Monte Carlo Dependency Estimation

Edouard Fouché & Klemens Böhm
Karlsruhe Institute of Technology (KIT)
{edouard.fouche,klemens.boehm}@kit.edu

ABSTRACT

Estimating dependency is a fundamental task in data management. Identifying the relevant variables leads to better understanding and improves both the runtime and outcome of data analysis. In this paper, we propose *Monte Carlo Dependency Estimation* (*MCDE*), a framework to estimate multivariate dependency. *MCDE* quantifies dependency as the average discrepancy between marginal and conditional distributions via Monte Carlo simulations. Based on this framework, we present *Mann-Whitney P* (*MWP*), a novel dependency estimator. We show that *MWP* satisfies a number of desirable properties and demonstrate the superiority of our estimator against the state-of-the-art multivariate dependency measures.

CCS CONCEPTS

• **Mathematics of computing → Multivariate statistics**; *Exploratory data analysis*; • **Information systems → Data mining**.

KEYWORDS

Dependency Estimation; Correlation Analysis; Anytime Algorithms

ACM Reference Format:
Edouard Fouché & Klemens Böhm. 2019. Monte Carlo Dependency Estimation. In *31st International Conference on Scientific and Statistical Database Management (SSDBM '19), July 23–25, 2019, Santa Cruz, CA, USA*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3335783.3335795

1 INTRODUCTION

1.1 Motivation

Estimating statistical relationships between variables is fundamental to any knowledge discovery process and has become an important topic in the database community [4, 9, 28]. Knowing the relationship between attributes, one can infer useful knowledge about unknown outcomes. For example, knowing that weight and arterial pressure correlate with the odds of contracting certain diseases may guide physicians, when predicting whether a patient will become sick within a year or not.

In the real world, data often comes as an open-ended, ever evolving stream of sensor signals. The signals can be noisy, redundant or generated at a varying speed. In this setting, the timely detection of changes in the stream is crucial; the early discovery of anomalies can, say, facilitate predictive maintenance and yield tremendous cost savings. We see the following requirements which any dependency estimator should satisfy. To our knowledge, any existing solution only fulfils some of them at best. In this article, we propose a new estimator which features all these characteristics:

R1: Multivariate. Bivariate measures only apply to two entities (i.e., variables, vectors). Estimating the dependency between more than two entities is useful as well, but existing attempts to generalize bivariate measures lack efficiency or effectiveness, as we will show.

R2: Efficient. Monitoring and respective computations should be 'at least as fast' as the stream. Next, one often is not only interested in one set of attributes, but potentially in all of them. Since the number of attribute combinations grows exponentially with the number of attributes, the efficiency of dependency estimators is crucial, with large data sets in particular.

R3: General-purpose. Dependency estimators should not be restricted to specific types of dependency, or they may miss important associations. Existing multivariate estimators are typically limited, e.g., to monotonous dependencies or to functional ones.

R4: Intuitive. A method is intuitive if its parameters are easy to set, i.e., users understand their impact on the estimation process. Existing solutions typically have unintuitive parameters, and the suggestion of 'good' parameter values often happens at the discretion of the inventors. Different values often yield very different results. This calls for a method that is intuitive to use.

R5: Non-parametric. Since real data can exhibit virtually any kind of distribution, it is not reasonable to use measures relying on parametric assumptions. The risk is to systematically miss relevant effects with wrong assumptions.

R6: Interpretable. The results of dependency estimators should be interpretable. In particular, the returned estimate should have a maximum and a minimum, so that one can interpret a given estimate from 'highly dependent' to 'independent'.

R7: Sensitive. Dependency estimation is not only about deciding whether a relationship exists, but also about quantifying its strength. Data points generally are observations sampled from a potentially noisy process. The same dependency should get a higher score when observed with more objects, as the size of the observed effect – the 'effect size' – is larger.

R8: Robust. Real-world data may be of poor quality. It is common to discretise attributes, for a more compact representation. Next, measuring devices often have a limited precision, such that values are rounded or trimmed, wrongly leading to points with exactly the same values. Such artefacts can have a negative influence on the estimation. In other words, estimators need to be robust against duplicates and imprecision.

R9: Anytime. Last but not least, users should be able to trade accuracy for a faster computation and to interrupt the estimation process at any time; a data set may be too large to allow for acceptable computation times, or the rate of incoming items from a data stream may vary. Thus, users should be able to set a 'budget' they are willing to spend, and the estimator should return approximate results, ideally with a known quality, in case of early termination.

1.2 Contributions

The contributions of this article are as follows:

We introduce *Monte Carlo Dependency Estimation (MCDE)*, a framework to estimate multivariate dependency. *MCDE* estimates the dependency of an attribute set as the average discrepancy between marginal and conditional distributions, via Monte Carlo simulations. In each iteration, a condition is applied on each dimension, in a process called *subspace slicing* [11]. A statistical test quantifies the discrepancy between the marginal and conditional distributions of a dimension taken at random. *MCDE* is abstract, since the underlying statistical test is left unspecified. We determine a lower bound for the quality of the estimation, allowing to trade a quantifiable level of accuracy for a computational advantage.

As a proof of concept, we instantiate within *MCDE* a new dependency measure, named *Mann-Whitney P* (*MWP*). *MWP* relies on the *Mann-Whitney U* test, a well-known non-parametric statistical test [15], to quantify the average discrepancy between the marginal and conditional distributions. We describe the implementation of *MWP* in detail. We compare our estimator to the state of the art. We benchmark each approach using an assortment of synthetic dependencies. In particular, we measure the statistical power and execution time of each approach. This will show that *MWP* fulfils all requirements while the existing ones do not.

We release our source code and experiments on GitHub[1], together with documentation, to ensure reproducibility.

Paper outline: Section 2 reviews related work. Section 3 describes *MCDE* and *MWP*. Section 4 evaluates *MWP* and compares it to the state of the art. Section 5 concludes. In Appendix, we provide the formal proofs and details about our algorithms and experiments.

2 RELATED WORK

Estimating the correlation has been of interest for more than a century. Many bivariate measures exist, e.g., [21, 24]. Some of them also target at quantifying the association between two vectors which are possibly multivariate [1, 8, 14, 25]. However, they can only measure the dependency between two entities – not several ones, i.e., they do not fulfil requirement **R1**. They also often have other drawbacks. For example, the *Pearson correlation coefficient* is parametric (**R5**) and targets at linear dependencies (**R3**).

There are attempts to extend bivariate dependency measures to the multivariate case. [22] propose an extension of Spearman's ρ to multivariate data (*MS*), but it is limited to monotonous relationships (**R3**). Several authors also propose multivariate extensions of *Mutual Information* [26]. For example, *Interaction Information* (*II*) [16] quantifies the 'synergy' or 'redundancy' in a set of variables. Similarly, *Total Correlation* (*TC*) [27] quantifies the total amount of information. However, information-theoretic measures are difficult to estimate, as they require knowledge about the underlying probability distributions. Density estimation methods, based on kernels, histograms or local densities, all require to set unintuitive parameters (**R4**) and may be computationally expensive (**R2**). Next, with many dimensions, density estimation becomes meaningless, due to the *curse of dimensionality* [2]. Information-theoretic measures also are difficult to interpret (**R6**), since they usually correspond to a quantity of bits or nats, that is theoretically unbounded.

More recently, *Cumulative Mutual Information* (*CMI*) [20], *Multivariate Maximal Correlation* (*MAC*) [19] and *Universal Dependency Score* (*UDS*) [18] have been proposed as multivariate dependency measures. They are remotely related to concepts from information theory, as they rely on the so-called *cumulative entropy* [5]. However, these measures are computationally expensive (**R2**) and not intuitive (**R4**). They also are difficult to interpret, because their theoretical maximum and minimum vary with the number of dimensions (**R6**). Another approach, *High Contrast Subspaces* (*HiCS*) [11], is the one most similar to *MCDE/MWP*. It introduces *subspace slicing* as a heuristic to quantify the potential of subspaces to contain outliers. Yet the suitability of *HiCS* as a dependency estimators is unknown so far. To our knowledge, the requirements **R7**, **R8** and **R9** have not been considered in the literature.

In Section 4, we compare our estimator *MWP* to the related work, namely *MS*, *TC*, *II*, *CMI*, *MAC*, *UDS* and *HiCS*, against each of our requirements, via experiments.

3 THE MCDE FRAMEWORK

Dependency estimation determines to which extent a relationship differs from randomness. In this spirit, *MCDE* quantifies a dependency, i.e., an extent of independence violation, based on marginal and conditional distributions.

3.1 Notation

A database DB is a set of dimensions/attributes/variables $D = \{s_1, \dots, s_d\}$ and a list of objects $B = (\vec{x}_1, \dots, \vec{x}_n)$, where $\vec{x}_i = \langle x_i^{s_j} \rangle_{j \in \{1,\dots,d\}}$ is a d-dimensional vector of real numbers. We call a subspace S a projection of the database on d' attributes, with $S \subseteq D$ and $d' \leq d$. To formalize our framework, we treat the attributes in D as random variables, i.e., a random variable X_{s_j} represents each attribute $s_j \in D$. Additionally, $p(X)$ is the joint probability density function (*pdf*) of a random vector $X = \langle X_{s_i} \rangle_{s_i \in S}$ and let $\hat{p}(X)$ denote the empirical estimation of this *pdf*. We use $p_{s_j}(X)$ and $\hat{p}_{s_j}(X)$ for the marginal *pdf* of s_j and its estimation. $\mathcal{P}(S)$ is the power set of S, i.e., the set of all attribute subsets. For any attribute subset $S' \in \mathcal{P}(S)$, its random vector is $X_{S'} = \langle X_{s_i} \rangle_{s_i \in S'}$, and its complement random vector is $\overline{X_{S'}} = X_{S \setminus S'} = \langle X_{s_i} \rangle_{s_i \in S \setminus S'}$.

3.2 Theory of MCDE

3.2.1 Measuring Dependency via Contrast. A set of variables is *independent* or *uncorrelated* if and only if all the variables are **mutually independent**. By treating the attributes of a subspace as random variables, we can define the independence assumption of a subspace as follows:

Definition 1 (Independence Assumption). *The independence assumption $\mathcal{A}$ of a subspace S holds if and only if the random variables $\{X_{s_i} : s_i \in S\}$ are mutually independent, i.e.:*

$$\mathcal{A}(S) \Leftrightarrow p(X) = \prod_{s_i \in S} p_{s_i}(X) \tag{1}$$

Under the independence assumption, the joint distribution of subspace S is **expected** to be equal to the product of its marginal distributions. We can define a degree of dependency based on the degree to which $\mathcal{A}$ does not hold:

[1]https://github.com/edouardfouche/MCDE-experiments

Definition 2 (Degree of Dependency). *The degree of dependency $\mathcal{D}$ of a subspace S is the discrepancy, abbreviated as 'disc', between the **observed** joint distribution $p^o(X)$ and the **expected** joint distribution $p^e(X) = \prod_{s_i \in S} p^o_{s_i}(X)$:*

$$\mathcal{D}(S) \equiv disc\left(p^o(X), p^e(X)\right) \tag{2}$$

While one can estimate the discrepancy between two probability distributions, using for instance the Kullback-Leibler divergence, this is not trivial here because $p^o(X)$ and $p^e(X)$ are a priori unknown. We work around this as follows:

Lemma 1. *The independence assumption $\mathcal{A}$ of a subspace S holds if and only if the joint pdf for all $S' \in \mathcal{P}(S)$ is equal to its joint conditional pdf given all other variables $S \setminus S'$:*

$$\mathcal{A}(S) \Leftrightarrow p(X_{S'}|\overline{X_{S'}}) = p(X_{S'}) \qquad \forall S' \in \mathcal{P}(S) \tag{3}$$

Lemma 1 provides an alternative definition of $\mathcal{A}$. However, it is still problematic: First, a multivariate density estimation is needed to estimate $p(X_{S'})$ and $p(X_{S'}|\overline{X_{S'}})$ with $|S'| \geq 1$ in the multivariate case. Second, even if one could estimate $p(X_{S'})$ and $p(X_{S'}|\overline{X_{S'}})$, estimating the densities for all $S' \in \mathcal{P}(S)$ is intractable. We instead relax the problem by considering only subspaces with $|S'| = 1$, i.e., we only look at the marginal *pdf* of single variables.

Definition 3 (Relaxed Independence Assumption). *The relaxed independence assumption $\mathcal{A}^*$ of a subspace S holds if and only if the marginal distribution $p_{s_i}(X)$ of each variable $s_i \in S$ equals $p_{s_i}(X|\overline{X_{s_i}})$, i.e., the conditional pdf of s_i:*

$$\mathcal{A}^*(S) \Leftrightarrow p_{s_i}(X|\overline{X_{s_i}}) = p_{s_i}(X) \qquad \forall s_i \in S$$

Lemma 2 (Independence Assumption Relaxation). *$\mathcal{A}(S) \Rightarrow \mathcal{A}^*(S)$, i.e., we can relax $\mathcal{A}$ into $\mathcal{A}^*$ for any subspace S.*

Loosely speaking, the relaxed independence assumption holds if and only if knowing the value of all variables but s_i does not bring any information about s_i.

$\mathcal{A}(S) \Rightarrow \mathcal{A}^*(S)$, then $\neg\mathcal{A}^*(S) \Rightarrow \neg\mathcal{A}(S)$, i.e., showing that $\mathcal{A}^*$ does not hold is sufficient but not necessary to show that $\mathcal{A}$ does not hold. Thus, we can define a relaxed degree of dependency $\mathcal{D}^*$ of a subspace S, as the discrepancy *disc* between the observed marginal distribution $p^o_{s_i}(X)$ and the expected one $p^e_{s_i}(X)$. Under the relaxed independence assumption $\mathcal{A}^*$, we have $p^e_{s_i}(X) = p^o_{s_i}(X|\overline{X_{s_i}})$. We define $\mathcal{D}^*$ as the expected value $\mathbb{E}[.]$ of this discrepancy:

Definition 4 (Relaxed Degree of Dependency).

$$\mathcal{D}^*(S) \equiv \mathop{\mathbb{E}}_{s_i \in S}\left[disc\left(p^o_{s_i}(X), p^o_{s_i}(X|\overline{X_{s_i}})\right)\right] \tag{4}$$

This definition is broad and contains a whole class of dependency estimators, e.g., [11]. This class of estimators aims at measuring a so-called notion of *contrast* of the subspace. $\mathcal{D}^*$ – or *contrast* – is a variant of $\mathcal{D}$ which is much easier to estimate: First, it relies on the comparison of marginal against conditional densities, i.e., multivariate density estimation is not required. Second, the number of degrees of freedom of $\mathcal{A}^*(S)$ increases linearly with $|S|$, while exponentially for $\mathcal{A}(S)$. Thus, $\mathcal{D}^*$ is less expensive to estimate.

By definition, $\mathcal{D}^*$ does not take into account the dependency between multivariate subsets, but only of each variable versus all others. However, we argue that this relaxation is not problematic, and even supports interpretability. In fact, the detection of dependency is only interesting as long as we can observe effects w.r.t. the marginal and conditional distributions: In real-world scenarios, one is typically looking for interpretable influences of particular variables – so-called 'targets' – on the system and vice versa.

3.2.2 *Estimating Conditional Distributions.*

The difficulty to estimate $\mathcal{D}^*$ is estimating the conditional distributions $p_{s_i}(X|\overline{X_{s_i}})$, because the underlying data distribution is unknown. As suggested in [11], we can simulate conditional distributions by applying a set of conditions to S, in a process called *subspace slicing*.

Definition 5 (Subspace Slice). *A slice c_i of subspace S w.r.t. dimension s_i is a set of $|S| - 1$ conditions $\left[l_{s_j}, u_{s_j}\right]$, which restricts the values of each dimension $s_j \in S \setminus s_i$:*

$$c_i = \left\{\left[l_{s_j}, u_{s_j}\right] : s_j \in S \setminus s_i\right\}$$
$$s.t.\ \forall \left[l_{s_j}, u_{s_j}\right] \in c_i, \left|\left\{\vec{x_k} : x^{s_j}_k \in \left[l_{s_j}, u_{s_j}\right]\right\}\right| = n'$$

*where $n' \in \{1, \dots, n\}$ is the number of objects per condition, and s_i is the **reference** dimension. We say that $\vec{x_k} \in c_i$ when $\vec{x_k}$ fulfils all the conditions in c_i. We define $\bar{c}_i$ as the set of complementary conditions:*

$$\bar{c}_i = \left\{(-\infty, l_{s_j}) \cup (u_{s_j}, \infty) : \left[l_{s_j}, u_{s_j}\right] \in c_i\right\} \tag{5}$$

$p_{s_i|c_i}(X)$ and $p_{s_i|\bar{c}_i}(X)$ denote the conditional pdf of the observation in the slice c_i and of its complement $\bar{c}_i$ respectively. $\mathcal{P}^c(S)$ is the set of all possible slices in subspace S.

We choose each interval in a slice at random and independently from each other. Under the independence assumption, the expected share of observations α in the slice is equal to $\alpha = (n'/n)^{|S|-1}$. Interestingly, n' can be determined given α and $|S|$:

$$n' = \left\lceil n \sqrt[|S|-1]{\alpha} \right\rceil \tag{6}$$

As a result, subspace slicing can be done in a **dimensionality-aware** fashion. When α is a constant, the expected number of objects per slice does not change between subspaces with different dimensionality. Thus, subspace slicing is a dynamic grid-based method, which does not suffer from the curse of dimensionality.

Definition 6 (Dimensionality-aware Slice). *A dimensionality-aware slice c^α_i of subspace S is a set of $|S| - 1$ conditions:*

$$c^\alpha_i = c_i \ s.t.\ n' = \left\lceil n \sqrt[|S|-1]{\alpha} \right\rceil \tag{7}$$

For brevity, we assume a fixed $\alpha \in (0, 1)$ and write $c_i \equiv c^\alpha_i$, and we omit '(X)' in $p_{s_j}(X)$ and $p_{s_j|c_j}(X)$ in the following.

The idea behind dimensionality-aware slicing is to simulate conditional distributions empirically. Under the $\mathcal{A}^*$-assumption, the conditional distribution $p_{s_i|c_i}$ equals the marginal distribution p_{s_i}, for any dimension s_i and slice c_i.

Lemma 3 ($\mathcal{A}^*$ and Conditional Distributions).

$$\mathcal{A}^*(S) \Leftrightarrow p_{s_i|c_i} = p_{s_i} \qquad \forall s_i \in S, \forall c_i \in \mathcal{P}^c(S) \tag{8}$$

3.2.3 *Discrepancy Estimation.*

In reality, one only has a limited number of observations, i.e., one only has access to empirical distributions. A solution is to use a statistical test $\mathcal{T}$:

$$disc\left(\hat{p}_{s_i}, \hat{p}_{s_i|c_i}\right) \equiv \mathcal{T}\left(\hat{p}_{s_i}, \hat{p}_{s_i|c_i}\right) \tag{9}$$

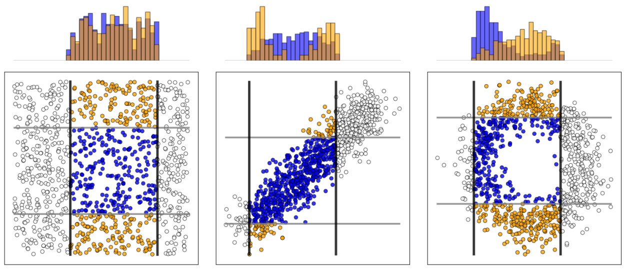

(a) Independent **(b) Linear** **(c) Circle**

Figure 1: Slicing in 2-D subspaces, with $\alpha = 0.5$

However, since the number of observations is finite, the observations from $\hat{p}_{s_i|c_i}$ are **included** in the set of observations from $\hat{p}_{s_i}$. This is problematic, as statistical tests assume the samples to be **distinct**. Plus, when $\alpha \approx 1$, $\hat{p}_{s_i|c_i}$ converges to $\hat{p}_{s_i}$, i.e., the two populations are nearly the same. Conversely, $\alpha \approx 0$ yields spurious effects, since the sample from $\hat{p}_{s_i|c_i}$ is then small. We solve the problem by observing that $p_{s_i|c_i}$ and $p_{s_i|\bar{c}_i}$ are equal under $\mathcal{A}^*$.

Theorem 1 ($\mathcal{A}^*$ and Complementary Conditions).

$$\mathcal{A}^*(S) \Leftrightarrow p_{s_i|\bar{c}_i} = p_{s_i|c_i} \qquad \forall s_i \in S, \forall c_i \in \mathcal{P}^c(S) \tag{10}$$

Hence, one can evaluate the assumption by looking instead at the discrepancies between the conditional distribution and its complementary conditional distribution. When doing so, the samples obtained from both distributions are distinct.

Dimensionality-aware slicing is defined based on α, the expected share of observations in the slice c_i. Thus, the expected share of observations $\bar{\alpha}$ in $\bar{c}_i$ equals $1 - \alpha$. This leads to setting $\alpha = 0.5$, so that $\bar{\alpha} = \alpha$. This choice is judicious for statistical testing, as equal sample sizes lead to higher statistical stability, and we get rid of α.

To further improve slicing, we also propose to restrict the domain of the reference dimension s_i to the same proportion α of objects. When doing so, statistical testing detects local effects in the marginal distributions better than when considering the full range. Next, this reduces the number of points in the two samples, leading to lower computational requirements of the underlying statistical test. Formally, we define the marginal restriction as follows:

Definition 7 (Marginal Restriction). *A marginal restriction is a condition on the reference dimension s_i, i.e., an interval $r_i : [l_{s_i}, u_{s_i}]$, such that $|\{\vec{x}_j : x_j^{s_i} \in r_i\}| = \lceil \alpha \cdot n \rceil$. We define $p_{s_i|c_i|r_i}$ as the restricted conditional distribution given c_i, r_i. $\mathcal{P}^r(S)$ is the set of all restrictions in subspace S.*

We illustrate slicing in Figure 1, with an independent subspace on the left-hand side and with subspaces with noisy dependencies on the right. The grey lines show a random slice c_x on the y-axis. Two black bold lines stand for the restriction r_x. The points in dark blue are in the restricted slice $c_x|r_x$ and the points in light orange are in $\bar{c}_x|r_x$. Using histograms, we plot along the x-axis the distribution of the points in both samples. From the histograms, we can see that the two distributions are relatively similar for Figure 1(a), while they are clearly different for Figures 1(b) and 1(c).

After each slicing operation, one obtains two object samples $B_{c_i|r_i}$ and $B_{\bar{c}_i|r_i}$ such that $B_{c_i|r_i} \cap B_{\bar{c}_i|r_i} = \emptyset$, and we use a statistical test $\mathcal{T}$ to estimate the discrepancy between $\hat{p}_{s_i|c_i|r_i}$ and $\hat{p}_{s_i|\bar{c}_i|r_i}$.

A statistical test $\mathcal{T}(B_1, B_2)$ on two samples B_1 and B_2 typically yields a p-value. Traditionally, one uses p-values to assess the *statistical significance*. Conversely, $p^c = 1 - p$ is a *confidence level* or probability to truly reject a false null hypothesis. The rationale behind $\mathcal{D}^*$ is to yield values quantifying the independence violation. We define our own notion of *contrast*, abbreviated as $\mathcal{C}$, as the expected value of the *confidence level* of a statistical test $\mathcal{T}$ between the samples from the conditional distributions for all the possible dimensions s_i, slices c_i and restrictions r_i:

Definition 8 (Contrast $\mathcal{C}$).

$$\mathcal{C}(S) \equiv \underset{\{c_i, r_i\} \in \mathcal{P}^{c \times r}}{\mathbb{E}} \left[\mathcal{T}\left(B_{c_i|r_i}, B_{\bar{c}_i|r_i}\right) \right] \tag{11}$$

where the test $\mathcal{T}$ yields p^c-values, and we draw c_i, r_i randomly and independently from each other from $\mathcal{P}^{c \times r} \equiv \mathcal{P}^c(S) \times \mathcal{P}^r(S)$, w.r.t. any reference dimension s_i in subspace S.

By definition, $\mathcal{T} \sim \mathcal{U}[0, 1]$ when the two samples are independent, and $\mathcal{T} \approx 1$ as the evidence against independence increases. The properties of $\mathcal{C}$ follow:

1. $\mathcal{C}$ converges to 1 as the dependency in S increases, since the p^c-values converge to 1 stochastically.
2. $\mathcal{C}$ converges to 0.5 when S is independent, since the distribution of the p^c-values converges to $\mathcal{U}[0, 1]$.
3. $\mathcal{C}$ is bounded between 0 and 1.

3.2.4 Monte Carlo Approximation. Unfortunately, $\mathcal{C}$ is impossible to compute exactly; one would need to know the distribution for every dimension, slice and restriction. Instead, we approximate $\mathcal{C}$ via Monte Carlo (MC) simulations, using M iterations. At each iteration, we choose the reference, slice and restriction at random. The *approximated contrast* $\hat{\mathcal{C}}$ is defined as follows:

Definition 9 (Approximated Contrast $\hat{\mathcal{C}}$).

$$\hat{\mathcal{C}}(S) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{T}\left(B_{[c_i|r_i]_m}, B_{[\bar{c}_i|r_i]_m}\right) \tag{12}$$

where $[c_i|r_i]_m$ means that we draw i, c_i and r_i randomly at iteration m, i.e., $i \leftarrow \{1, \ldots, |S|\}$ and $\{c_i, r_i\} \leftarrow \mathcal{P}^{c \times r}$.

Interestingly, we can bound the quality of the approximation. From Hoeffding's inequality [10], we derive a bound on $\hat{\mathcal{C}}$ w.r.t. $\mathcal{C}$, which decreases exponentially with increasing M:

Theorem 2 (Hoeffding's Bound of $\hat{\mathcal{C}}$).

$$\Pr\left(|\hat{\mathcal{C}} - \mathcal{C}| \geq \varepsilon\right) \leq 2e^{-2M\varepsilon^2} \tag{13}$$

where M is the number of MC iterations, and $0 < \varepsilon < 1 - \mathcal{C}$.

This is very useful. For instance, when $M = 200$, the probability of $\hat{\mathcal{C}}$ to deviate more than 0.1 from its expected value is less than $2e^{-4} \approx 0.04$, and this bound decreases exponentially with M. Thus, one can adjust the computational requirements of $\hat{\mathcal{C}}$, given the available resources or a desired quality level. In other words, users can set M intuitively, as it leads to an expected quality, and vice versa. Furthermore, M is the only parameter of *MCDE*. See Appendix A for our formal proofs.

3.3 Instantiation as *MWP*

To use the *MCDE* framework, one must instantiate a suitable statistical test as $\mathcal{T}$. To comply with our requirements, this test needs to be **non-parametric** (**R5**) and **robust** (**R8**). As a proof of concept, we instantiate $\mathcal{T}$ as a two-sided *Mann-Whitney U* test [15].

The *U* test has the following features which other statistical tests may lack. First, it is one of the most powerful statistical tests [17]: Its power-efficiency approaches 95.5% when comparing it to the *t*-test, as the number of observations *n* increases. But contrary to the *t*-test, the *U* test is **non-parametric**. Second, [6] shows that the *U* test is more efficient than the *Kolmogorov-Smirnov* test for large samples. Finally, the *U* test does not require continuous data, as it operates on ranks. Thus, it is **robust** and applicable to virtually any kind of ordinal measurements.

Our approach, *MWP*, is the instantiation of $\hat{\mathcal{C}}$ using a two-sided *U* test as the statistical test $\mathcal{T}$. The letter *P* emphasises that the test returns a p^c-value, as required by *MCDE*, i.e., whenever the evidence against independence increases, the values of the *U* test approach 1. *MWP* is the average of this test over *M* iterations:

Definition 10 (Mann-Whitney P (*MWP*)).

$$MWP = \frac{1}{M}\sum_{m=1}^{M} U\left(B_{[c_i|r_i]_m}, B_{[\bar{c}_i|r_i]_m}\right)$$

3.4 Algorithmic Considerations & Complexity

We now give an overview of our algorithm to efficiently compute the *MWP* score of a subspace *S*. See Algorithm 1.

First, we construct an index (Line 2), as a preprocessing step, to avoid the expensive repetition of sorting operations. Afterwards, for *M* iterations, we slice the data (Line 6) and compute the *U* test (Line 7). We can do this efficiently thanks to the index, because tuples are sorted. *MWP* is the average *U* test outcome over *M* iterations.

Algorithm 1 *MWP*

```
1: function MWP($S = \{s_i\}_{i \in \{1,\dots,d\}}$)
2:   $\mathcal{I} \leftarrow$ ConstructIndex($S$)
3:   $result \leftarrow 0$
4:   for $m \leftarrow 1$ to $M$ do
5:     $r \leftarrow$ random integer in $[1, d]$
6:     $slice \leftarrow$ Slice($\mathcal{I}, r$)
7:     $result \leftarrow result +$ U-test($\mathcal{I}, slice, r$)
8:   $result \leftarrow result/M$
9:   return $result$
```

To replace the statistical test, one only needs to change the algorithm behind U-test. The rest is part of the *MCDE* framework and does not require any adaptation. Since $|S| \ll n$, the overall complexity of *MWP* is in $O(n \cdot log(n) + M \cdot n)$. The index construction is asymptotically the most expensive step, as it is in $O(n \cdot log(n))$. However, one only needs to construct the index once. When the index for a given data set is available, one can compute *MWP* in linear time for the exponential number of subspaces. Thus, *MWP* scales well with the size of the data set. In Appendix B, we outline our algorithms for the index construction, slicing and the statistical test. We also provide a detailed complexity analysis.

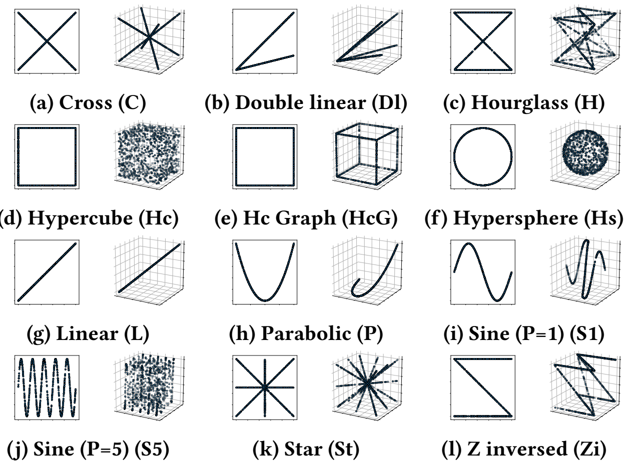

(a) Cross (C) (b) Double linear (Dl) (c) Hourglass (H)
(d) Hypercube (Hc) (e) Hc Graph (HcG) (f) Hypersphere (Hs)
(g) Linear (L) (h) Parabolic (P) (i) Sine (P=1) (S1)
(j) Sine (P=5) (S5) (k) Star (St) (l) Z inversed (Zi)

Figure 2: 12 selected multidimensional dependencies

4 EVALUATION

We implement every approach in Scala. We re-implement *MS* following [22], *TC* and *II* following [26] using Kraskov's [13] and Kozachenko & Leonenko estimators respectively, with $k = 4$. We use the R*-tree implementation from ELKI [23] to increase the efficiency of the k-NN queries. For *CMI*, *MAC*, *UDS* and *HiCS*, we use the implementation in [18]. Each algorithm runs single-threaded in a server with 32 cores at 2.2 GHz and 64GB RAM, with default parameters, if any. If not stated otherwise, the samples we use have $n = 1000$, $d = 3$, and we set $M = 50$ for *MWP* and *HiCS*. In most existing studies, such as in [19, 21], *n* usually is equal or lower.

Hereafter, we focus on the comparison of *MWP* to the state-of-the-art approaches. Appendix C contains a detailed evaluation of *MWP* w.r.t. parameters *d*, *n*, *M* and requirements **R7** and **R8**.

4.1 Methodology

To compare the approaches, we characterise the score they produce w.r.t. different dependencies of variable noise. Intuitively, noiseless dependencies should lead to higher scores than noisier ones.

4.1.1 Dependency Generation. For benchmarking, we use an assortment of 12 multivariate dependencies scaled to [0, 1], as showed in Figure 2. For each dependency, we repeatedly draw *n* objects with *d* dimensions, to which we add Gaussian noise with standard deviation σ, which we call *noise level.* We provide the pseudo-code for the generation of each dependency in Appendix B.

4.1.2 Evaluation Measures. A dependency estimator $\mathcal{E}$ is an operator $\mathcal{E}(S) \mapsto$ *score* which computes a *score* for a subspace *S*. We inspect the *score* of each estimator $\mathcal{E}$ against each dependency **X**, with increasing noise level σ. We consider 30 noise levels, distributed linearly from 0 to 1. For better comparability, we also include the independent subspace **I** in the experiments, where each attribute is i.i.d. in $\mathcal{U}[0, 1]$. For each dependency and each noise level, we draw 500 subspaces to compute the estimate. We record the average (avg) and standard deviation (std) for each estimator and, as in other multivariate studies [12, 18, 21], we compute the statistical power with confidence $\gamma = 0.95$, see Appendix C.

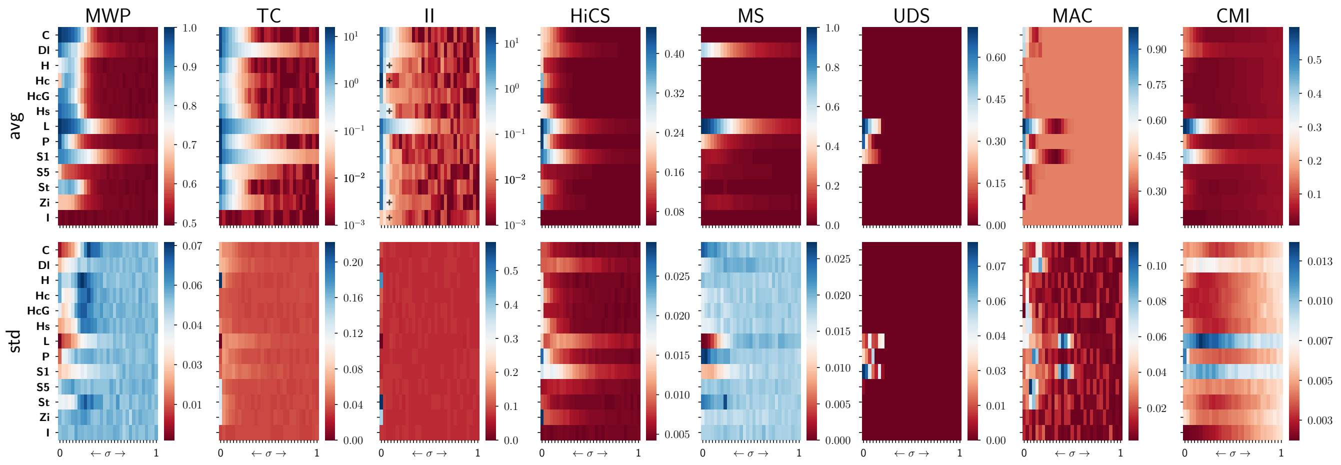
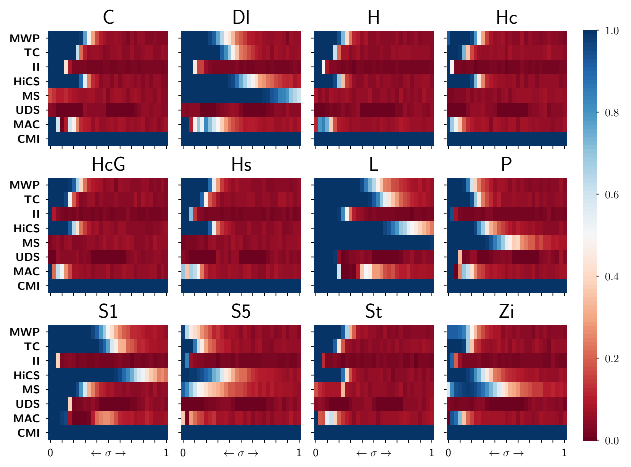
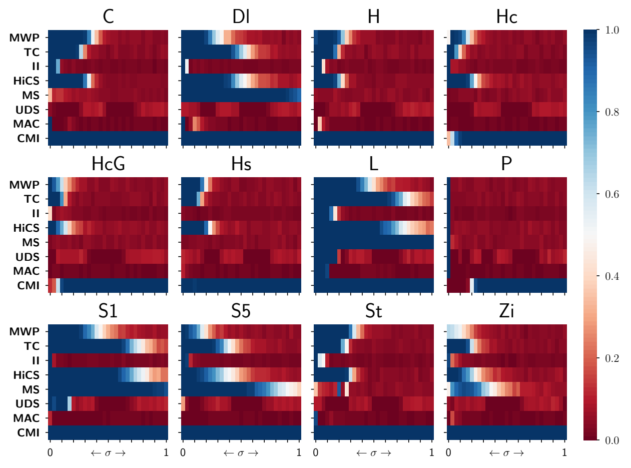


Figure 3: Distribution of dependency estimation scores, $d = 3$



(a) $d = 3$



(b) $d = 5$

Figure 4: Power against each dependency

4.2 Score Distribution and Statistical Power

We observe the distribution of the scores for each approach in Figure 3. Please note that the figures are best seen in colour. The expectation is that the scores and statistical power are high for noiseless dependencies, i.e., the left side of the plot is blue, and decrease gradually as we add noise. A noise level $\sigma = 1$ is comparably high, since the data is scaled to $[0, 1]$. Thus, the right side of the plot should be red, standing for low scores, or low power.

First, we see that the average score of *MWP* is most similar to *TC*. *TC* however is unbounded, and its scores follow a logarithmic scale. This means that the estimates of TC change very abruptly. We see that *MWP* scores are slightly smaller for noiseless dependencies with many ties w.r.t. marginal distributions, such as **H** and **Hc**, which we attribute to the correction for ties in the U test.

II can yield positive or negative values. We visualize the absolute value of *II* with a logarithmic scale. We mark the dependencies which obtain a positive score in their noiseless form with a plus sign. *II* assigns high scores to every noiseless dependency. However, the score decreases rapidly with noise, except for **L**.

HiCS shows a similar behaviour as *MWP*, except that the scores decrease faster, and that many dependencies start with a low score, even in the noiseless form, such as **C**, **Dl**, **H**, **Hs**, **S5** and **St**.

Next, *MS* and *UDS* are restricted to monotonous and bijective functional relationships respectively. They can detect only 3 out of 12 dependencies. *MAC* and *CMI* behave curiously. Their scores change noticeably only for **C**, **Dl**, **L**, **P** and **S1**. The values of *MAC* also change very abruptly and even non-monotonously with noise. For example, **L** and **S1** obtain lower scores with a noise level of 0.3 than with higher noise levels. *CMI* evolves smoothly. However, for many dependencies, including **I**, the score increases again with more noise: The shades on the right are lighter, which shows a bias towards noise, independently from the underlying relationship.

While *HiCS*, *UDS*, *MAC* and *CMI* are expected to be in $[0, 1]$, the theoretical maximum or minimum is never reached, even if our benchmark features both strong and weak dependencies. On the other hand, *MWP* and *MS* exploit all the values of their range, being $[0.5, 1]$ and $[0, 1]$ respectively. Thus, they are interpretable (**R6**).

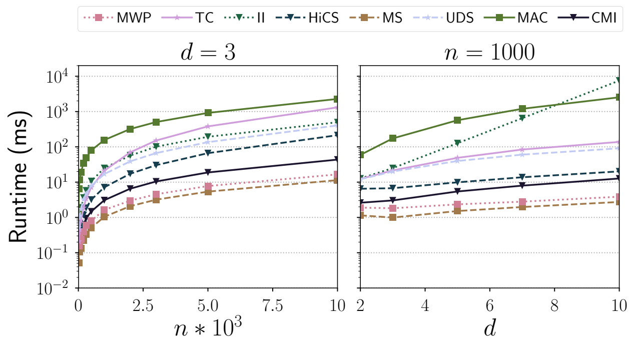


Figure 5: Execution time w.r.t. *n* and *d*

Figure 4 reveals that *MWP*, *TC* and *HiCS* achieve high power in any situation up to a certain extent of noise. *MWP* shows slightly more power with **C**, **H**, **Hc**, **HcG**, **Hs** and **St**. *II* can detect almost every dependency, but the power decreases rapidly with noise and dimensionality. *MS* detects **Dl**, **L**, **P**, **S1**, **S5** and **Zi**, but misses all other dependencies. *MAC* looks unstable, since its power evolves in a non-monotonous way and decreases with increasing dimensionality by much. In fact, it is not able to detect most dependencies for $d = 5$. *UDS* can only detect **L**, **P** and **S1**, a clear limitation. *CMI* has maximal power for each dependency and noise level for $d = 3$, which is unrealistic: *CMI* reaches its lowest score against the noiseless **I**, our baseline for power.

4.3 Scalability

We measure the average CPU time for each estimator against 500 independent data sets with growing n and d. Note that which data set we use only has a marginal effect on the measured time. For consistency, we use instantiations of **I** for every estimator. Figure 5 graphs the results. As we can see, *MWP* is the second fastest after *MS*. *HiCS* and *CMI* scale relatively well with n and d. There is a second group formed by *TC*, *II* and *UDS* one order of magnitude slower. However, *II* does not scale well with d. *MAC* is way behind all others. One should note that the runtime of *MWP* can be further improved via parallelisation and prior indexing.

4.4 Discussion

Our study shows that *MWP* fulfils all our requirements. We have compared *MWP* to a range of multivariate (**R1**) and non-parametric (**R5**) approaches. We have shown to which extent they are efficient (**R2**), general-purpose (**R3**), interpretable (**R6**), sensitive (**R7**) and robust (**R8**). Each approach, except *MWP* and *MS*, has at least one unintuitive parameter (**R4**): *TC* and *II* require $k \in \mathbb{N}$, *CMI* requires $Q \in \mathbb{N}$, *MAC* requires $\epsilon \in (0, 1)$, *UDS* requires $\beta \in \mathbb{N}$, *HiCS* requires $\alpha \in (0, 1)$. Next, only *MWP* and *HiCS* allow to trade accuracy for a computational advantage (**R9**). Table 1 summarizes our findings.

All in all, *MWP* is a state-of-the-art estimator: It is versatile, allowing quality-runtime trade-offs and parallelisation, which is useful when time is critical, e.g., in large data streams. At the same time, it shows excellent detection quality with no restriction on the dependency type, while being easy to use and interpret. *MWP* features a blend of properties that so far no competitor offers.

Table 1: Requirement fulfilment

Estimator	R1	R2	R3	R4	R5	R6	R7	R8	R9
MS	✓	++	✗	✓	✓	✓	✗	✓	✗
TC	✓	-	✓	✗	✓	✗	✓	✗	✗
II	✓	--	✗	✗	✓	✗	✗	✗	✗
CMI	✓	+	✗	✗	✓	✗	✗	✗	✗
MAC	✓	--	✗	✗	✓	✗	✗	✓	✗
UDS	✓	-	✗	✗	✓	✗	✗	✓	✗
HiCS	✓	+	✓	✗	✓	✗	✗	✗	✓
MWP	✓	++	✓	✓	✓	✓	✓	✓	✓

5 CONCLUSIONS & OUTLOOK

In this paper, we have introduced *MCDE*, a framework to estimate multivariate dependency, and its instantiation as *MWP*. We have shown that *MWP* fulfils all the requirements one would expect from a state-of-the-art dependency estimator. Compared to other approaches, it provides high statistical power on a large panel of dependencies, while being very efficient. Thus, *MCDE*/*MWP* is particularly promising for correlation monitoring in data streams.

As future work, we will study the deployment of *MCDE* in streaming scenarios. Our goal is to characterize the **anytime flexibility** of *MCDE* by refining the bound presented in Theorem 2 via further assumptions. It will also be interesting to consider different instantiations of the statistical test, e.g., by comparing recent modifications of the *Mann-Whitney U* test, such as [7] and [3]. Finally, the efficiency of *MCDE* in the streaming setting could be further improved via efficient insert and delete index operations.

ACKNOWLEDGMENTS

This work was supported by the DFG Research Training Group 2153: 'Energy Status Data – Informatics Methods for its Collection, Analysis and Exploitation' and the German Federal Ministry of Education and Research (BMBF) via Software Campus (01IS17042).

REFERENCES

[1] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, R. Devon Hjelm, and Aaron C. Courville. 2018. Mutual Information Neural Estimation. In *ICML (Proceedings of Machine Learning Research)*, Vol. 80. PMLR, 530–539. http://proceedings.mlr.press/v80/belghazi18a.html

[2] Richard E. Bellman. 1957. *Dynamic Programming*. Princeton University Press.

[3] Edgar Brunner and Ullrich Munzel. 2000. The Nonparametric Behrens-Fisher Problem: Asymptotic Theory and a Small-Sample Approximation. *Biometrical journal* 42, 1 (2000), 17–25. https://doi.org/10.1002/(SICI)1521-4036(200001)42:1<17::AID-BIMJ17>3.0.CO;2-U

[4] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. 1996. Data Mining: An Overview from a Database Perspective. *IEEE Trans. Knowl. Data Eng.* 8, 6 (1996), 866–883. https://doi.org/10.1109/69.553155

[5] Antonio Di Crescenzo and Maria Longobardi. 2009. On Cumulative Entropies. *Journal of Statistical Planning and Inference* 139, 12 (2009), 4072–4087.

[6] Wilfrid J Dixon. 1954. Power Under Normality of Several Nonparametric Tests. *The Annals of Mathematical Statistics* 25, 3 (1954), 610–614. https://www.jstor.org/stable/2236846

[7] Michael A. Fligner and George E. Policello II. 1981. Robust Rank Procedures for the Behrens-Fisher Problem. *J. Amer. Statist. Assoc.* 76, 373 (1981), 162–168. https://doi.org/10.1080/01621459.1981.10477623

[8] Arthur Gretton, Kenji Fukumizu, Choon Hui Teo, Le Song, Bernhard Schölkopf, and Alexander J. Smola. 2007. A Kernel Statistical Test of Independence. In *NIPS*. Curran Associates, Inc., 585–592. http://papers.nips.cc/paper/3201-a-kernel-statistical-test-of-independence

[9] Mark A. Hall and Geoffrey Holmes. 2003. Benchmarking Attribute Selection Techniques for Discrete Class Data Mining. *IEEE Trans. Knowl. Data Eng.* 15, 6 (2003), 1437–1447. https://doi.org/10.1109/TKDE.2003.1245283

[10] Wassily Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *J. Amer. Statist. Assoc.* 58, 301 (1963), 13–30. https://doi.org/10.1080/01621459.1963.10500830

[11] Fabian Keller, Emmanuel Müller, and Klemens Böhm. 2012. HiCS: High Contrast Subspaces for Density-Based Outlier Ranking. In *ICDE*. IEEE Computer Society, 1037–1048. https://doi.org/10.1109/ICDE.2012.88

[12] Justin B. Kinney and Gurinder S. Atwal. 2014. Equitability, mutual information, and the maximal information coefficient. *Proceedings of the National Academy of Sciences* 111, 9 (2014), 3354–3359. https://doi.org/10.1073/pnas.1309933111

[13] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. 2004. Estimating mutual information. *Phys. Rev. E* 69 (Jun 2004), 066138. Issue 6. https://doi.org/10.1103/PhysRevE.69.066138

[14] David López-Paz, Philipp Hennig, and Bernhard Schölkopf. 2013. The Randomized Dependence Coefficient. In *NIPS*. Curran Associates, Inc., 1–9. http://papers.nips.cc/paper/5138-the-randomized-dependence-coefficient

[15] Henry B Mann and Donald R Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* 18, 1 (1947), 50–60. https://www.jstor.org/stable/2236101

[16] William J. McGill. 1954. Multivariate information transmission. *Trans. of the IRE Professional Group on Information Theory (TIT)* 4 (1954), 93–111. https://doi.org/10.1109/TIT.1954.1057469

[17] Alexander M. Mood. 1954. On the Asymptotic Efficiency of Certain Nonparametric Two-Sample Tests. *The Annals of Mathematical Statistics* 25, 3 (1954), 514–522. https://www.jstor.org/stable/2236833

[18] Hoang Vu Nguyen, Panagiotis Mandros, and Jilles Vreeken. 2016. Universal Dependency Analysis. In *SDM*. SIAM, 792–800. https://doi.org/10.1137/1.9781611974348.89

[19] Hoang Vu Nguyen, Emmanuel Müller, Jilles Vreeken, Pavel Efros, and Klemens Böhm. 2014. Multivariate Maximal Correlation Analysis. In *ICML (JMLR Workshop and Conference Proceedings)*, Vol. 32. JMLR.org, 775–783. http://jmlr.org/proceedings/papers/v32/nguyenc14.html

[20] Hoang Vu Nguyen, Emmanuel Müller, Jilles Vreeken, Fabian Keller, and Klemens Böhm. 2013. CMI: An information-theoretic contrast measure for enhancing subspace cluster and outlier detection. *SDM* (2013), 198–206. https://doi.org/10.1137/1.9781611972832.22

[21] David N. Reshef, Yakir A. Reshef, Hilary K. Finucane, Sharon R. Grossman, Gilean McVean, Peter J. Turnbaugh, Eric S. Lander, Michael Mitzenmacher, and Pardis C. Sabeti. 2011. Detecting Novel Associations in Large Data Sets. *Science* 334, 6062 (2011), 1518–1524. https://doi.org/10.1126/science.1205438

[22] Friedrich Schmid and Rafael Schmidt. 2007. Multivariate extensions of Spearman's rho and related statistics. *Statistics & Probability Letters* 77, 4 (2007), 407–416. https://doi.org/10.1016/j.spl.2006.08.007

[23] Erich Schubert, Alexander Koos, Tobias Emrich, Andreas Züfle, Klaus Arthur Schmid, and Arthur Zimek. 2015. A Framework for Clustering Uncertain Data. *PVLDB* 8, 12 (2015), 1976–1979. https://doi.org/10.14778/2824032.2824115

[24] Charles Spearman. 1904. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology* 15, 1 (1904), 72–101. https://doi.org/10.2307/1412159

[25] Gábor J. Székely and Maria L. Rizzo. 2009. Brownian Distance Covariance. *Ann. Appl. Stat.* 3, 4 (2009), 1236–1265. https://doi.org/10.1214/09-AOAS312

[26] Nicholas Timme, Wesley Alford, Benjamin Flecker, and John M. Beggs. 2014. Synergy, Redundancy, and Multivariate Information Measures: An Experimentalist's Perspective. *Journal of Computational Neuroscience* 36, 2 (2014), 119–140. https://doi.org/10.1007/s10827-013-0458-4

[27] Satosi Watanabe. 1960. Information Theoretical Analysis of Multivariate Correlation. *IBM Journal of Research and Development* 4, 1 (1960), 66–82. https://doi.org/10.1147/rd.41.0066

[28] Yunyue Zhu and Dennis E. Shasha. 2002. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. In *VLDB*. Morgan Kaufmann, 358–369. http://dl.acm.org/citation.cfm?id=1287369.1287401

A FORMAL PROOFS

A.1 Lemma 1

Lemma 1. *The independence assumption $\mathcal{A}$ of a subspace S holds if and only if the joint pdf for all $S' \in \mathcal{P}(S)$ is equal to its joint conditional pdf given all other variables $S \setminus S'$:*

$$\mathcal{A}(S) \Leftrightarrow p(X_{S'}|\overline{X_{S'}}) = p(X_{S'}) \qquad \forall S' \in \mathcal{P}(S) \tag{3}$$

Proof. Since all variables in S are mutually independent, for any subspace $S' \in \mathcal{P}(S)$ we also have $p(X_{S'}) = \prod_{s_i \in S'} p_{s_i}(X)$:

$$\mathcal{A}(S) \Leftrightarrow \; p(X) = \prod_{s_i \in S} p_{s_i}(X)$$

$$\mathcal{A}(S) \Leftrightarrow \; p(X) = p(X_{S'}) * \prod_{s_i \in S \setminus S'} p_{s_i}(X) \qquad \forall S' \in \mathcal{P}(S)$$

$$\mathcal{A}(S) \Leftrightarrow \; \frac{p(X)}{p(\overline{X_{S'}})} = p(X_{S'}) \qquad \forall S' \in \mathcal{P}(S)$$

By the definition of the conditional pdf:

$$\mathcal{A}(S) \Leftrightarrow \; p(X_{S'}|\overline{X_{S'}}) = p(X_{S'}) \qquad \forall S' \in \mathcal{P}(S) \qquad \square$$

A.2 Lemma 2

Lemma 2 (Independence Assumption Relaxation). *$\mathcal{A}(S) \Rightarrow \mathcal{A}^*(S)$, i.e., we can relax $\mathcal{A}$ into $\mathcal{A}^*$ for any subspace S.*

Proof. Using Lemma 1:

$$\mathcal{A}(S) \Leftrightarrow p(X_{S'}|\overline{X_{S'}}) = p(X_{S'}) \qquad \forall S' \in \mathcal{P}(S)$$
$$\mathcal{A}(S) \Rightarrow p(X_{S^1}|\overline{X_{S^1}}) = p(X_{S^1}) \qquad \forall S^1 \in \mathcal{P}(S) : |S^1| = 1$$
$$\mathcal{A}(S) \Rightarrow p_{s_i}(X|\overline{X_{s_i}}) = p_{s_i}(X) \qquad \forall s_i \in S \qquad \square$$

A.3 Lemma 3

Lemma 3 ($\mathcal{A}^*$ and Conditional Distributions).

$$\mathcal{A}^*(S) \Leftrightarrow p_{s_i|c_i} = p_{s_i} \qquad \forall s_i \in S, \forall c_i \in \mathcal{P}^c(S) \tag{8}$$

Proof. By contradiction, using Lemma 2.

'$\Leftarrow$': From Lemma 2, assume $\mathcal{A}^*(S)$ and that

$$\exists s_j \in S : p_{s_j}(X|\overline{X_j}) \neq p_{s_j}$$
$$\Rightarrow \exists c_j \in \mathcal{P}^c(S) : p_{s_j|c_j} \neq p_{s_j}$$
$$\Rightarrow \text{Contradiction of Lemma 3}$$

'$\Rightarrow$': From Lemma 3, assume $\mathcal{A}^*(S)$ and that

$$\exists s_j \in S, \exists c_j \in P^c(S) : p_{s_j|c_j} \neq p_{s_j}$$
$$\Rightarrow p_{s_j}(X|\overline{X_{s_j}}) \neq p_{s_j}$$
$$\Rightarrow \text{Contradiction of Lemma 2} \qquad \square$$

A.4 Theorem 1

Theorem 1 ($\mathcal{A}^*$ and Complementary Conditions).

$$\mathcal{A}^*(S) \Leftrightarrow p_{s_i|\bar{c}_i} = p_{s_i|c_i} \qquad \forall s_i \in S, \forall c_i \in \mathcal{P}^c(S) \tag{10}$$

Proof. By contradiction, using Lemma 3.

'$\Leftarrow$': From Lemma 3, assume $\mathcal{A}^*(S)$ and that

$$\exists s_j \in S, \exists c_j \in P^c(S) : p_{s_j|c_j} \neq p_{s_j}$$

since $p_{s_j} = p_{s_j|c_j \cup \bar{c}_j}$,

$$\Rightarrow \exists s_j \in S, \exists c_j \in P^c(S) : p_{s_j|c_j} \neq p_{s_j|c_j \cup \bar{c}_j}$$
$$\Rightarrow \exists s_j \in S, \exists c_j \in P^c(S) : p_{s_j|c_j} \neq p_{s_j|\bar{c}_j}$$
$$\Rightarrow \text{Contradiction of Theorem 1}$$

'$\Rightarrow$': From Theorem 1, assume $\mathcal{A}^*(S)$ and that

$$\exists s_j \in S, \exists c_j \in P^c(S) : p_{s_j|c_j} \neq p_{s_j|\bar{c}_j}$$
$$\Rightarrow \exists s_j \in S, \exists c_j \in P^c(S) : p_{s_j|c_j \cup c_j} \neq p_{s_j|\bar{c}_j \cup c_j}$$

since $c_j \cup c_j = c_j$ and $p_{s_j} = p_{s_j|\bar{c}_j \cup c_j}$,

$$\Rightarrow \exists s_j \in S, \exists c_j \in P^c(S) : p_{s_j|c_j} \neq p_{s_j}$$
$$\Rightarrow \text{Contradiction of Lemma 3} \qquad \square$$

A.5 Theorem 2

Theorem 2 (Hoeffding's Bound of $\hat{C}$).

$$\Pr\left(|\hat{C} - C| \geq \varepsilon\right) \leq 2e^{-2M\varepsilon^2} \tag{13}$$

where M is the number of MC iterations, and $0 < \varepsilon < 1 - C$.

Proof. Let us first restate Theorem 1 from Hoeffding [10]:

Let $X_1, X_2, \ldots, X_n$ be independent random variables $0 \leq X_i \leq 1$ for $i \in \{1, \ldots, n\}$ and let $\bar{X} = \frac{1}{n}(X_1 + X_2 + \cdots + X_n)$ be their mean with expected value $E[\bar{X}]$. Then, for $0 < t < 1 - E[\bar{X}]$:

$$\Pr\left(\bar{X} - E[\bar{X}] \geq t\right) \leq e^{-2nt^2} \quad \Pr\left(\bar{X} - E[\bar{X}] \leq t\right) \leq e^{-2nt^2} \tag{14}$$

We can treat each MC iteration $m_1, m_2, \ldots, m_M$ as i.i.d. random variables $X_{m_1}, X_{m_2}, \ldots, X_{m_M}$ in $[0, 1]$ with mean $\hat{C}$ and expected value $E[\hat{C}] = C$ (cf. Definition 8). Thus, for $0 < \varepsilon < 1 - C$, we have $\Pr(|\hat{C} - C| \geq \varepsilon) \leq 2e^{-2M\varepsilon^2}$. □

B ALGORITHMS

B.1 Pseudo-code for the Index Construction

We outline the construction of the index in Algorithm 2. The index $\mathcal{I}$ is a one-dimensional structure containing the adjusted ranks and tying values corrections for each dimension. It consists of $|S|$ elements $\{I_1, \ldots, I_{|S|}\}$, where I_i is an array of 3-tuple $[(l_1^i, a_1^i, b_1^i), \ldots, (l_n^i, a_n^i, b_n^i)]$ ordered by s_i in ascending order. In this tuple, l^i are the row numbers of the values of s_i, a^i are the *adjusted* ranks and b^i the accumulated correction term of the standard deviation σ. We denote $I_i[j]$, $s_i[j]$ as the j-th elements of I_i and s_i; we refer to the components of $I_i[j]$ as l_j^i, a_j^i, b_j^i.

For each attribute s_i, we sort the values (Line 3) and perform a single pass over the sorted list to adjust the ranks and the correction of the standard deviation for ties. Thus, the index construction complexity is in $O(|S| \cdot (n \cdot log(n) + n))$.

Algorithm 2 ConstructIndex($S = \{s_i\}_{i \in \{1,\ldots,d\}}$)

```
1:  for $i = 1$ to $|S|$ do
2:      $r^i \leftarrow [0, \ldots, n-1]$
3:      $l^i \leftarrow$ sort $r^i$ by $s_i$ in ascending order
4:      $I_i \leftarrow \left[(l_1^i, r_1^i), \ldots, (l_n^i, r_n^i)\right]$
5:      $j \leftarrow 1$ ; $correction \leftarrow 0$
6:      while $j \leq n$ do
7:          $k \leftarrow j$ ; $t \leftarrow 1$ ; $adjust \leftarrow 0$
8:          while $(k < n-1) \wedge (s_i[l_k^i] = s_i[l_{k+1}^i])$ do
9:              $adjust \leftarrow adjust + r_k^i$
10:             increment $k$ and $t$
11:         if $k > j$ then
12:             $adjusted \leftarrow (adjust + r_k^i)/t$
13:             $correction \leftarrow correction + t^3 - t$
14:             for $m \leftarrow j$ to $k$ do
15:                 $I_i[m] \leftarrow (l_m^i, adjusted, correction)$
16:         else
17:             $I_i[j] \leftarrow (l_j^i, r_j^i, correction)$
18:         $j \leftarrow j + t$
19: return $\mathcal{I} : \{I_1, \ldots, I_{|S|}\}$ with $I_i : (l^i, a^i, b^i)$
```

B.2 Pseudo-code for Slicing

Algorithm 3 is the pseudo-code of the slicing process. We can slice the input data efficiently, because the tuples are already sorted in the index structure. We successively mask the row numbers based on a random condition for all but one reference attribute s_r. Since we visit each value at most once, the complexity is in $O(|S| \cdot n)$.

Algorithm 3 Slice($\mathcal{I} : \{I_1, \ldots .I_{|S|}\}, r$)

```
1: $slice \leftarrow$ Array of $n$ boolean values initialized to $true$
2: $slicesize \leftarrow \lceil n \cdot \sqrt[|S|-1]{\alpha} \rceil$
3: for $I_i \in \mathcal{I} \setminus I_r$ do
4:     $start \leftarrow$ random integer in $[1, n - slicesize]$
5:     $end \leftarrow start + slicesize$
6:     for $j \leftarrow 1$ until $start$ and $end + 1$ to $n$ do
7:         $slice[l_j^i] \leftarrow false$
8: return $slice$
```

B.3 Pseudo-code for the Statistical Test

Algorithm 4 is our approach to compute the statistical test. We determine a restriction $[start, end]$ on s_r and sum the adjusted ranks of the objects that belong to the slice. Thanks to the marginal restriction, we compute the statistical test in a subsample of size $n' < n$. Since the ranks in this subset may not start from 0, we adjust the sum of the ranks R_1 (Line 10). Then, we compute a correction term (Line 13) using the cumulative correction b^r to adjust σ for ties (Line 14). $\Phi^{1/2}$ is the cumulative distribution function of the half-Gaussian distribution. We compute the statistical test via a single pass, considering only elements between $start$ and end. Each operation requires constant time, so the complexity is in $O(n)$.

Algorithm 4 U-test($\mathcal{I} : \{I_1, \ldots .I_{|S|}\}, slice, r$)

```
1:  $start \leftarrow$ random integer in $[1, n \cdot (1 - \alpha)]$
2:  $end \leftarrow start + \lceil n \cdot \alpha \rceil$
3:  $R_1 \leftarrow 0$ ; $n_1 \leftarrow 0$
4:  for $j \leftarrow start$ to $end$ do
5:      if $slice[l_j^r] = true$ then
6:          $R_1 \leftarrow R_1 + a_j^r$
7:          $n_1 \leftarrow n_1 + 1$
8:  $n' \leftarrow end - start$
9:  if $n_1 = 0$ or $n_1 = n'$ then
10:     return 1
11: $U_1 \leftarrow R_1 - start \cdot n_1$
12: $n_2 \leftarrow n' - n_1$
13: $\mu \leftarrow (n_1 \cdot n_2)/2$
14: $correction \leftarrow (b_{end-1}^r - b_{start-1}^r)/(n' \cdot (n' - 1))$
15: $\sigma \leftarrow \sqrt{((n_1 \cdot n_2)/12) \cdot (n' + 1 - correction)}$
16: return $\Phi^{1/2}(|U_1 - \mu|/\sigma)$
```

B.4 Pseudo-code for Dependency Generation

We show in Algorithm 5 to 15 how to generate d-dimensional points for each dependency displayed in Figure 2. $\mathcal{U}[a, b]$ stands for the uniform distribution over the interval $[a, b]$, $\mathcal{N}(\mu, \sigma)$ refers to the normal distribution with mean μ and standard deviation σ and $\mathcal{B}(\mu)$ is a Bernoulli experiment with parameter μ. Let the notation $x \leftarrow \mathcal{U}[a,b]$ means drawing a value randomly from $\mathcal{U}[a,b]$.

Since each points p are generated independently from each other, it is enough to run the procedure n times to generate a d-dimensional data set with n points for a given dependency. Note that each dependency generator are scaled to $[0,1]$. We package our algorithms for data generation as an independent GitHub project[2].

Algorithm 5 CROSS(d)

```
1: $x \leftarrow \mathcal{U}[0,1]$ ; $p \leftarrow (x)$
2: for $j \leftarrow 2$ to $d$ do
3:     if $\mathcal{B}(0.5) = 0$ then $p \leftarrow p \cup (x)$
4:     else $p \cup (1-x)$
5: return $p$
```

Algorithm 6 DOUBLELINEAR(d)

```
1: $x \leftarrow \mathcal{U}[0,1]$ ; $p \leftarrow (x)$
2: for $j \leftarrow 2$ to $d$ do
3:     if $\mathcal{B}(0.5) = 0$ then $p \leftarrow p \cup (x)$
4:     else $p \leftarrow p \cup (x * 0.25)$
5: return $p$
```

Algorithm 7 HOURGLASS(d)

```
1: $x \leftarrow \mathcal{U}[0,1]$ ; $p \leftarrow (x)$
2: for $j \leftarrow 2$ to $d$ do
3:     $y \leftarrow$ random integer in $[1,4]$
4:     if $y = 1$ then $p \leftarrow p \cup (x)$
5:     else if $y = 2$ then $p \leftarrow p \cup (1-x)$
6:     else if $y = 3$ then $p \leftarrow p \cup (0)$
7:     else $p \leftarrow p \cup (1)$
8: return $p$
```

Algorithm 8 HYPERCUBE(d)

```
1: $flag \leftarrow$ random integer in $[1,d]$ ; $p \leftarrow ()$
2: for $j \leftarrow 1$ to $d$ do
3:     if $j = flag$ then $p \leftarrow p \cup (\mathcal{B}(0.5))$
4:     else $p \leftarrow p \cup (\mathcal{U}[0,1])$
5: return $p$
```

C EXPERIMENTS

C.1 Details on the Statistical Power

DEFINITION 11 (POWER). *The power of an estimator $\mathcal{E}$ w.r.t. $\boldsymbol{X}$ with σ, n and d is the probability of the score of $\mathcal{E}$ to be larger than a γ-th percentile of the scores w.r.t. $\boldsymbol{I}$:*

$$\Pr\left(\mathcal{E}\left(Inst^{\boldsymbol{X},\sigma}_{n\times d}\right) > \left\{\mathcal{E}\left(Inst^{\boldsymbol{I},0}_{n\times d}\right)\right\}^{P_\gamma}\right) \tag{15}$$

[2] https://github.com/edouardfouche/DataGenerator

Algorithm 9 HYPERCUBEGRAPH(d)

```
1: $flag \leftarrow$ random integer in $[1,d]$ ; $p \leftarrow ()$
2: for $j \leftarrow 1$ to $d$ do
3:     if $j = flag$ then $p \leftarrow p \cup (\mathcal{U}[0,1])$
4:     else $p \leftarrow p \cup (\mathcal{B}(0.5))$
5: return $p$
```

Algorithm 10 HYPERSPHERE(d)
* The procedure used here is known as Marsaglia's algorithm (1972)

```
1: $r \leftarrow 0$ ; $p \leftarrow ()$
2: for $j \leftarrow 1$ to $d$ do
3:     $v \leftarrow \mathcal{N}(0,1)$
4:     $p \leftarrow p \cup (v)$
5:     $r \leftarrow r + v^2$
6: for $j \leftarrow 1$ to $d$ do $p(j) \leftarrow p(j)/r^2 + 0.5$
7: return $p$
```

Algorithm 11 LINEAR(d)

```
1: $x \leftarrow \mathcal{U}[0,1]$ ; $p \leftarrow (x)$
2: for $j \leftarrow 2$ to $d$ do $p \leftarrow p \cup (x)$
3: return $p$
```

Algorithm 12 PARABOLIC(d)

```
1: $x \leftarrow \mathcal{U}[-1,1]$ ; $t \leftarrow (1)$ ; $p \leftarrow (x)$
2: for $j \leftarrow 2$ to $d$ do
3:     $p \leftarrow p \cup \sum[p]^2$ ; $t \leftarrow t \cup \sum[t]^2$
4: for $j \leftarrow 1$ to $d$ do $p(j) \leftarrow p(j)/t(d)$
5: return $p$
```

Algorithm 13 SINE(d, P)
* P is the number of periods of the sinusoid

```
1: $x \leftarrow \mathcal{U}[0,1]$ ; $p \leftarrow (x)$
2: for $j \leftarrow 2$ to $d$ do $p \leftarrow p \cup (\sin(\sum[p]) * 2\pi * P) + 1)/2$
3: return $p$
```

Algorithm 14 STAR(d)

```
1: $x \leftarrow \mathcal{U}[0,1]$ ; $p \leftarrow (x)$
2: for $j \leftarrow 2$ to $d$ do
3:     if $\mathcal{B}(0.5) = 0$ then $p \leftarrow p \cup (x)$
4:     else $p \leftarrow p \cup (1-x)$
5: if $\mathcal{B}(0.5) = 0$ then
6:     $flag \leftarrow$ random integer in $[1,d]$
7:     $p(flag) \leftarrow 0.5$
8: return $p$
```

Algorithm 15 ZINVERSED(d)

```
1: $x \leftarrow \mathcal{U}[0,1]$ ; $p \leftarrow (x)$
2: for $j \leftarrow 2$ to $d$ do
3:     $y \leftarrow$ random integer in $[1,3]$
4:     if $y = 1$ then $p \leftarrow p \cup (x)$
5:     else if $y = 2$ then $p \leftarrow p \cup (0)$
6:     else $p \leftarrow p \cup (1)$
7: return $p$
```

$Inst^{\mathbf{X},\sigma}_{n\times d}$ is a random instantiation of a subspace as dependency $\mathbf{X}$ with noise level σ, which has n objects and d dimensions. $\{x\}^{P_\gamma}$ stands for the γ-th percentile of the set $\{x\}$, i.e., a value v such that $\gamma\%$ of the values in $\{x\}$ are smaller than v.

Note that, since the attributes of $\mathbf{I}$ are independent, adding noise does not have any effect on dependence, so we set noise to 0 when instantiating $\mathbf{I}$. To estimate the power, we draw two sets of 500 estimates from $\mathbf{X}$, σ and $\mathbf{I}$ respectively:

$$\Sigma^{\mathcal{E}}_{\mathbf{X},\sigma} : \left\{\mathcal{E}\left(Inst^{\mathbf{X},\sigma}_{n\times d}\right)\right\}^{500}_{i=1} \qquad \Sigma^{\mathcal{E}}_{\mathbf{I}} : \left\{\mathcal{E}\left(Inst^{\mathbf{I},0}_{n\times d}\right)\right\}^{500}_{i=1}$$

Then, we count the elements in $\Sigma^{\mathcal{E}}_{\mathbf{X},\sigma}$ greater than $\left\{\Sigma^{\mathcal{E}}_{\mathbf{I}}\right\}^{P_\gamma}$:

$$power^{\mathbf{X},\sigma}_{n\times d,\gamma}(\mathcal{E}) = \frac{\left|\left\{x : x \in \Sigma^{\mathcal{E}}_{\mathbf{X},\sigma} \ \wedge \ x > \left\{\Sigma^{\mathcal{E}}_{\mathbf{I}}\right\}^{P_\gamma}\right\}\right|}{500}$$

One can interpret *power* as the probability to correctly reject the independence hypothesis with $\gamma\%$ confidence. In other words, the power quantifies how well a dependency measure, such as *MWP*, can differentiate between the independence $\mathbf{I}$ and a given dependency $\mathbf{X}$ with noise level σ. For our experiments, we choose $\gamma = 95$. In the case of *II*, values can be negative or positive, depending on whether the dependency is a 'synergy' or a 'redundancy'. For *II*, we measure power using the absolute value of its score.

C.2 Influence of size n and parameter M

Figure 6 shows that power globally increases with n, but it is still high for most dependencies with low n, provided noise is moderate. As we can see, the average score of *MWP* tends to increase with n, which explains the gain in power. In fact, that is because *MWP* is **sensitive** (**R7**), as we discuss in Section C.4. Similarly, power increases slightly as M increases, but the effect is visible only for **S5** and **Zi**. This increase of power is easily explained by the fact that the standard deviation of *MWP* decreases, which is what Theorem 2 predicted: with more iterations, the values concentrate around C.

In the end, we see that *MWP* is already useful for small n or small M, even though more iterations or more data samples yield higher power when data is noisy.

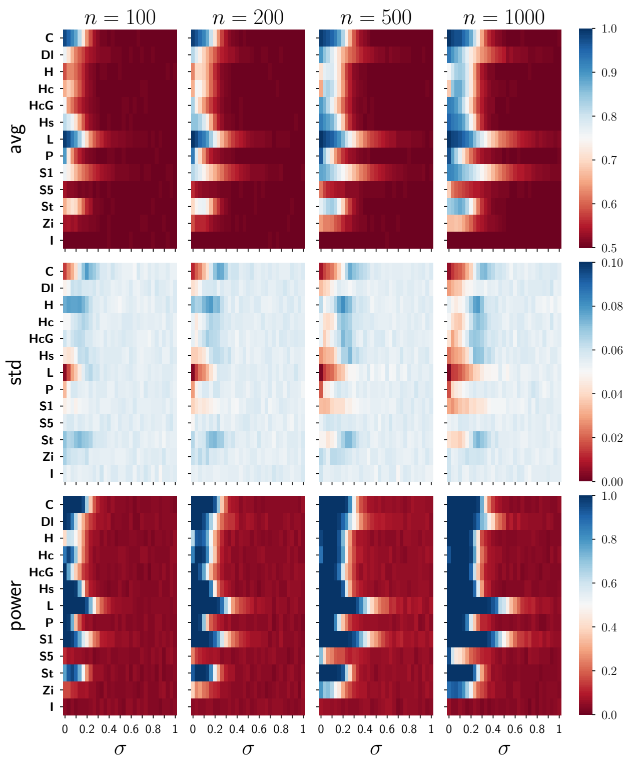
C.3 Influence of dimensionality d

Figure 7 graphs the evolution of *MWP* for $d = 2, 3, 5$. As we see, the average *MWP* decreases gradually for each dependency. The same level of noise does not seem to affect each estimate equally, also regarding dimensionality. For instance, the estimates of **L**, **P** and **S1** are larger at $d = 2$. While the estimates of **Hc**, **HcG**, **P** and **Zi** decrease with increasing d, they increase for **C** and **St**.

The standard deviation of *MWP* increases with noise and decreases with d. In particular, **L**, **C** and **Hs** have a low standard deviation. This means that fewer iterations are in fact required to estimate stronger dependencies.

The statistical power does not seem to vary much with dimensionality for most dependencies. It decreases with d for **Hc**, **HcG**, **Hs**, **P** and **Zi**, while it increases for **C**, **S5** and **St**.

All in all, each dependency yields a score larger than the independence **I** up to a certain level of noise, leading to high power. This indicates that *MWP* is **general-purpose** (**R3**).



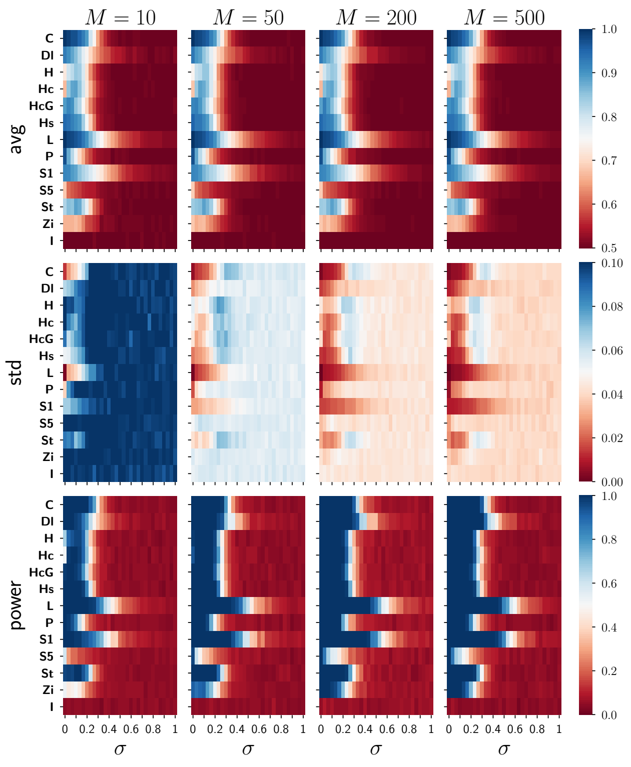


Figure 6: Power of *MWP* w.r.t. n and M

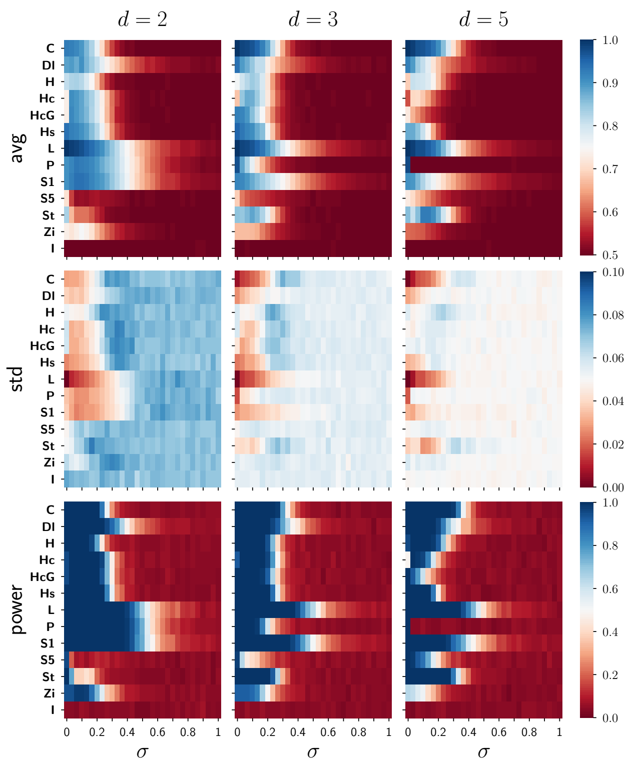


Figure 7: *MWP* w.r.t. dimensionality d

C.4 Sensitivity

R7 states that estimators should reflect the strength of the observed effect w.r.t. the number of observations. Figure 8 graphs the average score from 500 instances of each dependency with a minimal noise of 1/30. The average of *MWP* obtained for each dependency converges to 1 consistently with more samples, except for **I**, which stabilizes around 0.5. This means that *MWP* is **sensitive** (**R7**).

TC behaves similarly to *MWP*. However, it is not bounded. While the scores of *II* seem to increase with sample size, they decrease in terms of absolute value. *MS* is insensitive to changes of sample size. *HiCS*, *UDS*, *MAC* and *CMI* behave antagonistically: Their scores tends to go down as the sample size increases, even in the case of **I**. This implies that their minimum or maximum score varies with the sample size, highlighting also interpretability (**R6**) problems.

C.5 Robustness

We simulate data imperfections by discretising a 3-dimensional linear dependency into a number ω of discrete values from 100 to 1. With only one value, the space is completely redundant, i.e., its *contrast* should be minimal. We compare the power of *MWP* and of the other approaches against **L** and **I** for different levels of discretisation. Since *TC* and *II* rely on a nearest neighbours algorithm, they fail when the same observation is present more than k times, i.e., they are by design not robust; we exclude them from the analysis. Figure 9(a) displays the results.

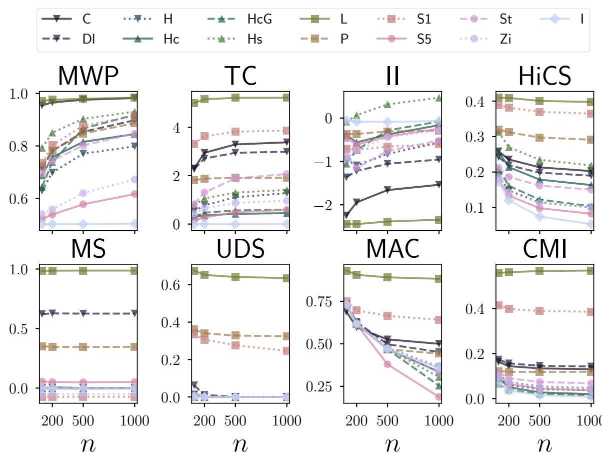
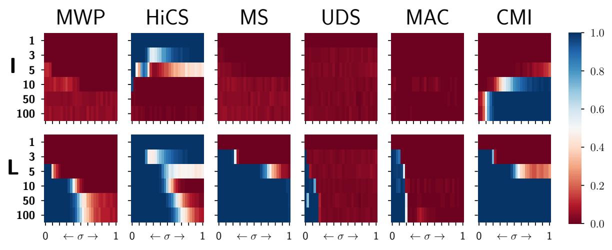


Figure 8: Average score w.r.t. n, $\sigma = 1/30$


(a) Power

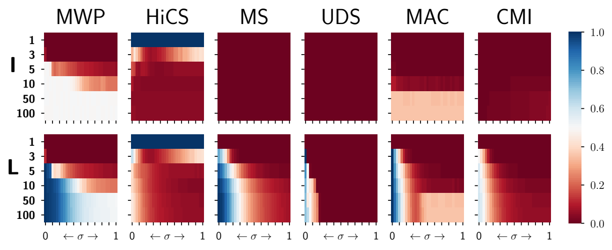

(b) Average

Figure 9: Power and average score w.r.t. ω

HiCS yields high power in the case of discrete values, even for **I**. Thus, *HiCS* is not robust. Also, the power of *CMI* wrongly increases as we add noise to **I**, provided that $\omega \geq 10$. This is why the power of *CMI* is high for every dependency in Figure 4. *CMI* rejects the independence for independent spaces, i.e., it is not robust. On the other hand, *MWP*, *MS*, *UDS* and *MAC* seem **robust** (**R8**).

In Figure 9(b), we see that the score of *CMI* tends to increase slightly for **I** as we add noise, whenever $\omega > 5$. Also, the score of *HiCS* increases for both **I** and **L** when $\omega \leq 5$. *MAC* converges to 0.4 as noise increases for $\omega > 10$. On the other hand, *MWP* converges to 0 as the space becomes discrete. This is an interesting feature of our estimator: discrete spaces are of lower interested, since the notion of *contrast* is not clearly defined there. It allows analysts to draw a line between discrete and real-valued attributes.