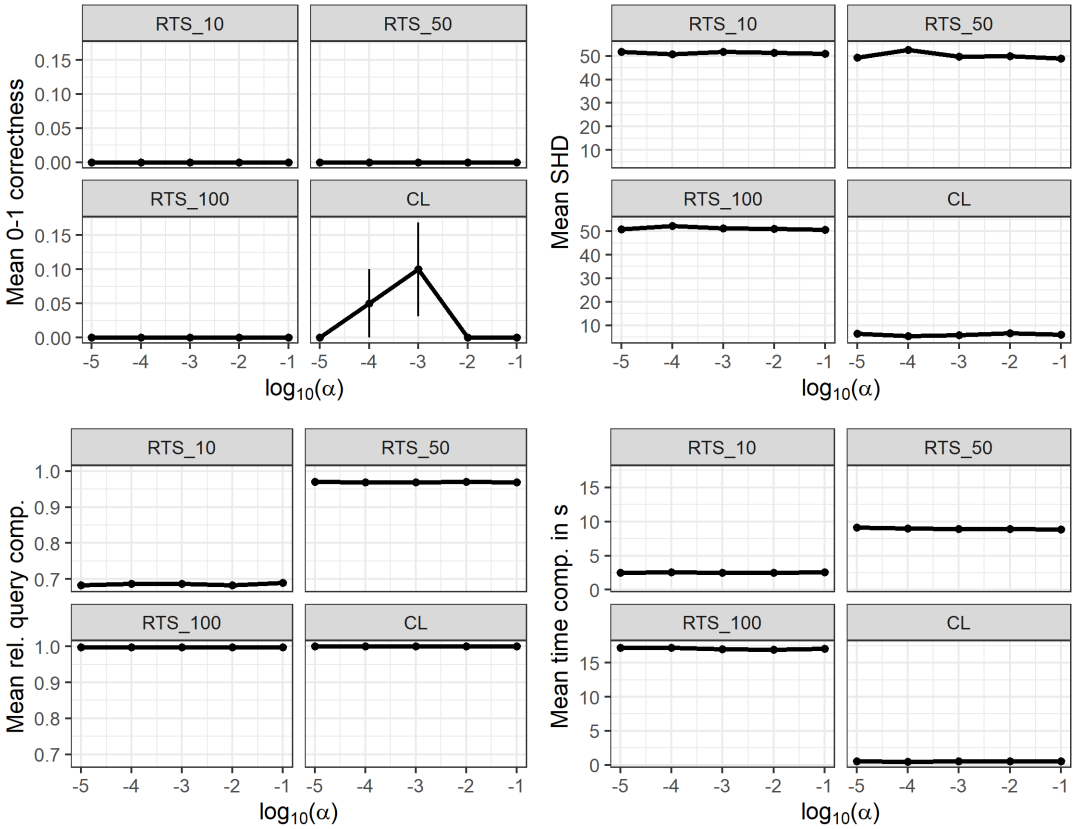


Figure 4.26.: The total execution time of `ReconstructTreeSample` with respect to the querying setup. Approach 1: Oracle-approach; approach 2: No-oracle-approach with directly reading in all observations at once; approach 3 : No-oracle-approach with reading in observations one by one using the Welford algorithm. The parameters are $|V| = 10$ and $\alpha = \alpha' = 5 \cdot 10^{-4}$.

We observe that the Chow-Liu algorithm becomes equally slow for approach 3, i.e. where a query costs the most.

For the following simulations, we go back to $|V| = 100$. We fix $n = 500$ or $n = 2500$ and vary α while $\alpha' = 5 \cdot 10^{-4}$ is fixed. We again vary n and κ and do again 20 replications as before. We then repeat by varying α' while fixing $\alpha = 5 \cdot 10^{-4}$.



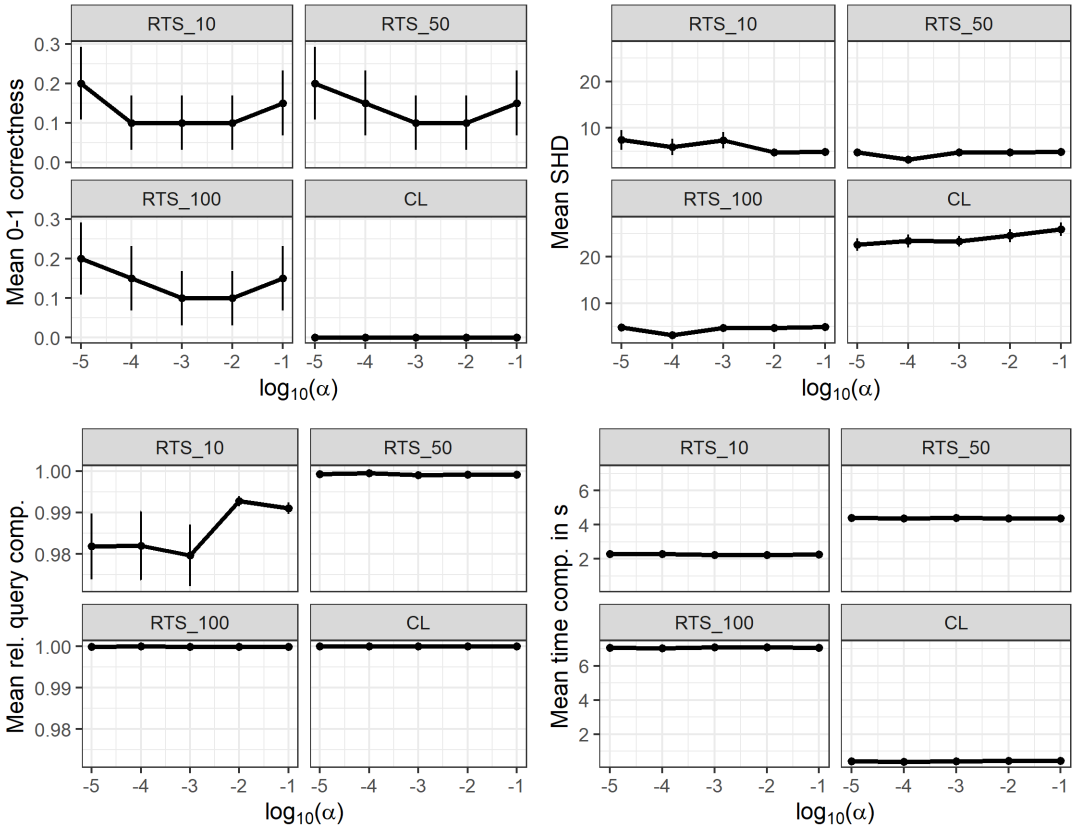
$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0 (0)	0 (0)	0 (0)	0 (0)
-4.00	0 (0)	0 (0)	0 (0)	0.05 (0.05)
-3.00	0 (0)	0 (0)	0 (0)	0.1 (0.07)
-2.00	0 (0)	0 (0)	0 (0)	0 (0)
-1.00	0 (0)	0 (0)	0 (0)	0 (0)

RTS_10	RTS_50	RTS_100	CL
52 (1)	49 (1)	51 (2)	6.5 (0.6)
51 (1)	53 (1)	52 (1)	5.4 (0.7)
52 (1)	50 (1)	51 (1)	5.8 (0.7)
51.4 (0.9)	50 (1)	51 (0.9)	6.7 (0.7)
51 (0.9)	49 (1)	51 (1)	6.1 (0.6)

$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.68 (0)	0.97 (0)	1 (0)	1 (0)
-4.00	0.69 (0)	0.97 (0)	1 (0)	1 (0)
-3.00	0.69 (0)	0.97 (0)	1 (0)	1 (0)
-2.00	0.68 (0)	0.97 (0)	1 (0)	1 (0)
-1.00	0.69 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.53 (0.03)	9.1 (0.1)	17.2 (0.2)	0.52 (0.02)
2.54 (0.03)	8.96 (0.09)	17.2 (0.2)	0.51 (0.01)
2.53 (0.02)	8.9 (0.1)	17 (0.2)	0.53 (0.02)
2.49 (0.03)	8.9 (0.1)	16.9 (0.2)	0.52 (0.01)
2.54 (0.03)	8.88 (0.06)	17 (0.1)	0.53 (0.02)

Figure 4.27.: Chain with $|V| = 100, n = 500$ and $\alpha' = 5 \cdot 10^{-4}$ using the Cholesky method.



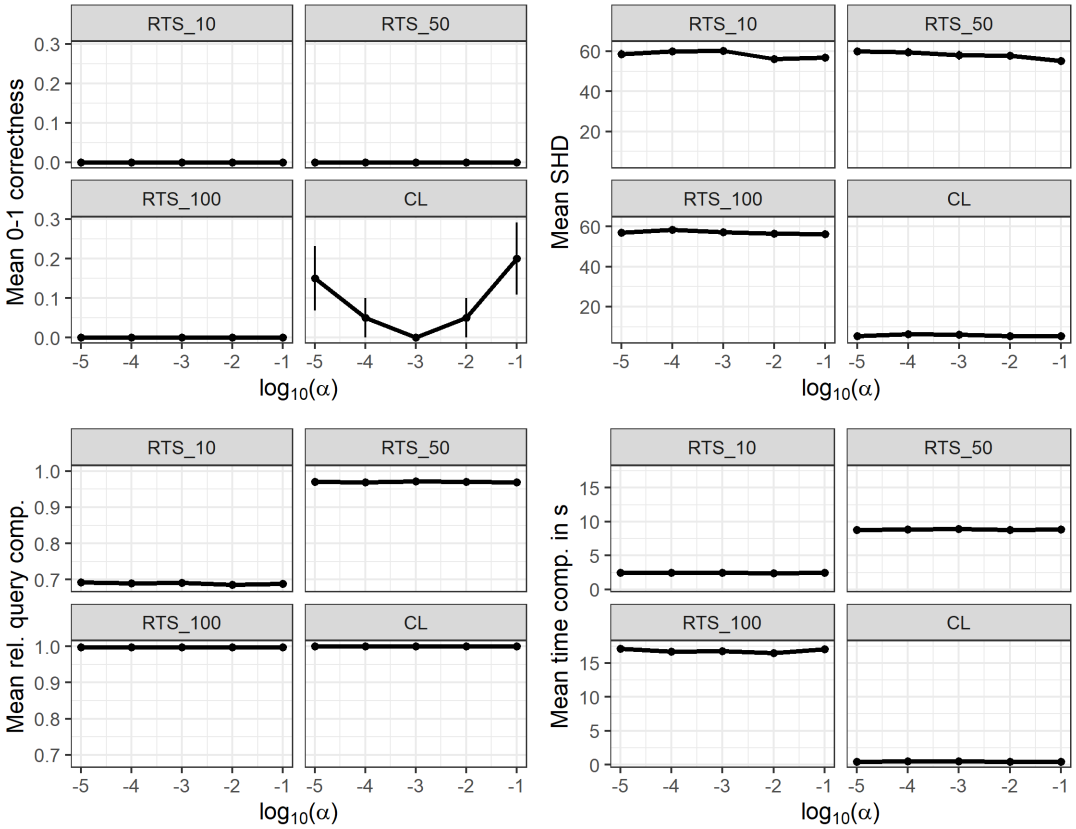
$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.2 (0.09)	0.2 (0.09)	0.2 (0.09)	0 (0)
-4.00	0.1 (0.07)	0.15 (0.08)	0.15 (0.08)	0 (0)
-3.00	0.1 (0.07)	0.1 (0.07)	0.1 (0.07)	0 (0)
-2.00	0.1 (0.07)	0.1 (0.07)	0.1 (0.07)	0 (0)
-1.00	0.15 (0.08)	0.15 (0.08)	0.15 (0.08)	0 (0)

RTS_10	RTS_50	RTS_100	CL
7 (2)	4.8 (0.7)	4.8 (0.7)	23 (1)
6 (2)	3.2 (0.6)	3.2 (0.6)	23 (1)
7 (2)	4.8 (0.7)	4.8 (0.7)	23 (1)
4.8 (0.7)	4.8 (0.7)	4.8 (0.7)	24 (1)
4.9 (0.7)	4.9 (0.7)	4.9 (0.7)	26 (1)

$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.98 (0.01)	1 (0)	1 (0)	1 (0)
-4.00	0.98 (0.01)	1 (0)	1 (0)	1 (0)
-3.00	0.98 (0.01)	1 (0)	1 (0)	1 (0)
-2.00	0.99 (0)	1 (0)	1 (0)	1 (0)
-1.00	0.99 (0)	1 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.3 (0.03)	4.41 (0.02)	7.07 (0.05)	0.41 (0.01)
2.28 (0.02)	4.39 (0.03)	7.04 (0.04)	0.4 (0.01)
2.25 (0.02)	4.41 (0.02)	7.1 (0.03)	0.41 (0.01)
2.24 (0.01)	4.37 (0.02)	7.11 (0.04)	0.44 (0.01)
2.25 (0.01)	4.39 (0.03)	7.07 (0.04)	0.44 (0.02)

Figure 4.28.: Star with $|V| = 100, n = 500$ and $\alpha' = 5 \cdot 10^{-4}$ using the Cholesky method.



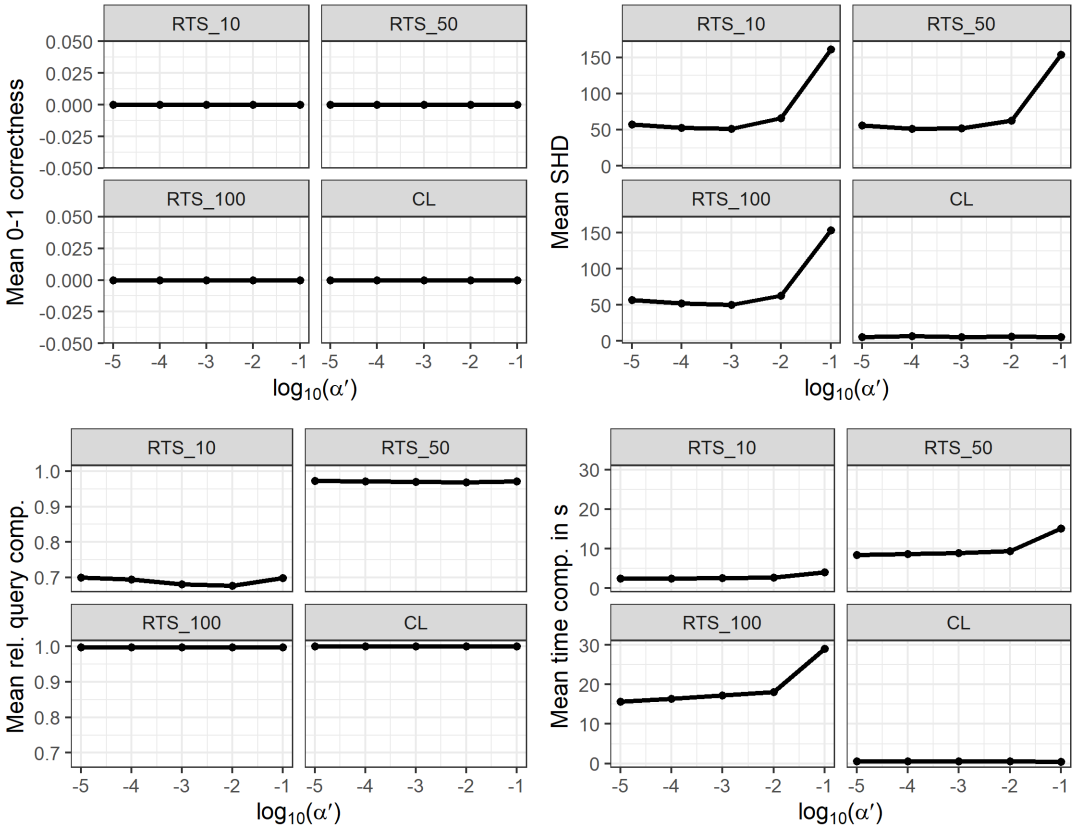
$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0 (0)	0 (0)	0 (0)	0.15 (0.08)
-4.00	0 (0)	0 (0)	0 (0)	0.05 (0.05)
-3.00	0 (0)	0 (0)	0 (0)	0 (0)
-2.00	0 (0)	0 (0)	0 (0)	0.05 (0.05)
-1.00	0 (0)	0 (0)	0 (0)	0.2 (0.09)

RTS_10	RTS_50	RTS_100	CL
58 (2)	60 (2)	57 (1)	5.4 (0.8)
60 (2)	60 (2)	58 (2)	6.4 (0.9)
60 (2)	58 (2)	57 (2)	6 (0.7)
56 (1)	58 (2)	57 (2)	5.3 (0.6)
57 (2)	55 (2)	56 (2)	5.4 (0.8)

$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.69 (0)	0.97 (0)	1 (0)	1 (0)
-4.00	0.69 (0)	0.97 (0)	1 (0)	1 (0)
-3.00	0.69 (0)	0.97 (0)	1 (0)	1 (0)
-2.00	0.69 (0)	0.97 (0)	1 (0)	1 (0)
-1.00	0.69 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.46 (0.04)	8.8 (0.1)	17.1 (0.4)	0.47 (0.01)
2.47 (0.04)	8.9 (0.1)	16.7 (0.3)	0.48 (0.01)
2.44 (0.05)	8.9 (0.1)	16.8 (0.3)	0.47 (0.01)
2.41 (0.04)	8.8 (0.1)	16.5 (0.3)	0.46 (0.01)
2.43 (0.05)	8.9 (0.1)	17 (0.4)	0.46 (0.01)

Figure 4.29.: Random trees with $|V| = 100, n = 500$ and $\alpha' = 5 \cdot 10^{-4}$ using the Cholesky method.



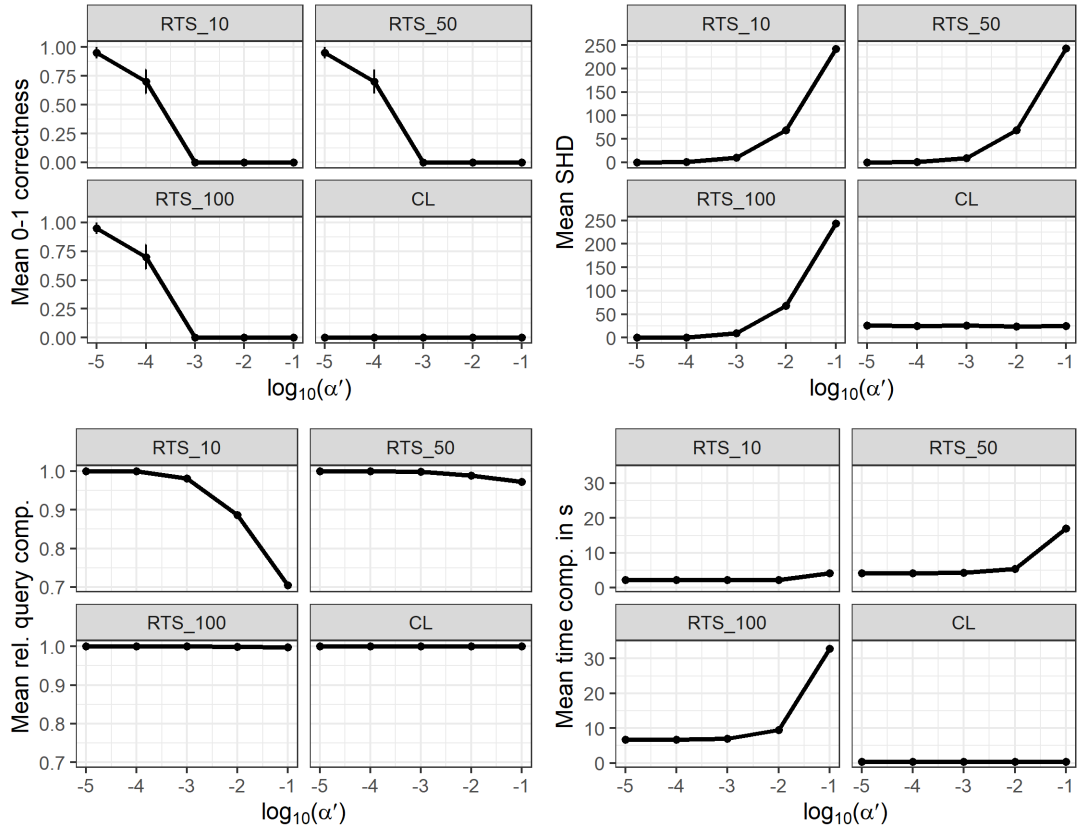
$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	0 (0)	0 (0)	0 (0)	0 (0)
-4.00	0 (0)	0 (0)	0 (0)	0 (0)
-3.00	0 (0)	0 (0)	0 (0)	0 (0)
-2.00	0 (0)	0 (0)	0 (0)	0 (0)
-1.00	0 (0)	0 (0)	0 (0)	0 (0)

RTS_10	RTS_50	RTS_100	CL
58 (2)	56 (2)	56.6 (0.9)	5.5 (0.7)
52 (1)	52 (1)	52 (1)	6.8 (0.6)
51 (2)	52 (1)	50 (1)	5.5 (0.5)
66 (2)	62 (2)	63 (2)	6.6 (0.6)
161 (3)	154 (4)	154 (3)	5.6 (0.6)

$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.7 (0)	0.97 (0)	1 (0)	1 (0)
-4.00	0.69 (0)	0.97 (0)	1 (0)	1 (0)
-3.00	0.68 (0)	0.97 (0)	1 (0)	1 (0)
-2.00	0.68 (0)	0.97 (0)	1 (0)	1 (0)
-1.00	0.7 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.44 (0.03)	8.4 (0.1)	15.7 (0.1)	0.51 (0.02)
2.47 (0.03)	8.7 (0.1)	16.4 (0.2)	0.51 (0.02)
2.53 (0.03)	8.9 (0.08)	17.2 (0.2)	0.51 (0.02)
2.65 (0.04)	9.4 (0.1)	18.1 (0.2)	0.51 (0.02)
4.05 (0.08)	15.1 (0.3)	29.1 (0.6)	0.48 (0.02)

Figure 4.30.: Chain with $|V| = 100, n = 500$ and $\alpha = 5 \cdot 10^{-4}$ using the Cholesky method.



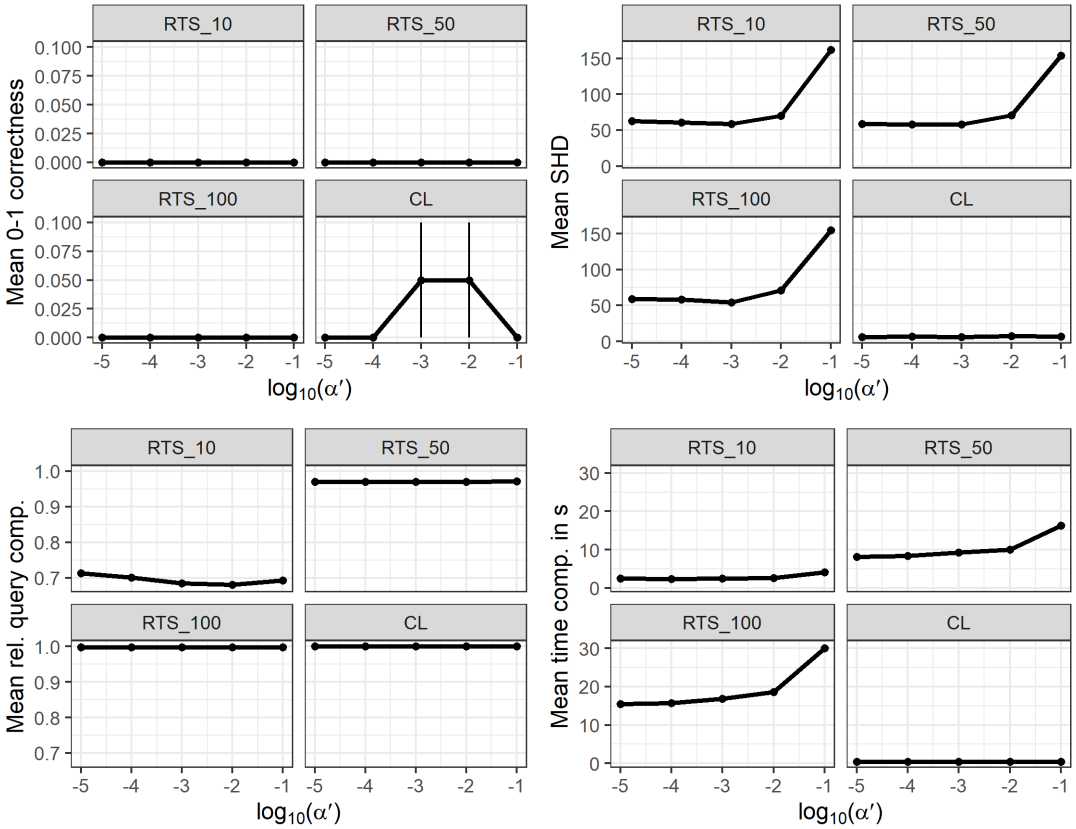
$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.95 (0.05)	0.95 (0.05)	0.95 (0.05)	0 (0)
-4.00	0.7 (0.1)	0.7 (0.1)	0.7 (0.1)	0 (0)
-3.00	0 (0)	0 (0)	0 (0)	0 (0)
-2.00	0 (0)	0 (0)	0 (0)	0 (0)
-1.00	0 (0)	0 (0)	0 (0)	0 (0)

RTS_10	RTS_50	RTS_100	CL
0.1 (0.1)	0.1 (0.1)	0.1 (0.1)	26 (2)
0.6 (0.2)	0.6 (0.2)	0.6 (0.2)	25 (2)
10 (1)	9.1 (0.8)	9.1 (0.8)	26 (2)
69 (2)	68 (2)	68 (2)	24 (2)
242 (3)	243 (3)	243 (3)	25 (2)

$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	1 (0)	1 (0)	1 (0)	1 (0)
-4.00	1 (0)	1 (0)	1 (0)	1 (0)
-3.00	0.98 (0)	1 (0)	1 (0)	1 (0)
-2.00	0.89 (0)	0.99 (0)	1 (0)	1 (0)
-1.00	0.71 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.19 (0.02)	4.2 (0.03)	6.69 (0.05)	0.39 (0.01)
2.2 (0.02)	4.22 (0.04)	6.72 (0.05)	0.39 (0.01)
2.23 (0.02)	4.33 (0.04)	7.03 (0.06)	0.42 (0.01)
2.19 (0.04)	5.38 (0.06)	9.4 (0.1)	0.41 (0.01)
4.18 (0.09)	17 (0.4)	32.8 (0.7)	0.41 (0.01)

Figure 4.31.: Star with $|V| = 100, n = 500$ and $\alpha = 5 \cdot 10^{-4}$ using the Cholesky method.



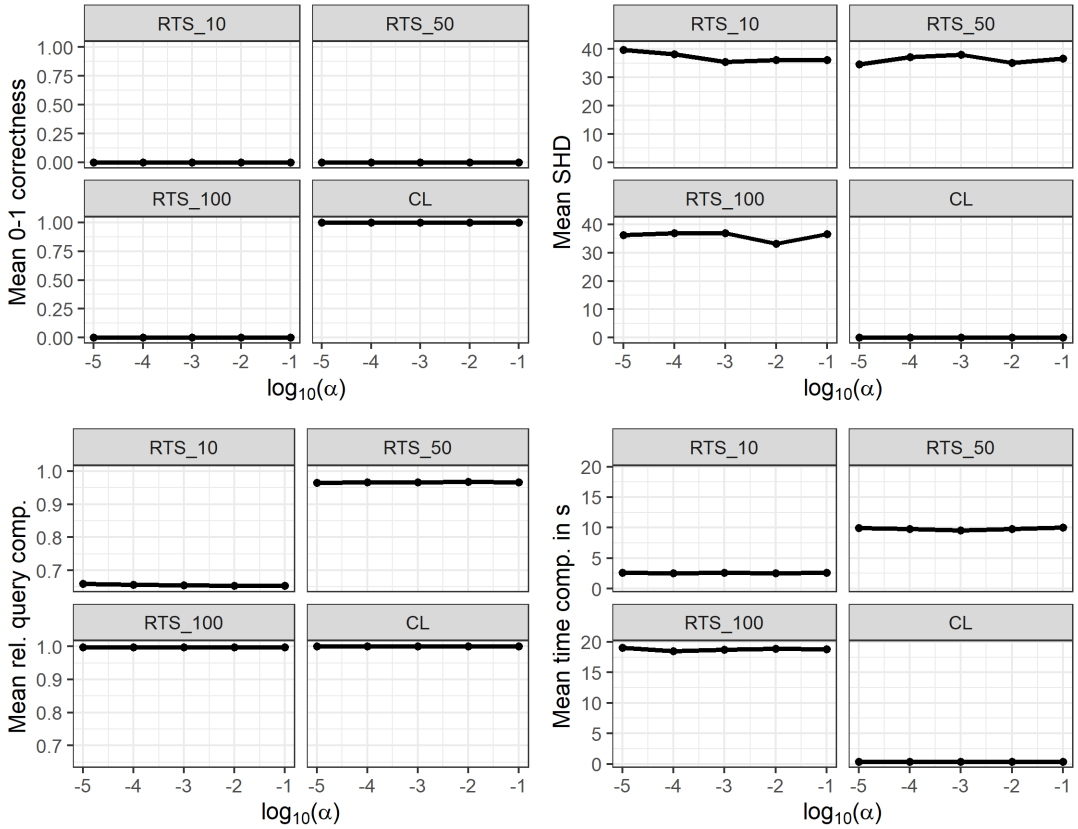
$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	0 (0)	0 (0)	0 (0)	0 (0)
-4.00	0 (0)	0 (0)	0 (0)	0 (0)
-3.00	0 (0)	0 (0)	0 (0)	0.05 (0.05)
-2.00	0 (0)	0 (0)	0 (0)	0.05 (0.05)
-1.00	0 (0)	0 (0)	0 (0)	0 (0)

RTS_10	RTS_50	RTS_100	CL
63 (2)	59 (2)	59 (2)	6.2 (0.8)
61 (1)	58 (1)	58 (2)	6.6 (0.8)
59 (2)	58 (2)	54 (1)	5.8 (0.6)
70 (2)	70 (2)	71 (2)	7.1 (0.8)
162 (4)	154 (3)	155 (4)	7 (0.7)

$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.71 (0)	0.97 (0)	1 (0)	1 (0)
-4.00	0.7 (0)	0.97 (0)	1 (0)	1 (0)
-3.00	0.68 (0)	0.97 (0)	1 (0)	1 (0)
-2.00	0.68 (0)	0.97 (0)	1 (0)	1 (0)
-1.00	0.69 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.45 (0.04)	8.1 (0.1)	15.4 (0.3)	0.46 (0.01)
2.41 (0.04)	8.4 (0.1)	15.7 (0.2)	0.47 (0.01)
2.5 (0.06)	9.3 (0.2)	16.8 (0.2)	0.45 (0.01)
2.66 (0.05)	10 (0.2)	18.6 (0.5)	0.45 (0.01)
4.1 (0.1)	16.3 (0.3)	30 (0.5)	0.46 (0.01)

Figure 4.32.: Random trees with $|V| = 100, n = 500$ and $\alpha = 5 \cdot 10^{-4}$ using the Cholesky method.



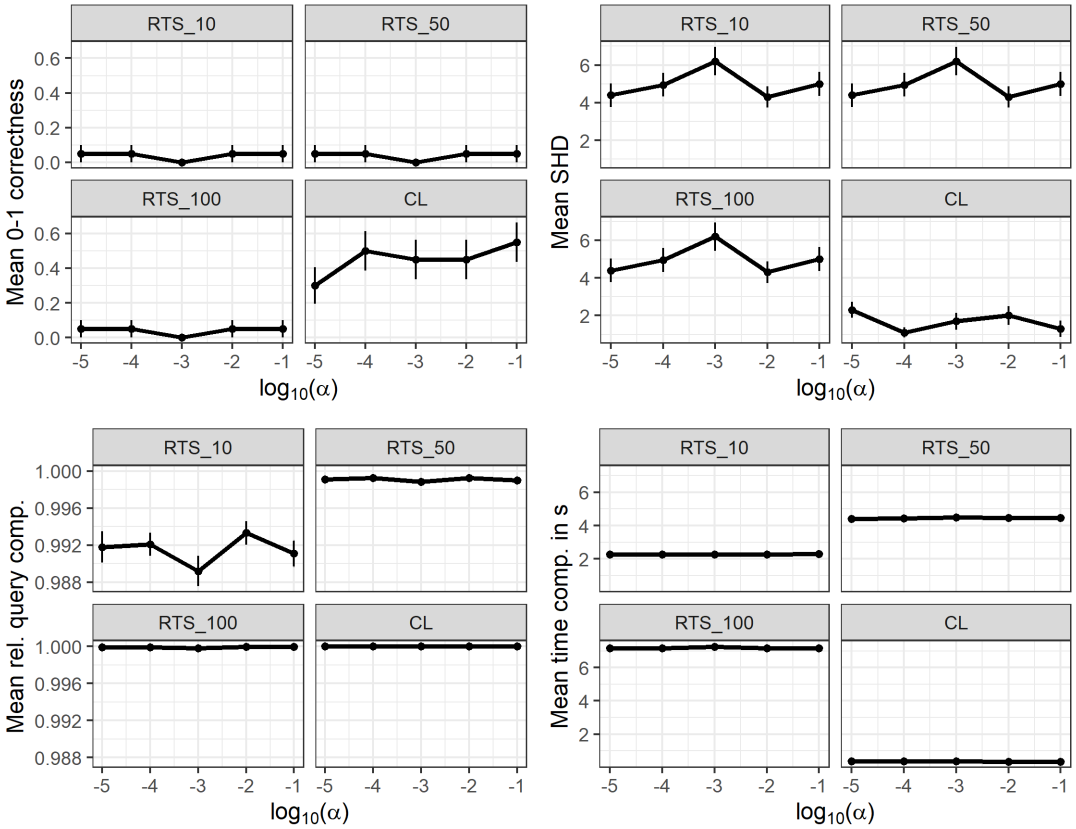
$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0 (0)	0 (0)	0 (0)	1 (0)
-4.00	0 (0)	0 (0)	0 (0)	1 (0)
-3.00	0 (0)	0 (0)	0 (0)	1 (0)
-2.00	0 (0)	0 (0)	0 (0)	1 (0)
-1.00	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
40 (1)	35 (1)	36.1 (0.8)	0 (0)
38 (1)	37 (1)	36.9 (0.5)	0 (0)
35.5 (0.9)	38 (1)	37 (0.9)	0 (0)
36 (1)	35.2 (0.9)	33.1 (0.8)	0 (0)
36 (1)	37 (1)	36 (1)	0 (0)

$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.66 (0)	0.97 (0)	1 (0)	1 (0)
-4.00	0.66 (0)	0.97 (0)	1 (0)	1 (0)
-3.00	0.65 (0)	0.97 (0)	1 (0)	1 (0)
-2.00	0.65 (0)	0.97 (0)	1 (0)	1 (0)
-1.00	0.65 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.61 (0.04)	9.9 (0.1)	19.1 (0.2)	0.38 (0)
2.57 (0.03)	9.8 (0.1)	18.5 (0.2)	0.38 (0)
2.6 (0.03)	9.6 (0.1)	18.7 (0.2)	0.38 (0)
2.57 (0.03)	9.8 (0.1)	18.9 (0.2)	0.38 (0)
2.6 (0.04)	10 (0.1)	18.8 (0.2)	0.38 (0)

Figure 4.33.: Chain with $|V| = 100, n = 2500$ and $\alpha' = 5 \cdot 10^{-4}$ using the Cholesky method.



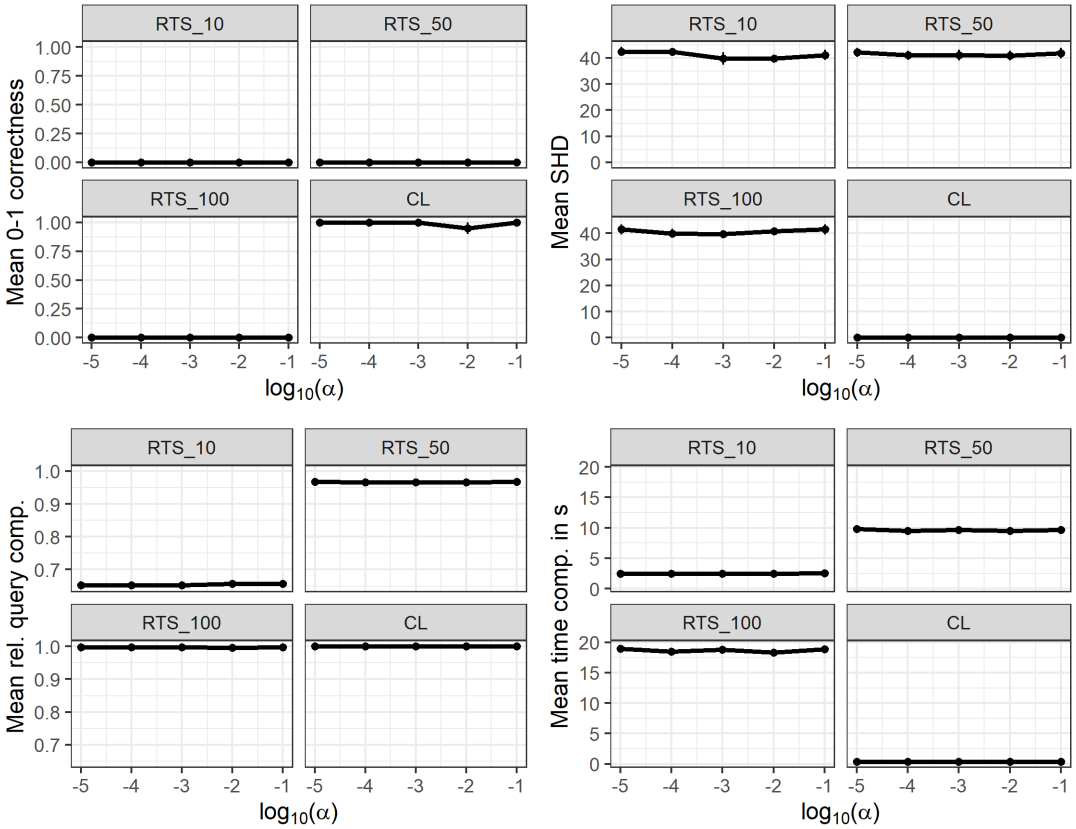
$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.05 (0.05)	0.05 (0.05)	0.05 (0.05)	0.3 (0.1)
-4.00	0.05 (0.05)	0.05 (0.05)	0.05 (0.05)	0.5 (0.1)
-3.00	0 (0)	0 (0)	0 (0)	0.4 (0.1)
-2.00	0.05 (0.05)	0.05 (0.05)	0.05 (0.05)	0.4 (0.1)
-1.00	0.05 (0.05)	0.05 (0.05)	0.05 (0.05)	0.6 (0.1)

RTS_10	RTS_50	RTS_100	CL
4.4 (0.6)	4.4 (0.6)	4.4 (0.6)	2.3 (0.4)
5 (0.6)	5 (0.6)	5 (0.6)	1.1 (0.3)
6.2 (0.8)	6.2 (0.8)	6.2 (0.8)	1.7 (0.4)
4.3 (0.6)	4.3 (0.6)	4.3 (0.6)	2 (0.5)
5 (0.6)	5 (0.6)	5 (0.6)	1.3 (0.4)

$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.99 (0)	1 (0)	1 (0)	1 (0)
-4.00	0.99 (0)	1 (0)	1 (0)	1 (0)
-3.00	0.99 (0)	1 (0)	1 (0)	1 (0)
-2.00	0.99 (0)	1 (0)	1 (0)	1 (0)
-1.00	0.99 (0)	1 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.27 (0.02)	4.41 (0.02)	7.17 (0.03)	0.38 (0.01)
2.27 (0.01)	4.44 (0.02)	7.17 (0.04)	0.36 (0.01)
2.26 (0.01)	4.48 (0.02)	7.24 (0.03)	0.36 (0)
2.26 (0.01)	4.47 (0.02)	7.17 (0.03)	0.36 (0.01)
2.28 (0.01)	4.46 (0.02)	7.17 (0.04)	0.36 (0.01)

Figure 4.34.: Star with $|V| = 100, n = 2500$ and $\alpha' = 5 \cdot 10^{-4}$ using the Cholesky method.



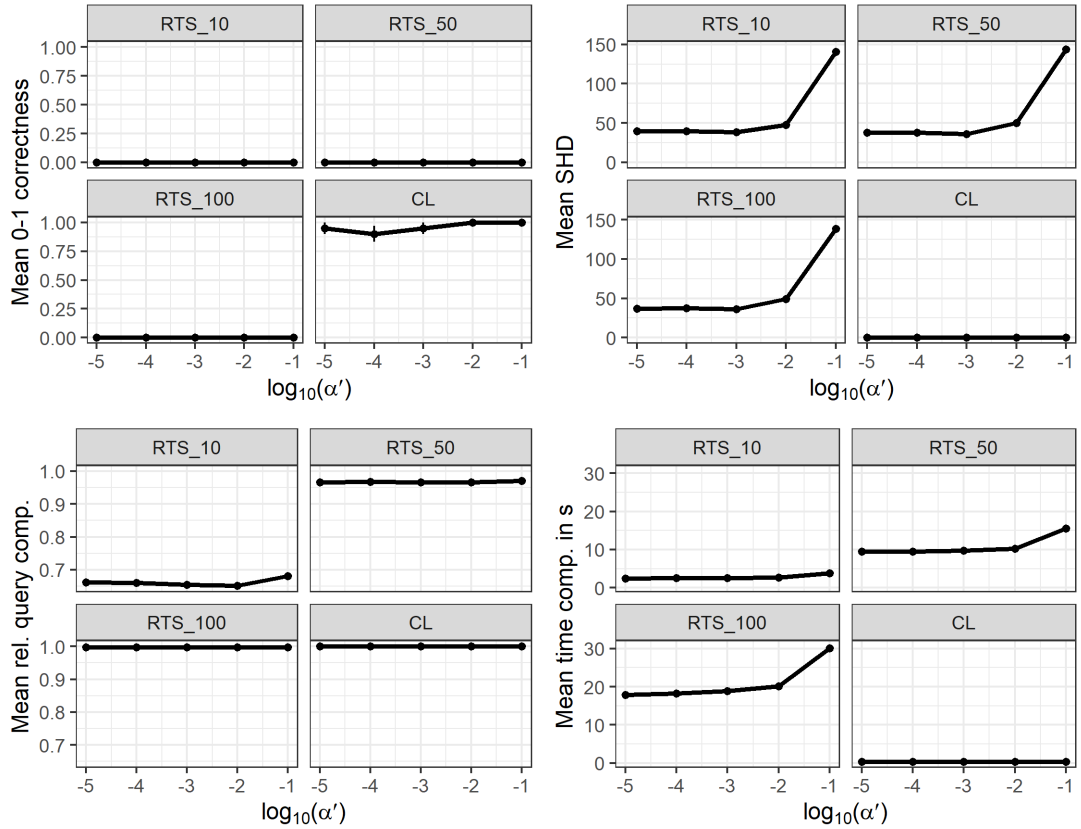
$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0 (0)	0 (0)	0 (0)	1 (0)
-4.00	0 (0)	0 (0)	0 (0)	1 (0)
-3.00	0 (0)	0 (0)	0 (0)	1 (0)
-2.00	0 (0)	0 (0)	0 (0)	0.95 (0.05)
-1.00	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
42 (2)	42 (2)	41 (2)	0 (0)
42 (1)	41 (2)	40 (2)	0 (0)
40 (2)	41 (2)	40 (2)	0 (0)
40 (1)	41 (2)	41 (1)	0.1 (0.1)
41 (2)	42 (2)	41 (2)	0 (0)

$\log_{10}(\alpha)$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.65 (0)	0.97 (0)	1 (0)	1 (0)
-4.00	0.65 (0)	0.97 (0)	1 (0)	1 (0)
-3.00	0.65 (0)	0.97 (0)	1 (0)	1 (0)
-2.00	0.66 (0)	0.97 (0)	1 (0)	1 (0)
-1.00	0.66 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.5 (0.03)	9.8 (0.2)	18.9 (0.4)	0.41 (0.01)
2.46 (0.05)	9.5 (0.1)	18.5 (0.4)	0.41 (0.01)
2.45 (0.03)	9.7 (0.2)	18.8 (0.3)	0.41 (0.01)
2.52 (0.04)	9.5 (0.2)	18.3 (0.3)	0.41 (0.01)
2.53 (0.04)	9.6 (0.2)	18.9 (0.4)	0.4 (0.01)

Figure 4.35.: Random trees with $|V| = 100, n = 2500$ and $\alpha' = 5 \cdot 10^{-4}$ using the Cholesky method.



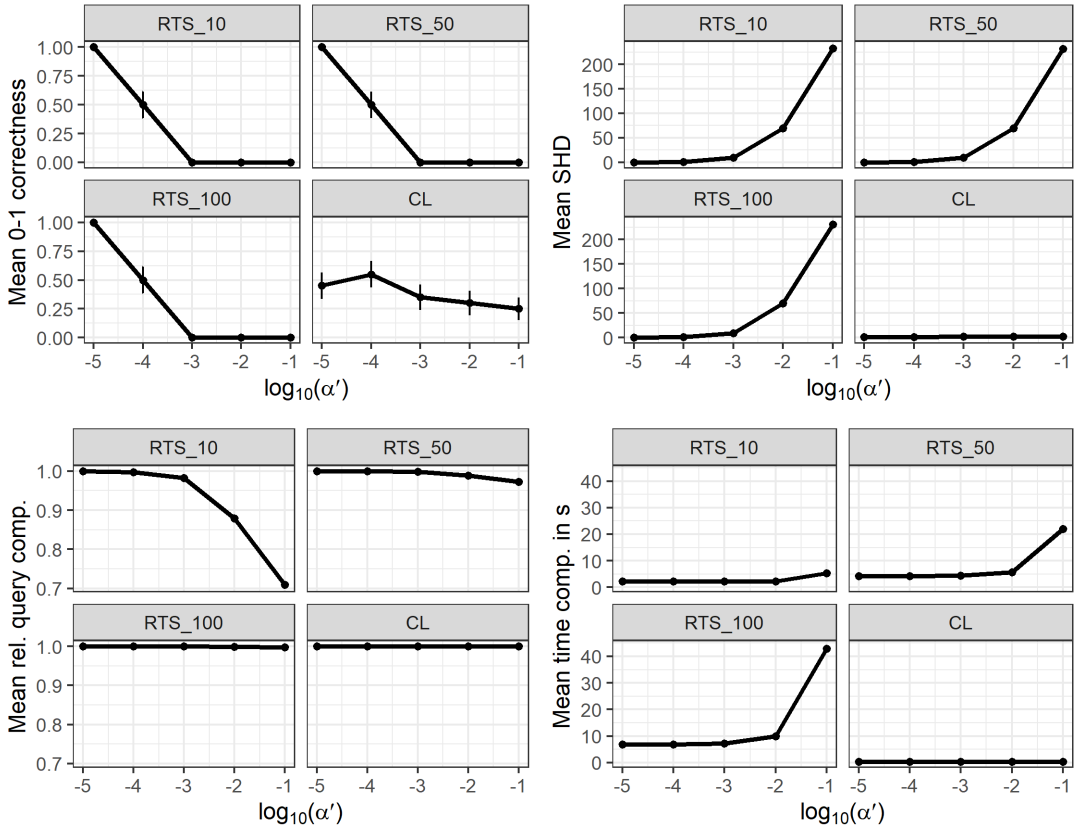
$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	0 (0)	0 (0)	0 (0)	0.95 (0.05)
-4.00	0 (0)	0 (0)	0 (0)	0.9 (0.07)
-3.00	0 (0)	0 (0)	0 (0)	0.95 (0.05)
-2.00	0 (0)	0 (0)	0 (0)	1 (0)
-1.00	0 (0)	0 (0)	0 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
40 (1)	38 (1)	36.9 (0.9)	0.1 (0.1)
39.4 (0.9)	38 (1)	37 (1)	0.2 (0.1)
38 (1)	35.9 (0.9)	36 (1)	0.1 (0.1)
48 (1)	50 (2)	49 (2)	0 (0)
140 (4)	144 (3)	138 (4)	0 (0)

$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.66 (0)	0.97 (0)	1 (0)	1 (0)
-4.00	0.66 (0)	0.97 (0)	1 (0)	1 (0)
-3.00	0.65 (0)	0.97 (0)	1 (0)	1 (0)
-2.00	0.65 (0)	0.97 (0)	1 (0)	1 (0)
-1.00	0.68 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.48 (0.03)	9.5 (0.1)	17.8 (0.2)	0.37 (0.01)
2.54 (0.03)	9.5 (0.1)	18.3 (0.2)	0.38 (0.01)
2.59 (0.04)	9.7 (0.1)	18.9 (0.3)	0.37 (0.01)
2.67 (0.03)	10.2 (0.1)	20.1 (0.2)	0.36 (0)
3.81 (0.08)	15.6 (0.2)	30.1 (0.5)	0.38 (0.01)

Figure 4.36.: Chain with $|V| = 100, n = 2500$ and $\alpha = 5 \cdot 10^{-4}$ using the Cholesky method.



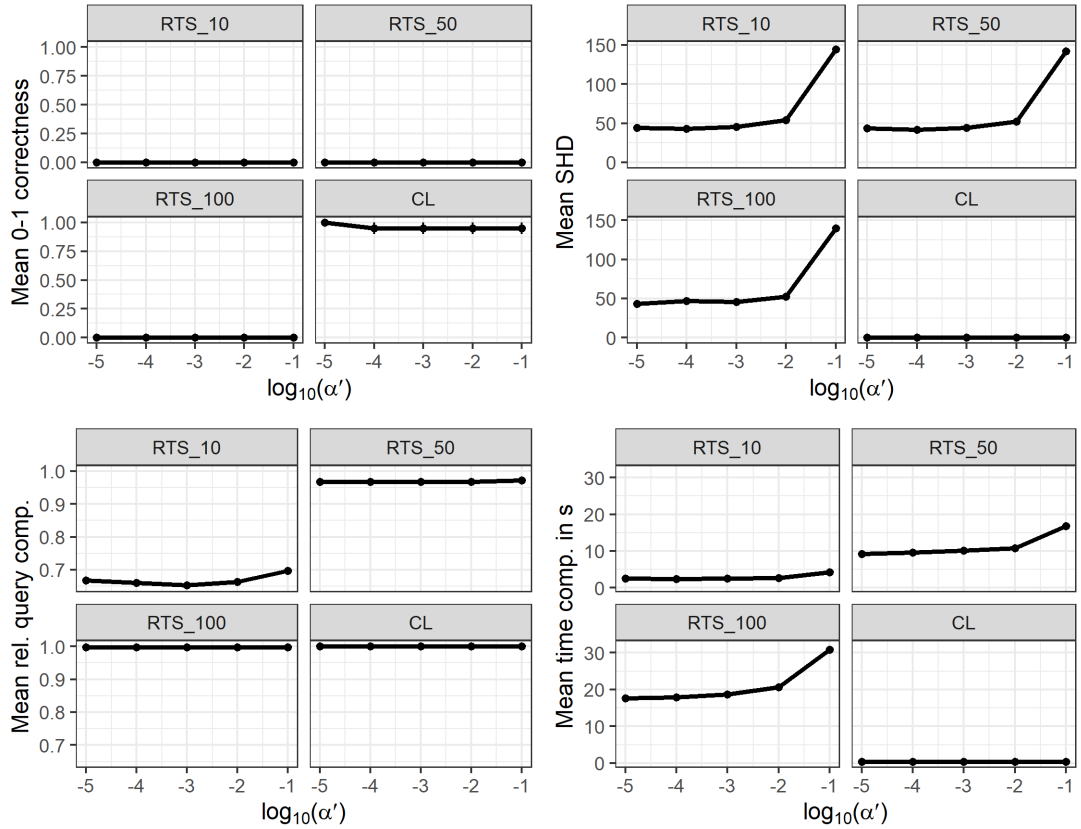
$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	1 (0)	1 (0)	1 (0)	0.4 (0.1)
-4.00	0.5 (0.1)	0.5 (0.1)	0.5 (0.1)	0.6 (0.1)
-3.00	0 (0)	0 (0)	0 (0)	0.3 (0.1)
-2.00	0 (0)	0 (0)	0 (0)	0.3 (0.1)
-1.00	0 (0)	0 (0)	0 (0)	0.2 (0.1)

RTS_10	RTS_50	RTS_100	CL
0 (0)	0 (0)	0 (0)	1.5 (0.4)
1.2 (0.3)	1.2 (0.3)	1.2 (0.3)	1.1 (0.3)
9.4 (0.9)	9.4 (0.9)	9.4 (0.9)	2.2 (0.5)
70 (2)	70 (2)	70 (2)	2.1 (0.4)
233 (2)	231 (3)	231 (3)	2.3 (0.4)

$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	1 (0)	1 (0)	1 (0)	1 (0)
-4.00	1 (0)	1 (0)	1 (0)	1 (0)
-3.00	0.98 (0)	1 (0)	1 (0)	1 (0)
-2.00	0.88 (0)	0.99 (0)	1 (0)	1 (0)
-1.00	0.71 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.21 (0.02)	4.2 (0.02)	6.77 (0.03)	0.37 (0.01)
2.18 (0.01)	4.26 (0.03)	6.85 (0.05)	0.36 (0.01)
2.18 (0.01)	4.4 (0.03)	7.12 (0.05)	0.35 (0.01)
2.22 (0.02)	5.62 (0.05)	9.9 (0.1)	0.35 (0.01)
5.26 (0.09)	22 (0.4)	42.9 (0.8)	0.35 (0.01)

Figure 4.37.: Star with $|V| = 100, n = 2500$ and $\alpha = 5 \cdot 10^{-4}$ using the Cholesky method.



$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	0 (0)	0 (0)	0 (0)	1 (0)
-4.00	0 (0)	0 (0)	0 (0)	0.95 (0.05)
-3.00	0 (0)	0 (0)	0 (0)	0.95 (0.05)
-2.00	0 (0)	0 (0)	0 (0)	0.95 (0.05)
-1.00	0 (0)	0 (0)	0 (0)	0.95 (0.05)

RTS_10	RTS_50	RTS_100	CL
44 (1)	43 (1)	43 (2)	0 (0)
43 (1)	42 (1)	47 (2)	0.1 (0.1)
46 (1)	44 (1)	46 (2)	0.1 (0.1)
54 (2)	52 (2)	52 (2)	0.1 (0.1)
144 (3)	142 (2)	140 (3)	0.1 (0.1)

$\log_{10}(\alpha')$	RTS_10	RTS_50	RTS_100	CL
-5.00	0.67 (0)	0.97 (0)	1 (0)	1 (0)
-4.00	0.66 (0)	0.97 (0)	1 (0)	1 (0)
-3.00	0.65 (0)	0.97 (0)	1 (0)	1 (0)
-2.00	0.66 (0)	0.97 (0)	1 (0)	1 (0)
-1.00	0.7 (0)	0.97 (0)	1 (0)	1 (0)

RTS_10	RTS_50	RTS_100	CL
2.53 (0.04)	9.3 (0.1)	17.7 (0.3)	0.41 (0.01)
2.47 (0.04)	9.6 (0.2)	17.9 (0.3)	0.4 (0.01)
2.52 (0.05)	10.1 (0.2)	18.7 (0.3)	0.42 (0.01)
2.71 (0.08)	10.8 (0.3)	20.6 (0.5)	0.42 (0.01)
4.2 (0.1)	16.8 (0.4)	30.8 (0.9)	0.41 (0.01)

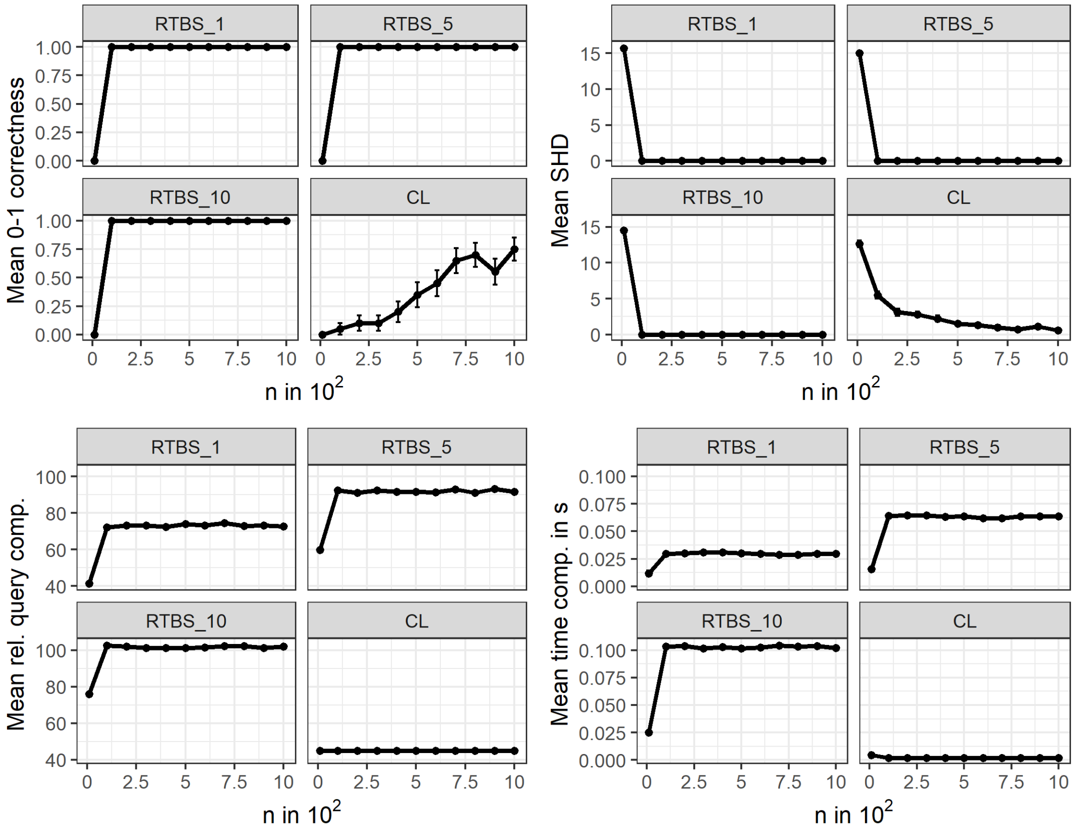
Figure 4.38.: Random trees with $|V| = 100, n = 2500$ and $\alpha = 5 \cdot 10^{-4}$ using the Cholesky method.

We do not see much difference between $n = 500$ and $n = 2500$. We would have expected that the optimal α' for the correctness decreases, but that is hard to see in the plots. Maybe, better choices of n show this effect or a finer scale for α resp. α' is required.

We also do not see much dependence on α. However, we observe that the correctness worsens and the execution time increases for larger values of α', irrespective of the underlying tree. The effect of α' on the execution time was not predicted by the theory; we only expected α to have an execution on the query complexity, which we have not observed for our particular observation.
Interestingly, the query complexity for the star decreases for decreasing α'; for the other tree structures the query complexity seems not to react to changes in α or α'.

4.5.2. Results for binary random variables

We also do some simulations for binary random variables, even though not as detailed as for Gaussians. We start similarly as for Gaussians by comparing `ReconstructTreeBinarySample` with the Chow-Liu algorithm for different values of α, α' and n. Again, we present the mean with one standard error. This time, we work with the absolute number of queries and not a relative version, because querying three-way tables and querying two-way tables have a different denominator. In contrast to Gaussians, we do the simulations with $|V| = 10$, instead of $|V| = 100$, as larger $|V|$ are computationally much more expensive for binary random vectors; we comment on that later as well.
We also remark that very small execution times for the Chow-Liu algorithm are displayed as 0 in the corresponding tables.



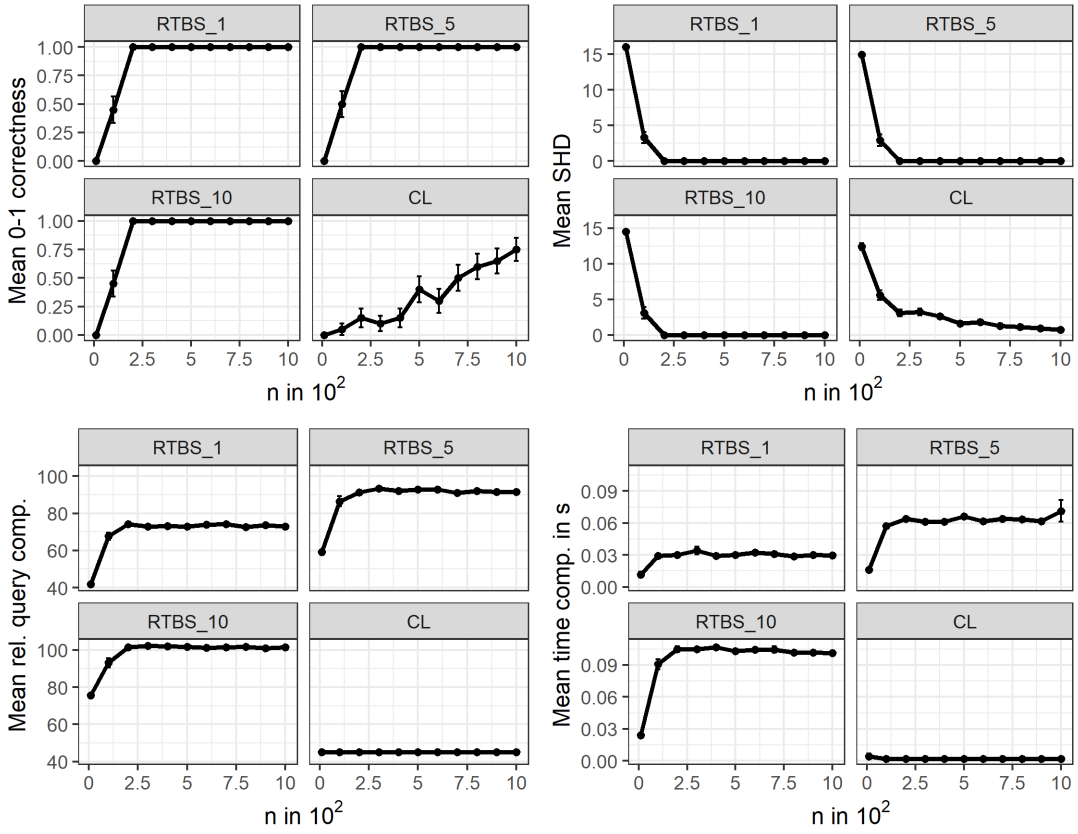
n	RTBS_1	RTBS_5	RTBS_10	CL
10	0 (0)	0 (0)	0 (0)	0 (0)
100	1 (0)	1 (0)	1 (0)	0.05 (0.05)
200	1 (0)	1 (0)	1 (0)	0.1 (0.07)
300	1 (0)	1 (0)	1 (0)	0.1 (0.07)
400	1 (0)	1 (0)	1 (0)	0.2 (0.09)
500	1 (0)	1 (0)	1 (0)	0.3 (0.1)
600	1 (0)	1 (0)	1 (0)	0.4 (0.1)
700	1 (0)	1 (0)	1 (0)	0.7 (0.1)
800	1 (0)	1 (0)	1 (0)	0.7 (0.1)
900	1 (0)	1 (0)	1 (0)	0.6 (0.1)
1000	1 (0)	1 (0)	1 (0)	0.8 (0.1)

RTBS_1	RTBS_5	RTBS_10	CL
15.7 (0.2)	15 (0.3)	14.5 (0.3)	12.7 (0.5)
0 (0)	0 (0)	0 (0)	5.5 (0.5)
0 (0)	0 (0)	0 (0)	3.1 (0.5)
0 (0)	0 (0)	0 (0)	2.8 (0.4)
0 (0)	0 (0)	0 (0)	2.2 (0.4)
0 (0)	0 (0)	0 (0)	1.5 (0.3)
0 (0)	0 (0)	0 (0)	1.3 (0.3)
0 (0)	0 (0)	0 (0)	1 (0.3)
0 (0)	0 (0)	0 (0)	0.7 (0.3)
0 (0)	0 (0)	0 (0)	1.1 (0.3)
0 (0)	0 (0)	0 (0)	0.6 (0.3)

n	RTBS_1	RTBS_5	RTBS_10	CL
10	41.4 (0.4)	59.8 (0.6)	76 (0.8)	45 (0)
100	72.1 (0.8)	92.3 (0.6)	102.5 (0.5)	45 (0)
200	73.2 (0.5)	91.1 (0.8)	101.9 (0.6)	45 (0)
300	73.2 (0.6)	92.3 (0.7)	101.2 (0.6)	45 (0)
400	72.4 (0.7)	91.8 (0.6)	101.3 (0.4)	45 (0)
500	74 (0.9)	91.5 (0.5)	101.2 (0.5)	45 (0)
600	73.2 (0.7)	91.4 (0.7)	101.4 (0.5)	45 (0)
700	74.6 (0.7)	92.8 (0.6)	102.4 (0.9)	45 (0)
800	73 (0.8)	91.2 (0.7)	102.2 (0.7)	45 (0)
900	73.3 (0.7)	93.2 (0.6)	101.3 (0.6)	45 (0)
1000	72.8 (0.8)	91.5 (0.7)	101.9 (0.6)	45 (0)

RTBS_1	RTBS_5	RTBS_10	CL
0.01 (0)	0.02 (0)	0.02 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)

Figure 4.39.: Chain with $|V| = 10$ and $\alpha = \alpha' = 0.05$ for binary random vectors.



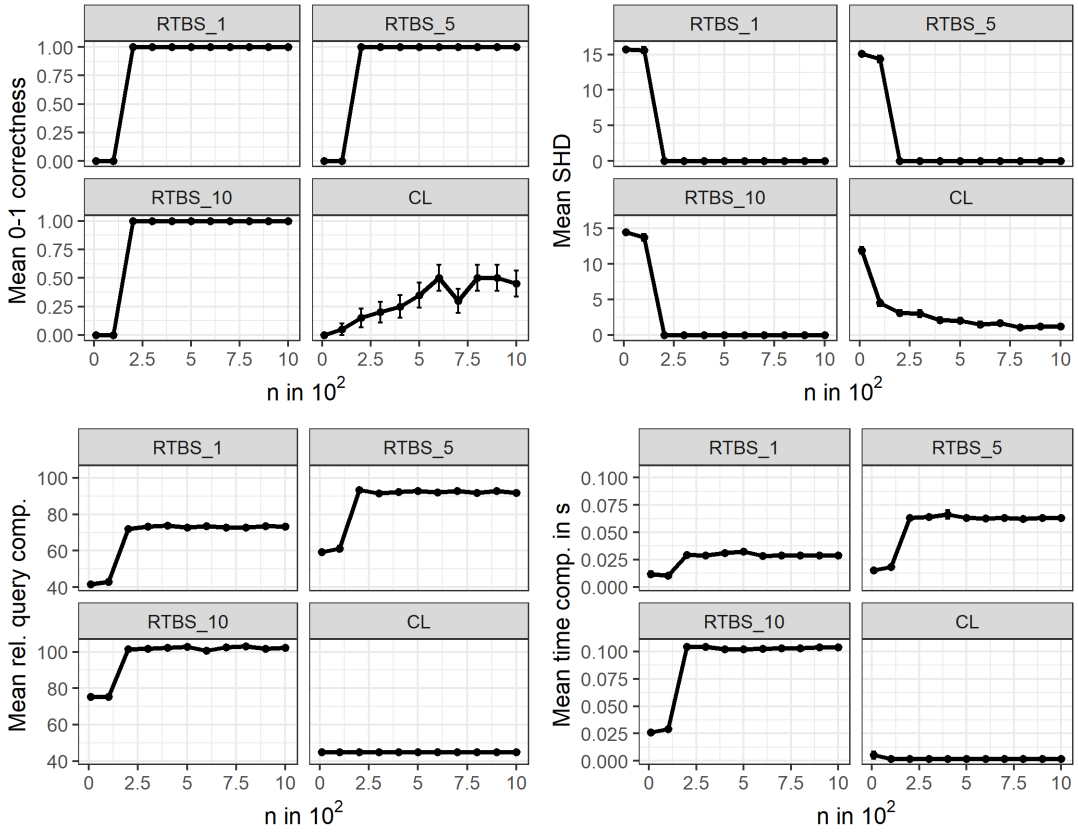
n	RTBS_1	RTBS_5	RTBS_10	CL
10	0 (0)	0 (0)	0 (0)	0 (0)
100	0.4 (0.1)	0.5 (0.1)	0.4 (0.1)	0.05 (0.05)
200	1 (0)	1 (0)	1 (0)	0.15 (0.08)
300	1 (0)	1 (0)	1 (0)	0.1 (0.07)
400	1 (0)	1 (0)	1 (0)	0.15 (0.08)
500	1 (0)	1 (0)	1 (0)	0.4 (0.1)
600	1 (0)	1 (0)	1 (0)	0.3 (0.1)
700	1 (0)	1 (0)	1 (0)	0.5 (0.1)
800	1 (0)	1 (0)	1 (0)	0.6 (0.1)
900	1 (0)	1 (0)	1 (0)	0.7 (0.1)
1000	1 (0)	1 (0)	1 (0)	0.8 (0.1)

RTBS_1	RTBS_5	RTBS_10	CL
16 (0)	14.9 (0.2)	14.5 (0.2)	12.4 (0.5)
3.3 (0.8)	2.9 (0.8)	3.1 (0.8)	5.6 (0.7)
0 (0)	0 (0)	0 (0)	3.1 (0.5)
0 (0)	0 (0)	0 (0)	3.2 (0.5)
0 (0)	0 (0)	0 (0)	2.6 (0.4)
0 (0)	0 (0)	0 (0)	1.6 (0.4)
0 (0)	0 (0)	0 (0)	1.8 (0.4)
0 (0)	0 (0)	0 (0)	1.3 (0.3)
0 (0)	0 (0)	0 (0)	1.1 (0.3)
0 (0)	0 (0)	0 (0)	0.9 (0.3)
0 (0)	0 (0)	0 (0)	0.7 (0.3)

n	RTBS_1	RTBS_5	RTBS_10	CL
10	41.9 (0.4)	59.3 (0.6)	75.7 (0.8)	45 (0)
100	68 (2)	86 (3)	93 (2)	45 (0)
200	74.3 (0.7)	91.3 (0.5)	101.6 (0.7)	45 (0)
300	73 (0.5)	93.2 (0.6)	102.2 (0.5)	45 (0)
400	73.2 (0.8)	92.1 (0.7)	102 (0.6)	45 (0)
500	73 (0.5)	92.7 (0.7)	101.8 (0.6)	45 (0)
600	73.8 (0.5)	92.8 (0.7)	101.2 (0.7)	45 (0)
700	74.2 (0.5)	90.9 (0.9)	101.5 (0.6)	45 (0)
800	72.7 (0.7)	92 (0.6)	101.8 (0.6)	45 (0)
900	73.8 (0.7)	91.6 (0.7)	101 (0.6)	45 (0)
1000	72.8 (0.6)	91.6 (0.9)	101.5 (0.7)	45 (0)

RTBS_1	RTBS_5	RTBS_10	CL
0.01 (0)	0.02 (0)	0.02 (0)	0 (0)
0.03 (0)	0.06 (0)	0.09 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.11 (0)	0 (0)
0.03 (0)	0.07 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.07 (0.01)	0.1 (0)	0 (0)

Figure 4.40.: Chain with $|V| = 10$ and $\alpha = \alpha' = 5 \cdot 10^{-4}$ for binary random vectors.



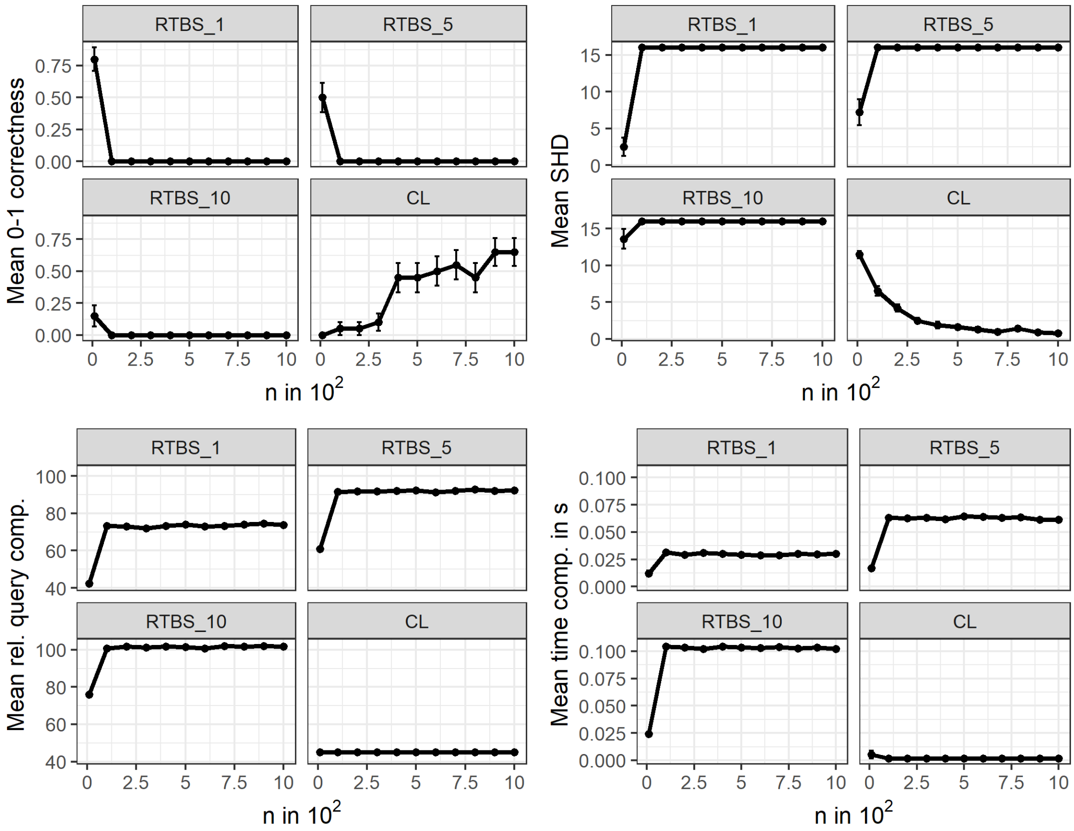
n	RTBS_1	RTBS_5	RTBS_10	CL
10	0 (0)	0 (0)	0 (0)	0 (0)
100	0 (0)	0 (0)	0 (0)	0.05 (0.05)
200	1 (0)	1 (0)	1 (0)	0.15 (0.08)
300	1 (0)	1 (0)	1 (0)	0.2 (0.09)
400	1 (0)	1 (0)	1 (0)	0.2 (0.1)
500	1 (0)	1 (0)	1 (0)	0.3 (0.1)
600	1 (0)	1 (0)	1 (0)	0.5 (0.1)
700	1 (0)	1 (0)	1 (0)	0.3 (0.1)
800	1 (0)	1 (0)	1 (0)	0.5 (0.1)
900	1 (0)	1 (0)	1 (0)	0.5 (0.1)
1000	1 (0)	1 (0)	1 (0)	0.4 (0.1)

RTBS_1	RTBS_5	RTBS_10	CL
15.7 (0.2)	15.1 (0.2)	14.5 (0.2)	11.9 (0.5)
15.6 (0.4)	14.3 (0.5)	13.7 (0.5)	4.5 (0.5)
0 (0)	0 (0)	0 (0)	3.1 (0.4)
0 (0)	0 (0)	0 (0)	3 (0.5)
0 (0)	0 (0)	0 (0)	2.1 (0.4)
0 (0)	0 (0)	0 (0)	2 (0.4)
0 (0)	0 (0)	0 (0)	1.5 (0.4)
0 (0)	0 (0)	0 (0)	1.7 (0.3)
0 (0)	0 (0)	0 (0)	1.1 (0.3)
0 (0)	0 (0)	0 (0)	1.2 (0.3)
0 (0)	0 (0)	0 (0)	1.2 (0.3)

n	RTBS_1	RTBS_5	RTBS_10	CL
10	41.5 (0.4)	59.2 (0.4)	75.3 (0.6)	45 (0)
100	43 (1)	61 (2)	75.4 (0.7)	45 (0)
200	72.1 (0.5)	93.4 (0.7)	101.5 (0.9)	45 (0)
300	73.4 (0.6)	91.6 (0.8)	101.7 (0.5)	45 (0)
400	73.8 (0.9)	92.2 (0.8)	102.3 (0.7)	45 (0)
500	72.9 (0.5)	92.8 (0.8)	102.8 (0.7)	45 (0)
600	73.7 (0.6)	92 (0.8)	100.6 (0.6)	45 (0)
700	72.7 (0.5)	92.8 (0.7)	102.6 (0.6)	45 (0)
800	72.8 (0.5)	91.8 (0.6)	103.2 (0.6)	45 (0)
900	73.5 (0.7)	92.8 (0.8)	101.8 (0.7)	45 (0)
1000	73.2 (0.7)	91.7 (0.7)	102.2 (0.6)	45 (0)

RTBS_1	RTBS_5	RTBS_10	CL
0.01 (0)	0.02 (0)	0.03 (0)	0.01 (0)
0.01 (0)	0.02 (0)	0.03 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.07 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)

Figure 4.41.: Chain with $|V| = 10$ and $\alpha = \alpha' = 5 \cdot 10^{-6}$ for binary random vectors.



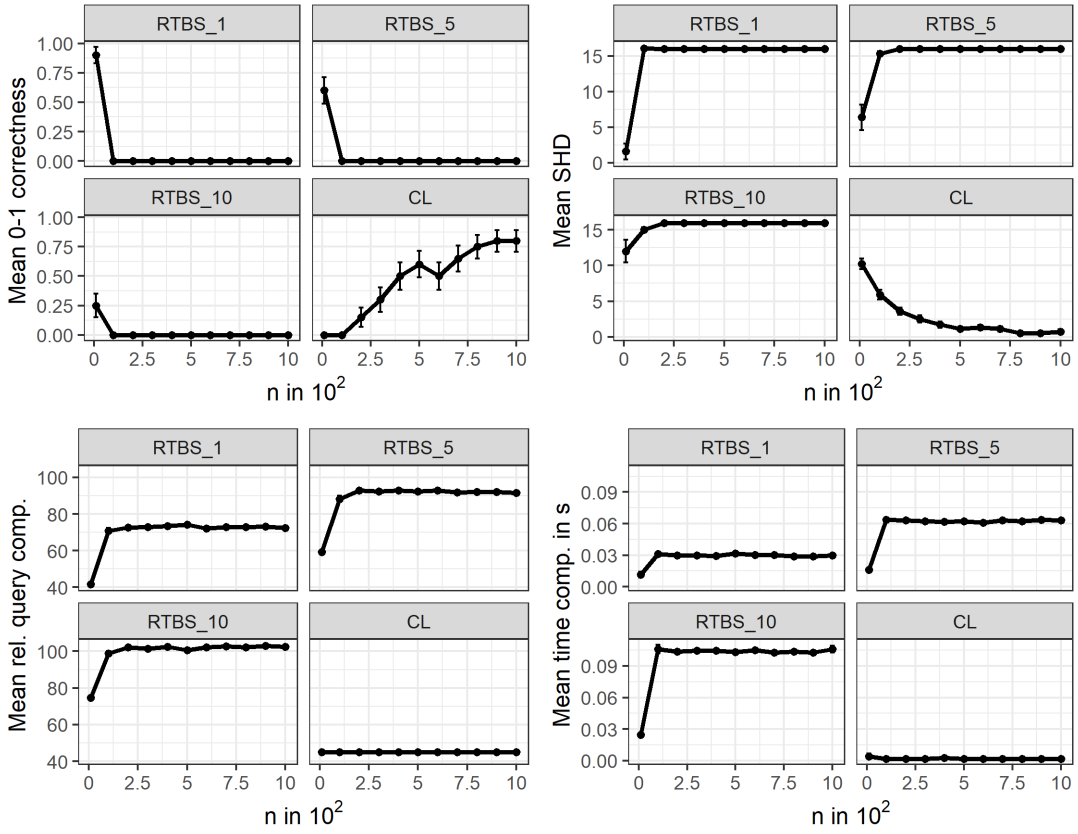
n	RTBS_1	RTBS_5	RTBS_10	CL
10	0.8 (0.09)	0.5 (0.1)	0.15 (0.08)	0 (0)
100	0 (0)	0 (0)	0 (0)	0.05 (0.05)
200	0 (0)	0 (0)	0 (0)	0.05 (0.05)
300	0 (0)	0 (0)	0 (0)	0.1 (0.07)
400	0 (0)	0 (0)	0 (0)	0.4 (0.1)
500	0 (0)	0 (0)	0 (0)	0.4 (0.1)
600	0 (0)	0 (0)	0 (0)	0.5 (0.1)
700	0 (0)	0 (0)	0 (0)	0.6 (0.1)
800	0 (0)	0 (0)	0 (0)	0.4 (0.1)
900	0 (0)	0 (0)	0 (0)	0.7 (0.1)
1000	0 (0)	0 (0)	0 (0)	0.7 (0.1)

RTBS_1	RTBS_5	RTBS_10	CL
2 (1)	7 (2)	14 (1)	11.4 (0.5)
16 (0)	16 (0)	16 (0)	6.5 (0.7)
16 (0)	16 (0)	16 (0)	4.2 (0.5)
16 (0)	16 (0)	16 (0)	2.5 (0.4)
16 (0)	16 (0)	16 (0)	1.9 (0.4)
16 (0)	16 (0)	16 (0)	1.6 (0.4)
16 (0)	16 (0)	16 (0)	1.3 (0.3)
16 (0)	16 (0)	16 (0)	1 (0.3)
16 (0)	16 (0)	16 (0)	1.4 (0.3)
16 (0)	16 (0)	16 (0)	0.9 (0.3)
16 (0)	16 (0)	16 (0)	0.8 (0.3)

n	RTBS_1	RTBS_5	RTBS_10	CL
10	42.3 (0.7)	60.9 (0.9)	76 (0.7)	45 (0)
100	73.2 (0.6)	91.5 (0.6)	100.8 (0.6)	45 (0)
200	73 (0.7)	91.8 (0.7)	101.8 (0.6)	45 (0)
300	72 (0.7)	91.9 (0.6)	101.3 (0.6)	45 (0)
400	73.3 (0.6)	92 (0.7)	101.8 (0.5)	45 (0)
500	74 (0.6)	92.4 (0.7)	101.4 (0.6)	45 (0)
600	72.9 (0.4)	91.4 (0.7)	100.8 (0.7)	45 (0)
700	73.3 (0.7)	92.1 (0.6)	101.9 (0.5)	45 (0)
800	74.1 (0.7)	92.8 (0.8)	101.7 (0.7)	45 (0)
900	74 (1)	92.2 (0.6)	102 (0.8)	45 (0)
1000	73.8 (0.6)	92.3 (0.6)	101.8 (0.7)	45 (0)

RTBS_1	RTBS_5	RTBS_10	CL
0.01 (0)	0.02 (0)	0.02 (0)	0.01 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)

Figure 4.42.: Star with $|V| = 10$ and $\alpha = \alpha' = 0.05$ for binary random vectors.



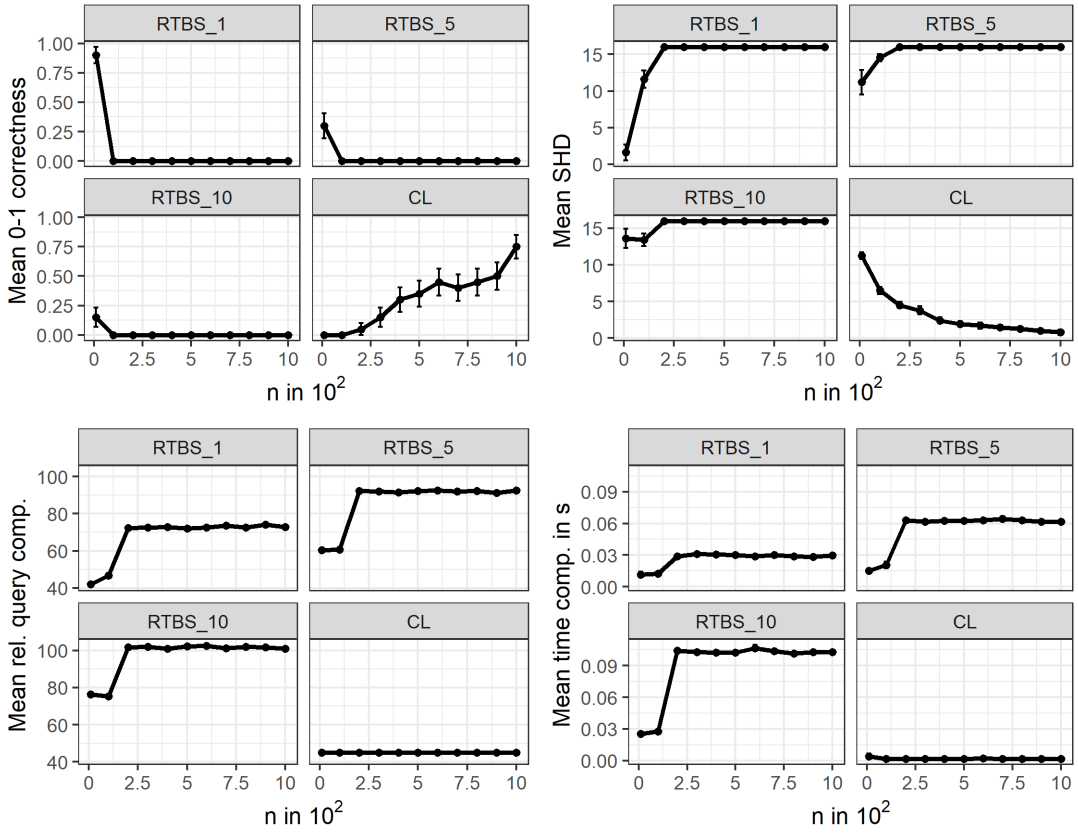
n	RTBS_1	RTBS_5	RTBS_10	CL
10	0.9 (0.07)	0.6 (0.1)	0.2 (0.1)	0 (0)
100	0 (0)	0 (0)	0 (0)	0 (0)
200	0 (0)	0 (0)	0 (0)	0.15 (0.08)
300	0 (0)	0 (0)	0 (0)	0.3 (0.1)
400	0 (0)	0 (0)	0 (0)	0.5 (0.1)
500	0 (0)	0 (0)	0 (0)	0.6 (0.1)
600	0 (0)	0 (0)	0 (0)	0.5 (0.1)
700	0 (0)	0 (0)	0 (0)	0.7 (0.1)
800	0 (0)	0 (0)	0 (0)	0.8 (0.1)
900	0 (0)	0 (0)	0 (0)	0.8 (0.09)
1000	0 (0)	0 (0)	0 (0)	0.8 (0.09)

RTBS_1	RTBS_5	RTBS_10	CL
2 (1)	6 (2)	12 (2)	10.2 (0.7)
16 (0.2)	15.3 (0.3)	15 (0.4)	5.9 (0.7)
16 (0)	16 (0)	16 (0)	3.6 (0.5)
16 (0)	16 (0)	16 (0)	2.5 (0.5)
16 (0)	16 (0)	16 (0)	1.7 (0.4)
16 (0)	16 (0)	16 (0)	1.1 (0.3)
16 (0)	16 (0)	16 (0)	1.3 (0.3)
16 (0)	16 (0)	16 (0)	1.1 (0.4)
16 (0)	16 (0)	16 (0)	0.5 (0.2)
16 (0)	16 (0)	16 (0)	0.5 (0.2)
16 (0)	16 (0)	16 (0)	0.7 (0.3)

n	RTBS_1	RTBS_5	RTBS_10	CL
10	41.6 (0.3)	59.2 (0.6)	74.8 (0.8)	45 (0)
100	71 (2)	88 (2)	99 (1)	45 (0)
200	72.7 (0.7)	93 (0.6)	102.2 (0.4)	45 (0)
300	73 (0.7)	92.3 (0.5)	101.3 (0.6)	45 (0)
400	73.6 (0.5)	92.9 (0.7)	102.4 (0.5)	45 (0)
500	74.3 (0.5)	92.3 (0.8)	100.7 (0.8)	45 (0)
600	72 (0.5)	93 (0.8)	102.2 (0.6)	45 (0)
700	73 (0.8)	91.8 (0.8)	102.7 (0.5)	45 (0)
800	72.9 (0.6)	92 (0.9)	102.3 (0.5)	45 (0)
900	73.2 (0.6)	92.1 (0.5)	103 (0.4)	45 (0)
1000	72.3 (0.7)	91.6 (0.6)	102.4 (0.6)	45 (0)

RTBS_1	RTBS_5	RTBS_10	CL
0.01 (0)	0.02 (0)	0.02 (0)	0 (0)
0.03 (0)	0.06 (0)	0.11 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.11 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.11 (0)	0 (0)

Figure 4.43.: Star with $|V| = 10$ and $\alpha = \alpha' = 5 \cdot 10^{-4}$ for binary random vectors.



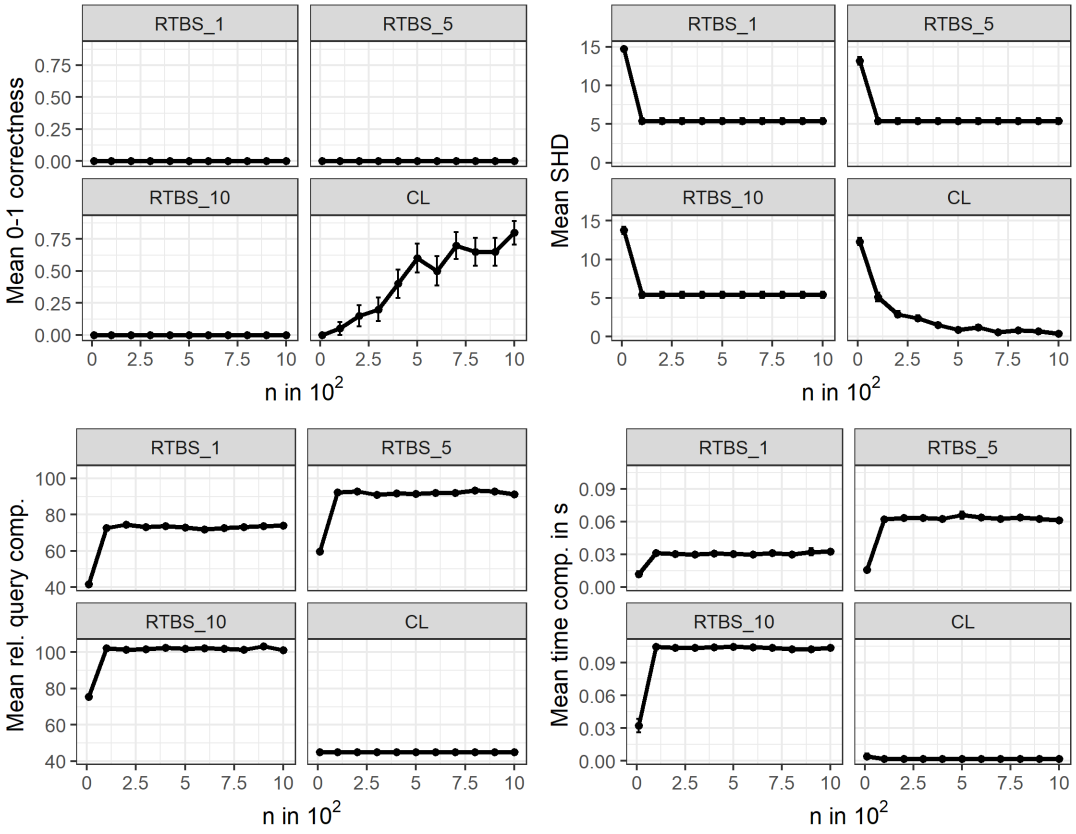
n	RTBS_1	RTBS_5	RTBS_10	CL
10	0.9 (0.07)	0.3 (0.1)	0.15 (0.08)	0 (0)
100	0 (0)	0 (0)	0 (0)	0 (0)
200	0 (0)	0 (0)	0 (0)	0.05 (0.05)
300	0 (0)	0 (0)	0 (0)	0.15 (0.08)
400	0 (0)	0 (0)	0 (0)	0.3 (0.1)
500	0 (0)	0 (0)	0 (0)	0.3 (0.1)
600	0 (0)	0 (0)	0 (0)	0.4 (0.1)
700	0 (0)	0 (0)	0 (0)	0.4 (0.1)
800	0 (0)	0 (0)	0 (0)	0.4 (0.1)
900	0 (0)	0 (0)	0 (0)	0.5 (0.1)
1000	0 (0)	0 (0)	0 (0)	0.8 (0.1)

RTBS_1	RTBS_5	RTBS_10	CL
2 (1)	11 (2)	14 (1)	11.2 (0.4)
12 (1)	14.6 (0.5)	13.4 (0.9)	6.5 (0.5)
16 (0)	16 (0)	16 (0)	4.5 (0.4)
16 (0)	16 (0)	16 (0)	3.8 (0.5)
16 (0)	16 (0)	16 (0)	2.4 (0.4)
16 (0)	16 (0)	16 (0)	1.9 (0.4)
16 (0)	16 (0)	16 (0)	1.7 (0.4)
16 (0)	16 (0)	16 (0)	1.5 (0.3)
16 (0)	16 (0)	16 (0)	1.3 (0.3)
16 (0)	16 (0)	16 (0)	1 (0.2)
16 (0)	16 (0)	16 (0)	0.8 (0.4)

n	RTBS_1	RTBS_5	RTBS_10	CL
10	41.9 (0.3)	60.4 (0.5)	76 (1)	45 (0)
100	47 (1)	61 (1)	75.2 (0.7)	45 (0)
200	72.2 (0.7)	92.2 (0.7)	101.7 (0.5)	45 (0)
300	72.6 (0.7)	91.9 (0.7)	101.9 (0.7)	45 (0)
400	72.8 (0.6)	91.5 (0.7)	101 (0.6)	45 (0)
500	72 (0.5)	92.2 (0.8)	102.2 (0.6)	45 (0)
600	72.7 (0.7)	92.6 (0.5)	102.4 (0.5)	45 (0)
700	73.6 (0.7)	92 (0.9)	101.2 (0.5)	45 (0)
800	72.7 (0.7)	92 (1)	102 (0.5)	45 (0)
900	74 (0.7)	91.2 (0.7)	101.7 (0.6)	45 (0)
1000	72.8 (0.5)	92.6 (0.7)	101 (0.5)	45 (0)

RTBS_1	RTBS_5	RTBS_10	CL
0.01 (0)	0.02 (0)	0.03 (0)	0 (0)
0.01 (0)	0.02 (0)	0.03 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.11 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)

Figure 4.44.: Star with $|V| = 10$ and $\alpha = \alpha' = 5 \cdot 10^{-6}$ for binary random vectors.



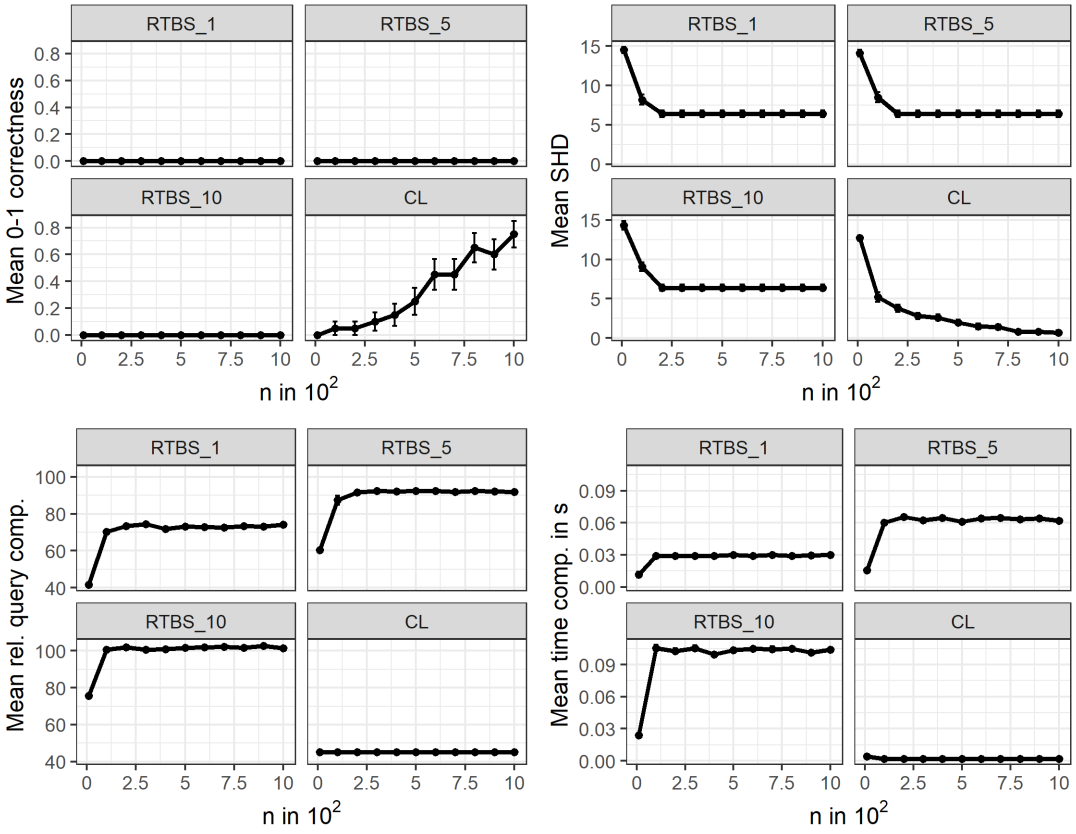
n	RTBS_1	RTBS_5	RTBS_10	CL
10	0 (0)	0 (0)	0 (0)	0 (0)
100	0 (0)	0 (0)	0 (0)	0.05 (0.05)
200	0 (0)	0 (0)	0 (0)	0.15 (0.08)
300	0 (0)	0 (0)	0 (0)	0.2 (0.09)
400	0 (0)	0 (0)	0 (0)	0.4 (0.1)
500	0 (0)	0 (0)	0 (0)	0.6 (0.1)
600	0 (0)	0 (0)	0 (0)	0.5 (0.1)
700	0 (0)	0 (0)	0 (0)	0.7 (0.1)
800	0 (0)	0 (0)	0 (0)	0.7 (0.1)
900	0 (0)	0 (0)	0 (0)	0.7 (0.1)
1000	0 (0)	0 (0)	0 (0)	0.8 (0.09)

RTBS_1	RTBS_5	RTBS_10	CL
14.7 (0.3)	13.2 (0.5)	13.8 (0.5)	12.3 (0.5)
5.4 (0.4)	5.4 (0.4)	5.4 (0.4)	5.1 (0.6)
5.4 (0.4)	5.4 (0.4)	5.4 (0.4)	2.9 (0.4)
5.4 (0.4)	5.4 (0.4)	5.4 (0.4)	2.4 (0.4)
5.4 (0.4)	5.4 (0.4)	5.4 (0.4)	1.5 (0.3)
5.4 (0.4)	5.4 (0.4)	5.4 (0.4)	0.9 (0.3)
5.4 (0.4)	5.4 (0.4)	5.4 (0.4)	1.2 (0.3)
5.4 (0.4)	5.4 (0.4)	5.4 (0.4)	0.6 (0.2)
5.4 (0.4)	5.4 (0.4)	5.4 (0.4)	0.8 (0.3)
5.4 (0.4)	5.4 (0.4)	5.4 (0.4)	0.7 (0.2)
5.4 (0.4)	5.4 (0.4)	5.4 (0.4)	0.4 (0.2)

n	RTBS_1	RTBS_5	RTBS_10	CL
10	41.5 (0.3)	59.6 (0.8)	75.4 (0.8)	45 (0)
100	72.8 (0.8)	92.2 (0.9)	102.2 (0.7)	45 (0)
200	74.4 (0.5)	92.8 (0.8)	101.5 (0.5)	45 (0)
300	73.3 (0.8)	91 (0.7)	101.6 (0.5)	45 (0)
400	73.7 (0.8)	91.8 (0.6)	102.6 (0.5)	45 (0)
500	72.8 (0.6)	91.6 (0.7)	101.9 (0.8)	45 (0)
600	71.9 (0.7)	92 (0.6)	102.2 (0.7)	45 (0)
700	72.8 (0.6)	92 (0.7)	102 (0.5)	45 (0)
800	73.2 (0.6)	93.3 (0.6)	101.3 (0.7)	45 (0)
900	73.8 (0.8)	92.8 (0.7)	103.3 (0.7)	45 (0)
1000	74 (0.6)	91.3 (0.5)	101.1 (0.6)	45 (0)

RTBS_1	RTBS_5	RTBS_10	CL
0.01 (0)	0.02 (0)	0.03 (0.01)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.07 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)

Figure 4.45.: Random trees with $|V| = 10$ and $\alpha = \alpha' = 0.05$ for binary random vectors.



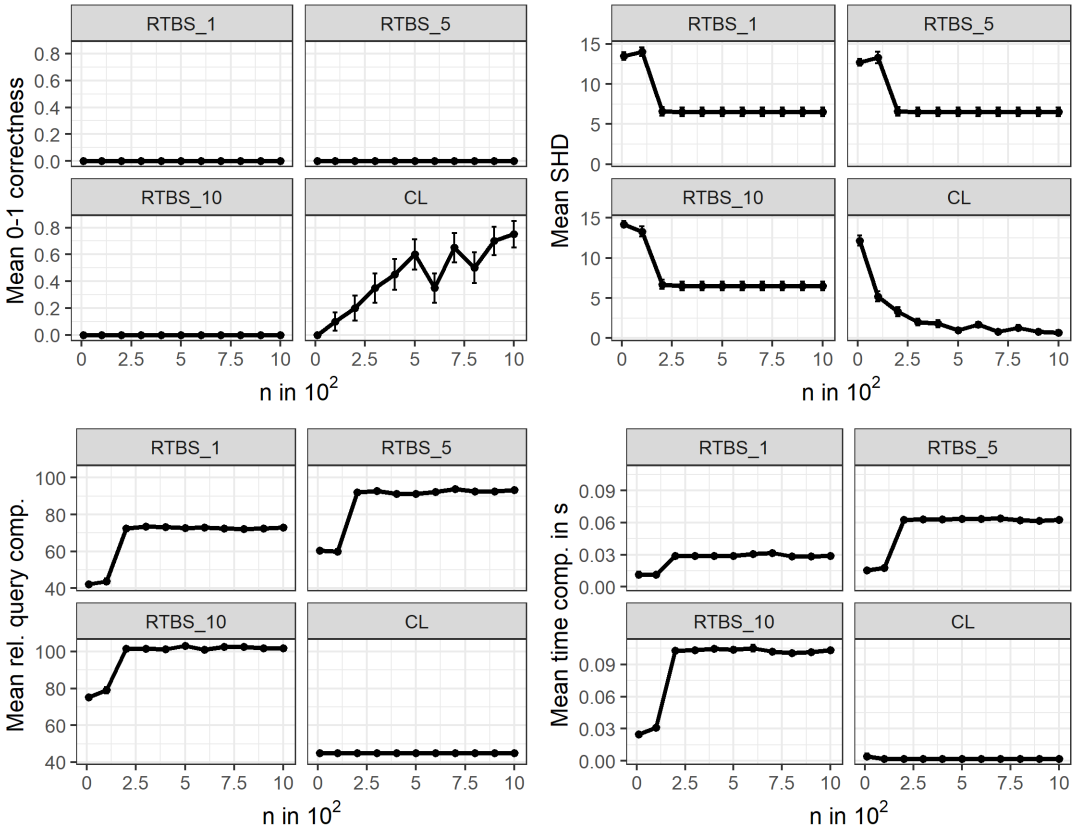
n	RTBS_1	RTBS_5	RTBS_10	CL
10	0 (0)	0 (0)	0 (0)	0 (0)
100	0 (0)	0 (0)	0 (0)	0.05 (0.05)
200	0 (0)	0 (0)	0 (0)	0.05 (0.05)
300	0 (0)	0 (0)	0 (0)	0.1 (0.07)
400	0 (0)	0 (0)	0 (0)	0.15 (0.08)
500	0 (0)	0 (0)	0 (0)	0.2 (0.1)
600	0 (0)	0 (0)	0 (0)	0.4 (0.1)
700	0 (0)	0 (0)	0 (0)	0.4 (0.1)
800	0 (0)	0 (0)	0 (0)	0.7 (0.1)
900	0 (0)	0 (0)	0 (0)	0.6 (0.1)
1000	0 (0)	0 (0)	0 (0)	0.8 (0.1)

RTBS_1	RTBS_5	RTBS_10	CL
14.5 (0.4)	14.1 (0.4)	14.3 (0.5)	12.7 (0.3)
8.2 (0.6)	8.5 (0.6)	9.1 (0.6)	5.2 (0.6)
6.4 (0.4)	6.4 (0.4)	6.4 (0.4)	3.8 (0.5)
6.4 (0.4)	6.4 (0.4)	6.4 (0.4)	2.8 (0.4)
6.4 (0.4)	6.4 (0.4)	6.4 (0.4)	2.6 (0.4)
6.4 (0.4)	6.4 (0.4)	6.4 (0.4)	2 (0.3)
6.4 (0.4)	6.4 (0.4)	6.4 (0.4)	1.5 (0.4)
6.4 (0.4)	6.4 (0.4)	6.4 (0.4)	1.4 (0.3)
6.4 (0.4)	6.4 (0.4)	6.4 (0.4)	0.8 (0.3)
6.4 (0.4)	6.4 (0.4)	6.4 (0.4)	0.8 (0.2)
6.4 (0.4)	6.4 (0.4)	6.4 (0.4)	0.7 (0.3)

n	RTBS_1	RTBS_5	RTBS_10	CL
10	41.7 (0.3)	60.4 (0.6)	75.5 (0.9)	45 (0)
100	70 (1)	87 (2)	100.7 (0.7)	45 (0)
200	73.4 (0.6)	91.6 (0.7)	102 (0.7)	45 (0)
300	74.3 (0.5)	92.4 (0.7)	100.8 (0.5)	45 (0)
400	71.9 (0.6)	92.1 (0.8)	100.9 (0.7)	45 (0)
500	73.2 (0.7)	92.3 (0.5)	101.8 (0.5)	45 (0)
600	72.8 (0.7)	92.4 (0.7)	102 (0.7)	45 (0)
700	72.6 (0.4)	91.9 (0.8)	102.2 (0.7)	45 (0)
800	73.5 (0.6)	92.4 (0.4)	101.7 (0.7)	45 (0)
900	73.1 (0.7)	92 (0.7)	102.6 (0.6)	45 (0)
1000	74 (0.5)	91.8 (0.8)	101.3 (0.5)	45 (0)

RTBS_1	RTBS_5	RTBS_10	CL
0.01 (0)	0.02 (0)	0.02 (0)	0 (0)
0.03 (0)	0.06 (0)	0.11 (0)	0 (0)
0.03 (0)	0.07 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.11 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)

Figure 4.46.: Random trees with $|V| = 10$ and $\alpha = \alpha' = 5 \cdot 10^{-4}$ for binary random vectors.



n	RTBS_1	RTBS_5	RTBS_10	CL
10	0 (0)	0 (0)	0 (0)	0 (0)
100	0 (0)	0 (0)	0 (0)	0.1 (0.07)
200	0 (0)	0 (0)	0 (0)	0.2 (0.09)
300	0 (0)	0 (0)	0 (0)	0.3 (0.1)
400	0 (0)	0 (0)	0 (0)	0.4 (0.1)
500	0 (0)	0 (0)	0 (0)	0.6 (0.1)
600	0 (0)	0 (0)	0 (0)	0.3 (0.1)
700	0 (0)	0 (0)	0 (0)	0.7 (0.1)
800	0 (0)	0 (0)	0 (0)	0.5 (0.1)
900	0 (0)	0 (0)	0 (0)	0.7 (0.1)
1000	0 (0)	0 (0)	0 (0)	0.8 (0.1)

RTBS_1	RTBS_5	RTBS_10	CL
13.5 (0.5)	12.7 (0.4)	14.2 (0.4)	12.2 (0.6)
14 (0.5)	13.3 (0.7)	13.3 (0.6)	5.2 (0.6)
6.6 (0.5)	6.6 (0.5)	6.7 (0.6)	3.3 (0.5)
6.5 (0.5)	6.5 (0.5)	6.5 (0.5)	2 (0.4)
6.5 (0.5)	6.5 (0.5)	6.5 (0.5)	1.8 (0.5)
6.5 (0.5)	6.5 (0.5)	6.5 (0.5)	1 (0.3)
6.5 (0.5)	6.5 (0.5)	6.5 (0.5)	1.7 (0.3)
6.5 (0.5)	6.5 (0.5)	6.5 (0.5)	0.8 (0.3)
6.5 (0.5)	6.5 (0.5)	6.5 (0.5)	1.3 (0.3)
6.5 (0.5)	6.5 (0.5)	6.5 (0.5)	0.8 (0.3)
6.5 (0.5)	6.5 (0.5)	6.5 (0.5)	0.7 (0.3)

n	RTBS_1	RTBS_5	RTBS_10	CL
10	42 (0.3)	60.5 (0.7)	75.2 (0.7)	45 (0)
100	44 (2)	60 (1)	79 (2)	45 (0)
200	72.4 (0.6)	92 (0.7)	101.4 (0.5)	45 (0)
300	73.4 (0.5)	92.9 (0.5)	101.6 (0.6)	45 (0)
400	73.3 (0.6)	91.3 (0.6)	101.2 (0.7)	45 (0)
500	72.8 (0.6)	91.2 (0.8)	103.1 (0.5)	45 (0)
600	73 (0.7)	92.4 (0.5)	101 (0.5)	45 (0)
700	72.4 (0.6)	93.8 (0.5)	102.7 (0.5)	45 (0)
800	72.2 (0.5)	92.4 (0.6)	102.5 (0.6)	45 (0)
900	72.6 (0.7)	92.7 (0.7)	102 (0.8)	45 (0)
1000	73 (0.7)	93.2 (0.6)	101.7 (0.6)	45 (0)

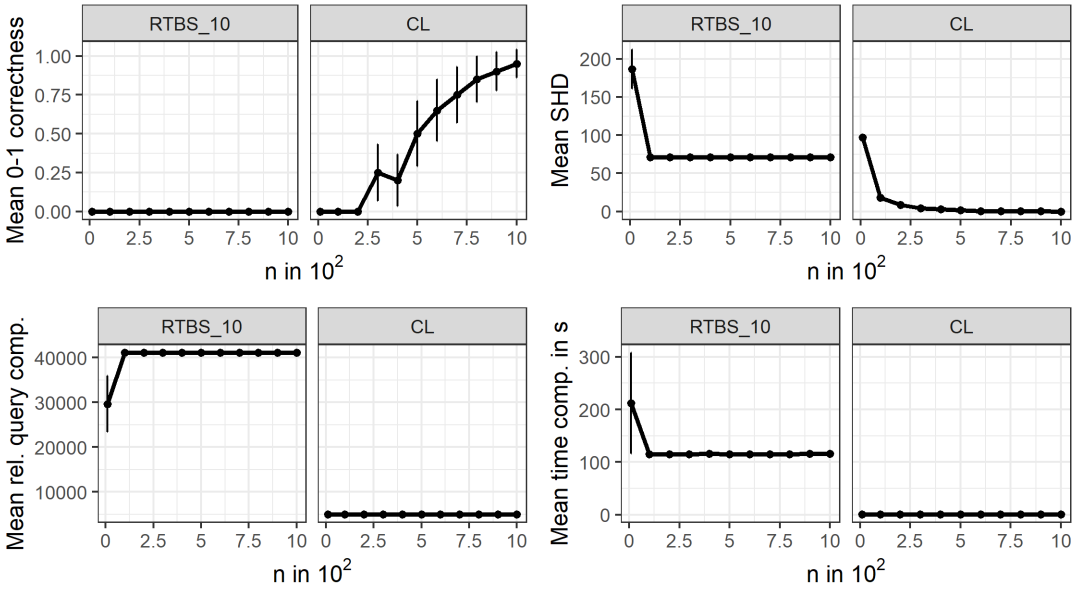
RTBS_1	RTBS_5	RTBS_10	CL
0.01 (0)	0.02 (0)	0.02 (0)	0 (0)
0.01 (0)	0.02 (0)	0.03 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.11 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)
0.03 (0)	0.06 (0)	0.1 (0)	0 (0)

Figure 4.47.: Random trees with $|V| = 10$ and $\alpha = \alpha' = 5 \cdot 10^{-6}$ for binary random vectors.

For chains, `ReconstructTreeBinarySample` nearly always returns the correct tree for all but the smallest sample sizes. The Chow-Liu algorithm is not that good, but also reaches very low levels of the SHD. The performance of `ReconstructTreeBinarySample` for chains seems to be very good for all choices of α and α'. This may either be due to the small number of vertices or our very restrictive approach to generate data from a binary distribution with a given underlying dependence structure.
For random trees, the performance in terms of correctness of `ReconstructTreeBinarySample` is not that good, the SHD only gets down to 5; in that instance the Chow-Liu algorithm is eventually better.
For stars, the performance of `ReconstructTreeBinarySample` is not very good, surprisingly, it is better for smaller n.

The query complexity of `ReconstructTreeBinarySample` is higher than that of the Chow-Liu algorithm, i.e. `ReconstructTreeBinarySample` needs more three-way tables than the Chow-Liu algorithm needs two-way tables.
Moreover, the Chow-Liu algorithm is orders of magnitude faster than `ReconstructTreeBinarySample`.

We now do another simulation with $|V| = 100$ and illustrate some of the shortcomings of `ReconstructTreeBinarySample`. Previously, we have seen that `ReconstructTreeBinarySample` needs to query approximately 100 three-way tables for $\kappa = 10$. Now, we set $|V| = 100$ and focus on $\kappa = 10$. We will see, how badly the query complexity scales for `ReconstructTreeBinarySample` with respect to the number of vertices. We have not included simulations for higher values of κ, given that this took an impractical amount of time and other resources.



n	RTBS_10	CL
100	0 (0)	0 (0)
1000	0 (0)	0 (0)
2000	0 (0)	0 (0)
3000	0 (0)	0.2 (0.2)
4000	0 (0)	0.2 (0.2)
5000	0 (0)	0.5 (0.2)
6000	0 (0)	0.7 (0.2)
7000	0 (0)	0.8 (0.2)
8000	0 (0)	0.8 (0.1)
9000	0 (0)	0.9 (0.1)
10000	0 (0)	0.95 (0.09)

RTBS_10	CL
190 (30)	97 (3)
71 (3)	18 (2)
71 (3)	8 (2)
71 (3)	4 (1)
71 (3)	2.9 (0.8)
71 (3)	1.5 (0.9)
71 (3)	0.7 (0.4)
71 (3)	0.5 (0.4)
71 (3)	0.3 (0.3)
71 (3)	0.2 (0.3)
71 (3)	0.1 (0.2)

n	RTBS_10	CL
100	30000 (6000)	4950 (0)
1000	41110 (30)	4950 (0)
2000	41100 (40)	4950 (0)
3000	41110 (30)	4950 (0)
4000	41130 (30)	4950 (0)
5000	41090 (30)	4950 (0)
6000	41130 (30)	4950 (0)
7000	41100 (40)	4950 (0)
8000	41100 (30)	4950 (0)
9000	41130 (30)	4950 (0)
10000	41110 (30)	4950 (0)

RTBS_10	CL
200 (100)	0.69 (0.05)
115 (0.8)	0.58 (0.04)
115.3 (0.7)	0.55 (0.04)
115.4 (0.8)	0.55 (0.05)
115.6 (0.7)	0.51 (0.04)
115.2 (0.8)	0.59 (0.06)
115.2 (0.8)	0.5 (0.03)
115.3 (0.8)	0.53 (0.05)
115.2 (0.8)	0.48 (0.03)
115.8 (0.9)	0.55 (0.06)
115.7 (0.9)	0.52 (0.04)

Figure 4.48.: Random trees with $|V| = 100$ and $\alpha = \alpha' = 5 \cdot 10^{-4}$ for binary random vectors.

We see that the correctness in terms of SHD of `ReconstructTreeBinarySample` is reasonably good, even though it is not as good as the Chow-Liu algorithm. However, the query complexity has risen to more than 40000. For $|V| = 10$ and

$\kappa = 1$, the query complexity was only around 40, for $\kappa = 10$ it was only around 100.
The time complexity has also increased to nearly 100 seconds per tree reconstruction from only around 0.03 seconds for $|V| = 10$.

These issues are very problematic in practice, even though we obtain comparable theorems for complexity similarly as for Gaussians; with our resources we were not able to study larger vertex sets and larger values of κ.

5. Discussion

In this thesis, we have studied the reconstruction of the concentration graph resp. the conditional independence graph of a given Gaussian or binary random vector if the graph is a tree. Based on a paper from Lugosi et al. (2021) we have derived several empirical algorithms, generally referred to as `ReconstructTreeSample` (or `ReconstructTreeBinarySample`), that allow us to reconstruct trees using only a few entries of the sample covariance matrix (or few three-way tables). We have framed this derivation with the notion of hypothesis testing and discussed suitable significance levels from the perspective of multiple testing and setting each significance level to a constant value. In both frameworks, we obtained consistency results on correctness and bounds on the query and time complexity.

`ReconstructTreeSample` is a divide-and-conquer algorithm, in each recursion level it cuts out a particular vertex from the underlying subtree which splits the subtree into several connected components. To obtain good complexity bounds, the cut vertex should be central, and for that `ReconstructTreeSample` uses the subprocedure `sCentralSample`, which has the job to return a central vertex in the tree. Each call of `sCentralSample` costs extra time and queries, but the procedure as a whole tries to leverage the trade-off between extra iterations and a more central vertex, which may lead to less overall time and query complexity.
We have also discussed whether there are other algorithms with a similar level of correctness and a lower query complexity. In that context, we have also conjectured that we can always find a bad enough matrix $\mathbf{\Sigma}$ such that the probability that the correct tree has been returned by `ReconstructTreeSample` is arbitrarely low for finite sample size.

In Chapter 3 we extended the results from Gaussians to binary random vectors. We chose the conditional G^2-statistic as the underlying test statistic due to its interpretation as a likelihood ratio test for testing conditional independencies. The conditional G^2-statistic cannot be computed using just the entries of the sample covariance matrix, due to this, we introduced a new oracle which queries three-way tables instead of entries from the covariance matrix. Using a concentration bound on the entropy from Antos and Kontoyiannis (2001), we obtained similar results on correctness, query and time complexity as for Gaussians.

In Chapter 4 we studied `ReconstructTreeSample` resp. `ReconstructTreeBi-`

`narySample` in simulations and compared them with the Chow-Liu algorithm. For Gaussians, we were able to show empirically that `ReconstructTreeSample` is able to obtain a reasonable non-worsening level of correctness such that the fraction of queried entries of the sample covariance matrix decreases for an increasing number of vertices. Moreover, the Chow-Liu algorithm turned out to be faster only for cases with a small number of vertices.
In terms of correctness, the Chow-Liu algorithm usually beat `ReconstructTreeSample`, nevertheless, the correctness of `ReconstructTreeSample` was much better than randomly guessing trees. For some cases, for example stars and a decreasing signicance level, `ReconstructTreeSample` was even better than the Chow-Liu algorithm. To summarize the empirical results for Gaussians, `ReconstructTreeSample` yields a reasonably accurate estimation of the underlying tree structure in a very efficient manner. This efficiency advantage is in particular visible for larger vertex sets.

In the binary case, `ReconstructTreeBinarySample` was also reasonably good in terms of correctness, and in some examples even better than the Chow-Liu algorithm. However, the query complexity of `ReconstructTreeBinarySample` was worse than that of the Chow-Liu algorithm. The reason for that is that there are $\binom{V}{3}$ unique three-way tables, but $\binom{V}{2}$ unique two-way tables. Thus, to beat the Chow-Liu algorithm in terms of query complexity, the vertex sets need to be much larger such that the fraction that `ReconstructTreeBinarySample` queries out of all three-way tables is significantly lower than the number of all two-way tables. However, these are theoretical considerations, in practice, `ReconstructTreeBinarySample` turns out to be too complex. Better approaches are surely needed. For that, one better relies on test statics or other decision rules that are of quadratic, and not cubic order with respect to number of vertices; one possible solution may be to directly calculate the sample partial correlation and use some thresholds to decide when it is zero or not. These decision rules have been discussed very briefly at the end of Lugosi et al. (2021), but more studies need to be done here; for example, it needs to be checked whether the correctness suffers under these alternative decision rules.

In this thesis, we have only focussed on Gaussian and binary distributions with respect to trees; it would be interesting to study further distribution types and graph structures. A crucial result in the derivation of `ReconstructTreeSample` resp. `ReconstructTreeBinarySample` was that the pairwise correlations for Gaussian and binary random vectors factorize over trees (see Lemma 2.11 and 2.13). This decomposition is also true in more general when the conditional expectation $\mathbb{E}[X_v|X_u]$ between two adjacent vertices X_v and X_u in the tree is an affine function of X_u (see Lugosi et al. (2021) and Zwiernik (2019)). Other approaches to

derive reconstruction algorithms using only a few entries may be based on Loh and Wainwright (2013), who discuss that zeros in generalized covariance matrices, i.e. covariance matrices augmented by further moments, correspond to conditional independencies of a given discrete graphical model.
Lugosi et al. (2021) have also discussed other tree structures, namely graphs with small 2-connected components and graphs with bounded treewidth, however, deriving results for these types of graphs is more complex. Ibid. also discusses that the reconstruction of graphs using only a few entries cannot be generalized to arbitrary graphs, because special types of graphs, e.g. trees, enjoy useful factorization and other properties, that general graphs do not necessarily have. Nevertheless, one may be interested to study other classes of graphs, e.g. triangulated graphs (or triangulations of graphs), which Loh and Wainwright (2013) consider.
Finally, less restrictive algorithms for generating data from binary random vectors with a given dependence structure also allow a better evaluation in simulations; our choice, which was based on truncating Gaussian random vectors, was difficult handle in practice.

A. R-Code

A.1. Comments on implementation

We have rented 6 t2.2xlarge-servers from Amazon Web Services. We have installed R-Studio on these servers using the templates of Aslett (2020). Our implementation can be found under `https://github.com/TomHochsprung/Master-Thesis`. Most of the code has been written in base R without using much external packages. Exceptions to this rule are

- the MASS-package for simulating from a multivariate normal (Venables and Ripley (2002));
- the data.table-package for the fread-function to read in external csv-files (see Dowle and Srinivasan (2021));
- the bindata-package to simulate binary random vectors (see Leisch et al. (2021), Leisch et al. (1998));
- the ggm-package for the function topSort which is used in the Cholesky method to generate covariance matrices (see Marchetti et al. (2020));
- the parallel-package to have a faster calculation for the oracle in the binary setting (see R Core Team (2021)), it is, however, not used directly when running the algorithms.

The plots and data-wrangling for the simulations results have been done on a local computer, because we did not manage to install relevant packages on the servers. The respective R-version on our local machine was 4.5. We have used

- the tidyverse-package for general data-wrangling tasks (see Wickham et al. (2019)
- the ggnetwork-package (see Briatte (2020)) and ggpubr-package (see Kassambara (2020)) for special graphics functions;
- the xtable-package for converting simulation results to LaTex-tables: xtable (see Dahl et al. (2019));
- the extrafont-package for fonts (see Chang (2014)).

B. Bibliography

Agrawal, R. (2020). Finite-sample concentration of the multinomial in relative entropy. *IEEE Transactions on Information Theory*, 66(10):6297–6302.

Anderson, T. W. (2003). *An introduction to Multivariate Statistical Analysis*. Wiley series in probability and mathematical statistics. Wiley, Hoboken, N.J., 3rd edition.

Antos, A. and Kontoyiannis, I. (2001). Convergence properties of functional estimates for discrete distributions. *Random Struct. Alg.*, 19:163–193.

Aslett, L. (2020). RStudio Server Amazon Machine Image (AMI). `https://www.louisaslett.com/RStudio_AMI/`. Last accessed on September 9th, 2021.

Banerjee, O., El Ghaoui, L., and d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516.

Briatte, F. (2020). *ggnetwork: Geometries to Plot Networks with 'ggplot2'*. R package version 0.5.8.

Chan, T. E., Stumpf, M. P., and Babtie, A. C. (2017). Gene regulatory network inference from single-cell data using multivariate information measures. *Cell Systems*, 5(3).

Chan, T. F., Golub, G. H., and LeVeque, R. J. (1983). Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician*, 37(3):242–247.

Chang, W. (2014). *extrafont: Tools for using fonts*. R package version 0.17.

Chow, C. K. and Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

Cook, M. B. (1951a). Bi-variate k-statistics and cumulants of their joint sampling distribution. *Biometrika*, 38(1/2):179–195.

Cook, M. B. (1951b). Two applications of bivariate k-statistics. *Biometrika*, 38(3/4):368–376.

Córdoba, I., Bielza, C., Larranaga, P., Varando, and Gherardo (2020). Sparse cholesky covariance parametrization for recovering latent structure in ordered data. *IEEE Access*, 8:154614–154624.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. The MIT Press, Cambridge, Massachusetts and London, 3rd edition.

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*. Wiley, Hoboken, N.J., 2nd edition.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Dahl, D. B., Scott, D., Roosen, C., Magnusson, A., and Swinton, J. (2019). *xtable: Export Tables to LaTeX or HTML*. R package version 1.8-4.

Dempster, A. P. (1972). Covariance selection. *Biometrics*, 28:157–175.

Dowle, M. and Srinivasan, A. (2021). *data.table: Extension of 'data.frame'*. R package version 1.14.0.

Drton, M. and Maathuis, M. H. (2017). Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4(1):365–393.

Drton, M. and Perlman, M. D. (2007). Multiple testing and error control in gaussian graphical model selection. *Statistical Science*, 22(3):430–449.

Dudoit, S., van der Laan, M. J., and Birkner, M. D. (2004). Multiple testing procedures for controlling tail probability error rates. *U.C. Berkeley Division of Biostatistics Working Paper Series*. Working Paper 166.

Fang, H.-r. and O'Leary, D. P. (2008). Modified Cholesky algorithms: a catalog with new approaches. *Mathematical Programming*, 115(2):319–349.

Ferguson, T. S. (1996). *A Course in Large Sample Theory*. Chapman & Hall, London and Weinheim and New York and Tokyo and Melbourne and Madras, 1st edition.

Fisher, R. A. (1924). The distribution of the partial correlation coefficient. *Metron*, 3:329–332.

Fisher, R. A. (1935). *The Design of Experiments*. Oliver & Boyd, Edinburgh and London, 1st edition.

Foygel, R. and Drton, M. (2010). Extended bayesian information criteria for gaussian graphical models. *Advances in Neural Information Processing Systems*, 23:2020–2028.

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

Geršgorin, S. (1931). Über die Abgrenzung der Eigenwerte einer Matrix. *Izv. Akad. Nauk SSSR Ser. Mat.*, 1:749–754.

Gibbs, A. L. and Su, F. E. (2002). On choosing and bounding probability metrics. *International Statistical Review*, 70(3):419–435.

Harary, F. (1969). *Graph Theory*. Addison-Wesley Publishing Company, Reading, Massachusetts and Menlo Park, California and London and Don Mills, Ontario.

Hochberg, Y. and Tamhane, A. C. (1987). *Multiple Comparison Procedures*. Wiley, Hoboken, N.J.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

Hoeffding, W. (1965). Asymptotically optimal tests for multinomial distributions. *The Annals of Mathematical Statistics*, 36(2):369–401.

Hoeffding, W. (1967). On probabilities of large deviations. *Berkeley Symposium on Mathematical Statistics and Probability*, 5(1):203–219.

Hojsgaard, S., Edwards, D., and Lauritzen, S. (2012). *Graphical Models with R*. Springer Science+Business Media, New York and Dordrecht and Heidelberg and London.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

Horn, R. A. and Johnson, C. R. (2013). *Matrix Analysis*. Cambridge University Press, Cambridge, 2nd edition.

Hotelling, H. (1953). New light on the correlation coefficient and its transforms. *Journal of the Royal Statistical Society. Series B (Methodological)*, 15(2):193–232.

Hsieh, C.-J., Dhillon, I. S., Ravikumar, P., and Banerjee, A. (2012). A divide-and-conquer procedure for sparse inverse covariance estimation. *Advances in Neural Information Processing Systems*, 25.

Hsieh, C.-J., Sustik, M. A., Dhillon, I. S., and Ravikumar, P. (2013). Big & quic: Sparse inverse covariance estimation for a million variables. *Advances in Neural Information Processing Systems*, 26.

Hwang, B., Lee, J. H., and Bang, D. (2018). Single-cell rna sequencing technologies and bioinformatics pipelines. *Experimental & Molecular Medicine*, 50(96).

Jefferys, W. H. and Berger, J. O. (1992). Ockham's razor and bayesian analysis. *American Scientist*, 80(1):64–72.

Kalisch, M. and Bühlmann, P. (2007). Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, (8):613–636.

Kallenberg, W. C. M. (1985). On moderate and large deviations in multinomial distributions. *The Annals of Statistics*, 13(4):1554–1580.

Kassambara, A. (2020). *ggpubr: 'ggplot2' Based Publication Ready Plots*. R package version 0.4.0.

Klenke (2020). *Probability Theory: A Comprehensive Course*. Universitext. Springer International Publishing, Cham, 3rd edition.

Knuth, D. E. (1998). *Seminumerical algorithms*, volume 2 of *The art of computer programming*. Addison-Wesley, Upper Saddle River, NJ and Boston and Indianapolis and San Francisco and New York and Toronto and Montréal and London and Munich and Paris and Madrid and Capetown and Sydney and Tokyo and Singapore and Mexico City, 3rd edition, 24th printing edition.

Kruskal Jr., J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

La Rocca, L. and Roverato, A. (2019). Discrete graphical models and their parameterization. In Maathuis, M., Drton, M., Lauritzen, S. L., and Wainwright, M., editors, *Handbook of Graphical Models*, Chapman & Hall/CRC handbooks of modern statistical methods, pages 191–216. CRC PRESS, Boca Raton.

Lauritzen, S. L. (1996). *Graphical models*, volume 17 of *Oxford statistical science series*. Clarendon Press, Oxford.

Lauritzen, S. L. and Dawid, A. P. (1993). Hyper markov laws in the statistical analysis of decomposable graphical models. *The Annals of Statistics*, 21(3):1272–1317.

Lehmann, E. L. and Romano, J. P. (2005). Generalizations of the familywise error rate. *The Annals of Statistics*, 33(3):1138–1154.

Leisch, F., Weingessel, A., and Hornik, K. (1998). On the generation of correlated artificial binary data. *Working Paper Series, SFB Adaptive Information Systems*, (13).

Leisch, F., Weingessel, A., and Hornik, K. (2021). *bindata: Generation of Artificial Binary Data*. R package version 0.9-20.

Letac, G. and Hélène Massam (2007). Wishart distributions for decomposable graphs. *The Annals of Statistics*, 35(3):1278–1323.

Liu, H., Han, F., Yuan, M., Lafferty, J., and Wasserman, L. (2012). High-dimensional semiparametric gaussian copula graphical models. *The Annals of Statistics*, 40(4):2293–2326.

Liu, H., Lafferty, J., and Wasserman, L. (2009). The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *Journal of Machine Learning Research*, 10:2295–2328.

Loh, P.-L. and Wainwright, M. (2013). Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses. *The Annals of Statistics*, 41(6):3022–3049.

Lugosi, G., Truszkowski, J., Velona, V., and Zwiernik, P. (2021). *Structure learning in graphical models by covariance queries*. Journal of Machine Learning Research. To appear.

MacKay, D. J. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, 7th edition.

Macke, J. H., Berens, P., Ecker, A. S., Tolias, A. S., and Bethge, M. (2009). Generating spike trains with specified correlation coefficients. *Neural computation*, 21:397–423.

Marchetti, G. M., Drton, M., and Sadeghi, K. (2020). *ggm: Graphical Markov Models with Mixed Graphs*. R package version 2.5.

Meinshausen, N. and Bühlmann, P. (2006). High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):1436–1462.

Neapolitan, R. E. (2004). *Learning Bayesian Networks*. Prentice Hall, Upper Saddle River, NJ.

Paninski, L. (2003). Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253.

Paulsen, V. I., Power, Stephen, C., and Smith, R. R. (1989). Schur products and matrix completions. *Journal of Functional Analysis*, 85:151–178.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers.

Pearl, J. and Paz, A. (1987). Graphoids: a graph based logic for reasoning about relevancy relations. *Advances in Artificial Intelligence - II*, pages 357–363.

Prüfer, H. (1918). Neuer Beweis eines Satzes über Permutationen. *Archiv der Mathematik und Physik (3)*, 27:142–144.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rice, J. A. (2007). *Mathematical Statistics and Data Analysis*. Brooks/Cole, Cengage Learning, Belmont, CA, 3rd edition.

Roverato, A. (2000). Cholesky decomposition of a hyper inverse wishart matrix. *Biometrika*, 87(1):99–112.

Roverato, A. (2017). *Graphical Models for Categorical Data*. Semstat Elements. Cambridge University Press, Cambridge and New York and Port Melbourne and Delhi and Singapore, 1st edition.

Serfling, R. J. (1980). *Approximation theorems of mathematical statistics*. Wiley, New York.

Shaffer, J. P. (1995). Multiple hypothesis testing. *Annual Review of Psychology*, 461(1):561–584.

Sokal, R. R. and Rohlf, F. J. (1995). *Biometry: The Principles and Practice of Statistics in Biological Research*. W. H. Freeman and Company, New York, 3rd edition.

Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction, and search*. Adaptive computation and machine learning. MIT Press, Cambridge, Massachussetts and London, 2nd edition edition.

Studený, M. (2005). *Probabilistic Conditional Independence Structures*. Information Science and Statistics. Springer-Verlag London Ltd, London and Berlin and Heidelberg.

Studený, M. (2019). Conditional independence and basic markov properties. In Maathuis, M., Drton, M., Lauritzen, S. L., and Wainwright, M., editors, *Handbook of Graphical Models*, Chapman & Hall/CRC handbooks of modern statistical methods, pages 3–38. CRC PRESS, Boca Raton.

Subrahmaniam, K. and Gajjar, A. V. (1980). Robustness to nonnormality of some transformations of the sample correlation coefficient. *Journal of Multivariate Analysis*, 10:60–77.

Tan, V. Y. F., Anandkumar, A., Tong, L., and Willsky, A. S. (2011). A large-deviation analysis of the maximum-likelihood learning of markov tree structures. *IEEE Transactions on Information Theory*, 57(3):1714–1735.

Tan, V. Y. F., Anandkumar, A., and Willsky, A. S. (2010). Learning gaussian tree models: Analysis of error exponents and extremal structures. *IEEE Transactions on Signal Processing*, 58(5):2701–2714.

Thorburn, W. M. (1918). The myth of occam's razor. *Mind*, 27(345-353).

Thorne, T. (2018). Approximate inference of gene regulatory network models from rna-seq time series data. *BMC Bioinformatics*, 19(127).

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning*, 15(1):31–78.

Uhler, C. (2019). Gaussian graphical models. In Maathuis, M., Drton, M., Lauritzen, S. L., and Wainwright, M., editors, *Handbook of Graphical Models*, Chapman & Hall/CRC handbooks of modern statistical methods, pages 217–238. CRC PRESS, Boca Raton.

van der Vaart, A. W. (1998). *Asymptotic Statistics*. Cambridge University Press, Cambridge.

Varga, R. S. (2004). *Geršgorin and his Circles*, volume 36 of *Springer series in computational mathematics*. Springer, Berlin and Heidelberg and New York.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition.

Welford, B. P. (1962). Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420.

Wermuth, N. (1980). Linear recursive equations, covariance selection, and path analysis. *Journal of the American Statistical Association*, 75(372):963–972.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., FranÃ§ois, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., MÌller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., and Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686.

Wu, B. Y. and Chao, K.-M. (2004). *Spanning trees and optimization problems*. Discrete mathematics and its applications. Chapman & Hall/CRC, Boca Raton, FL.

Yule, G. U. and Kendall, M. G. (1965). *An Introduction to the Theory of Statistics*. Griffin, London, 14th edition.

Zhang, X., Zhao, X.-M., He, K., Le Lu, Cao, Y., Liu, J., Hao, J.-K., Liu, Z.-P., and Chen, L. (2012). Inferring gene regulatory networks from gene expression data by path consistency algorithm based on conditional mutual information. *Bioinformatics*, 28(1):98–104.

Zwiernik, P. (2019). Latent tree models. In Maathuis, M., Drton, M., Lauritzen, S. L., and Wainwright, M., editors, *Handbook of Graphical Models*, Chapman & Hall/CRC handbooks of modern statistical methods, pages 265–288. CRC PRESS, Boca Raton.