# A Chasm Between Identity and Equivalence Testing with Conditional Queries*

**Jayadev Acharya[1], Clément L. Canonne[2], and Gautam Kamath[1]**

1 **Massachusetts Institute of Technology**
**32 Vassar Street, Cambridge MA, USA**
{jayadev,g}@csail.mit.edu
2 **Columbia University**
**500 W 120th Street, New York NY, USA**
ccanonne@cs.columbia.edu

## Abstract

A recent model for property testing of probability distributions [11, 9] enables tremendous savings in the sample complexity of testing algorithms, by allowing them to condition the sampling on subsets of the domain.

In particular, Canonne, Ron, and Servedio [9] showed that, in this setting, testing identity of an unknown distribution $D$ (i.e., whether $D = D^*$ for an explicitly known $D^*$) can be done with a *constant* number of samples, independent of the support size $n$ – in contrast to the required $\sqrt{n}$ in the standard sampling model. However, it was unclear whether the same held for the case of testing equivalence, where *both* distributions are unknown. Indeed, while Canonne, Ron, and Servedio [9] established a polylog$(n)$-query upper bound for equivalence testing, very recently brought down to $\tilde{O}(\log \log n)$ by Falahatgar et al. [13], whether a dependence on the domain size $n$ is necessary was still open, and explicitly posed by Fischer at the Bertinoro Workshop on Sublinear Algorithms [14]. In this work, we answer the question in the positive, showing that any testing algorithm for equivalence must make $\Omega\left(\sqrt{\log \log n}\right)$ queries in the conditional sampling model. Interestingly, this demonstrates an intrinsic qualitative gap between identity and equivalence testing, absent in the standard sampling model (where both problems have sampling complexity $n^{\Theta(1)}$).

Turning to another question, we investigate the complexity of support size estimation. We provide a doubly-logarithmic upper bound for the adaptive version of this problem, generalizing work of Ron and Tsur [22] to our weaker model. We also establish a logarithmic lower bound for the non-adaptive version of this problem. This latter result carries on to the related problem of non-adaptive uniformity testing, an exponential improvement over previous results that resolves an open question of Chakraborty, Fischer, Goldhirsh, and Matsliah [11].

**1998 ACM Subject Classification** F.2.2 Nonnumerical Algorithms and Problems, G.3 Probability and Statistics

**Keywords and phrases** property testing, probability distributions, conditional samples

**Digital Object Identifier** 10.4230/LIPIcs.APPROX-RANDOM.2015.449

* The full version of this paper can be found at [1].



18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) / 19th Int'l Workshop on Randomization and Computation (RANDOM'15).
Editors: Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim; pp. 449–466

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# 1 Introduction

*'No, Virginia, there is no constant-query tester.'*

Understanding properties and characteristics of an unknown probability distribution is a fundamental problem in statistics, and one that has been thoroughly studied. However, it is only since the seminal work of Goldreich and Ron [15] and Batu et al. [5] that the problem has been considered through the lens of theoretical computer science, more particularly in the setting of *property testing*.

Over the following decade, a flurry of subsequent work explored and delved into this new area, resulting in a better and often complete understanding of a number of questions in distributional property testing (see e.g. [15, 4, 6, 20, 24, 2, 7, 23, 17, 3, 12, 29] or [8] for a survey). In many cases, these culminated in provably sample-optimal algorithms. However, the standard setting of distribution testing, where one only obtains independent samples from an unknown distribution $D$, does not encompass *all* scenarios one may encounter. In recent years, stronger models have thus been proposed to capture more specific situations [16, 11, 9, 18, 10]: among these is the *conditional oracle model* [11, 9] which will be the focus of our work. In this setting, the testing algorithms are given the ability to sample from conditional distributions: that is, to specify a subset $S$ of the domain and obtain samples from $D_S$, the distribution induced by $D$ on $S$ (the formal definition of the model can be found in Definition 2.1). In particular, the hope is that allowing algorithms to have stronger interactions with the unknown underlying distributions might significantly reduce the number of samples they need, thereby sidestepping the strong lower bounds that hold in the standard sampling model.

## 1.1 Background and previous work

We focus in this paper on proving lower bounds for testing two extremely natural properties of distributions, namely *equivalence testing* ("are these two datasets identically distributed?") and *support size estimation* ("how many different outcomes can actually be observed?"). Along the way, we use some of the techniques we develop to obtain an upper bound on the query complexity of the latter. We state below the informal definition of these two problems, along with closely related ones (uniformity and identity testing). Hereafter, "oracle access" to a distribution $D$ over $[n] = \{1, \dots, n\}$ means access to samples generated independently from $D$.

**Uniformity testing:** granted oracle access to $D$, decide whether $D = \mathcal{U}$ (the uniform distribution on $[n]$) or is far from it;

**Identity testing:** granted oracle access to $D$ and the full description of a fixed $D^*$, decide whether $D = D^*$ or is far from it;

**Equivalence (closeness) testing:** granted independent oracle accesses to $D_1$, $D_2$ (both unknown), decide whether $D_1 = D_2$ or $D_1$, $D_2$ are far from each other.

**Support size estimation:** granted oracle access to $D$, output an estimate of the size of the support[1] $\mathrm{supp}(D) = \{\, x : D(x) > 0 \,\}$, accurate within a multiplicative factor.

It is not difficult to see that each of the first three problems generalizes the previous, and is therefore at least as hard. All of these tasks are known to require sample complexity $n^{\Omega(1)}$

[1] For this problem, it is typically assumed that all points in the support have probability mass at least $\Omega(1)/n$, as without such guarantee it becomes impossible to give any non-trivial estimate (consider for instance a distribution $D$ such that $D(i) \propto 1/2^{in}$).

in the standard sampling model (SAMP); yet, as prior work [11, 9] shows, their complexity decreases tremendously when one allows the stronger type of access to the distribution(s) provided by a conditional sampling oracle (COND). For the problems of uniformity testing and identity testing, the sample complexity even becomes a constant provided the testing algorithm is allowed to be *adaptive* (i.e. when the next queries it makes can depend on the samples it previously obtained).

### Testing uniformity and identity

Given the complete description of a distribution $D^*$ over $[n]$, a parameter $\varepsilon > 0$, and oracle access to a distribution $D$, identity testing asks to distinguish the case $D_1 = D^*$ from where their total variation distance $\mathrm{d}_{\mathrm{TV}}(D, D^*)$ is at least $\varepsilon$. This is a generalization of uniformity testing, where $D^*$ is taken to be the uniform distribution over $[n]$. The complexity of these tasks is well-understood in the sampling model; in particular, it is known that for both uniformity and identity testing $\Theta\big(\sqrt{n}/\varepsilon^2\big)$ samples are necessary and sufficient (see [15, 5, 20, 29] for the tight bounds on these problems).

The uniformity testing problem emphasizes the additional flexibility granted by conditional sampling: as Canonne, Ron, and Servedio [9] showed, in this setting only $\tilde{O}\big(1/\varepsilon^2\big)$ adaptive queries now suffice (and this is optimal, up to logarithmic factors). They further prove that identity testing has constant sample complexity as well, namely $\tilde{O}\big(1/\varepsilon^4\big)$ – very recently improved to a near-optimal $\tilde{O}\big(1/\varepsilon^2\big)$ by Falahatgar et al. [13]. The power of the COND model is evident from the fact that a task requiring polynomially many samples in the standard model can now be achieved with a number of samples *independent of the domain size* $n$.

Focusing on the case of non-adaptive algorithms, Chakraborty et al. [11] describe a $\mathrm{poly}(\log n, 1/\varepsilon)$-query tester for uniformity, showing that even without the full power of conditional queries one can still get an exponential improvement over the standard sampling setting. They also obtain an $\Omega(\log \log n)$ lower bound for this problem, and leave open the possibility of improving this lower bound up to a logarithmic dependence. The present work answers this question, establishing that any non-adaptive uniformity tester must perform $\Omega(\log n)$ conditional queries.

### Testing equivalence

A natural generalization of these two testing problems is the question of *equivalence testing*, defined as follows. Given oracle access to two unknown distributions $D_1$ and $D_2$ over $[n]$ and a parameter $\varepsilon > 0$, equivalence testing asks to distinguish between the cases $D_1 = D_2$ and $\mathrm{d}_{\mathrm{TV}}(D_1, D_2) > \varepsilon$. This problem has been extensively studied over the past decade, and its sample complexity is now known to be $\Theta(\max(n^{2/3}/\varepsilon^{4/3}, \sqrt{n}/\varepsilon^2))$ in the sampling model [5, 30, 12].

In the COND setting, Canonne, Ron, and Servedio showed that equivalence testing is possible with only $\mathrm{poly}(\log n, 1/\varepsilon)$ queries. Concurrent to our work, Falahatgar et al. [13] brought this upper bound down to $\tilde{O}\big((\log \log n)/\varepsilon^5\big)$, a *doubly exponential* improvement over the $n^{\Omega(1)}$ samples needed in the standard sampling model. However, these results still left open the possibility of a constant query complexity: given that both uniformity and identity testing admit constant-query testers, it is natural to wonder where equivalence testing lies[2].

This question was explicitly posed by Fischer at the Bertinoro Workshop on Sublinear Algorithms 2014 [14]: in this paper, we make decisive progress in answering it, ruling out

[2] It is worth noting that an $\Omega(\log^c n)$ lower bound was known for equivalence testing in a weaker version of the conditional oracle, PAIRCOND (where the tester's queries are restricted to being either $[n]$ or subsets of size 2 [9]).

the possibility of any constant-query tester for equivalence. Along with the upper bound of Falahatgar et al. [13], our results essentially settle the dependence on the domain size, showing that $(\log\log n)^{\Theta(1)}$ samples are both necessary and sufficient.

### Support size estimation

Finally, the question of approximating the support size of a distribution has been considered by Raskhodnikova et al. [21], where it was shown that obtaining *additive* estimates requires sample complexity almost linear in $n$. Subsequent work by Valiant and Valiant [28, 26] settles the question, establishing that an $n/\log n$ dependence is both necessary and sufficient. Note that the proof of their lower bound translates to multiplicative approximations as well, as they rely on the hardness of distinguishing a distribution with support $s \leq n$ from a distribution with support $s + \varepsilon n \geq (1+\varepsilon)s$. To the best of our knowledge, the question of getting a multiplicative-factor estimate of the support size of a distribution given conditional sampling access has not been previously considered. We provide upper and lower bounds for both the adaptive and non-adaptive versions of this problem.

## 1.2 Our results

In this work, we make significant progress in each of the problems introduced in the previous section, yielding a better understanding of their intrinsic query complexities. We prove four results pertaining to the sample complexity of equivalence testing, support size estimation, and uniformity testing in the COND framework. Our main result on the sample complexity of equivalence testing is presented in this extended abstract, the details of the other results are available in the full version of this paper [1].

Our main result considers the sample complexity of testing equivalence with adaptive queries under the COND model, resolving in the negative the question of whether constant-query complexity was achievable [14]. More precisely, we prove the following theorem:

▶ **Theorem 1.1** (Testing Equivalence). *Any* adaptive *algorithm which, given* COND *access to unknown distributions $D_1, D_2$ on $[n]$, distinguishes with probability at least $2/3$ between* (a) $D_1 = D_2$ *and* (b) $\mathrm{d}_{\mathrm{TV}}(D_1, D_2) \geq \frac{1}{4}$, *must have query complexity* $\Omega\big(\sqrt{\log\log n}\big)$.

Combined with the recent $\tilde{O}(\log\log n)$ upper bound of Falahatgar et al. [13], this almost settles the sample complexity of this question. Furthermore, as the related task of identity

**Table 1** Summary of results. Note that the lower bound (†) can also be easily derived from our lower bound on testing equivalence.

| Problem | COND model | Standard model |
|---|---|---|
| Are $D_1, D_2$ (both unknown) equivalent? (adaptive) | $\tilde{O}\Big(\frac{\log\log n}{\varepsilon^5}\Big)$ [13]<br>$\Omega\big(\sqrt{\log\log n}\big)$ [this work] | $\Theta\Big(\max\Big(\frac{n^{2/3}}{\varepsilon^{4/3}}, \frac{n^{1/2}}{\varepsilon^2}\Big)\Big)$ [12] |
| What is the support size of $D$? (adaptive) | $\tilde{O}\Big(\frac{\log\log n}{\varepsilon^3}\Big)$ [this work]<br>$\Omega\big(\sqrt{\log\log n}\big)$ [11] (†) | $\Theta\Big(\frac{n}{\log n}\Big)$ [26] |
| What is the support size of $D$? (non-adaptive) | $O(\mathrm{poly}(\log n, 1/\varepsilon))$ [this work]<br>$\Omega(\log n)$ [this work] | |
| Is $D$ uniform over the domain? (non-adaptive) | $\tilde{O}\Big(\frac{\log^5 n}{\varepsilon^6}\Big)$ [11]<br>$\Omega(\log n)$ [this work] | $\Theta\Big(\frac{\sqrt{n}}{\varepsilon^2}\Big)$ [15, 5, 20] |

testing *can* be performed with a constant number of queries in the conditional sampling model, this demonstrates an intriguing and intrinsic difference between the two problems. Our result can also be interpreted as showing a fundamental distinction from the usual sampling model, where both identity and equivalence testing have polynomial sample complexity.

Next, we establish a logarithmic lower bound on *non-adaptive* support size estimation, for any factor larger than a fixed constant. This improves on the result of Chakraborty et al. [11], which gave a doubly logarithmic lower bound for constant factor support-size estimation.

▶ **Theorem 1.2** (Non-Adaptive Support Size Estimation). *Any non-adaptive algorithm which, given* COND *access to an unknown distribution $D$ on $[n]$, estimates the size of its support up to a factor $\gamma$ must have query complexity $\Omega\left(\frac{\log n}{\log \gamma}\right)$, for any $\gamma \geq \sqrt{2}$.*

Moreover, the approach used to prove this theorem also implies an analogous lower bound on *non-adaptive* uniformity testing in the conditional model, answering a conjecture of Chakraborty et al. [11]:

▶ **Theorem 1.3** (Non-Adaptive Uniformity Testing). *Any non-adaptive algorithm which, given* COND *access to an unknown distribution $D$ on $[n]$, distinguishes with probability at least $2/3$ between* (a) $D = \mathcal{U}$ *and* (b) $\mathrm{d}_{\mathrm{TV}}(D,\mathcal{U}) \geq \frac{1}{4}$, *must have query complexity $\Omega(\log n)$.*

We note that these results complement polylog$(n)$-query upper bounds, the former of which we sketch in the full version of this paper, and the latter obtained by Chakraborty et al. [11]. This shows that both of these problems have query complexity $\log^{\Theta(1)} n$ in the non-adaptive case.

Finally, we also show an upper bound for *adaptive* support size estimation. Specifically, we provide a $\tilde{O}(\log \log n)$-query algorithm for support size estimation. This shows that the question becomes *double exponentially* easier when conditional samples are allowed.

▶ **Theorem 1.4** (Adaptive Support Size Estimation). *Let $\tau > 0$ be any constant. There exists an adaptive algorithm which, given* COND *access to an unknown distribution $D$ on $[n]$ which has minimum non-zero probability $\tau/n$ and accuracy parameter $\varepsilon$ makes $\tilde{O}\big((\log \log n)/\varepsilon^3\big)$ queries to the oracle and outputs a value $\tilde{\omega}$ such that the following holds. With probability at least $2/3$, $\tilde{\omega} \in [\frac{1}{1+\varepsilon} \cdot \omega, (1+\varepsilon) \cdot \omega]$, where $\omega = |\mathrm{supp}(D)|$.*

### 1.2.1 Relation to the Ron-Tsur model

Recent work of Ron and Tsur [22] studies a model which is slightly stronger than ours. In their setting, the algorithm still performs queries consisting of a subset of the domain. However, the algorithm is also given the promise that the distribution is uniform on a subset of the domain, and whenever a query set contains 0 probability mass the oracle explicitly indicates this is the case. Their paper provides a number of results for support size estimation in this model.

We point out two connections between our work and theirs. First, our $\Omega(\log n)$ lower bound for non-adaptive support size estimation (Theorem 1.2) leads to the same lower bound for the problem in the model of Ron and Tsur. Although lower bounds in the conditional sampling setting do not apply directly to theirs, we note that our construction and analysis still carry over. This provides a nearly tight answer to this question, which was left unanswered in their paper. Also, our $\tilde{O}(\log \log n)$-query algorithm for adaptive support size estimation (Theorem 1.4) can be seen as generalizing their result to the weaker

conditional sampling model (most significantly, when we are not given the promise that the distribution be uniform).

## 1.3 Techniques and proof ideas

We now provide an overview of the techniques and arguments used to prove our results.

### Lower bound on adaptive equivalence testing

In order to prove our main $\omega(1)$ lower bound on the query complexity of testing equivalence in the conditional sampling model, we have to deal with one main conceptual issue: *adaptivity*. While the standard sampling model does not, by definition, allow any choice on what the next query to the oracle should be, this is no longer the case for COND algorithms. Quantifying the power that this grants an algorithm makes things much more difficult. To handle this point, we follow the approach of Chakraborty et al. [11] and focus on a restricted class of algorithms they introduce, called "core adaptive testers" (see Section 2.2 for a formal definition). They show that this class of testers is equivalent to general algorithms for the purpose of testing a broad class of properties, namely those which are invariant to any permutation of the domain. Using this characterization, it remains for us to show that none of these structurally much simpler core testers can distinguish whether they are given conditional access to (a) a pair of random identical distributions $(D_1, D_1)$, or (b) two distributions $(D_1, D_2)$ drawn according to a similar process, which are far apart.

At a high level, our lower bound works by designing instances where the property can be tested if and only if the support size is known to the algorithm. Our construction randomizes the support size by embedding the instance into a polynomially larger domain. Since the algorithm is only allowed a small number of queries, Yao's Principle allows us to argue that, with high probability, a deterministic algorithm is unable to "guess" the support size. This separates queries into several cases. First, in a sense we make precise, it is somehow "predictable" whether or not a query will return an element we have previously observed. If we do, it is similarly predictable *which* element the query will return. On the other hand, if we observe a fresh element, the query set is either "too small" or "too large." In the former case, the query will entirely miss the support, and the sampling process is identical for both types of instance. In the latter case, the query will hit a large portion of the support, and the amount of information gleamed from a single sample is minimal.

At a lower level, this process itself is reminiscent of the lower bound construction of Canonne, Ron, and Servedio [9] on testing identity (with a PAIRCOND oracle), with one pivotal twist. As in their work, both $D_1$ and $D_2$ are uniform within each of $\omega(1)$ "buckets" whose size grows exponentially and are grouped into "bucket-pairs." Then, $D_2$ is obtained from $D_1$ by internally redistributing the probability mass of each pair of buckets, so that the total mass of each pair is preserved but each particular bucket has mass going up or down by a constant factor (see Section 3.1 for details of the construction). However, we now add a final step, where in both $D_1$ and $D_2$ the resulting distribution's support is *scaled by a random factor*, effectively reducing it to a (randomly) negligible fraction of the domain. Intuitively, this last modification has the role of "blinding" the testing algorithm: we argue that unless its queries are on sets whose size somehow match (in a sense formalized in Section 3.2) this random size of the support, the sequences of samples it will obtain under $D_1$ and $D_2$ are almost identically distributed. The above discussion crucially hides many significant aspects and technical difficulties which we address in Section 3. Moreover, we observe that the lower bound we obtain seems to be optimal with regard to our proofs techniques (specifically, to

the decision tree approach), and not an artifact of our lower bound instances. Namely, there appear to be conceptual barriers to strengthening our result, which would require new ideas.

### Lower bound on non-adaptive support size estimation

Turning to the (non-adaptive) lower bound of Theorem 1.2, we define two families of distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, where an instance is either a draw $(D_1, D_2)$ from $\mathcal{D}_1 \times \mathcal{D}_2$, or simply $(D_1, D_1)$. Any distribution in $\mathcal{D}_2$ has support size $\gamma$ times that of its corresponding distribution in $\mathcal{D}_1$. Yet, we argue that no non-adaptive *deterministic* tester making too few queries can distinguish between these two cases, as the tuple of samples it will obtain from $D_1$ or (the corresponding) $D_2$ is almost identically distributed (where the randomness is over the choice of the instance itself). To show this last point, we analyze separately the case of "small" queries (conditioning on sets which turn out to be much smaller than the actual support size, and thus with high probability will not even intersect it) and the "big" ones (where the query set $A$ is so big in front of the support size $S$ that a uniform sample from $A \cap S$ is essentially indistinguishable from a uniform sample from $A$). We conclude the proof by invoking Yao's Principle, carrying the lower bound back to the setting of non-adaptive *randomized* testers.

Interestingly, this argument essentially gives us Theorem 1.3 "for free:" indeed, the big-query-set case above is handled by proving that the distribution of samples returned on those queries is indistinguishable, both for $\mathcal{D}_1$ and $\mathcal{D}_2$, from samples obtained from the *actual* uniform distribution. Considering again the small-query-set case separately, this allows us to argue that a random distribution from (say) $\mathcal{D}_1$ is indistinguishable from uniform.

### Upper bound on support size estimation

Our algorithm for estimating the support size to a constant factor (Theorem 1.4) is simple in spirit, and follows a guess-and-check strategy. In more detail, it first obtains a "reference point" *outside* the support, to check whether subsequent samples it may consider belong to the support. Then, it attempts to find a *rough upper bound* on the size of the support, of the form $2^{2^j}$ (so that only $\log \log n$ many options have to be considered); by using its reference point to check if a uniform random subset of this size contains, as it should, at least one point from the support. Once such an upper bound has been obtained using this double-exponential strategy, a refined bound is then obtained via a binary search on the new range of values for the exponent, $\{2^{j-1}, \ldots, 2^j\}$. Not surprisingly, our algorithm draws on similar ideas as in [22, 25], with some additional machinery to supplement the differences in the models. Interestingly, as a side-effect, this upper bound shows our analysis of Theorem 1.1 to be tight up to a quadratic dependence. Indeed, the lower bound construction we consider (see Section 3.1) can be easily "defeated" if an estimate of the support size is known, and therefore cannot yield better than a $\Omega(\log \log n)$ lower bound. Similarly, this also shows that the adaptive lower bound for support size estimation of Chakraborty et al. [11] is also tight up to a quadratic dependence.

### Organization

The rest of the paper describes details and proofs of the results mentioned in the above discussion. In Section 2, we introduce the necessary definitions and some of the tools we shall use. Section 3 covers our main result on adaptive equivalence testing, Theorem 1.1. Further details on our other results are available in the full version of this paper.

# 2 Preliminaries

## 2.1 Notation and sampling models

All throughout this paper, we denote by $[n]$ the set $\{1, \ldots, n\}$, and by log the logarithm in base 2. A probability distribution over a (countable) domain $[n]$ is a non-negative function $D\colon [n] \to [0,1]$ such that $\sum_{x\in[n]} D(x) = 1$. We denote by $\mathcal{U}(S)$ the uniform distribution on a set $S$. Given a distribution $D$ over $[n]$ and a set $S \subseteq [n]$, we write $D(S)$ for the total probability mass $\sum_{x\in S} D(x)$ assigned to $S$ by $D$. Finally, for $S \subseteq [n]$ such that $D(S) > 0$, we denote by $D_S$ the conditional distribution of $D$ restricted to $S$, that is $D_S(x) = \frac{D(x)}{D(S)}$ for $x \in S$ and $D_S(x) = 0$ otherwise.

As is usual in distribution testing, in this work the distance between two distributions $D_1, D_2$ on $[n]$ will be the *total variation distance*:

$$\mathrm{d_{TV}}(D_1, D_2) \stackrel{\text{def}}{=} \frac{1}{2}\|D_1 - D_2\|_1 = \frac{1}{2}\sum_{x\in[n]} |D_1(i) - D_2(i)| = \max_{S\subseteq[n]} (D_1(S) - D_2(S)) \tag{1}$$

which takes value in $[0,1]$.

In this work, we focus on the setting of *conditional access* to the distribution, as introduced and studied in [11, 9]. We reproduce below the corresponding definition of a conditional oracle, henceforth referred to as COND:

▶ **Definition 2.1** (Conditional access model)**.** Fix a distribution $D$ over $[n]$. A COND *oracle for* $D$, denoted $\mathsf{COND}_D$, is defined as follows: the oracle takes as input a *query set* $S \subseteq [n]$, chosen by the algorithm, that has $D(S) > 0$. The oracle returns an element $i \in S$, where the probability that element $i$ is returned is $D_S(i) = D(i)/D(S)$, independently of all previous calls to the oracle.

Note that as described above the behavior of $\mathsf{COND}_D(S)$ is undefined if $D(S) = 0$, i.e., the set $S$ has zero probability under $D$. Various definitional choices could be made to deal with this. These choice do not do not make significant difference in most situations, as most (adaptive) algorithms can always include in their next queries a sample previously obtained; while our lower bounds can be thought of as putting exponentially small probability mass of elements outside the support. For this reason, and for convenience, we shall hereafter assume, following Chakraborty et al., that the oracle returns in this case a sample uniformly distributed in $S$.

Finally, recall that a *property* $\mathcal{P}$ of distributions over $[n]$ is a set consisting of all distributions that have the property. The distance from $D$ to a property $\mathcal{P}$, denoted $\mathrm{d_{TV}}(D, \mathcal{P})$, is then defined as $\inf_{D'\in\mathcal{P}} \mathrm{d_{TV}}(D, \mathcal{P})$. We use the standard definition of testing algorithms for properties of distributions over $[n]$, tailored for the setting of conditional access to an unknown distribution:

▶ **Definition 2.2** (Property tester)**.** Let $\mathcal{P}$ be a property of distributions over $[n]$. A *$t$-query* COND *testing algorithm for* $\mathcal{P}$ is a randomized algorithm $\mathcal{T}$ which takes as input $n$, $\varepsilon \in (0,1]$, as well as access to $\mathsf{COND}_D$. After making at most $t(\varepsilon, n)$ calls to the oracle, $\mathcal{T}$ either outputs ACCEPT or REJECT, such that the following holds:

[2] Recall that a non-adaptive tester is an algorithm whose queries do not depend on the answers obtained from previous ones, but only on its internal randomness. Equivalently, it is a tester that can commit "upfront" to all the queries it will make to the oracle.

- if $D \in \mathcal{P}$, $\mathcal{T}$ outputs ACCEPT with probability at least $2/3$;
- if $\mathrm{d}_{\mathrm{TV}}(D, \mathcal{P}) \geq \varepsilon$, $\mathcal{T}$ outputs REJECT with probability at least $2/3$.

We observe that the above definitions can be straightforwardly extended to the more general setting of *pairs* of distributions, where given independent access to two oracles $\mathsf{COND}_{D_1}$, $\mathsf{COND}_{D_2}$ the goal is to test whether $(D_1, D_2)$ satisfies a property (now a set of pairs of distributions). This will be the case in Section 3, where we will consider equivalence testing, that is the property $\mathcal{P}_{\mathrm{eq}} = \{ (D_1, D_2) : D_1 = D_2 \}$.

## 2.2 Adaptive Core Testers

In order to deal with adaptivity in our lower bounds, we will use ideas introduced by Chakraborty et al. [11]. These ideas, for the case of *label-invariant* properties[3] allow one to narrow down the range of possible testers and focus on a restricted class of such algorithms called *adaptive core testers*. These core testers do not have access to the full information of the samples they draw, but instead only get to see the relations (inclusions, equalities) between the queries they make and the samples they get. Yet, Chakraborty et al. [11] show that any tester for a label-invariant property can be converted into a core tester with same query complexity; thus, it is enough to prove lower bounds against this – seemingly – weaker class of algorithms.

We here rephrase the definitions of a core tester and the view they have of the interaction with the oracle (the *configuration* of the samples), tailored to our setting.

▶ **Definition 2.3** (Atoms and partitions)**.** Given a family $\mathcal{A} = (A_1, \ldots, A_t) \subseteq [n]^t$, the *atoms* generated by $\mathcal{A}$ are the (at most) $2^t$ distinct sets of the form $\bigcap_{r=1}^{t} C_r$, where $C_r \in \{A_r, [n] \backslash A_r\}$. The family of all such atoms, denoted $\mathrm{At}(\mathcal{A})$, is the *partition* generated by $\mathcal{A}$.

This definition essentially captures "all sets (besides the $A_i$'s) about which something can be learnt from querying the oracle on the sets of $\mathcal{A}$." Now, given such a sequence of queries $\mathcal{A} = (A_1, \ldots, A_t)$ and pairs of samples $\mathbf{s} = ((s_1^{(1)}, s_1^{(2)}), \ldots, (s_t^{(1)}, s_t^{(2)})) \in A_1^2 \times \cdots \times A_t^2$, we would like to summarize "all the label-invariant information available to an algorithm that obtains $((s_1^{(1)}, s_1^{(2)}), \ldots, (s_t^{(1)}, s_t^{(2)}))$ upon querying $A_1, \ldots, A_t$ for $D_1$ and $D_2$." This calls for the following definition:

▶ **Definition 2.4** ($t$-configuration)**.** Given $\mathcal{A} = (A_1, \ldots, A_t)$ and $\mathbf{s} = ((s_j^{(1)}, s_j^{(2)}))_{1 \leq j \leq t}$ as above, the *$t$-configuration of* $\mathbf{s}$ consists of the $6t^2$ bits indicating, for all $1 \leq i, j \leq t$, whether

- $s_i^{(k)} = s_j^{(\ell)}$, for $k, \ell \in \{1, 2\}$; and (relations between samples)
- $s_i^{(k)} \in A_j$, for $k \in \{1, 2\}$. (relations between samples and query sets)

In other terms, it summarizes which is the unique atom $S_i \in \mathrm{At}(\mathcal{A})$ that contains $s_i^{(k)}$, and what collisions between samples have been observed.

As aforementioned, the key idea is to argue that, without loss of generality, one can restrict one's attention to algorithms that only have access to $t$-configurations, and generate their queries in a specific (albeit adaptive) fashion:

▶ **Definition 2.5** (Core adaptive tester)**.** A *core adaptive distribution tester* for pairs of distributions is an algorithm $\mathcal{T}$ that acts as follows.

[3] Recall that a property is label-invariant (or *symmetric*) if it is closed under relabeling of the elements of the support. More precisely, a property of distributions (resp. pairs of distributions) $\mathcal{P}$ is label-invariant if for any distribution $D \in \mathcal{P}$ (resp. $(D_1, D_2) \in \mathcal{P}$) and permutation $\sigma$ of $[n]$, one has $D \circ \sigma \in \mathcal{P}$ (resp. $(D_1 \circ \sigma, D_2 \circ \sigma) \in \mathcal{P}$).

- In the $i$-th phase, based only on its own internal randomness and the configuration of the previous queries $A_1, \ldots, A_{i-1}$ and samples obtained $(s_1^{(1)}, s_1^{(2)}), \ldots, (s_{i-1}^{(1)}, s_{i-1}^{(2)})$ – whose labels it does not actually know, $\mathcal{T}$ provides:
  - a number $k_i^A$ for each $A \in \mathrm{At}(A_1, \ldots, A_{i-1})$, between 0 and $\left|A \setminus \{s_j^{(1)}, s_j^{(2)}\}_{1 \le j \le i-1}\right|$ ("how many *fresh, not-already-seen* elements of each particular atom $A$ should be included in the next query")
  - sets $K_i^{(1)}, K_i^{(2)} \subseteq \{1, \ldots, i-1\}$ ("which of the samples $s_1^{(k)}, \ldots, s(k)_{i-1}$ (whose label is unknown to the tester, but referred to by the index of the query it got them) will be included in the next query").
- based on these specifications, the next query $A_i$ is drawn (but not revealed to $\mathcal{T}$) by
  - drawing uniformly at random a set $\Lambda_i$ in

    $$\left\{ \Lambda \subseteq [n] \setminus \{s_j^{(1)}, s_j^{(2)}\}_{1 \le j \le i-1} \,:\, \forall A \in \mathrm{At}(A_1, \ldots, A_{i-1}),\, |\Lambda \cap A| = k_i^A \right\} .$$

    That is, among all sets, containing only "fresh elements," whose intersection with each atom contains as many elements as $\mathcal{T}$ requires.
  - adding the selected previous samples to this set:

    $$\Gamma_i \stackrel{\text{def}}{=} \left\{ s_j^{(1)} \,:\, j \in K_i^{(1)} \right\} \cup \left\{ s_j^{(2)} \,:\, j \in K_i^{(2)} \right\} \;; \quad A_i \stackrel{\text{def}}{=} \Lambda_i \cup \Gamma_i \;.$$

    This results in a set $A_i$, not fully known to $\mathcal{T}$ besides the samples it already got and decided to query again; in which the *labels* of the fresh elements are unknown, but the *proportions* of elements belonging to each atom are known.
- samples $s_i^{(1)} \sim (D_1)_{A_i}$ and $s_i^{(2)} \sim (D_2)_{A_i}$ are drawn (but not disclosed to $\mathcal{T}$). This defines the $i$-configuration of $A_1, \ldots, A_i$ and $(s_1^{(1)}, s_1^{(2)}), \ldots, (s_i^{(1)}, s_i^{(2)})$, which is revealed to $\mathcal{T}$. Put differently, the algorithm only learns (i) to which of the $A_\ell$'s the new sample belongs, and (ii) if it is one of the previous samples, in which stage(s) and for which of $D_1, D_2$ it has already seen it.

After $t = t(\varepsilon, n)$ such stages, $\mathcal{T}$ outputs either ACCEPT or REJECT, based only on the configuration of $A_1, \ldots, A_t$ and $(s_1^{(1)}, s_1^{(2)}), \ldots, (s_t^{(1)}, s_t^{(2)})$ (which is all the information it ever had access to).

Note that in particular, $\mathcal{T}$ does not know the labels of samples it got, nor the actual queries it makes: it knows all about their sizes and sizes of their intersections, but not the actual "identity" of the elements they contain.

## 2.3 On the use of Yao's Principle in our lower bounds

We recall Yao's Principle (e.g., see Chapter 2.2 of [19]), a technique which is ubiquitous in the analysis of randomized algorithms. Consider a set $S$ of instances of some problem: what this principle states is that the worst-case expected cost of a randomized algorithm on instances in $S$ is lower-bounded by the expected cost of the best deterministic algorithm on an instance drawn randomly from $S$.

As an example, we apply it in a standard way for the proofs of Theorems 1.2 and 1.3: instead of considering a randomized algorithm working on a fixed instance, we instead analyze a *deterministic* algorithm working on a *random* instance. (We note that, importantly, the randomness in the samples returned by the COND oracle is "external" to this argument, and these samples behave identically in an application of Yao's Principle.)

On the other hand, our application for the proof of Theorem 1.1 in Section 3 is slightly different, due to our use of adaptive core testers. Once again, we focus on deterministic

algorithms working on random instances, and the randomness in the samples is external and therefore unaffected by Yao's Principle. However, we stress that the randomness in the choice of the set $\Lambda_i$ is also external to the argument, and therefore unaffected – similar to the randomness in the samples, the algorithm has no control here. Another way of thinking about this randomness is via another step in the distribution over instances: after an instance (which is a pair of distributions) is randomly chosen, we permute the labels on the elements of the distribution's domain uniformly at random. We note that since the property in question is label-invariant, this does not affect its value. We can then use the model as stated in Section 2.2 for ease of analysis, observing that this can be considered an application of the principle of deferred decisions (as in Chapter 3.5 of [19]).

# 3 A Lower Bound for Equivalence Testing

We prove our main lower bound on the sample complexity of testing equivalence between unknown distributions. We construct two priors $\mathcal{Y}$ and $\mathcal{N}$ over *pairs* of distributions $(D_1, D_2)$ over $[n]$. $\mathcal{Y}$ is a distribution over pairs of distributions of the form $(D, D)$, namely the case when the distributions are identical. Similarly, $\mathcal{N}$ is a distribution over $(D_1, D_2)$ with $\mathrm{d}_{\mathrm{TV}}(D_1, D_2) \geq \frac{1}{4}$. We then show that no algorithm $\mathcal{T}$ making $O\left(\sqrt{\log \log n}\right)$ queries to $\mathsf{COND}^{D_1}, \mathsf{COND}^{D_2}$ can distinguish between a draw from $\mathcal{Y}$ and $\mathcal{N}$ with constant probability (over the choice of $(D_1, D_2)$, the randomness in the samples it obtains, and its internal randomness). We describe the construction of $\mathcal{Y}$ and $\mathcal{N}$ in Section 3.1, and provide a detailed analysis in Section 3.2. (Due to space constraints, some of the proofs are deferred to the full version of the paper.)

## 3.1 Construction

We now summarize how a pair of distribution is constructed under $\mathcal{Y}$ and $\mathcal{N}$. (Each specific step will be described in more detail in the subsequent paragraphs.)

1. **Effective Support**
   - **a.** Pick $k_b$ from the set $\{0, 1, \ldots, \frac{1}{2}\log n\}$ at random.
   - **b.** Let $b = 2^{k_b}$ and $m \stackrel{\text{def}}{=} b \cdot n^{1/4}$.
2. **Buckets**
   - **a.** $\rho$ and $r$ are chosen with $\sum_{i=1}^{2r} \rho^i = n^{1/4}$.
   - **b.** Divide $\{1, \ldots, m\}$ into intervals $B_1, \ldots, B_{2r}$ with $|B_i| = b \cdot \rho^i$.
3. **Distributions**
   - **a.** Assign probability mass $\frac{1}{2r}$ uniformly over $B_i$ to generate distribution $D_1$.
   - **b.** **(i)** Let $\pi_1, \ldots, \pi_r$ be independent 0/1 with $\Pr(\pi_i = 0) = \frac{1}{2}$.
     **(ii)** If $\pi_i = 0$, assign probability mass $\frac{1}{4r}$ and $\frac{3}{4r}$ over $B_{2i-1}$ and $B_{2i}$ respectively, else $\frac{3}{4r}$ and $\frac{1}{4r}$ respectively. This generates a distribution $D_2$.
3. **Support relabeling**
   - **a.** Pick a permutation $\sigma \in S_n$ of the *total* support $n$.
   - **b.** Relabel the symbols of $D_1$ and $D_2$ according to $\sigma$.
4. **Output:** Generate $(D_1, D_1)$ for $\mathcal{Y}$, and $(D_1, D_2)$ otherwise.

We now describe the various steps of the construction in greater detail.

**Effective support.** Both $D_1$ and $D_2$, albeit distributions on $[n]$, will have (common) *sparse* support. The support size is taken to be $m \stackrel{\text{def}}{=} b \cdot n^{1/4}$. Note that, from the above

definition, $m$ is chosen uniformly at random from products of $n^{1/4}$ with powers of 2, resulting in values in $[n^{1/4}, n^{3/4}]$.

In this step $b$ will act as a random scaling factor. The objective of this random scaling is to induce uncertainty in the algorithm's knowledge of the true support size of the distributions, and to prevent it from leveraging this information to test equivalence. In fact one can verify that the class of distributions induced for a single value of $b$, namely all distributions have the same value of $m$, then one can distinguish the $\mathcal{Y}$ and $\mathcal{N}$ cases with only $O(1)$ conditional queries.

**Buckets.** Our construction is inspired by the lower bound of Canonne, Ron, and Servedio [9, Theorem 8] for the more restrictive PAIRCOND access model. We partition the support in $2r$ consecutive intervals (henceforth referred to as *buckets*) $B_1, \ldots, B_{2r}$, where the size of the $i$-th bucket is $b\rho^i$. We note that $r$ and $\rho$ will be chosen such that $\sum_{i=1}^{2r} b\rho^i = bn^{1/4}$, i.e., the buckets fill the effective support.

**Distributions.** We output a pair of distributions $(D_1, D_2)$. Each distribution that we construct is uniform within any particular bucket $B_i$. In particular, the first distribution assigns the same mass $\frac{1}{2r}$ to each bucket. Therefore, points within $B_i$ have the same probability mass $\frac{1}{(2rb\rho^i)}$. For the $\mathcal{Y}$ case, the second distribution is identical to the first. For the $\mathcal{N}$ case, we pair buckets in $r$ consecutive *bucket-pairs* $\Pi_1, \ldots, \Pi_r$, with $\Pi_i = B_{2i-1} \cup B_{2i}$. For the second distribution $D_2$, we consider the same buckets as $D_1$, but repartition the mass $1/r$ *within* each $\Pi_i$. More precisely, in each pair, one of the buckets gets now total probability mass $\frac{1}{4r}$ while the other gets $\frac{3}{4r}$ (so that the probability of every point is either decreased by a factor $\frac{1}{2}$ or increased by $\frac{3}{2}$). The choice of which goes up and which goes down is done uniformly and independently at random for each bucket-pair determined by the random choices of $\pi_i$'s.

**Random relabeling.** The final step of the construction randomly relabels the symbols, namely is a random injective map from $[m]$ to $[n]$. This is done to ensure that no information about the individual symbol labels can be used by the algorithm for testing. For example, without this the algorithm can consider a few symbols from the first bucket and distinguish the $\mathcal{Y}$ and $\mathcal{N}$ cases. As mentioned in Section 2.3, for ease of analysis, the randomness in the choice of the permutation is, in some sense, deferred to the randomness in the choice of $\Lambda_i$ during the algorithm's execution.

**Summary.** A no-instance $(D_1, D_2)$ is thus defined by the following parameters: the support size $m$, the vector $(\pi_1, \ldots, \pi_m) \in \{0,1\}^r$ (which only impacts $D_2$), and the final permutation $\sigma$ of the domain. A yes-instance $(D_1, D_1)$ follows an identical process, however, $\pi$ has no influence on the final outcome. See Figure 1 for an illustration of such a $(D_1, D_2)$ when $\sigma$ is the identity permutation and thus the distribution is supported over the first $m$ natural numbers.

**Values for $\rho$ and $r$.** By setting $r = \frac{\log n}{8 \log \rho} + O(1)$, we have as desired $\sum_{i=1}^{2r} |B_i| = m$ and there is a factor $(1 + o(1))n^{1/4}$ between the height of the first bucket $B_1$ and the one of the last, $B_{2r}$. It remains to choose the parameter $\rho$ itself; we shall take it to be $2^{\sqrt{\log n}}$, resulting in $r = \frac{1}{8}\sqrt{\log n} + O(1)$. (Note that for the sake of the exposition, we ignore technical details such as the rounding of parameters, e.g. bucket sizes; these can be easily taken care of at the price of cumbersome case analyses, and do not bring much to the argument.)

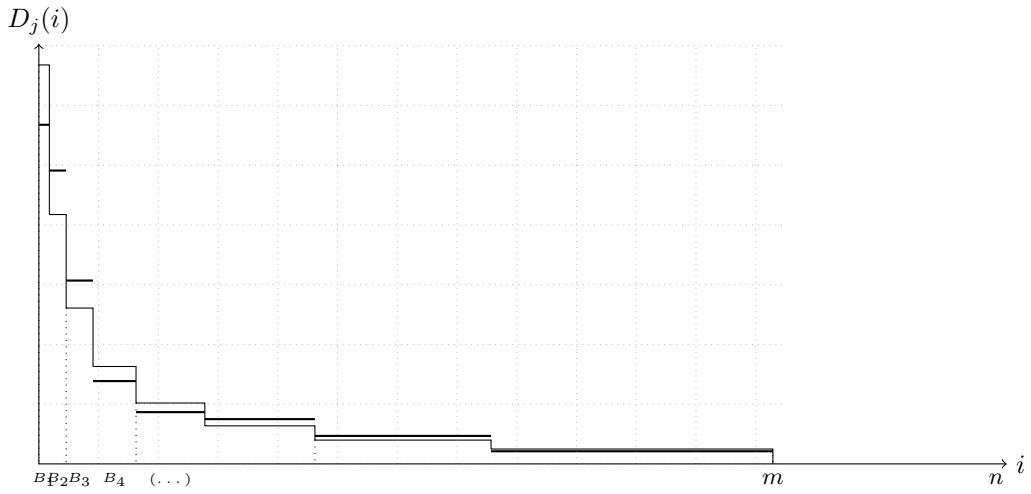


**Figure 1** A no-instance $(D_1, D_2)$ (before permutation).

## 3.2 Analysis

We now prove our main lower bound, by analyzing the behavior of core adaptive testers (as per Definition 2.5) on the families $\mathcal{Y}$ and $\mathcal{N}$ from the previous section. In Section 3.2.1, we argue that, with high probability, the sizes of the queries performed by the algorithm satisfy some specific properties. Conditioned upon this event, in Section 3.2.2, we show that the algorithm will get similar information from each query, whether it is running on a yes-instance or a no-instance.

Before moving to the heart of the argument, we state the following fact, straightforward from the construction of our no-instances:

▶ **Fact 3.1.** *For any $(D_1, D_2)$ drawn from $\mathcal{N}$, one has $\mathrm{d}_{\mathrm{TV}}(D_1, D_2) = 1/4$.*

Moreover, as allowing more queries can only increase the probability of success, we hereafter focus on a core adaptive tester that performs exactly $q = \frac{1}{10}\sqrt{\log\log n}$ (adaptive) queries; and will show that it can only distinguish between yes- and no-instances with probability $o(1)$.

### 3.2.1 Banning "bad queries"

As mentioned in Section 3.1, the draw of a yes- or no-instance involves a random scaling of the size of the support of the distributions, meant to "blind" the testing algorithm. Recall that a testing algorithm is specified by a decision tree, which at step $i$, specifies how many unseen elements from each atom to include in the query ($\{k_i^A\}$) and which previously seen elements to include in the query (sets $K_i^{(1)}, K_i^{(2)}$, as defined in Section 2.2), where the algorithm's choice depends on the observed configuration at that time. Note that, using Yao's Principle (as discussed in Section 2.3), these choices are deterministic for a given configuration – in particular, we can think of all $\{k_i^A\}$ and $K_i^{(1)}, K_i^{(2)}$ in the decision tree as being fixed. In this section, we show that all $k_i^A$ values satisfy with high probability some particular conditions with respect to the choice of distribution, where the randomness is over the choice of the support size. Due to space constraints, all proofs from this section are deferred to the full version.

First, we recall an observation from [11], though we modify it slightly to apply to configurations on pairs of distributions and we apply a slightly tighter analysis. This

essentially limits the number of states an algorithm could be in by a function of how many queries it makes.

▶ **Proposition 3.2.** *The number of nodes in a decision tree corresponding to a $q$-sample algorithm is at most $2^{6q^2+1}$.*

For the sake of the argument, we will introduce a few notions applying to the *sizes* of query sets: namely, the notions of a number being *small*, *large*, or *stable*, and of a vector being *incomparable*. Roughly speaking, a number is small if a uniformly random set of this size does not, in expectation, hit the largest bucket $B_{2r}$. On the other hand, it is large if we expect such a set to intersect many bucket-pairs (i.e., a significant fraction of the support). The definition of stable numbers is slightly more quantitative: a number $\beta$ is stable if a random set of size $\beta$, for each bucket $B_i$, either completely misses $B_i$ or intersects it in a number of points very close to the expected number (in this case, we say the set *concentrates* over $B_i$). Finally, a vector of values $(\beta_j)$ is incomparable if the union of random sets $S_1, \dots, S_m$ of sizes $\beta_1, \dots, \beta_m$ contains (with high probability) an amount of mass $D\left(\bigcup_j S_j\right)$ which is either much smaller or much larger than the probability $D(s)$ of any single element $s$.
We formalize these concepts in the definitions below. To motivate them, it will be useful to bear in mind that, from the construction described in Section 3.1, the expected intersection of a uniform random set of size $\beta$ with a bucket $B_i$ is of size $\beta b\rho^i/n$; while the expected probability mass from $B_i$ it contains (under either $D_1$ or $D_2$) is $\beta/(2rn)$.

▶ **Definition 3.3.** Let $q$ be an integer, and let $\varphi = \Theta(q^{5/2})$. A number $\beta$ is said to be *small* if $\beta < \frac{n}{b\rho^{2r}}$; it is *large* (with relation to some integer $q$) if $\beta \geq \frac{n}{b\rho^{2r-2\varphi}}$.

Note that the latter condition equivalently means that, in expectation, a set of large size will intersect at least $\varphi + 1$ bucket-pairs (as it hits an expected $2\varphi + 1$ buckets, since $\beta|B_{2r-2\varphi}|/n \geq 1$). From the above definitions we get that, with high probability, a random set of any fixed size will in expectation either hit many or no buckets:

▶ **Proposition 3.4.** *A number is either small or large with probability $1 - O\left(\frac{\varphi \log \rho}{\log n}\right)$.*

The next definition characterizes the sizes of query sets for which the expected intersection with any bucket is either close to 0 (less than $1/\alpha$, for some threshold $\alpha$), or very big (more than $\alpha$). (It will be helpful to keep in mind that we will eventually use this definition with $\alpha = \mathrm{poly}(q)$.)

▶ **Definition 3.5.** A number $\beta$ is said to be *$\alpha$-stable* (for $\alpha \geq 1$) if, for each $j \in [2r]$, $\beta \notin \left[\frac{n}{\alpha b\rho^j}, \frac{\alpha n}{b\rho^j}\right]$. A vector of numbers is said to be $\alpha$-stable if all numbers it contains are $\alpha$-stable.

▶ **Proposition 3.6.** *A number is $\alpha$-stable with probability $1 - O\left(\frac{r \log \alpha}{\log n}\right)$.*

The following definition characterizes the sizes of query sets which have a probability mass far from the probability mass of any individual element. (For the sake of building intuition, the reader may replace $\nu$ in the following by the parameter $b$ of the distribution.)

▶ **Definition 3.7.** A vector of numbers $(\beta_1, \dots, \beta_\ell)$ is said to be *$(\alpha, \tau)$-incomparable with respect to $\nu$* (for $\tau \geq 1$) if the two following conditions hold.

- $(\beta_1, \dots, \beta_\ell)$ is $\alpha$-stable.
- Let $\Delta_j$ be the minimum $\Delta \in \{0, \dots, 2r\}$ such that $\frac{\beta_j \nu \rho^{2r-\Delta}}{n} \leq \frac{1}{\alpha}$, or $2r$ if no such $\Delta$ exists. For all $i \in [2r]$, $\frac{1}{2rn}\sum_{j=1}^{\ell} \beta_j \Delta_j \notin \left[\frac{1}{\tau 2r\nu\rho^i}, \frac{\tau}{2r\nu\rho^i}\right]$.

Recall from the definition of $\alpha$-stability of a number that a random set of this size either has essentially no intersection with a bucket or "concentrates over it" (i.e., with high probability, the probability mass contained in the intersection with this bucket is very close to the expected value). The above definition roughly captures the following. For any $j$, $\Delta_j$ is the number of buckets that will concentrate over a random set of size $\beta_j$. The last condition asks that the total probability mass from $D_1$ (or $D_2$) enclosed in the union of $m$ random sets of size $\beta_1, \ldots, \beta_\ell$ be a multiplicative factor of $\tau$ from the individual probability weight $\frac{1}{2rb\rho^i}$ of a single element from any of the $2r$ buckets.

▶ **Proposition 3.8.** *Given that a vector of numbers of length $\ell$ is $\alpha$-stable, it is $(\alpha, q^2)$-incomparable with respect to $b$ with probability at least $1 - O\left(\frac{r \log q}{\log n}\right)$.*

We put these together to obtain the following lemma:

▶ **Lemma 3.9.** *With probability at least $1 - O\left(\frac{2^{6q^2+q}(r \log \alpha + \varphi \log \rho) + 2^{6q^2}(r \log q)}{\log n}\right)$, the following holds for the decision tree corresponding to a $q$-query algorithm:*

- *the size of each atom is $\alpha$-stable and either large or small;*
- *the size of each atom, after excluding elements we have previously observed,[4] is $\alpha$-stable and either large or small;*
- *for each $i$, the vector $(k_i^A)_{A \in \mathrm{At}(A_1,\ldots,A_i)}$ is $(\alpha, q^2)$-incomparable (with respect to $b$).*

**Proof.** From Proposition 3.2, there are at most $2^{6q^2+1}$ tree nodes, each of which contains one vector $(k_i^A)_A$, and at most $2^q$ atom sizes. The first point follows from Propositions 3.4 and 3.6 and applying the union bound over all $2^{6q^2+1} \cdot 2 \cdot 2^q$ sizes, where we note the additional factor of 2 comes from either including or excluding the old elements. The latter point follows from Proposition 3.8 and applying the union bound over all $2^{6q^2+1}$ $(k_i^A)$ vectors. ◀

### 3.2.2 Key lemma: bounding the variation distance between decision trees

In this section, we prove a key lemma on the variation distance between the distribution on leaves of any decision tree, when given access to either an instance from $\mathcal{Y}$ or $\mathcal{N}$. This lemma will in turn directly yield Theorem 1.1. Hereafter, we set the parameters $\alpha$ (the threshold for stability), $\varphi$ (the parameter for smallness and largeness) and $\gamma$ (an accuracy parameter for how well things concentrate over their expected value) as follows:[5] $\alpha \stackrel{\text{def}}{=} q^7$, $\varphi \stackrel{\text{def}}{=} q^{5/2}$ and $\gamma \stackrel{\text{def}}{=} 1/\varphi = q^{-5/2}$. (Recall further that $q = \frac{1}{10}\sqrt{\log \log n}$.)

▶ **Lemma 3.10.** *Conditioned on the events of Lemma 3.9, consider the distribution over leaves of any decision tree corresponding to a $q$-query adaptive algorithm when the algorithm is given a yes-instance, and when it is given a no-instance. These two distributions have total variation distance $o(1)$.*

[4] More precisely, we mean to say that for each $i \leq q$, for every atom $A$ defined by the partition of $(A_1, \ldots, A_i)$, the values $k_i^A$ and $|A \setminus \{s_1^{(1)}, s_1^{(2)}, \ldots, s_{i-1}^{(1)}, s_{i-1}^{(2)}\}| - k_i^A$ are $\alpha$-stable and either large or small;

[5] This choice of parameters is not completely arbitrary: combined with the setting of $q$, $r$ and $\rho$, they ensure a total bound $o(1)$ on variation distance and probability of "bad events" as well as a (relative) simplicity and symmetry in the relevant quantities.

**Proof.** This proof is by induction. We will have three inductive hypotheses, $\boldsymbol{E_1}(t), \boldsymbol{E_2}(t)$, and $\boldsymbol{E_3}(t)$. Assuming all three hold for all $t < i$, we prove $\boldsymbol{E_1}(i)$. Additionally assuming $\boldsymbol{E_1}(i)$, we prove $\boldsymbol{E_2}(i)$ and $\boldsymbol{E_3}(i)$.

Roughly, the first inductive hypothesis states that the query sets behave similarly to as if we had picked a random set of that size. It also implies that whether or not we get an element we have seen before is "obvious" based on past observances and the size of the query we perform. The second states that we never observe two distinct elements from the same bucket-pair. The third states that the next sample is distributed similarly in either a yes-instance or a no-instance. Note that this distribution includes both features which our algorithm can observe (i.e., the atom which the sample belongs to and if it collides with a previously seen sample), as well as those which it can not (i.e., which bucket-pair the observed sample belongs to). It is necessary to show the latter, since the bucket-pair a sample belongs to may determine the outcome of future queries.

More precisely, the three inductive hypotheses are as follows:

- $\boldsymbol{E_1}(i)$: In either a yes-instance or a no-instance, the following occurs: For an atom $S$ in the partition generated by $A_1, \ldots, A_i$, let $S' = S \setminus \{s_1^{(1)}, s_1^{(2)}, \ldots, s_{i-1}^{(1)}, s_{i-1}^{(2)}\}$. For every such $S'$, let $\ell^{S'}$ be the largest index $\ell \in \{0, \ldots, 2r\}$ such that $\frac{|S'|b\rho^\ell}{n} \leq \frac{1}{\alpha}$, or 0 if no such $\ell$ exists. We claim that $\ell^{S'} \in \{0, \ldots, 2r - \varphi - 2\} \cup \{2r\}$, and say $S'$ is small if $\ell^{S'} = 2r$ and large otherwise. Additionally:
  - for $j \leq \ell^{S'}, |S' \cap B_j| = 0$;
  - for $j > \ell^{S'}, |S' \cap B_j|$ lies in $[1 - i\gamma, 1 + i\gamma] \frac{|S'|b\rho^j}{n}$.

  Furthermore, let $p_1$ and $p_2$ be the probability mass contained in $\Lambda_i$ and $\Gamma_i$, respectively. Then $\frac{p_1}{p_1+p_2} \leq O\left(\frac{1}{q^2}\right)$ or $\frac{p_2}{p_1+p_2} \leq O\left(\frac{1}{q^2}\right)$ (that is, either almost all the probability mass comes from elements which we have not yet observed, or almost all of it comes from previously seen ones).
- $\boldsymbol{E_2}(i)$: No two elements from the set $\{s_1^{(1)}, s_1^{(2)}, \ldots, s_i^{(1)}, s_i^{(2)}\}$ belong to the same bucket-pair.
- $\boldsymbol{E_3}(i)$: Let $T_i^{\mathsf{yes}}$ be the random variable representing the atoms and bucket-pairs[6] containing $(s_i^{(1)}, s_i^{(2)})$, as well as which of the previous samples they intersect with, when the $i$-th query is performed on a yes-instance, and define $T_i^{\mathsf{no}}$ similarly for no-instances. Then $\mathrm{d_{TV}}(T_i^{\mathsf{yes}}, T_i^{\mathsf{no}}) \leq O\left(\frac{1}{q^2} + \frac{1}{\rho} + \gamma + \frac{1}{\varphi}\right) = o(1)$.

We state the lemmata, whose proofs are deferred to the full version of this paper:

▶ **Lemma 3.11.** *Assuming that $\boldsymbol{E_1}(t), \boldsymbol{E_2}(t), \boldsymbol{E_3}(t)$ hold for all $1 \leq t \leq i-1$, then $\boldsymbol{E_1}(i)$ holds with probability at least $1 - O\left(2^i \exp\left(-\frac{2\gamma^2\alpha}{3}\right)\right) = 1 - 2^{i-\Omega(q^2)}$.*

▶ **Lemma 3.12.** *Assuming that $\boldsymbol{E_1}(t), \boldsymbol{E_2}(t), \boldsymbol{E_3}(t)$ hold for all $1 \leq t \leq i-1$ and additionally $\boldsymbol{E_1}(i)$, then $\boldsymbol{E_2}(i)$ holds with probability at least $1 - O\left(\frac{i}{\varphi}\right)$.*

▶ **Lemma 3.13.** *Assuming that $\boldsymbol{E_1}(t), \boldsymbol{E_2}(t), \boldsymbol{E_3}(t)$ hold for all $1 \leq t \leq i-1$ and additionally $\boldsymbol{E_1}(i)$, then $\boldsymbol{E_3}(i)$ holds.*

Let $T^{\mathsf{yes}}$ be the random variable representing the $q$-configuration and the bucket-pairs containing each of the observed samples in a yes-instance, and define $T^{\mathsf{no}}$ similarly for a no-instance. We note that this random variable determines which leaf of the decision tree

[6] If a sample $s_i^{(k)}$ does not belong to any bucket (if the corresponding $i$-th query did not intersect the support), it is marked in $T_i^{\mathsf{yes}}$ with a "dummy label" to indicate so.

we reach. By a union bound over all $q$ queries of the algorithm, a coupling argument, and the triangle inequality, the above lemmata imply that the total variation distance between $T^{\mathsf{yes}}$ and $T^{\mathsf{no}}$ will be $O\left(2^q \exp\left(-\frac{2\gamma^2\alpha}{3}\right) + \frac{q^2}{\varphi} + \frac{1}{q} + \frac{q}{\rho} + q\gamma + \frac{q}{\varphi}\right) = o(1)$ (from our choice of $\alpha, \gamma, \varphi$), concluding the proof of Lemma 3.10. ◀

With this lemma in hand, the proof of the main theorem is straightforward:

**Proof of Theorem 1.1.** Conditioned on Lemma 3.9, Lemma 3.10 implies that the distribution over the leaves in a yes-instance vs. a no-instance is $o(1)$. Since an algorithm's choice to accept or reject depends deterministically on which leaf is reached, this bounds the difference between the conditional probability of reaching a leaf which accepts. Since Lemma 3.9 occurs with probability $1 - o(1)$, the difference between the unconditional probabilities is also $o(1)$. ◀

**Acknowledgments.** Clément Canonne would like to thank Dana Ron and Rocco Servedio for the many helpful discussions and remarks that influenced the lower bound construction of Section 3.

## References


1. Jayadev Acharya, Clément L. Canonne, and Gautam Kamath. A chasm between identity and equivalence testing with conditional queries. *ArXiV*, abs/1411.7346, April 2015.
2. Jayadev Acharya, Hirakendu Das, Ashkan Jafarpour, Alon Orlitsky, and Shengjun Pan. Competitive closeness testing. In *Proceedings of 24th COLT*, pages 47–68, 2011.
3. Jayadev Acharya, Hirakendu Das, Ashkan Jafarpour, Alon Orlitsky, Shengjun Pan, and Ananda Theertha Suresh. Competitive classification and closeness testing. In *Proceedings of 25th COLT*, pages 1–18, 2012.
4. Tuğkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *Proceedings of FOCS*, pages 442–451, 2001.
5. Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing closeness of discrete distributions. *Journal of the ACM*, 60(1):1–25, 2013.
6. Tuğkan Batu, Ravi Kumar, and Ronitt Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of STOC*, pages 381–390, New York, NY, USA, 2004. ACM.
7. Arnab Bhattacharyya, Eldar Fischer, Ronitt Rubinfeld, and Paul Valiant. Testing monotonicity of distributions over general partial orders. In *Proceedings of ITCS*, pages 239–252, 2011.
8. Clément L. Canonne. A Survey on Distribution Testing: your data is Big, but is it Blue? *Electronic Colloquium on Computational Complexity (ECCC)*, TR15-063, April 2015.
9. Clément L. Canonne, Dana Ron, and Rocco A. Servedio. Testing probability distributions using conditional samples. *SIAM Journal on Computing*, 44(3), 2015.
10. Clément L. Canonne and Ronitt Rubinfeld. Testing probability distributions underlying aggregated data. In *Proceedings of ICALP*, pages 283–295, 2014.
11. Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. In *Proceedings of ITCS*, pages 561–580, New York, NY, USA, 2013. ACM.
12. Siu-On Chan, Ilias Diakonikolas, Gregory Valiant, and Paul Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of SODA*, pages 1193–1203. Society for Industrial and Applied Mathematics (SIAM), 2014.

**13** Moein Falahatgar, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapathi, and Ananda Theertha Suresh. Faster algorithms for testing under conditional sampling. In *Proceedings of 28th COLT*, 2015.

**14** Eldar Fischer. List of Open Problems in Sublinear Algorithms: Problem 66. `http://sublinear.info/66`. Suggested by Fischer at Bertinoro Workshop on Sublinear Algorithms 2014.

**15** Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, TR00-020, March 2000.

**16** Sudipto Guha, Andrew McGregor, and Suresh Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In *Proceedings of SODA*, pages 733–742. Society for Industrial and Applied Mathematics (SIAM), 2006.

**17** Piotr Indyk, Reut Levi, and Ronitt Rubinfeld. Approximating and Testing $k$-Histogram Distributions in Sub-linear Time. In *Proceedings of PODS*, pages 15–22, 2012.

**18** Reut Levi, Dana Ron, and Ronitt Rubinfeld. Testing properties of collections of distributions. *Theory of Computing*, 9(8):295–347, 2013.

**19** Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1995.

**20** Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.

**21** Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam Smith. Strong lower bounds for approximating distributions support size and the distinct elements problem. *SIAM Journal on Computing*, 39(3):813–842, 2009.

**22** Dana Ron and Gilad Tsur. The power of an example: Hidden set size approximation using group queries and conditional sampling. *ArXiV*, abs/1404.5568, 2014.

**23** Ronitt Rubinfeld. Taming Big Probability Distributions. *XRDS*, 19(1):24–28, September 2012.

**24** Ronitt Rubinfeld and Rocco A. Servedio. Testing monotone high-dimensional distributions. *Random Structures and Algorithms*, 34(1):24–44, January 2009.

**25** L. Stockmeyer. On approximation algorithms for #P. *SIAM Journal on Computing*, 14(4):849–861, 1985.

**26** Gregory Valiant and Paul Valiant. A CLT and tight lower bounds for estimating entropy. *Electronic Colloquium on Computational Complexity (ECCC)*, TR10-179, 2010.

**27** Gregory Valiant and Paul Valiant. Estimating the unseen: A sublinear-sample canonical estimator of distributions. *Electronic Colloquium on Computational Complexity (ECCC)*, TR10-180, 2010.

**28** Gregory Valiant and Paul Valiant. The power of linear estimators. In *Proceedings of FOCS*, pages 403–412, October 2011. See also [26] and [27].

**29** Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. In *Proceedings of FOCS*, 2014.

**30** Paul Valiant. Testing symmetric properties of distributions. *SIAM Journal on Computing*, 40(6):1927–1968, 2011.