Distributed and Parallel Databases
https://doi.org/10.1007/s10619-020-07295-x

A framework for dependency estimation in heterogeneous data streams

Edouard Fouché[1] · Alan Mazankiewicz[1] · Florian Kalinke[1] · Klemens Böhm[1]

Abstract
Estimating dependencies from data is a fundamental task of Knowledge Discovery. Identifying the relevant variables leads to a better understanding of data and improves both the runtime and the outcomes of downstream Data Mining tasks. Dependency estimation from static numerical data has received much attention. However, real-world data often occurs as heterogeneous data streams: On the one hand, data is collected online and is virtually infinite. On the other hand, the various components of a stream may be of different types, e.g., numerical, ordinal or categorical. For this setting, we propose Monte Carlo Dependency Estimation (MCDE), a framework that quantifies multivariate dependency as the average statistical discrepancy between marginal and conditional distributions, via Monte Carlo simulations. MCDE handles heterogeneity by leveraging three statistical tests: the Mann–Whitney U, the Kolmogorov–Smirnov and the Chi-Squared test. We demonstrate that MCDE goes beyond the state of the art regarding dependency estimation by meeting a broad set of requirements. Finally, we show with a real-world use case that MCDE can discover useful patterns in heterogeneous data streams.

Keywords Data Mining · Multivariate Statistics · Exploratory Data Analysis

1 Introduction

The discovery of relationships between attributes is fundamental to many Data Mining applications, e.g., Feature Selection [32], Clustering [31] or Outlier Detection [21], and it is a prominent topic in the database community [6, 18, 46]. Identifying groups of dependent features is critical to deal with high dimensionality and helps to filter out the features irrelevant for the task at hand. One typically

✉ Edouard Fouché
edouard.fouche@kit.edu

Klemens Böhm
klemens.boehm@kit.edu

1 Karlsruhe Institute of Technology (KIT), Am Fasanengarten 5, 76131 Karlsruhe, Germany

estimates the 'relevance' of a feature by estimating its 'dependence' on a given target to predict. To do so, one often leverages well-known 'dependency estimators' such as the *Pearson correlation coefficient*, or *Mutual Information* [37].

Most dependency estimators only deal with numerical data, and one assumes that all relevant observations are available during estimation. In the real world, however, data often consists of an open-ended, ever-evolving stream of measurements or indicators of various types, e.g., numerical, ordinal, or categorical observations. Such data sources are known as *Heterogeneous Data Streams* (H-DS) [45]. Knowledge Discovery in H-DS is known as one of the most challenging problems in Data Mining [8]. The streaming nature of the data constrains system design in several ways, laid out by Domingos and Hulten [9] as follows:

- **Efficiency (C1)** The system must spend a short constant time and a constant amount of memory to process each incoming record.
- **Single Scan (C2)** The system may perform at most one scan over the data since access to past observations is often unavailable or impractical.
- **Adaptation (C3)** Whenever the data distribution changes over time—a phenomenon that is known as 'concept drift' [1]—the system must adapt, e.g., by forgetting outdated information.
- **Anytime (C4)** The system must be available at any point in time, with an output ideally equivalent (or nearly identical) to the one of a non-streaming system, operating without the streaming constraints.

Most approaches designed for streams can only handle numerical data. Since data may be of other types as well, heterogeneity is on our list of constraints:

- **Heterogeneity (C5)** The system must handle not only numerical types but ideally all data types such as strings, categories, ordinal values.

That being said, and orthogonally to the streaming setting, modern dependency estimators have their own desirable features:

- **Multivariate (F1)** 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 generalise bivariate measures lack efficiency or effectiveness [43].
- **General-purpose (F2)** Dependency estimators should not restrict to specific types of dependencies. Otherwise, they may miss relevant attribute relationships. Existing multivariate estimators are typically limited to, say, monotonous or functional dependencies.
- **Intuitive (F3)** A method is intuitive if its parameters are easy to set, i.e., users understand their impact on the estimation. Existing solutions tend to have unintuitive parameters, and the suggestion of 'good' parameter values happens (or does not happen) at the discretion of the inventors.

- **Non-parametric (F4)** 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 miss relevant effects systematically.
- **Interpretable (F5)** 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 and compare two given estimates.
- **Sensitive (F6)** Dependency estimation is not only about detecting the existence of a relationship, 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 observations, as the size of the observed effect—the 'effect size'—is larger.
- **Robust (F7)** Real-world data may be of poor quality. Measuring devices often have limited precision, so that values are rounded or trimmed, leading to points with the same values. It is also common to discretise attributes, for a more compact representation. Such artefacts can have a negative influence on the estimation. Thus, estimators need to be robust against duplicates and imprecision.

To our knowledge, no existing solution satisfies all these requirements. In this paper, we propose an approach that fulfils them all.

2 Contributions

We present Monte Carlo Dependency Estimation (MCDE), a framework which satisfies both the constraints of heterogeneous data streams and the desirable features of dependency estimation. Over a given time window, MCDE estimates the dependency of an attribute set as the average discrepancy between marginal and conditional distributions, via Monte Carlo (MC) simulations. In each MC simulation, MCDE applies a condition on each attribute. Then a statistical test quantifies the discrepancy between the marginal and conditional distributions. We determine a lower bound for the quality of our estimates, which only depends on the number of MC simulations. Such bound allows users to trade estimation accuracy for a computational advantage.

We explore three instantiations of MCDE, i.e., three new dependency measures, dubbed Mann–Whitney-P (MWP), Kolmogorov–Smirnov-P (KSP) and Chi-Squared-P (CSP), which base on the corresponding statistical test. We show that using them in combination allows dealing with heterogeneous data. We describe their implementation and compare them in our experiments.

We introduce index structures for MWP, KSP, and CSP, to speed up contrast estimation. Our indexes support insertion/deletion operations for efficient estimation in streaming settings, e.g., over a sliding window.

We feature a use case against real-world data from Bioliq®, a pyrolitic power plant [33], and show how one can leverage MCDE to discover interesting and useful patterns. We release our source code and experiments on GitHub,[1] with documentation to ensure reproducibility.

This work extends our previous study [14]. A first novelty is that we propose instantiations based on further statistical tests, i.e., KSP and CSP. Next, we explicitly address the H-DS setting: The streaming index operations and our real-world use case are new in this article.

Outline Section 3 reviews the related work. Section 4 describes MCDE and its instantiations as MWP, KSP and CSP and introduces our index operations for the streaming setting. Section 5 evaluates our approach and presents our use case. Section 6 concludes. We refer the reader to [14] for formal proofs and comparison with other methods.

3 Related work

Estimating correlation has been of interest for more than a century. Many bivariate measures exist, e.g., [35, 38]. Some of them also target at quantifying the association between two vectors which are possibly multivariate [2, 17, 25, 39]. However, they can only quantify the dependency between two entities—not between several ones (**F1**). They also may have other drawbacks. The *Pearson correlation coefficient*, for instance, is parametric (**F4**), targets at linear dependencies (**F2**) and is only applicable to numerical data (**C5**).

There are attempts to extend bivariate dependency measures to the multivariate case. Schmid and Schmidt [36] propose an extension of Spearman's ρ to multivariate data, but it is limited to monotonous relationships (**F2**). Several authors also propose multivariate extensions of *Mutual Information* [40]. For example, *Interaction Information* [27] quantifies the 'synergy' or 'redundancy' in a set of variables. Similarly, *Total Correlation* [44] 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, either based on kernels, histograms or local densities, all require to set unintuitive parameters (**F3**) and may be computationally expensive (**C1**). Next, with many attributes, density estimation becomes meaningless due to the *curse of dimensionality* [3]. Information-theoretic measures also are difficult to interpret (**F5**), since they usually correspond to a number of bits or nats, which is theoretically unbounded.

More recently, *Cumulative Mutual Information* [31], *Multivariate Maximal Correlation* [30], and *Universal Dependency Score* [28] were proposed as multivariate dependency measures. They are remotely related to concepts from information theory, as they rely on the so-called *Cumulative Entropy* [7]. However, these measures are computationally expensive (**C1**) and unintuitive (**F3**). They also are difficult to interpret, because their theoretical maximum and minimum vary with the number of attributes (**F5**).

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

Another approach, *High Contrast Subspaces* (HiCS) [21], is somewhat similar to ours. It uses *subspace slicing* as a heuristic to quantify the potential of subspaces to contain outliers. Nevertheless, HiCS only addresses static numerical data, and is not suitable as a dependency estimator.

The current state of the art to handle heterogeneity (**C5**) is to rely on discretisation, using methods such as the one proposed in [12]. Then one can compute an information-theoretic measure, as in [29]. However, any discretisation inevitably results in an information loss and may not work as dimensionality increases.

A new line of work focuses on estimating *Mutual Information* on numerical data streams. MISE [22] is a data summarisation technique to estimate *Mutual Information* over arbitrary time windows. Vollmer et al. [42] provide dynamic data structures to maintain *Mutual Information* over a sliding window. Vollmer and Böhm [41] extend this method to propose an anytime estimator for *Mutual Information*. However, the resulting estimates inherit the qualities and caveats of *Mutual Information*.

4 The MCDE framework

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

4.1 Preliminaries

An H-DS is a set of attributes/dimensions/variables $D = \{s_1, \dots, s_d\}$ and an open list of observations $B = (\mathbf{x}_1, \mathbf{x}_2, \dots)$, where $\mathbf{x}_i = \langle x_{ij} \rangle_{j \in \{1,\dots,d\}}$ is a vector of values with d attributes, and we see an attribute $s_j = (x_{1j}, x_{2j}, \dots)$ as an open list of values. Since the stream is virtually infinite, we use the sliding window model: At any time $t > 1$, we only keep the latest w observations, $W_t = \left(\mathbf{x}_{t-w}, \dots, \mathbf{x}_t\right)$. We assume, without loss of generality, that observations arrive at equidistant time steps. Note that one could easily adapt our methods to other summarisation techniques, such as the landmark window or reservoir sampling [16].

We call a subspace S a projection of the current window W on d' attributes, with $S \subseteq D$ and $d' \leq d$. To formalise our framework, we treat an attribute $s_i \in D$ as a random variable X_{s_i}. We also make the distinction between numerical, ordinal and categorical attributes:

- We say that s_i is of numerical type ($s_i \in Num$) if one can see X_{s_i} as a continuous variable on a given interval.
- We say that s_i is of ordinal type ($s_i \in Ord$) if one can see X_{s_i} as a discrete variable, i.e., it can take a finite number of ordered values.
- We say that s_i is of categorical type ($s_i \in Cat$) if one can see X_{s_i} as a categorical variable, with a fixed number of nominal categories.

Naturally, knowing whether a given attribute is of numerical, ordinal or categorical type requires domain knowledge. Typically, ordinal attributes have many tying values, while values from a numerical attribute are unique, given enough precision. On the other hand, values from categorical attributes might not be numeric and do not have any meaningfully ordering.

$p(X)$ is the joint probability distribution function (*pdf*) of a random vector $X = \langle X_{s_i} \rangle_{s_i \in S}$, and $\hat{p}(X)$ denotes the empirical estimation of this distribution. We use $p_{s_i}(X)$ and $\hat{p}_{s_i}(X)$ for the marginal *pdf* and its estimation for each variable s_i. $\mathcal{P}(S)$ is the power set of S, i.e., the set of all attribute subsets. For any 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 $\overline{X_{S'}} = X_{S \setminus S'} = \langle X_{s_i} \rangle_{s_i \in S \setminus S'}$. In our algorithms, '$\oplus$' and '$\wedge$' stand for concatenation and element-wise logical conjunction.

4.2 Theory behind MCDE

4.2.1 Quantifying dependency via contrast

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

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:

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 $p^e(X)$:

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

The discrepancy is a random variable. While one can estimate it between two probability distributions, for instance, using 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 subspace S states that the joint distribution for all $S' \subset S$ is equal to its conditional distribution on $S \setminus S'$:*

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

See [14] *for proofs of this and the following lemmas.*

Lemma 1 provides an alternative definition of $\mathcal{A}$. However, it still has issues: First, multivariate density estimation is required to estimate $p(X_{S'})$ and $p(X_{S'}|\overline{X_{S'}})$ with $|S'| \geq 1$. 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 distribution of single variables.

Definition 3 *(Relaxed Independence Assumption)* The relaxed independence assumption $\mathcal{A}^*$ of a subspace S states that 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 distribution of s_i:

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

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 the values of all variables but s_i do not reveal any information on s_i.

Next, $\mathcal{A}(S) \Rightarrow \mathcal{A}^*(S)$, 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, namely the discrepancy *disc* of 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 includes a whole class of dependency estimators, e.g., [21], which aim at quantifying the so-called *contrast* of a 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 distributions, i.e., multivariate density estimation is not required. Second, the number of degrees of freedom of $\mathcal{A}^*(S)$ increases linearly with $|S|$, but exponentially for $\mathcal{A}(S)$. Thus, estimating $\mathcal{D}^*$ instead of $\mathcal{D}$ is in line with the strict efficiency requirements of data streams.

By definition, $\mathcal{D}^*$ does not take the dependency between multivariate subsets into account, but only of each variable versus all others. However, we argue that this relaxation is not problematic, and it even supports interpretability. In real-world scenarios, the detection of dependency is only interesting as long as we can observe effects w.r.t. the marginal and conditional distributions: one is typically looking for interpretable influences of particular variables on the system and vice versa [19].

4.2.2 Estimating conditional distributions

The difficulty when estimating $\mathcal{D}^*$ is estimating the conditional distributions, because the underlying data distributions are unknown. As proposed in [21], one can simulate conditional distributions by applying a set of conditions to S, in a process called *subspace slicing*. We handle heterogeneity by differentiating between numerical, ordinal and categorical attributes:

Definition 5 *(Subspace Slice)* A slice c_i in a subspace S w.r.t. attribute s_i is a list of $|S| - 1$ conditions C_j which restricts the values of each $s_j \in S \setminus s_i$:

$$c_i = \left(C_1, \dots, C_{i-1}, C_{i+1}, \dots, C_{|S|}\right), \quad \text{where}$$
$$C_j = \begin{cases} \left[l_j, u_j\right] \; s.t. \; \left|\left\{\mathbf{x}_k : x_{kj} \in \left[l_j, u_j\right]\right\}\right| = w' & \text{if } s_j \in Num \\ \left[l_j \dots u_j\right] \; s.t. \; \left|\left\{\mathbf{x}_k : x_{kj} \in \left[l_j \dots u_j\right]\right\}\right| = w' & \text{if } s_j \in Ord \\ \left\{v_j : v_j \in s_j\right\} \; s.t. \; \left|\left\{\mathbf{x}_k : x_{kj} \in \left\{v_j : v_j \in s_j\right\}\right\}\right| = w' & \text{if } s_j \in Cat \end{cases}$$
$$\forall j \in \{1, \dots, |S|\} \setminus i$$

where $\left[l_j, u_j\right]$ is a continuous interval, $\left[l_j \dots u_j\right]$ is a discrete interval, and $\left\{v_j : v_j \in s_j\right\}$ is a set of values of s_j. $w' < w$ is the number of observations per condition. We call s_i the *reference* attribute, the only attribute without a condition. We write that $\mathbf{x}_k \in c_i$ if $\mathbf{x}_k$ fulfils all the conditions in c_i. We define $\overline{c}_i$ as the complementary slice, i.e., it contains all observations which are not in c_i. $p_{s_i|c_i}(X)$ and $p_{s_i|\overline{c}_i}(X)$ denote the conditional distribution of the observations in the slice c_i and its complement $\overline{c}_i$ respectively. $\mathcal{P}^c(S)$ is the set of all possible slices in subspace S.

We choose each condition in a slice randomly, but so that they contain w' observations. Note that ordinal and categorical attributes (e.g., gender) may have many tying values. In such a case, a random condition might not precisely have w' elements. Our solution is to take a random condition containing at least w' observations and remove elements from the condition until only w' observations remain.

We set $w' = \left\lceil w \sqrt[|S|-1]{\alpha} \right\rceil$ with $\alpha \in (0, 1)$, so that, under the independence assumption, the expected share of observations in the slice equals α. As a result, subspace slicing happens in a *dimensionality-aware* fashion. When α is a constant, the expected number of observations per slice does not change between subspaces with different dimensionalities. Thus, subspace slicing is a dynamic grid-based method based on the dimensionality of the subspace.

Under the $\mathcal{A}^*$-assumption, the conditional distribution $p_{s_i|c_i}$ is equal to the marginal distribution p_{s_i}, for any attribute s_i and slice c_i. For brevity, we omit '(X)' in $p_{s_i}(X)$ and $p_{s_i|c_j}(X)$ in the following.

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

$$\mathcal{A}^*(S) \Leftrightarrow p_{s_i|c_i} = p_{s_i} \quad \forall s_i \in S,\ \forall c_i \in \mathcal{P}^c(S) \tag{5}$$

4.2.3 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 $\mathcal{T}$:

$$disc\left(\hat{p}_{s_i}, \hat{p}_{s_i|c_i}\right) \equiv \mathcal{T}\left(\hat{p}_{s_i}, \hat{p}_{s_i|c_i}\right) \tag{6}$$

However, the number of observations is finite, and the observations that we use to estimate $\hat{p}_{s_i|c_i}$ are part of the ones used to estimate $\hat{p}_{s_i}$ so far. 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 $\hat{p}_{s_i|c_i}$ then is small. We solve the problem by observing that $p_{s_i|c_i}$ and $p_{s_i|\overline{c}_i}$, the conditional distribution from the complimentary slice $\overline{c}_i$, are equal under $\mathcal{A}^*$.

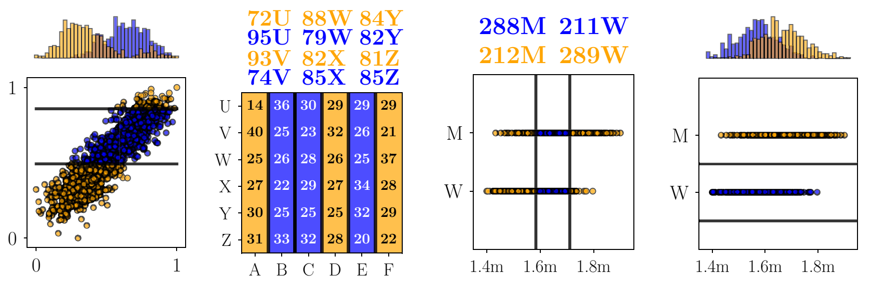
Theorem 1 ($\mathcal{A}^*$ and Complementary Conditions)

$$\mathcal{A}^*(S) \Leftrightarrow p_{s_i|\overline{c}_i} = p_{s_i|c_i} \quad \forall s_i \in S,\ \forall c_i \in \mathcal{P}^c(S) \tag{7}$$

Hence, one can evaluate $\mathcal{A}^*$ by looking at the discrepancies between the conditional distribution and its complementary conditional distribution. When doing so, the samples obtained from both distributions are distinct.

Our dynamic slicing scheme is defined based on a parameter α, the expected share of observations in the slice c_i. Thus, the expected share of observations $\overline{\alpha}$ in $\overline{c}_i$ equals $1 - \alpha$. As a result, we set $\alpha = 0.5$ so that $\overline{\alpha} = \alpha$. This choice is judicious for statistical testing, as having samples of equal size leads to higher statistical stability, and we get rid of parameter α.

We illustrate slicing in heterogeneous subspaces in Fig. 1, with an exemplary numerical and categorical subspace in the left half and a heterogeneous subspace in the right half. The black lines show a random slice. The points in dark blue are in c_i, and the points in light orange are in $\overline{c}_i$. Figure 1a represents a numerical linear dependency. We can see from the histograms that, after slicing, the distribution of the points in each sample are very different. Figure 1b depicts the absolute frequencies of observing various symptoms $\{U \ldots Z\}$ in different groups of patients $\{A \ldots F\}$. Since there is no ordering within symptoms and patient groups, slicing consists of selecting categories at random, here $\{B, C, E\}$. By comparing the absolute frequencies after slicing, we can determine whether there is a statistical association between groups and symptoms. Naturally, the statistical test that we use to estimate the discrepancy between $p_{s_i|\overline{c}_i}$ and $p_{s_i|c_i}$ might differ depending on the type of the reference attribute, as we will discuss later.



(a) Numerical **(b)** Categorical **(c)** Heterogeneous i. **(d)** Heterogeneous ii.

Fig. 1 Slicing in numerical, categorical, and heterogeneous subspaces, with $d = 2$

Different attribute types can also be part of the same subspace, as we show in Fig. 1c and d. We graph the height from a sample of individuals of two sexes. When we slice on the x-axis, the slice is a numerical interval. On the y-axis, in turn, the slice is a category drawn at random.

Intuitively, ordinal attributes share features from both numerical and categorical attributes: There exists an ordering between values, but typically also a large number of tied values. In this case, we recommend to use a similar slicing methodology as for numerical attributes, by selecting a discrete interval (see Definition 5), and a statistical test that is robust to tying values.

4.2.4 Properties of contrast

A statistical test $\mathcal{T}(B_1, B_2)$ between two samples B_1 and B_2 typically yields a p-value, which one uses to determine the *statistical significance* of a hypothesis. Conversely, $\overline{p} = 1 - p$ is known as the *confidence level*. The rationale behind estimating the degree of dependency $\mathcal{D}^*$ is to yield values quantifying the independence violation. We define *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 attributes s_i and slices c_i:

Definition 6 (*Contrast* $\mathcal{C}$)

$$\mathcal{C}(S) \equiv \mathop{\mathbb{E}}_{c_i \in \mathcal{P}^c(S)} \left[\mathcal{T}\left(B(c_i), B(\overline{c}_i)\right) \right] \tag{8}$$

where $\mathcal{T}$ yields $\overline{p}$-values, and $B(c_i), B(\overline{c}_i)$ are the samples resulting from slicing. We draw the conditions in c_i randomly and independently, w.r.t. any reference attribute 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.
2. $\mathcal{C}$ converges to 0.5 when S is independent, since $\mathcal{T} \sim \mathcal{U}[0, 1]$.
3. $\mathcal{C}$ is bounded between 0 and 1.

4.2.5 Monte Carlo approximation

It is impossible to compute $\mathcal{C}$ exactly; one would need to know the distribution of $B(c_i)$ and $B(\overline{c}_i)$ for every slice. Instead, we approximate $\mathcal{C}$ via Monte Carlo (MC) simulations, with M iterations. In each iteration, we choose the reference attribute s_i and slice c_i at random. We define the *approximated contrast* $\widehat{\mathcal{C}}$:

Definition 7 (Approximated contrast $\widehat{\mathcal{C}}$)

$$\widehat{\mathcal{C}}(S) = \frac{1}{M} \sum_{m=1}^{M} \left[\mathcal{T}\big(B(c_i), B(\overline{c}_i)\big) \ : \ c_i \overset{m}{\sim} \mathcal{P}^c(S) \right] \tag{9}$$

where $c_i \overset{m}{\sim} \mathcal{P}^c(S)$ means that we draw c_i randomly from $\mathcal{P}^c(S)$ in iteration m.

Interestingly, we can bound the quality of the approximation. From Hoeffding's inequality [20], we derive a bound on $\widehat{\mathcal{C}}$ w.r.t. $\mathcal{C}$, which decreases exponentially with increasing M:

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

$$\Pr\left(\left| \widehat{\mathcal{C}} - \mathcal{C} \right| \geq \varepsilon \right) \leq 2e^{-2M\varepsilon^2} \tag{10}$$

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

This bound is very useful. For instance, when $M = 200$, the probability of $\widehat{\mathcal{C}}$ 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 $\widehat{\mathcal{C}}$ given the available resources, the desired quality level, or the rate of arrival of new observations in a stream. In other words, users can set M intuitively, as it leads to an expected quality level and vice versa. Observe that M is our only parameter.

4.2.6 Instantiating MCDE

Finally, one must instantiate a suitable statistical test as $\mathcal{T}$. Ideally, this test is non-parametric (**F4**) and suitable for the type of the reference attribute (numerical, ordinal, categorical). To facilitate meaningful experiments, we investigate instantiations of MCDE based on three well-known non-parametric tests: the Kolmogorov–Smirnov, the Mann–Whitney U and the Chi-Squared test. We call the respective instantiations KSP, MWP and CSP.

The Kolmogorov–Smirnov test assumes the data to be continuous, i.e., it should be adequate for numerical attributes. The Mann–Whitney U test explicitly handles tying values. Thus it might work well with ordinal attributes. Lastly, the Chi-Squared test bases on frequencies from a finite number of categories, i.e., we hypothesise it to be suitable for categorical attributes.

Algorithm 1 MCDE ($S = \{s_1,...,s_d\}$)

```
1: $\mathcal{I} \leftarrow$ ConstructIndex($S$) ; $result \leftarrow 0$
2: for $m \leftarrow 1$ to $M$ do
3:     $r \leftarrow$ random integer in $[1, d]$
4:     $slice \leftarrow$ Slice($\mathcal{I}, r$)
5:     $result \leftarrow result$ + Test($\mathcal{I}, slice, r$)
6: return $(result/M) \in (0, 1)$
```

Algorithm 1 summarises the general idea behind MCDE for any arbitrary subspace $S = \{s_1, \ldots, s_d\}$ of dimensionality $|S| = d$. In practice, we can significantly improve the efficiency of slicing operations, which require the values of each attribute to be ordered, with an index structure (Line 1). Afterwards, for M iterations, we slice the data (Line 4) and carry out the statistical test (Line 5). The final estimate is the average of the test outcomes over M iterations. In what follows, we present the specifics of constructindex, slice and test for each instantiation of MCDE.

4.3 Instantiation as Mann–Whitney-P (MWP)

We first consider the instantiation of $\mathcal{T}$ as a two-sided Mann–Whitney U test [26]. An advantage of this test is that it does not assume the data to be continuous, as it operates on ranks. So it is robust and applicable to virtually any kind of numeric or ordinal measurements.

4.3.1 Marginal restriction

In a nutshell, the Mann–Whitney U test compares the difference between the median of the two samples. However, it is known that the ability of this test to detect dependency—the so-called 'statistical power'—declines in the case of unequal variance of the two samples [11, 47]. To alleviate this issue, we include an additional step into the slicing process. It restricts the domain of the reference attribute s_i to a share α of observations. Formally, we define the marginal restriction as follows:

Definition 8 *(Marginal Restriction)* A marginal restriction is a condition on the reference attribute s_i, i.e., an interval $r_i : [l_i, u_i]$ or $r_i : [l_i \ldots u_i]$, so that $|\{\mathbf{x}_j : x_{ji} \in r_i\}| = \lceil \alpha \cdot w \rceil = \lceil w' \rceil$ and the subspace slice becomes $c_i \cup r_i$.

We illustrate in Fig. 2 the impact of the marginal restriction. We show in the left half a circular dependency and in the other half a linear dependency. Two grey lines show a marginal restriction (in Fig. 2b, d), and two vertical dashed lines stand for the median of each sample. After slicing, both samples have highly unequal variance (see Fig. 2a, c). However, in Fig. 2a, the median of both distributions are nearly equal. Thus, this dependency remains undetected. The marginal restriction solves this problem, as we see in Fig. 2b. However, there is almost no difference between Fig. 2c and d. Intuitively, the marginal restriction 'breaks the symmetry'

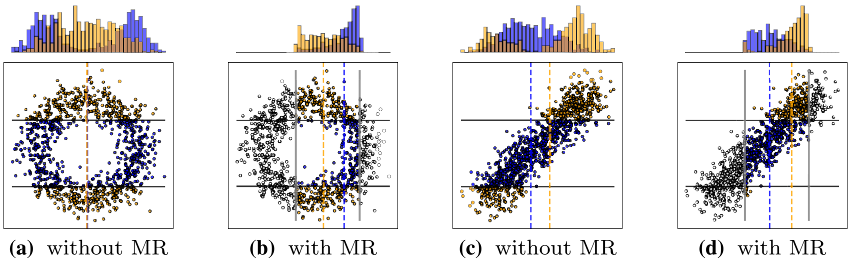


Fig. 2 Marginal restriction (MR) w.r.t. a circular and linear dependency ($d = 2$)

between both distributions. Because of that, the MWP estimator with marginal restriction has higher statistical power against certain dependency types.

4.3.2 Implementation details

Algorithm 2 is the pseudo-code for the index construction. The index $\mathcal{I}$ is a one-dimensional structure containing the adjusted ranks and tying value corrections for each attribute. It consists of $|S| = d$ elements $\{I_1, \dots, I_d\}$, where I_i is an array of 4-tuples $[(l_{1i}, \tilde{x}_{1i}, a_{1i}, b_{1i}), \dots, (l_{wi}, \tilde{x}_{wi}, a_{wi}, b_{wi})]$ ordered by s_i in ascending order. In this index, l_i are the row numbers, $\tilde{x}_i$ are the sorted values of s_i, a_i are the *adjusted* ranks and b_i the accumulated correction terms. $I_i[j]$ stands for the j-th tuple of I_i, and l_{ji}, $\tilde{x}_{ji}$, a_{ji}, b_{ji} are its components.

Algorithm 2 MWP-ConstructIndex($S = \{s_1, \dots, s_d\}$)

```
1:  for $i = 1$ to $d$ do
2:      $r_i \leftarrow [0, \dots, w-1]$
3:      $(l_i, \tilde{x}_i) \leftarrow$ sort $(r_i, x_i)$ by $s_i$ in ascending order, break ties randomly
4:      $I_i \leftarrow [(l_{1i}, \tilde{x}_{1i}, r_{1i}), \dots, (l_{wi}, \tilde{x}_{wi}, r_{wi})]$
5:      $j \leftarrow 1$ ; $correction \leftarrow 0$
6:      while $j \leq w$ do
7:          $k \leftarrow j$ ; $t \leftarrow 1$ ; $adjust \leftarrow 0$
8:          while $(k < w-1) \wedge (s_i[l_{ki}] = s_i[l_{k+1,i}])$ do
9:              $adjust \leftarrow adjust + r_{ki}$
10:             increment $k$ and $t$
11:         if $k > j$ then
12:             $adjusted \leftarrow (adjust + r_{ki})/t$
13:             $correction \leftarrow correction + t^3 - t$
14:             for $m \leftarrow j$ to $k$ do $I_i[m] \leftarrow (l_{mi}, \tilde{x}_{mi}, adjusted, correction)$
15:         else  $I_i[j] \leftarrow (l_{ji}, \tilde{x}_{ji}, r_{ji}, correction)$
16:         $j \leftarrow j + t$
17: return $\mathcal{I} : \{I_1, \dots, I_d\}$ with $I_i : (l_i, \tilde{x}_i, a_i, b_i)$
```

Algorithm 3 shows the slicing process. We successively mask the row numbers based on a random condition for all but one reference attribute s_r. Additionally, we ensure that the condition boundaries do not split any tying values and that each condition has w' observations. The algorithm returns a *slice*, i.e., a list of w Boolean values, so we write $slice \in \mathbb{Z}_2^w$.

Algorithm 3 MWP-SLICE($\mathcal{I} : \{I_1, \ldots, I_d\}, r \in \{1, \ldots, d\}$)

```
1: $w' \leftarrow \lceil w \cdot \sqrt[d-1]{\alpha} \rceil$
2: for $I_i \in \mathcal{I} \setminus I_r$ do
3:     $slice_i \leftarrow$ Array of $w$ Boolean values initialised to $false$
4:     $start \leftarrow$ random integer in $[1, w - w']$
5:     $end \leftarrow start + w'$
6:     while $r_{start,i} = r_{start-1,i}$ do $start = start - 1$
7:     while $r_{end,i} = r_{end+1,i}$ do $end = end + 1$
8:     for $j \leftarrow start$ to $end$ do $slice[l_{ji}] \leftarrow true$
9:     if $end - start > w'$ then
10:        $nb \leftarrow end - start - w'$
11:        $exclude \leftarrow$ draw $nb$ sample from $[start, end]$ without replacement
12:        for $el \in exclude$ do $slice[el] \leftarrow false$
13: $slice_r \leftarrow$ Array of $w$ Boolean values initialised to $true$
14: $slice \leftarrow slice_1 \wedge \cdots \wedge slice_d$
15: return $slice \in \mathbb{Z}_2^w$
```

Algorithm 4 MWP-TEST($\mathcal{I} : \{I_1, \ldots, I_d\}, slice \in \mathbb{Z}_2^w, r \in \{1, \ldots, d\}$)

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

Algorithm 4 implements the statistical test based on our index. We determine a restriction $[start, end]$ on s_r and sum the adjusted ranks of the observations in the slice. 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.

4.4 Instantiation as Kolmogorov–Smirnov-P (KSP)

We now describe another instantiation of MCDE, which uses the two-sample Kolmogorov–Smirnov (KS) test as a statistical test for $\mathcal{T}$. The (two-sample) KS test is non-parametric and tests the equality of two continuous one-dimensional probability distributions. It is adequate in case the reference attribute is numerical. However, the KS test has less power in the presence of ties [24]. So it may not work well with ordinal attributes.

In a nutshell, the two-sample Kolmogorov–Smirnov statistic D is the supremum of the absolute differences of the empirical cumulative distribution functions $F_1(x)$ and $F_2(x)$ of Samples 1 and 2 with n_1 and n_2 elements:

$$D = \sup_x \left|F_1(x) - F_2(x)\right| \tag{11}$$

HiCS [21] employed this test statistic to quantify the contrast of a subspace. However, to comply with the MCDE framework, one must first derive the corresponding $\overline{p}$-value. The $\overline{p}$-values are not trivial to obtain, because the distribution of D does not have any known closed form, and the time required for an exact computation increases with n_1 and n_2 in particular.

We approximate the $\overline{p}$-values using the asymptotic Kolmogorov–Smirnov distribution proposed in [10]:

$$\Pr\left(D\sqrt{\frac{n_1 \cdot n_2}{n_1 + n_2}} \leq x\right) = 1 - 2\sum_{i=1}^{\infty}(-1)^{i-1}e^{-2i^2x^2} \tag{12}$$

Empirically, we found that summing up the first 1000 terms of the expansion provides enough accuracy for our application, without much impact on the execution time. Using this approximation is common practice within most modern statistical software, such as R [34].

Algorithm 5 is the pseudo-code for the index construction. The difference to MWP is that we do not need to adjust ranks or precompute tie correction, because the Kolmogorov–Smirnov test assumes no tying values. The resulting data structure contains the indexes l_i and the values x_i of each attribute s_i, ordered by s_i in ascending order.

Algorithm 5 KSP-ConstructIndex($S = \{s_{1,\ldots,}s_{\mathrm{d}}\}$)

```
1: for $i = 1$ to $d$ do
2:     $r_i \leftarrow [0, \ldots, w-1]$
3:     $(l_i, \tilde{x}_i) \leftarrow$ sort $(r_i, x_i)$ by $s_i$ in ascending order, break ties randomly
4:     $I_i \leftarrow [(l_{1i}, \tilde{x}_{1i}), \ldots, (l_{wi}, \tilde{x}_{wi})]$
5: return $\mathcal{I} : \{I_1, \ldots, I_d\}$ with $I_i : (l_i, \tilde{x}_i)$
```

Similarly, Algorithm 6 is responsible for slicing but does not require any further step to handle ties. Algorithms 2 and 5, as well as Algorithms 3 and 6 behave in the same way whenever the data does not have ties.

Algorithm 6 KSP-SLICE($\mathcal{I} : \{I_1, \ldots, I_d\}, r \in \{1, \ldots, d\}$)

```
1: $w' \leftarrow \lceil w \cdot \sqrt[d-1]{\alpha} \rceil$
2: for $I_i \in \mathcal{I} \setminus I_r$ do
3:     $slice_i \leftarrow$ Array of $w$ Boolean values initialised to $false$
4:     $start \leftarrow$ random integer in $[1, w - w']$
5:     $end \leftarrow start + w'$
6:     for $j \leftarrow start$ to $end$ do $slice[l_{ji}] \leftarrow true$
7: $slice_r \leftarrow$ Array of $w$ Boolean values initialised to $true$
8: $slice \leftarrow slice_1 \wedge \cdots \wedge slice_d$
9: return $slice \in \mathbb{Z}_2^w$
```

Algorithm 7 implements the KS test. We compute the statistic D, i.e., the maximal difference of the two empirical cumulative distribution functions in Lines 7–10. Then we approximate the $\overline{p}$-value with Eq. 12 in Line 12.

Algorithm 7 KSP-TEST($\mathcal{I} : \{I_1, \ldots, I_d\}, slice \in \mathbb{Z}_2^w, r \in \{1, \ldots, d\}$)

```
1: $n_1 \leftarrow |\{i : i \in [1 \ldots w] \wedge slice[l_{ir}] = true\}|$
2: $n_2 \leftarrow w - n_1$
3: if $n_1 = 0$ or $n_2 = 0$ return 1
4: $u \leftarrow 1/n_1$ ; $v \leftarrow 1/n_2$
5: $\zeta_1 \leftarrow 0$ ; $\zeta_2 \leftarrow 0$
6: $D \leftarrow 0; \phi \leftarrow 0$
7: for $i \leftarrow 1$ to $w$ do
8:     if $slice[l_{ir}] = true$ then $\zeta_1 \leftarrow \zeta_1 + u$
9:     else $\zeta_2 \leftarrow \zeta_2 + v$
10:    $D \leftarrow max\{D, |\zeta_2 - \zeta_1|\}$
11: $z \leftarrow D\sqrt{n_1 n_2/(n_1 + n_2)}$
12: for $i \leftarrow 1$ to 1000 do $\phi \leftarrow \phi + (-1)^{i-1} e^{-2i^2 z^2}$
13: return $(1 - 2\phi) \in (0, 1)$
```

4.5 Instantiation as Chi-Squared-P (CSP)

The Chi-Squared test, also known as Pearson's Chi-Squared test, perhaps is the most famous non-parametric statistical test. In short, it determines whether there is a significant difference between the expected frequencies and the observed frequencies of a set of categories.

For a reference variable $s_i \in Cat$ with categories $A = \{a_1, \ldots, a_{|A|}\}$, we sketch the contingency table w.r.t. the two samples $B(c_i)$ and $B(\overline{c}_i)$ in Table 1, where o^i_j is the absolute frequency of Category a_j in Sample i, and we have:

$$\sum_{i=1}^{|A|} o^j_i = o^j \quad j \in \{1, 2\} \tag{13}$$

Table 1 Exemplary contingency table between two samples

	a_1	a_2	...	$a_{\|A\|}$	Total
Sample 1: $B(c_i)$	o_1^1	o_2^1	...	$o_{\|A\|}^1$	o^1
Sample 2: $B(\overline{c}_i)$	o_1^2	o_2^2	...	$o_{\|A\|}^2$	o^2
Total	o_1	o_2	...	$o_{\|A\|}$	w

$$\sum_{j=1}^{2} o_i^j = o_i \quad i \in \{1, \dots, |A|\} \tag{14}$$

Then we can compute the test statistic Q as follows:

$$Q = \sum_{i=1}^{|A|} \sum_{j=1}^{2} \frac{\left(o_i^j - e_i^j\right)^2}{e_i^j} \tag{15}$$

where $e_i^j = o_i \cdot o^j / w$ is the expected absolute frequency.

Under independence, Q follows the χ^2 distribution with cumulative distribution function $\chi_k^2 : \mathbb{R}^+ \mapsto [0, 1]$, where $k = |A| - 1$ is the number of degrees of freedom. Thus, $\chi_k^2(Q)$ leads to the $\overline{p}$-value that we use for CSP.

Similarly to the other instantiations, we improve the execution time by constructing an index. It contains the position of each occurrence of a categorical value binned into its respective category. Algorithm 8 is our pseudo-code for its construction. We can construct the index in linear time with a single pass over each attribute, as it does not require any sorting.

Algorithm 8 CSP-ConstructIndex $(S = \{s_{1,\dots,}s_{\mathrm{d}}\})$

```
1: for $i = 1$ to $d$ do
2:     Define $I_i$ as a mapping of categories to $\{positions \subset \{0, \dots, w-1\}, counts \in \mathbb{N}^+\}$
3:     for $x_{ji} \in s_i$ do
4:         if $I_i[x_{ji}] \neq \emptyset$ do $I_i[x_{ji}] \leftarrow \{I_i[x_{ji}].positions \oplus x, I_i[x_{ji}].counts + 1\}$
5:         else $I_i[x_{ji}] \leftarrow \{j, 1\}$
6: return $\mathcal{I} : \{I_1, \dots, I_d\}$ where $I_i : el \in s_i \mapsto \mathbb{N}^+$
```

Algorithm 9 is our slicing procedure for CSP. The main difference to MWP and KSP is that the values of the index do not have any meaningful ordering. Thus, slicing consists of selecting a random set of categories per attribute. The algorithm ensures that exactly w' observations are part of each condition.

Algorithm 9 CSP-SLICE($\mathcal{I} : \{I_1, \dots, I_d\}, r \in \{1, \dots, d\}$)

```
 1: $w' \leftarrow \lceil w \cdot \sqrt[d-1]{\alpha} \rceil$
 2: for $I_i \in \mathcal{I} \setminus I_r$ do
 3:     $slice_i \leftarrow$ Array of $w$ Boolean values initialised to $false$
 4:     $slicesize \leftarrow 0$ ; $positions \leftarrow \emptyset$
 5:     $categories \leftarrow \mathcal{I}_i.keys$
 6:     while $slicesize < w'$ do
 7:         $category \leftarrow$ draw a random category from $categories$
 8:         $categories \leftarrow categories \setminus category$
 9:         $slicesize \leftarrow slicesize + \mathcal{I}_i[category].counts$
10:         $positions \leftarrow positions \oplus \mathcal{I}_i[category].positions$
11:     if $slicesize > w'$ then
12:         Delete $slicesize - w'$ random elements from $positions$
13:     for $pos \in positions$ do
14:         $slice_i[pos] \leftarrow true$
15: $slice_r \leftarrow$ Array of $w$ Boolean values initialised to $true$
16: $slice \leftarrow slice_1 \wedge \cdots \wedge slice_d$
17: return $slice \in \mathbb{Z}_2^w$
```

Finally, Algorithm 10 shows how to compute the Chi-Squared statistic, based on the information from the index and a subspace slice.

Algorithm 10 CSP-TEST($\mathcal{I} : \{I_1, \dots, I_d\}, slice \in \mathbb{Z}_2^w, r \in \{1, \dots, d\}$)

```
 1: $Q = 0$ ; $k = 0$
 2: $o^1 = |\{pos \in [0 \dots w-1] : slice[pos] = true\}|$
 3: $o^2 = w - o_1$
 4: for $\{positions, counts\} \in I_r$ do
 5:     $o_x^1 = |\{pos \in positions : slice[pos] = true\}|$
 6:     $o_x^2 = counts - o_x^1$
 7:     $o_x = o_x^1 + o_x^2$
 8:     $e_x^1 = o_x \cdot o^1 / w$
 9:     $e_x^2 = o_x \cdot o^2 / w$
10:     $k = k + 1$
11:     $Q = Q + (o_x^1 - e_x^1)^2 / e_x^1 + (o_x^2 - e_x^2)^2 / e_x^2$
12: return $\chi_{k-1}^2(Q) \in (0, 1)$
```

4.6 MCDE in heterogeneous data streams (H-DS)

4.6.1 Heterogeneity

For simplicity, we have described KSP, MWP and CSP, assuming a homogeneous data set, being numerical, ordinal and categorical respectively.

Each of our algorithms treats the attributes of a subspace independently. So we can easily extend Algorithm 1 to the heterogeneous setting, as we show in Algorithm 11. We construct the index of each attribute depending on its type (Lines 1–3) and use the corresponding slicing methodology (Lines 7–9). The resulting slice is the element-wise conjunction for each type (Line 10). The type of the reference attribute determines which test we should use (Lines 11–13). Independently from the underlying statistical test, the $\overline{p}$-values have the properties described in Sect. 4.2.4. So the final MCDE score is the average of the $\overline{p}$-values over each iteration.

Algorithm 11 HETEOROGENEOUS-MCDE $(S = \{s_1,...,s_d\})$

```
1: $\mathcal{I}_N \leftarrow$ KSP-CONSTRUCTINDEX($\{s_i \in S : s_i \in Num\}$)
2: $\mathcal{I}_O \leftarrow$ MWP-CONSTRUCTINDEX($\{s_i \in S : s_i \in Ord\}$)
3: $\mathcal{I}_C \leftarrow$ CSP-CONSTRUCTINDEX($\{s_i \in S : s_i \in Cat\}$)
4: $result \leftarrow 0$
5: for $m \leftarrow 1$ to $M$ do
6:     $r \leftarrow$ random integer in $[1, d]$
7:     $slice_N \leftarrow$ KSP-SLICE($\mathcal{I}_N, r$)
8:     $slice_O \leftarrow$ MWP-SLICE($\mathcal{I}_O, r$)
9:     $slice_C \leftarrow$ CSP-SLICE($\mathcal{I}_C, r$)
10:    $slice \leftarrow slice_N \oplus slice_O \oplus slice_C$
11:    if $s_r \in Num$ do $result \leftarrow result$ + KSP-TEST($\mathcal{I}_N, slice, r$)
12:    if $s_r \in Ord$ do $result \leftarrow result$ + MWP-TEST($\mathcal{I}_O, slice, r$)
13:    if $s_r \in Cat$ do $result \leftarrow result$ + CSP-TEST($\mathcal{I}_C, slice, r$)
14: return $(result/M) \in (0,1)$
```

4.6.2 Adaptation to the streaming setting

To deal with streams, we adopt the well-known sliding window model, i.e., we only consider the w most recent observation. A naive way to support this model is to recompute the index at the arrival of each new observation. Instead, we propose efficient insertion and deletion operations for our indexes.

Furthermore, to maintain a dependency estimate over time, we propose to perform a number M of MC iterations periodically and report the exponential moving average:

$$MCDE_t = \gamma \cdot MCDE(W_{t-\Delta}) + (1 - \gamma) \cdot MCDE(W_t) \tag{16}$$

where γ is the so-called decaying factor, and Δ is the step size.

We update the MWP index in Algorithm 12 in two steps: STEP 1: INSERT/DELETE and STEP 2: REFRESH. Our algorithm maintains two data structures: a queue, which determines for each new point the value of the point to delete in the current window, in a first-in-first-out fashion, and a variant of our static index which supports binary search. In the first step, we store the values for each attribute in a queue, in chronological order. Then we find the positions where to insert the new point and where to delete the oldest point in the index via binary search. In the second step, we recompute the adjusted ranks and cumulative correction as in Algorithm 2.

Using a data structure like a binary tree, STEP 1 only has logarithmic complexity, while STEP 2 has linear complexity. Besides this, one must perform STEP 1 for each new point, but STEP 2 only once before slicing. So when re-estimating contrast only every Δ-th step, we perform STEP 1 for each point, but STEP 2 lazily. As a result, we can update the index in $O(d \cdot log(w))$ in STEP 1 for each new observation and postpone STEP 2, which is in $O(d \cdot w)$, to contrast estimation. Updating the KSP index

Table 2 Algorithmic complexity

	MWP	KSP	CSP
Index Construction	$O(d \cdot w \cdot log(w))$	$O(d \cdot w \cdot log(w))$	$O(d \cdot w)$
Slicing	$O(d \cdot w)$	$O(d \cdot w)$	$O(d \cdot w)$
Test	$O(w)$	$O(w)$	$O(w)$
Update (STEP 1)	$O(d \cdot log(w))$	$O(d \cdot log(w))$	$O(1)$
Update (STEP 2)	$O(d \cdot w)$	$O(d \cdot w)$	$O(d \cdot w)$

is somewhat simpler because KSP does not handle tying values. The CSP index is unsorted and thus STEP 1 only requires constant time. We summarise the complexity of each step in Table 2 and refer the interested reader to our source code.[2]

For efficiency, Algorithm 12 simultaneously inserts and deletes observations. Note that one could also perform the insert and delete operations via two independent methods, i.e., effectively handling time-based windows, in which observations may arrive at arbitrary time steps, or in batches.

Algorithm 12 MWP-UPDATE($\mathcal{I} : \{I_1, \dots, I_d\}, \mathbf{x}_{new} = \langle x_{new,i} \rangle_{i \in \{1,\dots,d\}}$)

```
1: for $i = 1$ to $d$ do
2:     {STEP 1: INSERT/DELETE}
3:     $queue_i.insert(x_{new,i})$ ; $x_{old,i} = queue_i.pop()$
4:     $offset_i = offset_i + 1$
5:     $insert = binarysearch(I_i, x_{new,i})$
6:     $delete = binarysearch(I_i, x_{old,i})$
7:     if $insert < delete$ for $x \leftarrow insert$ to $delete$ do $I_i[x+1] = I_i[x]$
8:     else for $x \leftarrow delete$ to $insert$ do $I_i[x] = I_i[x+1]$
9:     $I_i[insert] = (w + offset_i, x_{new,i}, -1, -1)$
10:    {STEP 2: REFRESH}
11:    for $pos \leftarrow 1$ to $w$ do $I_i[pos] = (l_{pos,i} - offset_i, \tilde{x}_{pos,i}, pos, 0)$
12:    $offset_i \leftarrow 0$ ; $j \leftarrow 1$ ; $correction \leftarrow 0$
13:    while $j \leq w$ do
14:        $k \leftarrow j$ ; $t \leftarrow 1$ ; $adjust \leftarrow 0$
15:        while $(k < w - 1) \wedge (s_i[l_{ki}] = s_i[l_{k+1,i}])$ do
16:            $adjust \leftarrow adjust + a_{ki}$
17:            increment $k$ and $t$
18:        if $k > j$ then
19:            $adjusted \leftarrow (adjust + a_{ki})/t$
20:            $correction \leftarrow correction + t^3 - t$
21:            for $m \leftarrow j$ to $k$ do $I_i[m] \leftarrow (l_{mi}, \tilde{x}_{mi}, adjusted, correction)$
22:        else $I_i[j] \leftarrow (l_{ji}, \tilde{x}_{ji}, a_{ji}, correction)$
23:        $j \leftarrow j + t$
24: return $\mathcal{I} : \{I_1, \dots, I_d\}$ with $I_i : (l_i, \tilde{x}_i, a_i, b_i)$
```

[2] https://github.com/edouardfouche/MCDE-EXTENDED.

5 Experiments

To evaluate our dependency estimators, i.e., MWP, KSP and CSP, we look at the scores they produce w.r.t. an assortment of dependencies. See Fig. 3; we scale the dependencies to [0, 1]. 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*. Intuitively, noise-free dependencies should lead to higher scores than noisier ones.

We also show that MCDE is robust and handles heterogeneity by simulating numerical, ordinal, and categorical attributes. To simulate ordinal attributes, we discretise numerical attributes into a number Ω of values from 1 to 20. To simulate categorical attributes, we randomly permute the discretised values, to mimic the absence of an order.

Similarly to other studies [23, 28, 35], we compute the statistical power, defined as follows:

Definition 9 (Power) The power of an estimator $\mathcal{E}$ w.r.t. dependency O 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. the independent subspace I:

$$\Pr\left(\mathcal{E}\left(Inst^{\mathrm{O},\sigma}_{n\times d}\right) > \left\{\mathcal{E}\left(Inst^{\mathrm{I},0}_{n\times d}\right)\right\}^{P_\gamma}\right) \tag{17}$$

$Inst^{\mathrm{O},\sigma}_{n\times d}$ is a random instantiation of a subspace as dependency O 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 so that $\gamma\%$ of the values in $\{x\}$ are smaller than v.

The attributes of the independent subspace I are i.i.d. in $\mathcal{U}[0,1]$. Note that, since the attributes of I are independent, adding noise does not have any effect on

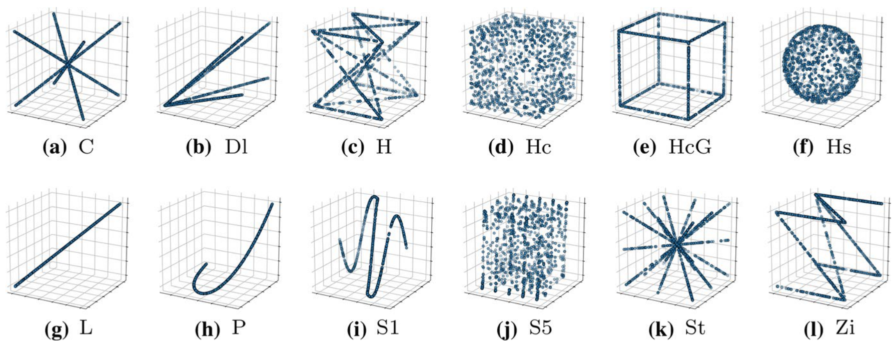


Fig. 3 An assortment of 12 dependencies (displayed here with three dimensions, $\sigma = 0$). *C* cross, *Dl* double linear, *H* Hourglass, *Hc* hypercube, *HcG* hypercube Graph, *Hs* hypersphere, *L* linear, *P* parabolic, *S1* Sine (P=1), *S5* Sine (P=5), *St* Star, *Zi* Z inversed

dependence, so we set noise to 0 when instantiating I. To estimate the power, we draw two sets of 500 estimates from O, σ and I respectively:

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

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

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

One can interpret ‘power’ as the probability to correctly reject the independence hypothesis with $\gamma\%$ confidence. I.e., the power quantifies how well a dependency measure can differentiate between the independence I and a given dependency O with noise level σ. For our experiments, we set $\gamma = 95$, $n = 1000$. We let the noise σ vary linearly from 0 to 1, with 30 distinct levels. We consider dependencies with dimensionality d from 2 to 20.

In our figures, O_Ω denotes each dependency, where O stands for the dependency type (e.g., L stands for ‘Linear’), and Ω is the discretisation level, i.e., the number of distinct values; O means that the attributes are numerical. d=x indicates the dimensionality, and O^*_Ω indicates that the attributes are categorical, with a number Ω of nominal values.

5.1 Specifics of attribute types

We first observe how MCDE handles numerical, ordinal, and categorical attributes. Figure 4 displays the empirical distribution of MWP, KSP and CSP iterations w.r.t. a 2-dimensional independent (I) and a linearly dependent (L) space. We visualise each distribution as a histogram from 10 000 independent iterations. The vertical dashed lines show the means of the distributions, and we display a scatter plot illustrating the respective scenario.

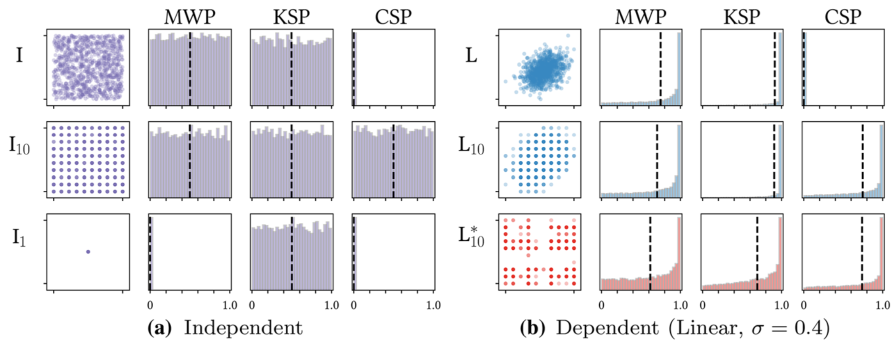


Fig. 4 Distribution of the contrast estimation iterations ($d = 2$)

According to Fig. 4a, MWP, KSP and CSP values are uniformly distributed in the case of an independent subspace, with a few exceptions: First, the CSP values are all close to 0 in the case of numerical attributes. In this setting, each point is unique, i.e., the Chi-Squared test treats each observation as a distinct category, and thus does not have power. We do the same observation with L (in Fig. 4b). Second, we see that the values of MWP and CSP are also 0 with I_1, which corresponds to the desired behaviour. Every observation in the subspace is equal, so contrast is undefined. KSP assumes that values are continuous, and thus does not handle this case.

Also, we can see from Fig. 4b that the KSP values are generally closer to 1 with L and L_{10}, which indicates more power than MWP and CSP. However, we can see that the CSP distribution does not change between L_{10} and L^*_{10}, while the mean of the MWP and KSP distribution decreases significantly. Thus, CSP detects dependency from categorical attributes better than MWP/KSP.

5.2 Statistical power of MWP, KSP and CSP

We first look at the statistical power of MWP, KSP and CSP with confidence level $\gamma = 0.95$ against a linear dependency of increasing dimensionality d, discretisation level Ω, and noise level σ. From Fig. 5, we see that MWP without marginal

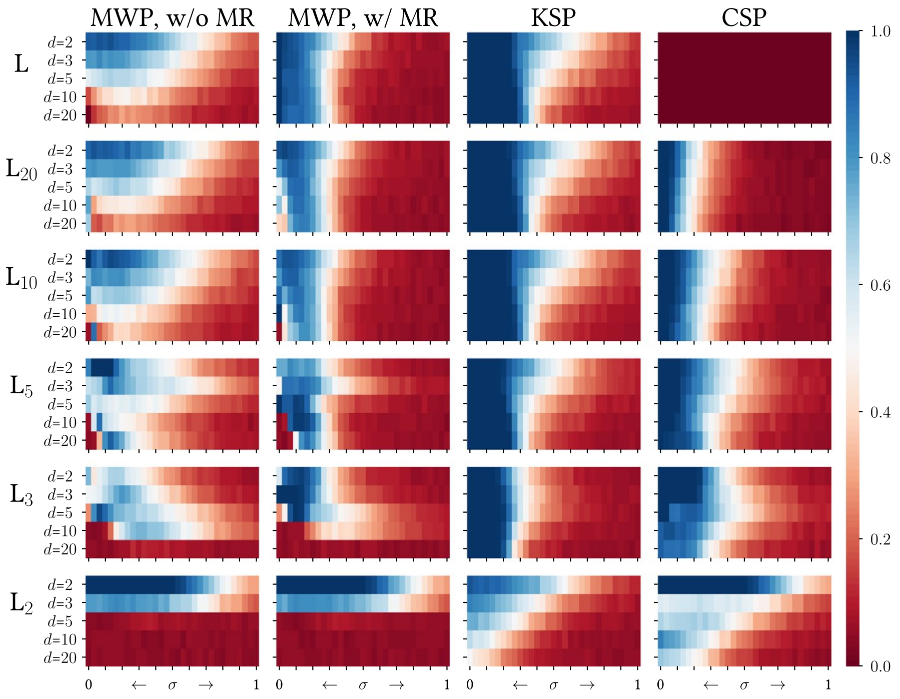


Fig. 5 Power against continuous and discrete linear distributions (d from 2 to 20)

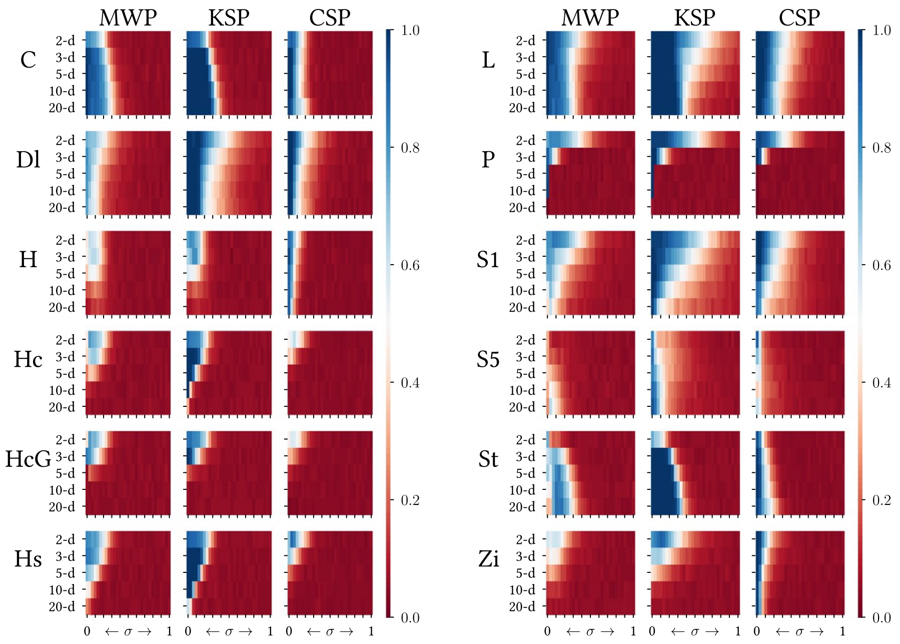


Fig. 6 Power of MWP/KSP/CSP against a dependency assortment (*d* from 2 to 20)

restriction does not work well in high-dimensional and highly discretised spaces. The marginal restriction alleviates this problem to some extent, in particular for numerical subspaces. In fact, as dimensionality increases, it becomes more and more likely that the points selected in the slice are 'in the centre' of the distribution. The mean of the points in the slice and outside of the slice become nearly equal, leading to the low power of the Mann–Whitney U test, and thus a slight performance decrease of MWP. This calls for further research on the MWP slicing scheme, or alternatives to the Mann–Whitney U test. Nonetheless, the results indicate that MWP with marginal restriction works well against numerical attributes.

Next, we see that KSP has high power in every case, although slightly decreasing with Ω. CSP does not work with numerical spaces but has more power in discrete spaces. CSP works best with categorical attributes.

We now compare the power of MWP, KSP, and CSP against the assortment of dependencies from Fig. 3. CSP does not apply to numerical attributes. Thus, for comparability, we discretise the values with $\Omega = 10$. We can see from Fig. 6 that KSP consistently has more power than MWP. CSP generally has less power than KSP but can detect categorical dependencies.

To summarise, our experiments show that MWP has a slight performance decrease in high-dimensional discrete spaces. KSP seems to perform better, but its statistical power decreases with discretisation. Overall, we recommend to use KSP against both numerical and ordinal data but to use CSP for categorical data. MWP still is a valid alternative with numerical attributes.

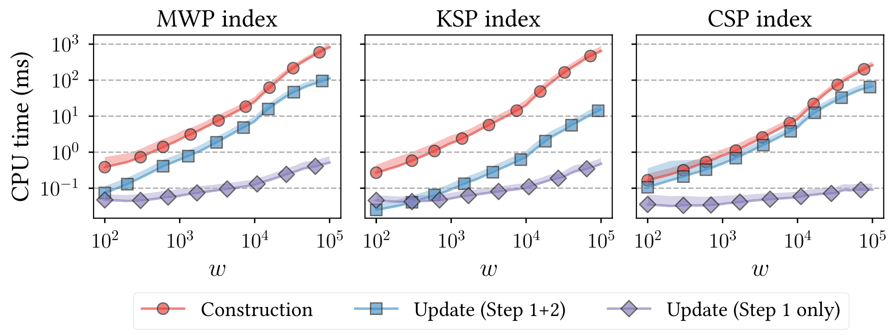


Fig. 7 Time required for index construction and update w.r.t. window size *w*

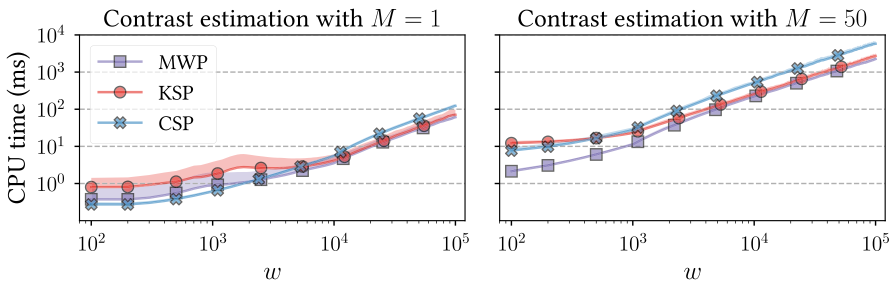


Fig. 8 Time required for contrast estimation w.r.t. window size *w*

5.3 Scalability

We evaluate the scalability of index construction for each approach, by increasing the size of the sliding window w from 10^2 to 10^5 in an independent space I with three dimensions. The red line (‘Construction’) is the average time for creating the index with window size w (Algorithms 2/5/8). The other lines show the average time to insert a new point into the window.

In Fig. 7, we can see that the construction time of each index increases almost linearly with increasing window size w. The KSP index is less expensive to update than MWP, regarding STEP 2 in particular. The first step of the CSP index update is very efficient, as it requires more or less constant time, cf. Table 2. We can see that only performing STEP 1 in our update operations, while delaying STEP 2 to the contrast estimation step, reduces the execution time by up to 3 orders of magnitude, compared to standard index construction. So we can significantly speed up the monitoring of MCDE contrast using the index update operations.

In Fig. 8, we compare the execution time of contrast estimation for MWP, KSP, and CSP, with increasing window size w. We can see that the three approaches have a comparable execution time. KSP is slightly slower for small window sizes because the $\overline{p}$ -values are more expensive to obtain than with the other approaches. However, as the window size increases, KSP and MWP have the same execution time. CSP contrast estimation appears to be slightly slower as the window size increases but does scale as well.

5.4 Deployment to the streaming setting

We monitor contrast in a subspace in which the dependency gradually evolves. We simulate this setting by concatenating 100 three-dimensional linear dependencies with 1000 observations each and a level of noise σ linearly increasing from 0 to 1. We use the approach described in Sect. 4.6.2, Eq. 16 and instantiate MCDE with MWP. We estimate the dependency over a sliding window of size $w = 1000$ and with a decaying factor $\gamma = 0.99$.

We let the step size Δ vary from 1 to 1000 and the number of iterations M from 1 to 500. We compare each configuration to a baseline, which is the most expensive configuration ($\Delta = 1, M = 500$), without the benefit of our update operations. When $\Delta > 1$, we simply set the current contrast estimate to the value from the latest estimation. We define the following measures:

- The **Absolute Error** is the average absolute difference between the values obtained with the tested configuration and the values from the baseline.
- The **Relative Time** is the ratio of the time required by the tested configuration over the time required by the baseline.
- The **Index Speedup** is the ratio between the time required by the tested configuration without/with our index update operations.

We can see from Fig. 9 that the absolute error decreases with Δ, while the relative execution time increases. The speedup obtained by our index operations is mainly responsible for this gain of efficiency. As we increase the number of iterations M, the errors tend to decrease, but at the same time, contrast estimation dominates the overall execution time. In such cases, we see less benefit from our efficient insert/delete operations.

We identify the configuration $M = 50, \Delta = 50$ as a sweet spot: For an absolute error as small as ≈ 0.01, the computational burden is reduced by up to 100 times, with a consistent index speedup of 3. We mark this configuration with a star $*$.

5.5 Pattern discovery

We now apply MCDE to a real-world multivariate time series. We collected the data during a 4-day production campaign at Bioliq, a pyrolysis plant in the surroundings of Karlsruhe [33]. It contains one measurement per second, i.e., 345 600 observations, from a selection of 20 physical sensors in various components of the plant. We monitored the evolution of dependency between each sensor pair with $w = 1000, \Delta = 50, M = 50$, as just explained. We obtained the evolution of dependency between the 20 sensors using a single CPU core in about 2 hours. Note that it would be easy to shorten the computation time significantly by parallel processing.

We have presented the results of our monitoring technique to the plant operators. They have identified several patterns which they deemed 'interesting', i.e., patterns yielding insights that could help with plant operation.

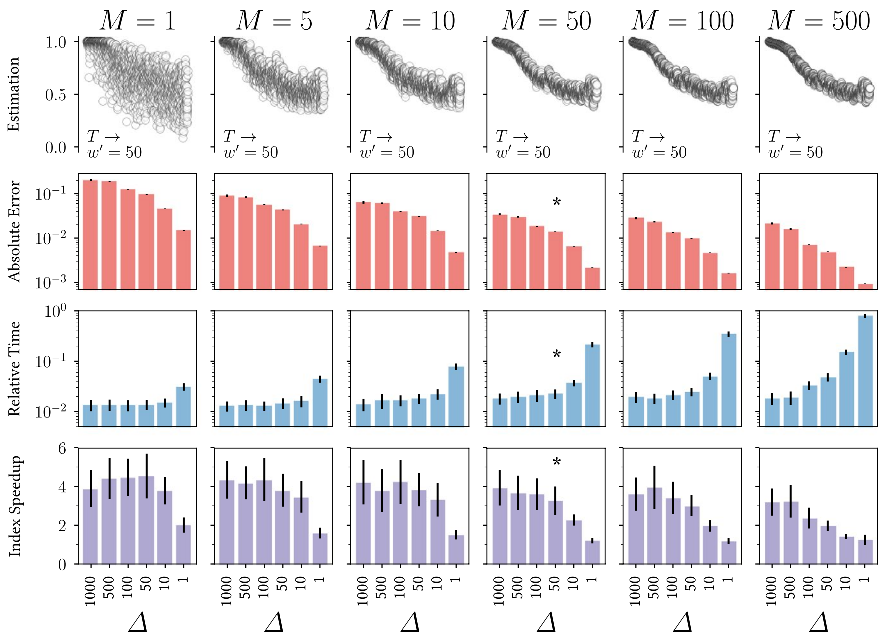


Fig. 9 Quality and speed of contrast estimation with concept drift (* ≡ sweet spot)

Figure 10 displays one of these patterns. It is the result of monitoring two sensors, namely the pressure at the reactor input, and the oxygen concentration at the output. As we see, the dependency between these two measures changes significantly over time. Some of the changes, marked in the figure from 1 to 4, appear to represent different stages in the production process. A better understanding of the dynamics of the physical measures involved in the reaction will help the plant owners to operate smoothly and efficiently.

5.6 Discussion

Our experiments show that MCDE fulfils all H-DS requirements and has the desirable features of a framework for dependency estimation.

First, we can see that MCDE is efficient (**C1**), and, thanks to the index update operations, one can use it in combination with the sliding window model to mine data streams in a single scan (**C2**) and adapt (**C3**) the contrast estimation over time, taking concept drift into account. Second, one can also reduce or increase the number of MC iterations *M* to trade between accuracy and computation time, in an anytime (**C4**) fashion. So our method also is intuitive to use (**F3**) and interpretable (**F5**). The different instantiations of MCDE allow dealing with various attribute types within the same data stream (**C5**). Last, MCDE is non-parametric (**F4**) by design. Our experiments against an assortment of dependencies show that it is multivariate (**F1**), general-purpose (**F2**) and robust (**F7**). MCDE is sensitive (**F6**) because

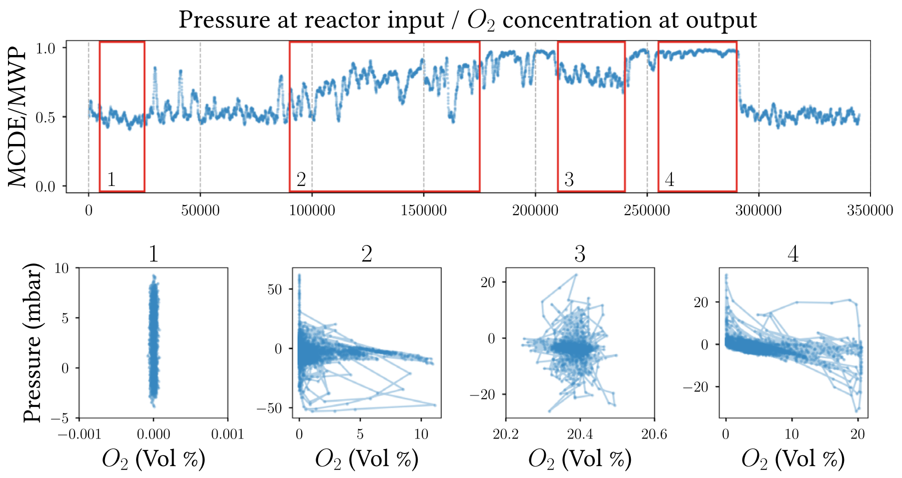


Fig. 10 Example of an interesting dependency pattern in the pyrolysis data

estimates are the average of statistical $\overline{p}$-values. Our previous work [15] has shown the superiority of MWP over other methods.

6 Conclusions

In this paper, we have proposed a framework to estimate multivariate dependency in heterogeneous data streams. It fulfils all requirements one would expect from a state-of-the-art dependency estimator. MCDE provides high statistical power on a large panel of dependencies while being very efficient. Furthermore, we introduced index operations for the streaming setting and illustrated the benefits of our framework against a real-world use case.

Future work could focus on improving our monitoring scheme: While updating the MCDE score via an exponential moving average appears to be a valid option, MCDE could benefit from a more flexible update mechanism, using for instance a sliding window of adaptive size [4]. Finally, we have only considered three instantiations of MCDE; it would be interesting to study the integration of further statistical tests, e.g., see [13] and [5].

Acknowledgements We thank Projekt DEAL for providing Open Access funding. We thank the anonymous reviewers for their comments; they helped us improve the quality of our manuscript. 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). We thank the pyrolysis team of the Bioliq process for providing the data for our real-world use case (see also https://www.bioliq.de).

Compliance with ethical standards

Conflict of interest The authors declare that they do not have any conflict of interest.

References

1. Barddal, J. P., Gomes, H. M., Enembreck, F.: A survey on feature drift adaptation. In: Proceedings of the ICTAI, pp. 1053–1060. IEEE Computer Society (2015)
2. Belghazi, M.I., Baratin, A., Rajeswar, S., Ozair, S., Bengio, Y., Hjelm, R.D., Courville, A.C.: Mutual information neural estimation. In: Proceedings of Machine Learning Research ICML, vol. 80, pp. 530–539. PMLR (2018)
3. Bellman, R.E.: Dynamic Programming. Princeton University Press, Princeton (1957)
4. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the SDM, pp. 443–448. SIAM (2007)
5. Brunner, E., Munzel, U.: The nonparametric Behrens-fisher problem: asymptotic theory and a small-sample approximation. Biom. J. **42**(1), 17–25 (2000)
6. Chen, M., Han, J., Yu, P.S.: Data mining: an overview from a database perspective. IEEE Trans. Knowl. Data Eng. **8**(6), 866–883 (1996)
7. Crescenzo, A.D., Longobardi, M.: On cumulative entropies and lifetime estimations. In: Proceedings of the of Lecture Notes in Computer Science IWINAC, vol. 5601, pp. 132–141. Springer (2009)
8. Ditzler, G., Roveri, M., Alippi, C., Polikar, R.: Learning in nonstationary environments: a survey. IEEE Comp. Int. Mag. **10**(4), 12–25 (2015)
9. Domingos, P., Hulten, G.: A general framework for mining massive data streams. J. Comp. Graph. Stat. **12**(4), 945–949 (2003)
10. Durbin, J.: Distribution Theory for Tests Based on Sample Distribution Function. SIAM, New Delhi (1973)
11. Fagerland, M.W., Sandvik, L.: The Wilcoxon-Mann-Whitney test under scrutiny. Stat. Med. **28**(10), 1487–1497 (2009)
12. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the IJCAI, pp. 1022–1029. Morgan Kaufmann (1993)
13. Fligner, M.A., Policello, G.E.: Robust rank procedures for the Behrens-Fisher problem. J. Am. Stat. Assoc. **76**(373), 162–168 (1981)
14. Fouché, E., Böhm, K.: Monte Carlo dependency estimation. In: Proceedings of the SSDBM, pp. 13–24. ACM (2019)
15. Fouché, E., Komiyama, J., Böhm, K.: Scaling multi-armed bandit algorithms. In: Proceedings of the KDD, pp. 1449–1459. ACM (2019)
16. Gama, J.: A survey on learning from data streams: current and future trends. Prog. AI **1**(1), 45–55 (2012)
17. Gretton, A., Fukumizu, K., Teo, C.H., Song, L., Schölkopf, B. Smola, A.J.: A Kernel statistical test of independence. In: Proceedings of the NIPS, pp. 585–592. Curran Associates, Inc (2007)
18. Hall, M.A., Holmes, G.: Benchmarking attribute selection techniques for discrete class data mining. IEEE Trans. Knowl. Data Eng. **15**(6), 1437–1447 (2003)
19. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Statistics, 2nd edn. Springer, New York (2009)
20. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**(301), 13–30 (1963)
21. Keller, F., Müller, E., Böhm, K.: HiCS: High contrast subspaces for density-based outlier ranking. In: Proceedings of the ICDE, pp. 1037–1048 (2012)
22. Keller, F., Müller, E., Böhm, K.: Estimating mutual information on data streams. In: Proceedings of the SSDBM, pp. 3:1–3:12. ACM (2015)

23. Kinney, J.B., Atwal, G.S.: Equitability, mutual information, and the maximal information coefficient. Proc. Natl. Acad. Sci. **111**(9), 3354–3359 (2014)
24. Lehmann, E.L., Romano, J.P.: Testing Statistical Hypotheses, 3rd edn. Springer, New York (2008)
25. López-Paz, D., Hennig, P., Schölkopf, B.: The randomized dependence coefficient. In: Proceedings of the NIPS, pp. 1–9 (2013)
26. Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. Ann. Math. Stat. **18**(1), 50–60 (1947)
27. McGill, W.: Multivariate information transmission. Trans. IRE Prof. Group Inf. Theory **4**(4), 93–111 (1954)
28. Nguyen, H.V., Mandros, P., Vreeken, J.: Universal dependency analysis. In: Proceedings of the SDM, pp. 792–800. SIAM (2016)
29. Nguyen, H.V., Müller, E., Andritsos, P., Böhm, K.: Detecting correlated columns in relational databaseswith mixed data types. In: Proceedings of the SSDBM, pp. 30:1–30:12. ACM (2014a)
30. Nguyen, H. V., Müller, E., Vreeken, J., Efros, P., Böhm, K.: Multivariate maximal correlation analysis. In: Proceedings of the ICML, pp. 775–783 (2014b)
31. Nguyen, H.V., Müller, E., Vreeken, J., Keller, F., Böhm, K.: CMI: an information-theoretic contrast measure for enhancing subspace cluster and outlier detection. In: Proceedings of the SDM, pp. 198–206. SIAM (2013)
32. Peng, H., Long, F., Ding, C.H.Q.: Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. **27**(8), 1226–1238 (2005)
33. Pfitzer, C., Dahmen, N., Tröger, N., Weirich, F., Sauer, J., Günther, A., Müller-Hagedorn, M.: Fast pyrolysis of wheat straw in the bioliq pilot plant. Energy Fuels **30**(10), 8047–8054 (2016)
34. Core Team, R.: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna (2019)
35. Reshef, D.N., Reshef, Y.A., Finucane, H.K., Grossman, S.R., McVean, G., Turnbaugh, P.J., Lander, E.S., Mitzenmacher, M., Sabeti, P.C.: Detecting novel associations in large data sets. Science **334**, 6062 (2011)
36. Schmid, F., Schmidt, R.: Multivariate extensions of Spearman's rho and related statistics. Stat. Probab. Lett. **77**(4), 407–416 (2007)
37. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(3), 379–423 (1948)
38. Spearman, C.: The proof and measurement of association between two things. Am. J. Psychol. **15**(1), 72–101 (1904)
39. Székely, G.J., Rizzo, M.L.: Brownian distance covariance. Ann. Appl. Stat. **3**(4), 1236–1265 (2009)
40. Timme, N., Alford, W., Flecker, B., Beggs, J.M.: Synergy, redundancy, and multivariate information measures: an experimentalist's perspective. J. Comput. Neurosci. **36**(2), 119–140 (2014)
41. Vollmer, M., Böhm, K.: Iterative estimation of mutual information with error bounds. In: Proceedings of the EDBT, pp. 73–84. OpenProceedings.org (2019)
42. Vollmer, M., Rutter, I., Böhm, K.: On complexity and efficiency of mutual information estimation on static and dynamic data. In: Proceedings of the EDBT, pp. 49–60. OpenProceedings.org (2018)
43. Wang, Y., Romano, S., Nguyen, V., Bailey, J., Ma, X., Xia, S.: Unbiased multivariate correlation analysis. In: Proceedings of the AAAI, pp. 2754–2760 (2017)
44. Watanabe, S.: Information theoretical analysis of multivariate correlation. IBM J. Res. Dev. **4**(1), 66–82 (1960)
45. Yang, C., Zhou, J.:. HClustream: A novel approach for clustering evolving heterogeneous data stream. In: Proceedings of the ICDM Workshops, pp. 682–688 (2006)
46. Zhu, Y., Shasha, D.E.: StatStream: statistical monitoring of thousands of data streams in real time. In: Proceedings of the VLDB, pp. 358–369 (2002)
47. Zimmerman, D.W.: A warning about the large-sample Wilcoxon-Mann-Whitney test. Underst. Stat. **2**(4), 267–280 (2003)