Electronic Colloquium on Computational Complexity, Report No. 198 (2025)

Interactive Proofs For Distribution Testing With Conditional Oracles

Ari Biswas — *University of Warwick*
Mark Bun — *Boston University*
Clément Canonne — *University of Sydney*
Satchit Sivakumar — *Boston University*

Abstract

We revisit the framework of interactive proofs for distribution testing, first introduced by Chiesa and Gur (ITCS 2018), which has recently experienced a surge in interest, accompanied by notable progress (e.g., Herman and Rothblum, STOC 2022, FOCS 2023; Herman, RANDOM 2024). In this model, a data-poor verifier determines whether a probability distribution has a property of interest by interacting with an all-powerful, data-rich but untrusted prover bent on convincing them that it has the property. While prior work gave sample-, time-, and communication-efficient protocols for testing and estimating a range of distribution properties, they all suffer from an inherent issue: for most interesting properties of distributions over a domain of size N, the verifier must draw at least $\Omega(\sqrt{N})$ samples of its own. While sublinear in N, this is still prohibitive for large domains encountered in practice.

In this work, we circumvent this limitation by augmenting the verifier with the ability to perform an exponentially smaller number of more powerful (but reasonable) *pairwise conditional* queries, effectively enabling them to perform "local comparison checks" of the prover's claims. We systematically investigate the landscape of interactive proofs in this new setting, giving polylogarithmic query and sample protocols for (tolerantly) testing all *label-invariant* properties, thus demonstrating exponential savings without compromising on communication, for this large and fundamental class of testing tasks.

Contents

ISSN 1433-8092

1 Introduction

Distribution testing, as introduced by Batu et al. [2000], is a mature subfield of property testing [Goldreich et al., 1998, Rubinfeld and Sudan, 1996] aimed at investigating statistical properties of an unknown distribution given sample access to it. Given a property (a set of distributions) and a proximity parameter $\tau \in (0, 0.1]$, distribution testing algorithms output Accept if the distribution is in the property (or close to it), or Reject if the distribution is τ-far from the property, both with high probability. Closeness and farness are quantified with respect to a prespecified notion of distance, typically total variation distance. The primary motivation behind distribution testing is to design testing algorithms for deciding properties with sample complexity sub-linear in the domain size N (which is demonstrably more efficient than learning the distribution, which requires drawing $\Theta(N)$ samples). Accordingly, over the last two decades, researchers have extensively studied the sample complexity of numerous distribution properties, such as simple uniformity testing [Goldreich and Ron, 2011] (testing whether a distribution is uniform over its entire domain), support size decision problem [Raskhodnikova et al., 2009, Valiant and Valiant, 2011, Wu and Yang, 2019, Ferreira Pinto Jr. and Harms, 2025] (testing whether a distribution's support is within some pre-specified range), and many more: see, e.g., [Goldreich, 2017, Chapter 11] and Rubinfeld [2012], Canonne [2020, 2022] for a more thorough introduction to distribution testing. Unfortunately, although distribution testing is often more efficient than learning the distribution, it is still prohibitively expensive for practical use. For example, it is known that generalized uniformity testing (testing whether a distribution is uniform over its support) over a domain of size N requires $\Omega\left(N^{2/3}\right)$ samples [Batu and Canonne, 2017, Diakonikolas et al., 2018], which can be impractical for large domain sizes. Even simple uniformity testing requires $\Omega(\sqrt{N})$ samples [Paninski, 2008], and its *tolerant* testing version (which asks to distinguish distributions *close* to uniform from those which are far) needs $\Omega\left(N/\log N\right)$ samples [Valiant and Valiant, 2017].

In the face of these limitations, a nascent line of work [Chiesa and Gur, 2018, Herman and Rothblum, 2022, 2023, Herman, 2024] has asked a related question: *with testing being hard by itself, what is the complexity of* verifying *the properties of a distribution given sample access to it?* Here, in addition to drawing samples from the distribution, the tester is allowed to interactively communicate with an omniscient but *untrusted* prover that knows the distribution in its entirety. The idea here is to leverage the provers extra knowledge about the distribution, with the hope that checking the provers' claims is easier than naively testing the property. While this model of verifiable computation has only recently been explored in the context of distribution testing, it has been an active area of research in other areas of theoretical computer science for over 40 years (see for e.g. [Goldwasser et al., 1985, Micali, 2000, Rothblum et al., 2013, Goldwasser et al., 2015, Berman et al., 2018, Arun et al., 2024]). It models settings where a centralized organization (for example, a company turning billions of dollars of profit) has the ability to collect large amounts of data and learn distributions to high precision, while end-users may not have the same ability. At the same time, the company might have incentives to lie, and so verifying whether the company is being truthful is important in this setting. The work of Chiesa and Gur [2018] shows that the verification of *any* distribution property over domain $[N]$ can be reduced to identity testing[1], with communication *superlinear* in the domain

[1] Identity testing refers to the task of testing if a distribution is exactly equal to a pre-specified reference distribution

size. Follow up work [Herman and Rothblum, 2022, 2023] recovers this result for the broad class of *label-invariant properties*, while only requiring communication *sub-linear* in the domain size. More specifically, the work of Herman and Rothblum [2022, 2023] show that for label-invariant properties, verification requires only $O\left(\sqrt{N}\right)$ samples and $\widetilde{O}\left(\sqrt{N}\right)$ communication, even though, as mentioned earlier, testing some properties in this class could require $\Theta(N/\log N)$ samples. Here, a property is *label-invariant* (also known as *symmetric*) if the names of the elements themselves are not significant to the decision outcome (see Definition 2.1). Testing if a distribution is uniform over its support (also known as generalized uniformity testing) is an example of a label-invariant property.

Unfortunately, while a significant improvement over unaided testing, requiring $O\left(\sqrt{N}\right)$ samples from the verifier can still be prohibitive when considering massive domains. Further, there is a matching sample complexity lower bound – verification of even basic label-invariant properties such as checking if a distribution is uniform over its entire domain requires $\Omega(\sqrt{N})$ samples. To summarise: For most properties, with access to *only* samples from a distribution, it is impossible for any tester to do better than drawing $\Omega\left(\sqrt{N}\right)$ samples, with or without the help of a prover. To bypass these limitations and develop more practical algorithms, in this work we study verifiers that can make a very small number of calls to a more powerful *conditional sampling* oracle. These oracles were introduced in the context of distribution testing [Chakraborty et al., 2013, Canonne et al., 2014]; allowing the tester to condition that samples from the oracle come from a subset S of the domain, of their choosing. The oracle responds with a sample with probability re-normalised over S. If no element in S is supported, the oracle responds with FAIL. Since specifying an arbitrary set may considered be unrealistic for practical purposes, a commonly studied restriction is the pairwise conditional sampling model (PCond), where the specified sets are restricted to be of size exactly 2 or the entire domain (thus, just a regular sample from the distribution). These oracles can be thought of as allowing for local comparisons between the probabilities of two points. While access to a PCond oracle can be significantly helpful for problems like simple uniformity testing, it is unclear from prior work whether it results in more efficient testing for the general class of label-invariant properties. Verification with access to a PCond oracle (or any type of conditional sampling oracle) has also, to the best of our knowledge, not been explored. In our quest to find practical algorithms that work for large domains, we thus ask the following question.

> *Can label-invariant properties be verified in a (query, sample and communication)-efficient way when the tester has access to a* PCond *oracle?*

1.1 Our Results

Our main result is an *exponential* query complexity separation between testing and verification for testing label-invariant properties with access to a PCond oracle. A detailed accounting of our results and comparison to existing work can be found in Table 1. A description follows.

One might have initially hoped that the power of a PCond oracle allows us to test label-invariant properties efficiently, even without the help of a prover. Indeed, with access to the full power of the Cond model (where arbitrary subsets S can be queried and a sample conditional on S is obtained), Chakraborty et al. [2013] show that this class of properties over a domain of size N can be tested with $O(\text{poly}\log N)$ queries to the Cond oracle. Our first result dashes this hope – we show a lower bound on the number of PCond queries required to test label-invariant properties with constant proximity

or is τ-far from it.

	Query Complexity Without Prover	Query Complexity With Prover	Communication	Rounds
Samp	$\widetilde{\Omega}\left(\frac{N}{\log N}\right)$ Valiant and Valiant [2017]	$\widetilde{O}\left(\sqrt{N}\right)$ [Herman and Rothblum, 2022, 2023]	$\widetilde{O}\left(\sqrt{N}\right)$ [Herman and Rothblum, 2022, 2023]	2 [Herman and Rothblum, 2022, 2023]
PCond	$\boldsymbol{\Omega\left(N^{1/3}\right)}$ (Theorem 3.7)	$\mathrm{poly}\boldsymbol{(\log N, \frac{1}{\tau})}$	$\boldsymbol{\widetilde{O}\left(\sqrt{N}\right)}$ (Theorem 5.3)	$\mathrm{poly}\boldsymbol{(\log N, \frac{1}{\tau})}$

Table 1: Results on testing and verifying label-invariant properties under different types of access to the distribution. We state the best known lower bounds for label invariant properties. For Samp the lower bound is for entropy estimation, whereas for PCond it is the support size decision problem described in Section 4. The upper bounds apply for all label-invariant properties. Our results are highlighted in bold.

parameter τ, demonstrating that the PCond oracle is not much better in the worst case than the sampling oracle for this class of properties. Specifically, we show that a simple variant of the support size distinguishing problem for distributions over a domain of size N requires $\Omega(N^{1/3})$ queries to a PCond oracle (the exact same as with access to only a sampling oracle). Thus, unaided, there exist (label-invariant) properties for which the PCond oracle is not much better than just sample access.

Theorem 1.1: [Informal Version of Corollary 3.7] There exists a label-invariant property Π such that every tester with access to a PCond oracle for Π with proximity parameter $\tau \leq 1/2$ and failure probability 0.01 must make $\Omega\left(N^{1/3}\right)$ queries.

The above lower bound motivates the investigation of verification with access to a PCond oracle. As mentioned earlier, [Chiesa and Gur, 2018, Proposition 3.4] showed that with super-linear communication complexity, there exists a reduction from verification to identity testing. Instantiating this reduction with an identity tester using PCond oracles [Narayanan, 2021, Theorem 1.5], we get that there exists an interactive proof system for every property with super-linear communication complexity that makes only $O(\sqrt{\log N}/\tau^2)$ queries to the PCond oracle. However, super-linear communication is also prohibitive for practical algorithms; the proof systems by Herman and Rothblum [2022, 2023] require the prover to only communicate $\widetilde{O}(\sqrt{N})$ domain elements, but still achieve the sample complexity of identity testing (for the class of label-invariant properties). Could we also hope to achieve such communication while maintaining similar query complexity as that of identity testing? The main result of this paper is an affirmative answer to this question. Specifically, we give an interactive proof system for tolerantly verifying *any* label-invariant property that has communication complexity $\widetilde{O}(\sqrt{N})$ and query complexity $\mathrm{poly}(\log N)$ (suppressing the dependence on the proximity parameter).

Theorem 1.2: [Informal Label-Invariant Tolerant Verification Theorem (Theorem 5.3)] Fix a label-invariant property Π over a domain $[N]$ and proximity parameters $\tau_c, \tau_f \in (0, 1/2]$. There exists a polylogarithmic (in N) round interactive protocol Π between an honest verifier T, and an omniscient untrusted prover $\mathsf{P}^{\mathcal{D}}$, where the verifier has `PCond` access to $\mathcal{D}$, such that at the end of the interaction the verifier satisfies the following conditions:

1. **Completeness:** If the prover follows the protocol as prescribed, and $d_{\mathrm{TV}}(D, \Pi) \leq \tau_c$, then
$$\Pr\left[\mathrm{out}\left[\Pi\left(\mathsf{P}^{\mathcal{D}}, \mathsf{T}^{(\mathcal{D})}; \tau_c, N\right)\right] = \mathsf{Accept}\right] \geq 2/3$$
2. **Soundness:** If $d_{\mathrm{TV}}(D, \Pi) \geq \tau_f$, then for any prover $\widetilde{\mathsf{P}^{\mathcal{D}}}$
$$\Pr\left[\mathrm{out}\left[\Pi\left(\widetilde{\mathsf{P}}^{\mathcal{D}}, \mathsf{T}^{(\mathcal{D})}; \tau_c, N\right)\right] = \mathsf{Reject}\right] \geq 2/3$$

The complexity of the verifier is as follows:

1. **Query Complexity + Sample Complexity**: $O\left(\mathrm{poly}(\log N, 1/(\tau_f - \tau_c))\right)$
2. **Communication Complexity:** $\widetilde{O}\left(\sqrt{N}\,\mathrm{poly}(1/(\tau_f - \tau_c))\right)$

In the process of proving this result, we give protocols for more basic primitives that may be of independent interest. Most significantly, we give an interactive proof system that is able to calculate the approximate probability mass of any point[2] in the domain using communication complexity $\widetilde{O}(\sqrt{N})$ and query complexity $O(\mathrm{poly}(\log N, 1/\tau))$. As we will explain in the techniques section to follow, this is a key technical workhorse in our protocol for verifying label-invariant properties.

[2]Provided it does not have prohibitively small probability mass in its neighborhood.

Theorem 1.3: [Informal Version of Theorem 5.1] Fix a domain $[N]$. For every $\tau > 0$ and $\delta \in (0, 1/2)$, there exists a $O(\text{ poly}(\log N, \log 1/\delta, \frac{1}{\tau}))$-round interactive proof system such that the verifier T with access to a $\mathtt{PCond}$ oracle satisfies the following.

1. **Completeness:** For every distribution $\mathcal{D}$, if the prover $\mathsf{P}^{\mathcal{D}}$ is honest, then

$$\Pr\left[\mathsf{T} \text{ outputs } (y^\star, \widetilde{\mathcal{D}}\left[y^\star\right]) \text{ s.t } \frac{\widetilde{\mathcal{D}}\left[y^\star\right]}{\mathcal{D}\left[y^\star\right]} \in \left[\frac{1}{(1+\tau)}, 1+\tau\right]\right] \geq 1-\delta$$

2. **Soundness:** For any cheating prover $\widetilde{\mathsf{P}}^{\mathcal{D}}$, then

$$\Pr\left[\mathsf{T} \text{ outputs } \mathsf{Reject} \vee \mathsf{T} \text{ outputs } (y^\star, \widetilde{\mathcal{D}}\left[y^\star\right]) \text{ s.t } \frac{\widetilde{\mathcal{D}}\left[y^\star\right]}{\mathcal{D}\left[y^\star\right]} \in [1/(1+\tau)^2, (1+\tau)^2]\right] \geq 1-\delta$$

The complexity of the verifier is as follows:

1. **Query Complexity + Sample Complexity**: $O\left(\text{poly}(\log N, 1/(\tau))\right)$
2. **Communication Complexity:** $\widetilde{O}\left(\sqrt{N}\,\text{poly}(1/(\tau))\right)$

1.2 Technical Overview

In this section, we give an overview of our protocol for verifying label-invariant properties.

Unlabelled Bucket Histogram: There is now a long line of work on the testing and verification of label-invariant properties [Batu et al., 2000, Valiant, 2008, Chakraborty et al., 2013, Herman and Rothblum, 2022, 2023], and a key object used in this work is the unlabelled *approximate* τ-bucket histogram of a distribution. Bucketing corresponds to partitioning the interval $[0, 1]$ into smaller multiplicative probability intervals (see Definition 2.4). The τ-bucket histogram divides the interval $[0, 1]$ into $\widetilde{O}\left(\log N/\tau\right)$ buckets where the ℓ^{th} bucket is a set of domain elements with individual probability mass in the range $(\tau(1+\tau)^{\ell}/N, \tau(1+\tau)^{\ell+1}/N]$. The *approximate* unlabelled bucket histogram of a distribution then corresponds to a list of $\widetilde{O}(\log N/\tau)$ fractions, where the ℓ^{th} element of the list is the fraction of domain points whose probability lies in the range specified by the ℓ^{th} bucket (see Definition 2.5). It is well known (see [Valiant, 2008]) that the *approximate* unlabelled τ-bucket histogram of a distribution is a sufficient statistic to (tolerantly) test any label-invariant property with proximity parameter(s) $O\left(\tau\right)$. Thus, similar to prior work [Herman and Rothblum, 2022, 2023, Herman, 2024], our protocol also focuses on efficiently verifying an unlabelled τ-bucket histogram given to us by the prover.

Using Pairwise Comparisons to Learn Bucket Histogram: Note that the unlabelled bucket histogram is a distribution over the buckets of the τ-bucket histogram and is hence a distribution over a domain of size $\widetilde{O}(\log \frac{N}{\tau})$. Hence, by standard results in distribution learning, $\widetilde{O}(\log N/\tau^2)$ samples from this bucket distribution would be sufficient to learn it. However, sampling from this bucket distribution is non-trivial since a sample from the original distribution $\mathcal{D}$ does not come with information about histogram bucket index.

While sampling from the bucket distribution might be hard with `Samp` access to the distribution, one might be more optimistic about the possibility of sampling from the τ-approximate histogram with `PCond` queries. In particular, one approach that we might take is the following: the verifier draws a dataset of size $\widetilde{O}(\log N/\tau^2)$ (enough to learn the histogram), and sends these samples to the prover, who responds with the bucket index of each sample. From how a τ-histogram is defined, if x and y belong to buckets i and j respectively, then this implies that the ratio of the probability masses of x and y under $\mathcal{D}$ is guaranteed to be in the interval $\left[\frac{1}{(1+\tau)^{|j-i|}}, (1+\tau)^{|j-i|}\right]$. As `PCond` access allows us to conditionally sample from a set restricted to two domain elements, it allows us to approximately learn the ratio of their probability masses up to a multiplicative constant (see Lemma 2.11). Equipped with this power, for each pair $x \neq y$ from the set of drawn samples, the verifier uses the `PCond` oracle to check if the learned ratios align with the provers claims. If the prover were to lie significantly, then for at least one pair of samples, the claimed ratio would significantly different from the learned ratio. Unfortunately, this simplistic strategy comes with two pitfalls. Firstly, assuming the above strategy was sound, naively comparing elements with arbitrary bucket indices could require $\Omega(N)$ `PCond` queries if the elements being compared had significantly different probability masses (see the Compare Algorithm, wherein K could be as large as N). This would be as bad as learning the distribution itself. Secondly, and more importantly, the above strategy is *not* sound. It does not catch a prover that "slides" all samples into different buckets *in the same way*, i.e., it lies about *every* bucket index by the same offset. As an example, consider two distributions: $\mathcal{D}_1$ is the uniform distribution over $N^{1/4}$ domain elements and $\mathcal{D}_2$ is the uniform distribution over $\sqrt{N}$ domain elements. The bucket histograms of both involve a unit mass on a single (but different) bucket. However, pairwise comparisons between samples taken from either distribution will always reveal a ratio that is approximately 1, since the probabilities of all elements in the support of both distributions are identical. Hence, given distribution $\mathcal{D}_1$, the prover can output the bucket histogram of distribution $\mathcal{D}_2$, and it is impossible for a verifier to catch it purely by using the test described above.

A remedy to these obstacles lies in the following observation. If we had a good estimate of the probability mass of *a single point* y in the domain, then we could resolve the soundness issue discussed above. We simply use the `PCond` oracle to learn the ratio of probabilities between each of our samples and y. Using this ratio, and knowledge of the approximate mass of y, we can compute estimates of the probabilities of all samples. This would squash the sliding attack described above. To deal with the first issue (that of the probability mass of y being very far from that of a sample), we would need to do more than learn just one value of y. Instead, we could learn the mass of a point y_j, for every bucket j that has large enough mass. This way, for any sample x in the tester's set of samples (which are likely to come from buckets with sufficiently large mass), we can find some y_j that is in a near-by bucket with high probability.

Verifying the probability of points: Given that we simply need to identify the probability of a few points in the domain, one might expect that this could be done even without access to a prover — this would give a query-efficient tester for label-invariant properties. However, this ends up being a surprisingly challenging task. Indeed, our lower bound in Theorem 1.1 shows that this is impossible (if we wanted to bypass the poly(N) lower bound). This indicates a power of an untrusted prover; it is able to certify the probability of a few points in the distribution support. Indeed, the proof system with super-linear communication Chiesa and Gur [2018] achieves something stronger-it certifies the entire distribution. Since we only need to certify the mass of at most $O(\log N)$ points, we ask if this be done in a more communication efficient way?

Support Size Verification: Inspired by the "sliding" cheating prover from earlier, we consider the orthogonal but related problem of verifying the support size of a flat distribution.[3] We will subsequently show that a protocol for this problem can be combined with the ability of a `PCond` oracle to learn neighbourhoods around points, enabling us to solve the probability approximation problem for "relevant" domain elements.

Given a support size claim represented by four numbers A', A, B, B', corresponding to the claim that $A' < A \leq \text{Supp}(\mathcal{D}) \leq B < B'$, our hope is to accept if the claimed support size range is accurate, and reject if the true support is larger than B' or smaller than A'. Given different values of A and B, we develop a number of tests to verify with *sub-linear* communication, the support size assuming the distribution is uniform over its support[4]. We summarize the ideas below.

If the claimed support size upper bound B is small (that is, $O\left(\sqrt{N}\right)$), we could ask the prover to send us the claimed support of the distribution. If the prover lies, and the true support is actually much larger, then taking a few samples from the distribution would give us a domain element outside the claimed support, thus catching the lie of the prover. If the true support is much smaller than A, then taking a number of uniform samples from the claimed support sent by the prover would result in a sample outside the support of the distribution, which could be easily detected with a few `PCond` queries (see Lemma 2.12). This gives us a protocol with a constant number of queries, and communication complexity roughly $O(B)$.

On the other hand, if the claimed support size lower bound A is large (that is, $\omega\left(\sqrt{N}\right)$), then asking the prover to send the support is not communication-efficient. Our approach instead involves using uniform samples from the domain. The first test is to catch provers that lie that the support is much larger than it really is. It involves drawing $O(N/A)$ uniform samples S_1 from the domain, and sending them to the prover. We ask the prover to send back a sample in this subset that is in the support of the distribution. If the true support is much smaller than A, there are (with high probability) no samples in the support of the distribution in S_1, and we can ensure the prover does not cheat by checking whether any element it sends back is in the support using a constant number of `PCond` queries. The second test catches provers that lie that the support is much smaller than it really is. It involves drawing $O(N/B')$ samples $z_1, \ldots, z_m$ uniformly from the domain and permuting them with one sample x taken from the distribution. We ask the prover to identify the index of x in the permuted set. If the true support is smaller than B, then w.h.p, we expect that none of $z_1, \ldots, z_s$ are in the support of the distribution and hence, the honest prover can identify x exactly. On the other hand, if the support is larger than B', we expect that at least one of $z_1, \ldots, z_s$ is in the support of the distribution, and the prover is unable to tell what the sample inserted by the prover was (since it could be x or z_i). This gives us a protocol with a constant number of queries, and communication complexity roughly $O(N/A)$. Balancing parameters in these tests to optimize the communication complexity, we get an overall protocol for support size verification with communication complexity roughly $\sqrt{N}$.

We emphasize that the above outline is a simplification of the truth, and sweeps important details under the rug. Recall that our main goal is to certify the histogram of a distribution. In an attempt to do so, we will use the support size protocol above repeatedly as a sub-routine. This requires bounding the soundness and completeness errors in a meaningful way. As described above, these protocols do not have low enough soundness and completeness error to be used as sub-routines. Additionally, the above description assumes that distributions are exactly flat. In practice this will not be the case, and we need to be able to handle distributions where the probability ratio between any two elements in the support is upper bounded by some constant α (nearly flat). In Section 4, we

[3] A distribution is *flat* if it is uniform over its support.

[4] We relax this condition in the main protocol, but assuming uniformity makes the description more intuitive.

show how to analyse the protocols above to facilitate amplification of soundness and completeness errors, and make the protocol work for the more general class of α-flat distributions.

Estimating Probability of a Point using Support Size Verification: Finally, we explain how we can use the support size protocols to estimate the probability of a point. Prior work [Canonne et al., 2015] using the `PCond` oracle has shown that it can be used to estimate the mass within a multiplicative $(1+\tau)$-neighbourhood of any point[5] (see The `EstimateNeighborhood` procedure of [Canonne et al., 2015, Lemma 3]). We ask the prover to send us a point $y^\star$ [6] with sufficiently large mass in its neighbourhood (by an averaging argument, at least one histogram bucket needs to have $\tau/\log N$ mass, ensuring that such a point exists, see Claim 5.2), and tell us the bucket the $y^\star$ belongs to. We then use the `PCond` oracle to estimate the mass within the multiplicative neighbourhood of $y^\star$. The learned mass of the neighborhood divided by the prover's claimed probability mass[7] of $y^\star$ gives us bounds on the number of elements in $y^\star$'s neighbourhood. Additionally, by definition the neighbourhood of $y^\star$ will be nearly flat. Hence, we have reduced the problem to a claim about the support size of a nearly-uniform distribution over a subset of the domain. Observe that we can sample from the distribution restricted to this bucket using `PCond` queries—since there is sufficient mass in the neighbourhood of the point, $O(\text{poly}\log N)$ samples from the distribution will contain at least one sample from the bucket, and we can use `PCond` queries between the samples and $y^\star$ to find out which sample it is (see Sampling From Neighbourhoods With Moats). If the prover was telling the truth (or rather did not lie egregiously), then the support size claim holds and the verifier will accept, thereby giving us a point and its approximate probability mass[8]. If the prover significantly lied, then the support size claim will be false and the prover will be caught. We note that the final analysis needs to handle some additional subtleties, since the `PCond` oracle is itself only approximate and does not provide perfect comparisons (which can affect the sampling), and additionally the bucket boundaries and the neighborhood of the provided point do not overlap precisely. We refer to Sections 4.3 and 5 for the details. Once we have estimated the probability of important points, we can use the ratio learning techniques discussed earlier to certify the prover's claim about the bucket histogram of the distribution.

1.3 Other Related Work:

Interactive Proofs for Distribution Testing: As mentioned earlier, interactive proofs for verifying distribution properties were first introduced in the work of Chiesa and Gur [2018]. Follow-up work [Herman and Rothblum, 2022, 2023] studied interactive proofs for verifying *label-invariant* properties, focusing on sublinear communication, and doubly efficient protocols (i.e. computationally-efficient and sample-efficient generation of a proof). Herman [2024] studied *public-coin* interactive proofs for testing label-invariant properties, where the verifier has no private randomness. Herman and Rothblum [2024] gives communication-efficient and sample-efficient interactive proofs for more general distribution properties that can be decided by uniform polynomial-size circuits with bounded depth. Herman and Rothblum [2025] introduces *computationally sound* interactive proofs and shows communication-, time-, and sample-efficient protocols for any distribution property that can be decided in polynomial time given explicit access to the full list of distributiom probabilities. All of

[5] As long as the point does not have prohibitively small probability mass under the distribution.

[6] Recall that in the final protocol, we will ask the prover to send us points from every bucket with sufficiently large mass. For simplicity, we consider a single point in this description.

[7] The bucket index gives a lower and upper bound on the true probability mass of $y^\star$. This is sufficient to catch a cheating prover.

[8] The claimed mass of $y^\star$ is derived from the bucket sent by the prover.

the above works assume that the verifier only has sample access to the distribution, and the verifier sample complexity in all of them is $\Omega(\sqrt{N})$ (where N is the size of the domain).

Interactive Proofs for Learning: A related but orthogonal line of work [Goldwasser et al., 2021, Mutreja and Shafer, 2023, Gur et al., 2024, Caro et al., 2024b,a] focuses on interactive proofs for verifying learning problems- for a specific hypothesis class H, given access to an untrusted prover, the goal is for a verifier to output an accurate hypothesis from H for an underlying unknown distribution D if the resource-rich prover is honest, and to reject if the prover lies egregiously. Different types of resource asymmetry between the prover and the verifier are explored in these papers – including differing number of samples, different computational complexities, different types of access to the underlying function (sample vs query), and differing access to computational resources (classical vs quantum computation and communication).

Distribution testing under conditional oracles: Our work focuses on interactive proofs for *verifying* distribution properties under conditional sampling models. There is a long line of work on *testing* with access to conditional samples. Chakraborty et al. [2013], Canonne et al. [2014] introduced the conditional sampling (`Cond`) model and its more restricted variants (`PCond`, `ICond`). They gave algorithms for uniformity testing, tolerant uniformity testing, identity testing etc. in these conditional sampling models with query and sample complexity significantly better than the `Samp` model. Follow-up work shows improved bounds for identity testing (and its tolerant version), tolerant uniformity testing, and new algorithms for other tasks such as equivalence testing and support size problem in the `Cond` model [Falahatgar et al., 2015, Narayanan, 2021, Chakraborty et al., 2023, 2024]. There is also a line of work studying the power of non-adaptive queries in the conditional sampling model [Acharya et al., 2015, Kamath and Tzamos, 2019]. The `PCond` model was studied in detail by Narayanan [2021] who gave optimal bounds for identity testing and tolerant uniformity testing in this model, improving on results from Canonne et al. [2015]. Testing under other types of conditional sampling has also been studied in the literature including subcube conditioning, where the distribution is supported on the hypercube, and the tester is allowed to ask for conditional samples from subcubes [Bhattacharyya and Chakraborty, 2018, Canonne et al., 2021, Chen et al., 2021, Kumar et al., 2023, Chakrabarty et al., 2025], and coordinate conditional sampling [Blanca et al., 2025] (a version of subcube conditioning where all but one coordinate is fixed to a specific configuration and a sample is obtained from the remaining coordinate). Testing under other types of access to the distribution such as Probability Mass Function queries or Cumulative Distribution Function queries has also been studied in the literature [Batu et al., 2002, Rubinfeld and Servedio, 2005, Guha et al., 2009, Canonne and Rubinfeld, 2014, Onak and Sun, 2018].

2 Preliminaries

General Notation We use $\Delta(\mathcal{X})$ to denote the set of all probability distributions over some set $\mathcal{X}$, and $[N]$ as shorthand for the set $\{1, \ldots, N\}$. We use the notation $x \leftarrow\!\$\ \mathcal{D}$ to indicate x was sampled according to $\mathcal{D}$. Given a set $S \subseteq \mathcal{X}$, we use $\mathcal{D}[S] \stackrel{\text{def}}{=} \sum_{y \in S} \mathcal{D}[y] \stackrel{\text{def}}{=} \sum_{y \in S} \Pr_{x \leftarrow\!\$\, \mathcal{D}}[x = y]$ to denote the probability of hitting set S when sampling from $\mathcal{D}$, and $\mathcal{D}[y]$ is the probability of seeing y when sampling according to $\mathcal{D}$. When we say distance of a distribution $\mathcal{D}$ from a property $\Pi \subseteq \Delta(\mathcal{X})$, we mean $d_{\mathrm{TV}}(\mathcal{D}, \Pi) \stackrel{\text{def}}{=} \min_{\mathcal{D}' \in \Pi} d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}')$, where $d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}')$ denotes the total variation distance between two distributions. We use $\mathrm{Supp}(\mathcal{D})$ to denote the support of a distribution $\mathcal{D}$. For any set $\mathcal{X}$, we use $\mathcal{S}_{\mathcal{X}}$ to denote the set of permutations over the set. Throughout, we use tilde notation $\widetilde{\cdot}$ with lighter font to denote an untrusted prover's claims. For example, we use $\widetilde{\mathcal{D}}[x]$ to denote a

prover's claim about the probability mass of $x \in [N]$ under $\mathcal{D}$, and $\mathcal{D}[x]$ as the true mass. Similarly, $\widetilde{\text{Bucket}}_\tau(x)$ will denote the provers claim about the histogram bucket index (see Definition 2.4), versus $\text{Bucket}_\tau(x)$ which denotes the true index under $\mathcal{D}$.

Definition 2.1: [Label-Invariant Properties] A property $\Pi \subseteq \Delta(\mathcal{X})$ is said to be *label-invariant* (or *symmetric*) if it is closed under permutations of the domain: that is, if $\mathcal{D} \in \Pi$ and $\pi \in \mathcal{S}_\mathcal{X}$, then the distribution $\mathcal{D}_\pi$ defined by

$$\mathcal{D}_\pi[x] \stackrel{\text{def}}{=} \mathcal{D}[\pi(x)], \qquad x \in \mathcal{X},$$

is also in Π.

The set of distributions over $[N]$ with support size less than 20 is an example of a label-invariant property. Next, we give the definition for the *true* histogram of a distribution $\mathcal{D}$, followed by the τ-approximate histogram, which can be viewed as a discretisation of the true histogram of a distribution. For label-invariant properties it makes more sense to use the following notion of distance instead of total variation distance.

Definition 2.2: [Relabelling Distance] Given two distributions $\mathcal{D}, \mathcal{D}' \in \Delta([N])$, we define the relabelling distance between $\mathcal{D}$ and $\mathcal{D}'$ as

$$\texttt{RL}(\mathcal{D}, \mathcal{D}') \stackrel{\text{def}}{=} \min_{\pi \in \mathcal{S}_{[N]}} d_{\text{TV}}(\mathcal{D}, \mathcal{D}'_\pi) = \min_{\pi \in \mathcal{S}_{[N]}} \frac{1}{2} \sum_{x \in [N]} |\mathcal{D}[x] - \mathcal{D}'[\pi(x)]|$$

Definition 2.3: [Histograms Of Distributions] Define the normalised *histogram* of $\mathcal{D} \in \Delta([N])$ as the non-negative function $h_\mathcal{D} : [0,1] \to [0,1]$

$$h_\mathcal{D}(z) \;=\; \frac{1}{N}\Big|\{i \in [N] : \mathcal{D}[i] = z\}\Big| \quad \text{for each } z \in [0,1].$$

Thus, $h_\mathcal{D}(z)$ counts the fraction of domain elements with probability mass z. For a function any non-negative function h, we use $\text{Supp}(h)$ to denote the set $\{x \in [0,1] : h(x) > 0\}$. Observe that

$$\int_0^1 h_\mathcal{D}(z)\, dz = 1.$$

Definition 2.4: [(N,τ)-Bucketing] Fix $\mathcal{D} \in \Delta([N])$ and $\tau \in (0,1]$. Let $L_\tau = \left\lceil \frac{\log(N/\tau)}{\log(1+\tau)} \right\rceil = O(\log(N/\tau))$ denote the number of buckets. We denote with sets $\{B_j^\mathcal{D}\}_{j \in [L_\tau]}$ the τ-partitioning (bucketing) of the elements in the support of $\mathcal{D}$ where

$$B_\ell^{(\mathcal{D},\tau)} = \left\{ x \in [N] : \frac{\tau(1+\tau)^{\ell-1}}{N} < \mathcal{D}[x] \le \frac{\tau(1+\tau)^\ell}{N} \right\} \text{ for all } \ell \in \{1, \ldots, L_\tau\}$$

and

$$B_0^{(\mathcal{D},\tau)} = \left\{ x \in [N] : \mathcal{D}[x] \le \frac{\tau}{N} \right\}$$

where $[L_\tau] = \{0, 1, \ldots, L\}$.

Throughout this document, for any $x \in [N]$, we use $\text{Bucket}_\tau(x) \in [L_\tau]$ to denote the bucket to which x belongs. When the context is clear, we will often drop the superscript in $B_j^{(\mathcal{D},\tau)}$, and write just B_j. Note that the buckets above are disjoint, and cover the entire $[0,1]$ interval. So for any $z \in [0,1]$ it can belong to exactly *one* bucket. This allows to give the following definition for an approximate histogram.

Definition 2.5: [Approximate Histogram of a Distribution] Given a (N,τ)-bucketing of a distribution $\mathcal{D}$: $\{B_j\}_{j\in[L_\tau]}$, we define the normalised τ-histogram of $\mathcal{D}$ with the following non-negative function $h_{\mathcal{D}}^{(\tau)} : [0,1] \to [0,1]$, where

$$h_{\mathcal{D}}^{(\tau)}(z) \;=\; \begin{cases} \mathcal{D}\left[B_0\right], & \text{if } z \le \frac{\tau}{N}, \\ \mathcal{D}\left[B_\ell\right], & \text{if } z \in \left(\frac{\tau(1+\tau)^{\ell-1}}{N}, \frac{\tau(1+\tau)^{\ell}}{N}\right], \ \ell \in \{1,\ldots,L_\tau\}. \end{cases}$$

Figure 1 illustrates how the (N,τ) bucketing of the domain a distribution induces an approximate histogram. Observe in that above definition, the domain of $h_{\mathcal{D}}^{(\tau)}$ has at most $L_\tau + 1$ values i.e. we have discretised the true histogram into $L_\tau + 1$ values. Thus, the above histogram can be succinctly represented with the following collection of items: $\{p_j\}_{j\in[L_\tau]}$, where for all $j \in [L_\tau]$

$$p_j \overset{\text{def}}{=} \mathcal{D}\left[B_j^{(\mathcal{D},\tau)}\right] \overset{\text{def}}{=} \sum_{x \in B_j^{(\mathcal{D},\tau)}} \mathcal{D}\left[x\right]$$

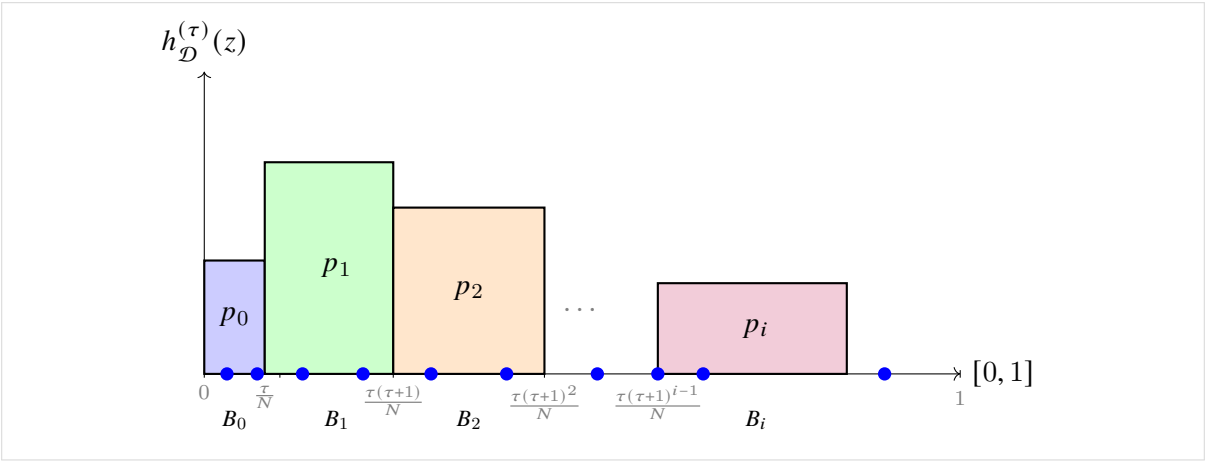


Figure 1: The approximate histogram $h_{\mathcal{D}}^{(\tau)}$ induced by a (N,τ)-partitioning of $[N]$ according to $\mathcal{D}$. The blue dots denote domain elements $x \in [N]$ positioned on the interval $[0,1]$ according their probability mass $\mathcal{D}[x]$. Buckets $(B_1,\ldots,B_L)$ denote a disjoint τ-partitioning of domain, where B_i is the set of domain elements x, such that $\mathcal{D}[x] \in \left(\frac{\tau(1+\tau)^{i-1}}{N}, \frac{\tau(1+\tau)^{i}}{N}\right]$. The mass of bucket i is denoted with $p_i = \sum_{x\in B_i} \mathcal{D}[x]$.

In the rest of the document, we use this succinct notation to represent approximate histograms of a distribution. As histograms are defined by non-negative functions over $[0,1]$, it is possible to define distances between histograms with the Earth-Mover Distance Function.

Definition 2.6: [Earth-Mover Distance and Relative Earth-Mover Distance] Let h and h' be two non-negative functions over $[0,1]$ such that

$$\sum_{x \in \text{Supp}(h)} h(x) = \sum_{x \in \text{Supp}(h')} h'(x).$$

The *Earth-Mover Distance (EMD)* between h and h', denoted $\text{EMD}(h, h')$, is defined as

$$\text{EMD}(h, h') = \min_{m} \sum_{x \in S(h)} \sum_{y \in S(h')} m(x,y) \cdot |x - y|,$$

where the minimum is taken over all non-negative functions $m : \text{Supp}(h) \times \text{Supp}(h') \to \mathbb{R}$ such that:

$$\sum_{y \in \text{Supp}(h)} m(x,y) = h(x), \qquad \forall x \in \text{Supp}(h),$$
$$\sum_{x \in \text{Supp}(h')} m(x,y) = h'(y), \qquad \forall y \in \text{Supp}(h').$$

The mental model, and the source of the metrics nomenclature is the following: Imagine we start with a non negative function h which has a hole for each $z \in [0,1]$, filled with $h(z)$ volume of dirt. In the case of approximate histograms, the holes with dirt are the buckets with non-zero mass, and the support size of h is the number of buckets. Given a histogram h', the earth-mover distance between h and h' is the minimum volume of dirt that must be moved from the support of h to the support of h' to transform h into h'. The following lemma is due to [Goldreich and Ron, 2020, Lemma 5] links distances between histograms to re-labelling distances.

Lemma 2.7: [Relationship Between Re-labelling Distance And Earth Mover Distance] For any two distributions $\mathcal{D}, \mathcal{D}' \in \Delta([N])$, let $h_{\mathcal{D}}$ and $h'_{\mathcal{D}}$ denote the histogram for $\mathcal{D}$ and $\mathcal{D}'$ respectively. Then we have

$$\texttt{RL}(\mathcal{D}, \mathcal{D}') = \frac{1}{2}\text{EMD}\left(h_{\mathcal{D}}, h'_{\mathcal{D}}\right)$$

Oracle Access Models Testers, provers, and verifiers in this paper take as input parameters such as the data domain size N, error tolerance, and completeness and soundness failure probabilities. They are also given oracle access to the distribution $\mathcal{D}$ comprising the problem instance as follows. A conditional oracle for distribution $\mathcal{D} \in \Delta([N])$, denoted by $\texttt{Cond}$, takes as input a set $S \subseteq [N]$ with the constraint $\mathcal{D}(S) > 0$, and outputs an $x \in S$ with probability $\mathcal{D}_S[x] = \frac{\mathcal{D}[x]}{\mathcal{D}[S]}$. If the testing algorithm sends S such that $\mathcal{D}(S) = 0$, then the oracle outputs $\textsf{FAIL}$. A $\texttt{PCond}$ oracle for a distribution $\mathcal{D} \in \Delta(N)$ is a conditional oracle with the restriction that inputs to the oracle must be of size exactly two or N.

The round complexity counts the number of rounds of interaction between the prover and the verifier, where one round consists of a message from the verifier, *and* the prover's response. For any algorithm A, we use parentheses $(\cdot)$ to distinguish between $\mathsf{A}^{\mathcal{D}}$ having full knowledge of the distribution, and $\mathsf{A}^{(\mathcal{D})}$ only being able to access $\mathcal{D}$ via oracles. Before describing proof systems, we recap the definition of a tolerant testing algorithm *without* a prover.

Definition 2.8: [Tolerant BPP Tester] A BPP tolerant tester for property $\Pi \subseteq \Delta([N])$ is a testing algorithm $\mathsf{T}^{(\mathcal{D})}(N, \tau_c, \tau_s)$ such that for any $\tau_c, \tau_s \in (0, 1]$ we have

- **Completeness**: For every $\mathcal{D} \in \Delta([N])$ such that $d_{\mathrm{TV}}(\mathcal{D}, \Pi) \leq \tau_c$,

$$\Pr\left[\mathsf{T}^{(\mathcal{D})}(N, \tau) = \mathsf{Yes}\right] \geq \frac{2}{3}$$

- **Soundness**: For every $\mathcal{D} \in \Delta([N])$ such that $d_{\mathrm{TV}}(\mathcal{D}, \Pi) > \tau_s$, we have

$$\Pr\left[\mathsf{T}^{(\mathcal{D})}(N, \tau) = \mathsf{Yes}\right] \leq \frac{1}{3}$$

Proof Systems The complexity of a proof system is measured by the number of oracle calls a verifier makes. In this work, to make accounting more granular, we report independent and identically distributed (iid) samples from the distribution and `PCond` queries separately. The total query complexity is the sum of the two measures. The communication complexity is measured in the number of bits[9] that the prover and verifier exchange throughout the proof.

Definition 2.9: [Tolerant IP Tester] A tolerant IP for property $\Pi \subseteq \Delta([N])$ is testing algorithm $\mathsf{T}^{(\mathcal{D})}(\tau_c, \tau_s, N)$ that interactively exchanges messages with an omniscient but untrusted prover $\mathsf{P}^{\mathcal{D}}(\tau_c, \tau_s, N)$, such that at the end of the interaction, for any $\tau_c, \tau_s \in (0, 1]$ we have

- **Completeness**: For every $\mathcal{D} \in \Delta([N])$, such that $d_{\mathrm{TV}}(\mathcal{D}, \Pi) \leq \tau_c$, there exists a prover P such that

$$\Pr\left[\Pi\left(\mathsf{P}^{\mathcal{D}}, \mathsf{T}^{(\mathcal{D})}; N, \tau\right) = \mathsf{Yes}\right] \geq \frac{2}{3}$$

- **Soundness**: For every $\mathcal{D} \in \Delta([N])$ such that $d_{\mathrm{TV}}(\mathcal{D}, \Pi) > \tau_s$, for *every* prover strategy $\widetilde{\mathsf{P}}$ we have

$$\Pr\left[\Pi\left(\widetilde{\mathsf{P}}^{\mathcal{D}}, \mathsf{T}^{(\mathcal{D})}; N, \tau\right) = \mathsf{Yes}\right] \leq \frac{1}{3}$$

Throughout this document, a distribution that is uniformly distributed over its support is referred to as a flat distribution.

[9] As any domain element can be represented in $\log_2 N$ bits, when suppressing log factors in asymptotic expressions, the communication complexity can be read as the number of domain elements exchanged.

Definition 2.10: [Nearly Flat Distributions] Given a parameter $\kappa \geq 1$, a probability distribution $\mathcal{D}$ over $[N]$ is said to be *κ-flat* if the probabilities of any two elements of its support are within a factor κ; that is,

$$\max_{x \in \mathrm{Supp}(\mathcal{D})} \mathcal{D}[x] \leq \kappa \cdot \min_{x \in \mathrm{Supp}(\mathcal{D})} \mathcal{D}[x] \,.$$

(Note that 1-flat distributions are exactly the flat distributions.) We further refer to 2-flat distributions as *nearly flat*.

2.1 Useful Tools

In this section, we describe a couple of useful lemmas that we repeatedly use later in the document.

Lemma 2.11: [Compare Algorithm] There is an algorithm `COMPARE` which, provided `PCond` oracle for an arbitrary probability distribution $\mathcal{D} \in \Delta([N])$, has the following guarantees. On input $\gamma, \beta \in (0,1]$ and $K \geq 1$, as well as two distinct elements $x, y \in [N]$, the algorithm makes $O\left(\frac{K \log 1/\beta}{\gamma^2}\right)$ queries to the oracle and outputs either High, Low, or a value $\alpha > 0$, such that the following holds.

- If $\frac{\mathcal{D}[x]}{\mathcal{D}[y]} > K$, then with probability at least $1-\beta$ the algorithm outputs either High, or a value $\alpha > 0$ such that $\left|\frac{\mathcal{D}[x]}{\mathcal{D}[y]} - \alpha\right| \leq \gamma$;
- If $\frac{\mathcal{D}[x]}{\mathcal{D}[y]} < 1/K$, then with probability at least $1-\beta$ the algorithm outputs either Low, or a value $\alpha > 0$ such that $\left|\frac{\mathcal{D}[x]}{\mathcal{D}[y]} - \alpha\right| \leq \gamma$;
- Otherwise, with probability at least $1-\beta$ it outputs a value $\alpha > 0$ such that $\left|\frac{\mathcal{D}[x]}{\mathcal{D}[y]} - \alpha\right| \leq \gamma$.

The proof follows from the guarantees of [Canonne et al., 2015, Lemma 2], specialized for the case of `PCond`.

Algorithm 1 `IsInSupport` algorithm.

Require: Given $x \in [N]$, and $y \in [N]$, failure probability parameter $\beta \in (0,1)$, oracle access to $\texttt{PCond}^{\mathcal{D}}$ for a nearly flat distribution $\mathcal{D}$.

Ensure: Check if $y \in \mathrm{Supp}(\mathcal{D})$

1: Set $T = \lceil \log_{3/2}(1/\beta) \rceil$
2: **for all** $1 \leq t \leq T$ **do**
3: $\quad s \leftarrow\!\$\ \texttt{PCond}^{\mathcal{D}}(\{x, y\})$
4: $\quad$ **if** $s = y$ **then return** Yes
5: **return** No

Checking Membership in Support

Lemma 2.12: The algorithm `IsInSupport` has the following guarantees. On inputs $x, y \in [N]$ and $\beta \in (0,1)$, it makes $O(\log(1/\beta))$ queries to the $\texttt{PCond}^{\mathcal{D}}$ oracle. Further, if $\mathcal{D}$ is nearly flat(Definition 2.10), then

- If $y \in \mathrm{Supp}(\mathcal{D})$, it returns Yes with probability at least $1-\beta$;
- If $y \notin \mathrm{Supp}(\mathcal{D})$ and $x \in \mathrm{Supp}(\mathcal{D})$, it returns No with probability 1.

Proof. The claimed query complexity is immediate from our setting of T, observing that the only `PCond` queries are on Line 3. Turning to the correctness, assume that $\mathcal{D}$ is nearly flat.

If $x = y$, the claims are trivially true and vacuous, respectively; we can thus assume $x \neq y$. Then, on the one hand, if $y \in \mathrm{Supp}(\mathcal{D})$, then (regardless of whether x is in $\mathrm{Supp}(\mathcal{D})$ or not), $\mathcal{D}[y] \geq \frac{1}{2} \cdot \mathcal{D}[x]$, and so

$$\Pr\left[\texttt{PCond}^{\mathcal{D}}(\{x,y\}) = x\right] = \frac{\mathcal{D}[x]}{\mathcal{D}[x] + \mathcal{D}[y]} \leq \frac{2}{3}.$$

The probability that none of the T independent queries returned y is then at most $2/3)^T \leq \beta$. On the other hand, if $y \notin \mathrm{Supp}(\mathcal{D})$ but $x \in \mathrm{Supp}(\mathcal{D})$, then $\Pr\left[\texttt{PCond}^{\mathcal{D}}(\{x,y\}) = x\right] = 1$, and so the algorithm returns No with probability 1. □

3 Lower Bounds For Pairwise Conditional Testers

In this section, we give a label-invariant property over a domain $[N]$ where any `PCond`-tester needs $\mathrm{poly}(N)$ queries. To do this, we will start by defining testing for promise problems and over meta-distributions.

Definition 3.1: [Testing for Promise Problem] A BPP tester for property promise $\Pi^{\mathsf{Yes}}, \Pi^{\mathsf{No}} \subseteq \Delta([N])$ is a testing algorithm T such that for any failure probability $\beta \in (0,1]$ we have

- **Yes Case**: For every $\mathcal{D} \in \Pi^{\mathsf{Yes}}$,
$$\Pr\left[\mathsf{T}^{(\mathcal{D})}(N,\beta) = \mathsf{Yes}\right] \geq 1-\beta$$
- **No Case**: For every $\mathcal{D} \in \Pi^{\mathsf{No}}$, we have
$$\Pr\left[\mathsf{T}^{(\mathcal{D})}(N,\beta) = \mathsf{No}\right] \geq 1-\beta$$

Definition 3.2: [Testing for Meta-Distributions] Given distribution subsets $\Pi^{\mathsf{Yes}}, \Pi^{\mathsf{No}} \subseteq \Delta([N])$, and distributions Q_{Yes} and Q_{No} over $\Pi^{\mathsf{Yes}}, \Pi^{\mathsf{No}}$ respectively, a BPP tester for Q_{Yes}, Q_{No} is a testing algorithm T such that for any $\beta \in (0,1]$, we have that

- **Yes Case**:
$$\Pr_{\mathcal{D} \sim Q_{\mathsf{Yes}}, \mathsf{T}} \left[\mathsf{T}^{(\mathcal{D})}(N, \beta) = \mathsf{Yes}\right] \geq 1 - \beta$$
- **No Case**:
$$\Pr_{\mathcal{D} \sim Q_{\mathsf{No}}, \mathsf{T}} \left[\mathsf{T}^{(\mathcal{D})}(N, \beta) = \mathsf{No}\right] \geq 1 - \beta$$

The main part of the argument will be to define a "hard promise problem." At the end of this section, we will argue that this also gives us a lower bound for a specific fixed label-invariant property.

Let K be a sufficiently large constant. The hard promise problem we consider will be the following:

$$\Pi_{\mathsf{Yes}} \stackrel{\mathsf{def}}{=} \{\mathcal{D} \in \Delta([N]) : |\mathrm{Supp}(\mathcal{D})| = N^{2/3}/K \wedge \mathcal{D} \text{ is flat}\} \tag{1}$$

$$\Pi_{\mathsf{No}} \stackrel{\mathsf{def}}{=} \{\mathcal{D} \in \Delta([N]) : |\mathrm{Supp}(\mathcal{D})| = N^{2/3} \wedge \mathcal{D} \text{ is flat}\} \tag{2}$$

We will let the meta-distributions Q_{Yes} and Q_{No} be the uniform distributions over each of the above properties.

We now state a theorem of Valiant that argues how one can use moment matching techniques to prove lower bounds for `Samp` testers for label-invariant properties. We state a modification of the theorem that applies for distinguishing between two label-invariant properties, and follows from the same proof as the original theorem.

Theorem 3.3: [Modified Version of Wishful Thinking Theorem [Valiant, 2008]] Fix label-invariant properties $\Pi_{\mathsf{Yes}}, \Pi_{\mathsf{No}}$. Given a positive integer m, a distance parameter τ, and two distributions $p_{\mathsf{Yes}} \in \Pi_{\mathsf{Yes}}, p_{\mathsf{No}} \in \Pi_{\mathsf{No}}$, suppose the following conditions hold:

1. No large elements:
$$\|p_{\mathsf{Yes}}\|_\infty, \|p_{\mathsf{No}}\|_\infty \leq \frac{1}{500m},$$
2. Moments (approximately) match:
$$\sum_{j=2}^{\infty} \frac{|m^{\mathsf{Yes}}(j) - m^{\mathsf{No}}(j)|}{\lfloor j/2 \rfloor! \sqrt{1 + \max\{m^{\mathsf{Yes}}(j), m^{\mathsf{No}}(j)\}}} \leq \frac{1}{24},$$
where $m^{\mathsf{Yes}}(j) = m^j \|p_{\mathsf{Yes}}\|_j^j$, $m^{\mathsf{No}}(j) = m^j \|p_{\mathsf{No}}\|_j^j$.

Then, any BPP tester for property promise $\Pi_{\mathsf{Yes}}, \Pi_{\mathsf{No}}$, with failure probability $\frac{1}{3}$ needs at least m samples.

We now connect testing for meta-distributions with testing for property promise problems.

Lemma 3.4: Let $(\Pi_{\mathsf{Yes}}, \Pi_{\mathsf{No}}) \subseteq \Delta([N])$ be the properties defined in Equation 1, and let Q_{Yes} and Q_{No} be the uniform distribution over $\Pi_{\mathsf{Yes}}, \Pi_{\mathsf{No}}$ respectively. If there exists a BPP tester for $Q_{\mathsf{Yes}}, Q_{\mathsf{No}}$ with failure probability β and `Samp` access to the distribution that uses ℓ samples from the distribution, then there exists a BPP tester for property promise $\Pi^{\mathsf{Yes}}, \Pi^{\mathsf{No}}$ with failure probability β and `Samp` access to the distribution that uses ℓ samples from the distribution.

Proof. We reduce from distinguishing Π_{Yes} and Π_{No} in the worst case to distinguishing meta-distributions Q_{Yes} and Q_{No}. Let A be the algorithm that distinguishes Q_{Yes} and Q_{No} with probability at least $1 - \beta$.

Fix any distribution $P \in \Pi_{\mathsf{Yes}} \cup \Pi_{\mathsf{No}}$. Consider a random permutation $\pi : [N] \to [N]$ of the domain. Note that $\pi(P)$ is distributed according to Q_{Yes} (if $P \in \Pi_{\mathsf{Yes}}$) or Q_{No} (if $P \in \Pi_{\mathsf{Yes}}$).

Let $S \leftarrow\!\!\$\ P^{\otimes \ell}$, and consider an algorithm A' (with blackbox access to A) that first applies π to each element of S. Observe that this process is equal to sampling the distribution from either Q_{Yes} or Q_{No} and then taking ℓ samples from the resulting distribution. Now, if Algorithm A' uses tester A on the samples $\pi(S)$, tester A can tell with probability at least $1 - \beta$ over the randomness of the meta-distribution, the randomness of the samples S, and the internal randomness of the algorithm whether the distribution was Q_{Yes} or Q_{No}. Hence, A' can distinguish with probability at least $1 - \beta$ over its internal randomness (which includes the random permutation), and the randomness of the sample S whether P was in Π_{Yes} or Π_{No}. □

Next, we use the wishful thinking theorem to prove a lower bound on the hard property promise problem above in the `Samp` model.

Lemma 3.5: Fix $\beta \leq \frac{1}{3}$. Any BPP tester for property promise $\Pi_{\mathsf{Yes}}, \Pi_{\mathsf{No}}$ (the "hard promise problem" described at the start of the section) with failure probability β and `Samp` access to the distribution needs $\Omega(N^{1/3})$ samples.

Proof. We will leverage the "wishful thinking theorem" (Theorem 3.3) in order to prove this lower bound.

We will choose $p_{\mathsf{Yes}} \in \Pi_{\mathsf{Yes}}$ and $p_{\mathsf{No}} \in \Pi_{\mathsf{No}}$ to be two distributions with maximally overlapping support, that is the support of the former is a subset of the latter. Let $m = C \cdot N^{1/3}$ for a sufficiently small constant $C > 0$.

We will verify the conditions of the theorem one at a time:

1. Firstly, $\|p_{\mathsf{Yes}}\|_\infty = \frac{K}{N^{2/3}} \leq \frac{1}{500CN^{1/3}} = \frac{1}{500m}$, $\|p_{\mathsf{No}}\|_\infty = \frac{1}{N^{2/3}} \leq \frac{1}{500CN^{1/3}} = \frac{1}{500m}$, where both inequalities hold for sufficiently large N.

2. Finally, we verify the condition on moments. Firstly, we compute $m^{\mathsf{Yes}}(j) = m^j \|p_{\mathsf{Yes}}\|_j^j = \frac{m^j K^{j-1}}{N^{2/3(j-1)}}$, and $m^{\mathsf{No}}(j) = m^j \|p_{\mathsf{No}}\|_j^j = \frac{m^j}{N^{2/3(j-1)}}$.

$$\begin{aligned}
\sum_{j=2}^{\infty} \frac{|m^{\mathsf{Yes}}(j) - m^{\mathsf{No}}(j)|}{\lfloor j/2 \rfloor! \sqrt{1 + \max(m^{\mathsf{Yes}}(j), m^{\mathsf{No}}(j))}} &\leq \sum_{j=2}^{\infty} |m^{\mathsf{Yes}}(j) - m^{\mathsf{No}}(j)| \\
&= \sum_{j=2}^{\infty} \frac{m^j K^{j-1}}{N^{2/3(j-1)}} \left(1 - \frac{1}{K^{j-1}}\right) \\
&\leq \sum_{j=2}^{\infty} \frac{m^j K^{j-1}}{N^{2/3(j-1)}} \leq \sum_{j=2}^{\infty} \frac{C^j K^{j-1} N^{j/3}}{N^{2/3(j-1)}} \leq \sum_{j=2}^{\infty} C^j K^{j-1} \leq \frac{C^2 K}{1 - CK} \leq \frac{1}{24},
\end{aligned}$$

where the first inequality follows by reducing the denominator to 1, the third inequality follows by substituting for m, the fourth inequality follows because $N^{j/3} \leq N^{2/3(j-1)}$ for $j \geq 2$, the fifth inequality follows by summing the geometric series and using the fact that $CK < 1$ for sufficiently small C, and the last inequality follows since C is a sufficiently small constant.

Then, invoking Theorem 3.3, we get that distinguishing between Π_{Yes} and Π_{No} requires at least m samples, thereby proving the lower bound. □

Finally, we will argue that for this hard promise problem, we can simulate PCond queries, thereby proving a lower bound for testers with access to a PCond oracle. Intuitively, the main idea is that since the distributions in both Π_{Yes} and Π_{No} are flat and the supports are small, PCond queries are not useful: they only provide information when the query set $\{x, y\}$ include both one element outside the support and one inside, and if the number of samples is small then finding such a $\{x, y\}$ can only happen with vanishing probability.

Theorem 3.6: [Hard Problem For PCond Tester] Every BPP tester using a PCond oracle for property promise Π_{Yes} and Π_{No} with failure probability 0.01 must make $\Omega\left(N^{1/3}\right)$ queries.

Proof. We give a simulation argument to show how PCond queries can be simulated without using a PCond oracle. Consider the following meta-distributions: Q_{Yes} is the uniform distribution over Π_{Yes} and Q_{No} the uniform distribution over Π_{No}. Let $\mathcal{D}$ be the distribution drawn from the meta-distribution. The simulation argument below will be agnostic to which meta-distribution it is drawn from.
Consider the following oracle $\texttt{PCond}^{\text{Sim}}$ that simulates oracle PCond given access to a sample set S.

$\texttt{PCond}^{\text{Sim}}$

Input: Description of domain $[N]$, Sample (Multi-set) $S \subseteq [N]$, query points z_1, z_2.
Output: $x \in \{z_1, z_2, \perp\}$.

- If $z_1, z_2 \notin S$, output $\perp$.
- If $z_1, z_2 \in S$, with probability $1/2$ output z_1 and with probability $1/2$ output z_2.
- If $z_1 \in S$, $z_2 \notin S$ output z_1, and if $z_1 \notin S$, $z_2 \in S$, output z_2.

Figure 2: Simulation of PCond Oracle with only access to samples from $\mathcal{D}$

Any algorithm A that uses a PCond oracle can be thought of without loss of generality in the following way: it takes S consisting of k i.i.d. samples from the distribution upfront, and then adaptively makes PCond queries $q_1 = (y_1, y_1'), \ldots, q_m = (y_m, y_m')$ (with some computations in between).

For reasonable values of m and k, we will now argue that any such algorithm can be simulated with the $\texttt{PCond}^{\text{Sim}}$ oracle, where the input S is the sample drawn by algorithm A. In particular, we will consider $m, k \leq C' N^{1/3}$ for some constant C'.

Fix any sample S for the rest of the argument. Define the event E_r as follows:

$$E_r = \bigcap_{i=1}^{r} \left(y_i \in S \vee y_i \notin \text{Supp}\left(\mathcal{D}\right)\right) \wedge \left(y_i' \in S \vee y_i' \notin \text{Supp}\left(\mathcal{D}\right)\right) .$$

We first note that if event E_r happens, then the output of $\texttt{PCond}^{\text{Sim}}$ when run on $[N], S, y_i, y_i'$ is identically distributed to the output of $\texttt{PCond}$ when run on $[N], S, y_i, y_i'$ for all $i \in [r]$. To see this, we consider the different cases: when $y_i, y_i' \notin S$, then by the definition of event E_r, $y_i, y_i' \notin \text{Supp}(\mathcal{D})$, and so the output of $\texttt{PCond}$ would be $\perp$ (identical to the output of $\texttt{PCond}^{\text{Sim}}$). If $y_i, y_i' \in S$, then by the definition of the promise problem, we have that $\mathcal{D}$ is uniform over its support and so the output of $\texttt{PCond}$ is equiprobable between y_i and y_i', identical to the distribution of the output of $\texttt{PCond}^{\text{Sim}}$. And finally, if one of y_i, y_i' is in S and the other is not in S (say WLOG that $y_i \in S$), then by the definition of E_r, $y_i' \notin \text{Supp}(\mathcal{D})$, and so the output of $\texttt{PCond}$ would be y_i, identical to the output of $\texttt{PCond}^{\text{Sim}}$.

Next, we bound the probability of $\overline{E_r}$ (to argue that this bad event happens with low probability).

$$\Pr(\overline{E_r}) \leq \Pr(\overline{E_r} \mid E_1, \ldots, E_{r-1}) + \sum_{i=1}^{r-1} \Pr(\overline{E_i}) \tag{3}$$

Consider the first term in the RHS above. Conditioning on $E_1, \ldots, E_{r-1}$ occurring, we have that for $\overline{E_r}$ to occur, we need that either y_r or y_r' belong to $\text{Supp}(\mathcal{D})$ but do not belong to sample S. Let $S' = S \cup \{y_1, \ldots, y_{r-1}, y_1', \ldots, y_{r-1}'\}$ (S' is defined as a set and hence does not have repeats). Note that we always have $|S'| \leq N/2$ (all sets that can occur satisfy this condition since $m, k \leq C'N^{1/3}$). Observe that since we conditioned on $E_1, \ldots, E_{r-1}$, each element of S' either belongs to S or is not in $\text{Supp}(\mathcal{D})$. Next, by symmetry, observe that all elements in $[N] \setminus S'$ have equal probability of being in the support of $\mathcal{D}$ conditioned on $E_1, \ldots, E_{r-1}$, and since $N - |S'| \geq N/2$ that probability is upper bounded by $\frac{|\text{Supp}(\mathcal{D})| - |S|}{N/2} \leq \frac{|\text{Supp}(\mathcal{D})|}{N/2} \leq \frac{N^{2/3}}{N/2} = \frac{2}{N^{1/3}}$. By a union bound, we get that $\Pr(\overline{E_r} \mid E_1, \ldots, E_{r-1}) \leq \frac{4}{N^{1/3}}$.

Hence, substituting back in Equation 3, we get that $\Pr(\overline{E_m}) \leq \frac{4m}{N^{1/3}}$. Thus, with probability at least $1 - \frac{4m}{N^{1/3}}$ over the choice of the distribution and the samples from the distribution, we get that Algorithm A run with $\texttt{PCond}^{\text{Sim}}$ perfectly simulates Algorithm A run with $\texttt{PCond}$. Also observe that A run with $\texttt{PCond}^{\text{Sim}}$ is a BPP tester that makes only $\texttt{Samp}$ queries.

Now, for an appropriately chosen constant $C' > 0$, assume by way of contradiction that algorithm A making $C'N^{1/3}$ queries to the $\texttt{PCond}$ oracle is a BPP tester for property promise Π_{Yes} and Π_{No} with failure probability at most 0.01. This implies that algorithm A making $C'N^{1/3}$ queries to the $\texttt{PCond}$ oracle is a BPP tester for Q_{Yes} and Q_{No} with failure probability at most 0.01. By the above argument A making $C'N^{1/3}$ queries to the $\texttt{PCond}^{\text{Sim}}$ oracle is a BPP tester for Q_{Yes} and Q_{No} with failure probability at most $0.01 + \frac{4m}{N^{1/3}} \leq 0.02$ where the last inequality holds for sufficiently small C' using the fact that $m \leq C'N^{1/3}$. This guarantees us that there is a BPP tester for Q_{Yes} and Q_{No} that makes only $\texttt{Samp}$ queries with failure probability at most 0.02 that uses $C'N^{1/3}$ samples from the distribution.

However, by Lemmas 3.4 and 3.5, we get that any BPP tester with $\texttt{Samp}$ access to the distribution that distinguishes Q_{Yes} and Q_{No} with probability at least 0.98 over the draw of the distribution, the samples from the distribution, and the internal randomness of the algorithm needs $\Omega(N^{1/3})$ samples. This gives a contradiction, thereby proving the theorem. □

Finally, observe that $d_{TV}(\Pi_{\mathsf{Yes}}, \Pi_{\mathsf{No}}) > \frac{1}{2}$ for sufficiently large support sizes, which also means we have proved the following result.

Corollary 3.7: Every BPP tester using a $\texttt{PCond}$ oracle for property Π_{Yes} with proximity parameter $\tau \leq 1/2$ and failure probability 0.01 must make $\Omega\left(N^{1/3}\right)$ queries.

Remark 1 We note that with the techniques above, the lower bound of $\Omega(N^{1/3})$ is the best we can hope to prove by varying the support sizes in the promise problems. In choosing these support sizes $s_1 < s_2$ in defining $\Pi_{\mathsf{Yes}}, \Pi_{\mathsf{No}}$, we trade-off between two parameter settings, the error probability of the simulation oracle (where we simulate `PCond` queries using `Samp` in the proof of Theorem 3.6), which is $O(ms_2/N)$, where m is the number of `PCond` queries made, and the lower bound obtained for the `Samp` model, which would be $m = \Omega(\sqrt{s_1})$. Balancing these inequalities (and also noting that the error probability needs to be sufficiently small) gives that setting $s_1, s_2 \approx N^{2/3}$ is optimal, motivating our choices and giving a lower bound of $\Omega(N^{1/3})$. It is an open question whether this lower bound can be improved by tweaking our approach or by other techniques.

4 Support Size Verification

In this section, we describe a succinct proof system for the two-sided support size decision problem, which may be of independent interest. This proof system, when used in conjunction with the The `EstimateNeighborhood` procedure of [Canonne et al., 2015, Lemma 3], allows a tester to accurately approximate the mass of "important" points in *any* "relevant/heavy" bucket of the (N, τ)-histogram for the distribution. This ability to estimate the mass of points in heavy enough buckets is critical for verifying the prover's claimed histogram, which then allows us to verify any label-invariant property. Our key technical contribution here is to ensure that the communication complexity of the proof system is sub-linear in the domain size N while also ensuring polylogarithmic in N query and sample complexities. To ensure low communication, we give two different protocols to apply based on the parameter range of interest.

4.1 Proofs For Testing Support Size For Distributions With Large Support

To start with, we describe with Figure 3, a proof system for distributions with large support, which requires us to run two separate proof systems sequentially for the two sides of the test (testing whether the claimed support is too large or too small).

Uniform Protocol

Common Input: Description of domain $[N]$, parameters $\alpha, \delta \in (0, 1]$, and A', A, B, B' with $\alpha \leq \frac{3}{2}\frac{B'-B}{B}$.
Verifier Input: `PCond` access to an $(1+\alpha)$-flat distribution $\mathcal{D} \in \Delta([N])$.
Prover Input: Full description of $\mathcal{D}$.

Draw $x \leftarrow\!\!\$\ \mathcal{D}$.

Test 1: "Detect Small Support"

- Compute the values p_{Reject}, q_{Accept} as in Claim 4.2, and set

$$\Delta_1 = (1 - q_{\mathsf{Accept}}) - p_{\mathsf{Reject}} = \Omega\left(\left(\frac{A - A'}{A}\right)^2\right)$$

- Run sequentially the proof system **Test 1** of Fig. 4, on input x, A, A' a total of

$$T_1 \stackrel{\text{def}}{=} O\left(\frac{\log(1/\delta)}{\Delta_1^2}\right)$$

times, with fresh randomness for each of the T_1 runs. Let $0 \le t_1 \le T_1$ be the number of runs in which the proof system returns Accept.

- Set the outcome of this test to Accept if $t_1 < (p_{\mathsf{Reject}} + \frac{\Delta_1}{2})T_1$, and to Reject otherwise.

Test 2: "Detect Large Support"

- Compute the values p'_{Reject}, q'_{Accept} as in Claim 4.3, and set

$$\Delta_2 = (1 - q'_{\mathsf{Accept}}) - p'_{\mathsf{Reject}} = \Omega\left(\frac{(B' - B)^2}{B'^2}\frac{B}{B'}\right)$$

- Run sequentially the proof system **Test 2** of Fig. 5, on input B, B' a total of

$$T_2 \stackrel{\text{def}}{=} O\left(\frac{\log(1/\delta)}{\Delta_2^2}\right)$$

times, with fresh randomness for each of the T_2 runs. Let $0 \le t_2 \le T_2$ be the number of runs in which the proof system returns Accept.

- Set the outcome of this test to Accept if $t_2 < (p'_{\mathsf{Reject}} + \frac{\Delta_2}{2})T_2$, and to Reject otherwise.

Final Output: The verifier outputs Accept if and only if both tests above were set to Accept.

Figure 3: Proof System For Support Size Range For Distributions With Large Support

Lemma 4.1: [Support Size Difference For Large Support Distributions] Fix $N \in \mathbb{N}$, and let $0 \leq \alpha \leq 1$. Fix integer parameters $A' < A < B < B'$ such that $A \geq \sqrt{N}$ and $\alpha \leq \frac{3}{2} \cdot \frac{B'-B}{B}$, and failure probability $\delta \in (0,1]$. Given $\mathtt{PCond}$ access to a $(1+\alpha)$-flat distribution $\mathcal{D} \in \Delta([N])$, the interactive proof system described in Figure 3 decides the following promise problem completeness and soundness errors are at most δ:

$$\Pi_{\mathsf{Yes}} \stackrel{\mathsf{def}}{=} \{\mathcal{D} \in \Delta([N]) : A \leq |\mathrm{Supp}(\mathcal{D})| \leq B \wedge \mathcal{D} \text{ is } (1+\alpha)\text{-flat}\} \tag{4}$$

$$\Pi_{\mathsf{No}} \stackrel{\mathsf{def}}{=} \{\mathcal{D} \in \Delta([N]) : (|\mathrm{Supp}(\mathcal{D})| \leq A' \vee |\mathrm{Supp}(\mathcal{D})| \geq B') \wedge \mathcal{D} \text{ is } (1+\alpha)\text{-flat}\} \tag{5}$$

The proof system has communication complexity

$$O\left(\max\left(\frac{N}{A} \cdot \left(\frac{A}{A-A'}\right)^4, \frac{N}{B'} \cdot \left(\frac{B'}{B}\right)^2 \left(\frac{B'}{B'-B}\right)^4 \log\frac{1}{\delta}\right)\right)$$

sample complexity

$$O\left(\left(\frac{B'}{B}\right)^2 \left(\frac{B'}{B'-B}\right)^4 \log\frac{1}{\delta}\right),$$

query complexity

$$\widetilde{O}\left(\left(\frac{A}{A-A'}\right)^4 \log\frac{1}{\delta}\right),$$

and round complexity

$$O\left(\max\left(\left(\frac{A}{A-A'}\right)^4, \left(\frac{B'}{B}\right)^2 \left(\frac{B'}{B'-B}\right)^4\right) \log\frac{1}{\delta}\right).$$

Proof. The various complexities immediately follow from those of Claims 4.2 and 4.3, after repetition for $T_1 = O\left(\left(\frac{A}{A-A'}\right)^4 \log\frac{1}{\delta}\right)$ and $T_2 = O\left(\left(\frac{B'}{B}\right)^2 \left(\frac{B'}{B'-B}\right)^4 \log\frac{1}{\delta}\right)$ rounds, respectively. The stated completeness and soundness errors follow from (sequential) repetition with a threshold rule, along with a standard Hoeffding bound.[10] Amplifying both **Test 1** and **Test 2** by sequential repetition to correctly estimate their rejection probability, except with probability $\frac{\delta}{2}$, and taking a union bound over the two, yields the claimed statement. □

Test 1

Common Input: Description of domain $[N]$, integers parameters $A' < A$.
Verifier Input: $\mathtt{PCond}$ access to a nearly flat distribution $\mathcal{D} \in \Delta([N])$, element $x \in \mathrm{Supp}(\mathcal{D})$.
Prover Input: Full description of $\mathcal{D}$.

1. Set $c \stackrel{\mathsf{def}}{=} \frac{A-A'}{2A}$, $s \stackrel{\mathsf{def}}{=} c\frac{N}{A}$, $\beta \stackrel{\mathsf{def}}{=} c^2 e^{-c}$.
2. $\mathsf{T} \to \mathsf{P}^{\mathcal{D}}$: Send set Y where $Y = (y_1, \ldots, y_s) \leftarrow\!\!\$\ \mathsf{Uniform}[N]$, and ask the prover to send

[10] Namely, if a single test has rejection probabilities p, q with $q \geq p + \Delta$ in the Yes and No cases, respectively, repeating the test sequentially (with fresh randomness) a total of $O(\log(1/\delta)/\Delta^2)$ times and thresholding at $p + \frac{\Delta}{2}$ results in a correct outcome with probability at least $1 - \frac{\delta}{2}$.

back any $\widetilde{y} \in Y$ such that $\widetilde{y} \in \mathrm{Supp}(\mathcal{D})$.

3. $\mathsf{T} \leftarrow \mathsf{P}^{\mathcal{D}}$: Sends back $\widetilde{y} \in \mathrm{Supp}(\mathcal{D})$. If no such $\widetilde{y}$ exists, then output FAIL.

4. T computation: If prover outputs FAIL, then output Reject. Otherwise, check $\widetilde{y} \in \mathrm{Supp}(\mathcal{D})$ using the IsInSupport algorithm with reference x and probability of failure β: if it returns Yes, output Accept, and Reject otherwise.

Figure 4: First Proof System For Support Size Range For Distributions With Large Support

.

Claim 4.2: [Detecting Small Support] The proof system given in Fig. 4 has the following guarantees. On input access to a nearly flat distribution $\mathcal{D}$, along with an element $x \in \mathrm{Supp}(\mathcal{D})$ and integers $A' < A \leq N$, it has communication complexity $O(N/A)$, query complexity $O\left(\log \frac{A}{A-A'}\right)$ (and sample complexity 0), and decides between (1) $|\mathrm{Supp}(\mathcal{D})| \geq A$ and (2) $|\mathrm{Supp}(\mathcal{D})| \leq A'$ with completeness and soundness errors at most p_{Reject} and q_{Accept}, explicit values such that

$$1 - q_{\mathsf{Accept}} > p_{\mathsf{Reject}} + \Omega\left(\left(\frac{A-A'}{A}\right)^2\right).$$

Moreover, it has round complexity 1.

Proof. The communication complexity follows from our choice of $s = O(N/A)$, and the query complexity from the call to IsInSupport algorithm, which has query complexity

$$O\left(\log \frac{1}{\beta}\right) = O\left(\log \frac{1}{c} + c\right) = O\left(\log \frac{1}{c}\right) = O\left(\log \frac{A}{A-A'}\right)$$

by our setting of parameters and Lemma 2.12. Turning to correctness, we analyze the completeness and soundness claims separately.

Completeness: assume that $|\mathrm{Supp}(\mathcal{D})| \geq A$. We have the probability that a uniformly sampled point is in the support is at least $\frac{A}{N}$. If any of the samples $y_1, \ldots, y_s$ is in the support of $\mathcal{D}$, then the honest prover is able to get the verifier to accept in **Test 1** by sending one of these samples, unless the call to IsInSupport algorithm then fails (which since the distribution is nearly flat happens with probability at most β by Lemma 2.12). Therefore, by a union bound, the completeness error for the first test is upper bounded as follows:

$$\begin{aligned}
&\Pr[\mathsf{T} \text{ outputs Reject}] \\
&= \Pr_{y_1,\ldots,y_s \leftarrow\$ [N]}[\{y_1,\ldots,y_s\} \cap \mathrm{Supp}(\mathcal{D}) = \emptyset] \\
&\quad + \Pr_{y_1,\ldots,y_s \leftarrow\$ [N]}[\{y_1,\ldots,y_s\} \cap \mathrm{Supp}(\mathcal{D}) \neq \emptyset,\ \texttt{IsInSupport} \text{ algorithm incorrect}] \\
&\leq \left(1 - \frac{A}{N}\right)^s + \beta \\
&\leq \left(1 - \frac{A}{N}\right)^{c\frac{N}{A}} + \beta \\
&\leq e^{-c} + c^2 e^{-c} = (1+c^2)e^{-c} = p_{\mathsf{Reject}}.
\end{aligned}$$

recalling our choice of s, β, and using the inequality $1 - x \leq e^{-x}$.

Soundness: conversely, assume that $|\mathrm{Supp}(\mathcal{D})| \leq A'$. **Test 1** can only accept if at least one of the uniformly chosen $y_1, \ldots, y_s$ falls in the support of $\mathcal{D}$ (otherwise it will always reject, as the IsInSupport algorithm has one-sided error in that case). By a union bound over all s uniform samples, this leads to

$$\begin{aligned}\Pr[\mathsf{T} \text{ outputs } \mathsf{Accept} \text{ in } \textbf{Test 1}] &\leq \Pr_{y_1,\ldots,y_{s_1} \leftarrow\$ \mathsf{Uniform}[N]}\left[\exists j \in [s] \text{ s.t. } y_j \in \mathrm{Supp}(\mathcal{D})\right] \\ &\leq \frac{A'}{N} s = q_{\mathsf{Accept}}\,.\end{aligned}$$

Equivalently, multiplying and dividing q_{Accept} by A, and using the fact that $1 - 2c = \frac{A'}{A}$ from the setting of c,

$$\Pr[\mathsf{T} \text{ outputs } \mathsf{Reject}] \geq 1 - \frac{A'}{A} \cdot \frac{A}{N} s = 1 - c \cdot \frac{A'}{A} = 1 - c(1-2c) = 1 - c + 2c^2 = q_{\mathsf{Reject}}\,.$$

Now, using the fact that $e^{-c} \leq 1 - c + c^2/2$

$$1 - c + 2c^2 - (1 + c^2)e^{-c} \geq \frac{c^2}{2}$$

which means the gap between the rejection probability in the completeness and soundness cases is at least

$$q_{\mathsf{Reject}} - p_{\mathsf{Reject}} \geq \frac{c^2}{2} = \Omega\left(\left(\frac{A - A'}{A}\right)^2\right)$$

□

Test 2

Common Input: Description of domain $[N]$, integers parameters $B < B'$.
Verifier Input: PCond access to an $(1+\alpha)$-flat distribution flat distribution $\mathcal{D} \in \Delta([N])$, where $\alpha \leq \frac{3}{2}\frac{B'-B}{B}$.
Prover Input: Full description of $\mathcal{D}$.

1. Set $c \stackrel{\mathsf{def}}{=} \frac{B'-B}{18B'}$, $s \stackrel{\mathsf{def}}{=} c\frac{N}{B'}$.
2. T: Draw $x \leftarrow\$ \mathcal{D}$ and $(z_1, \ldots, z_s) \leftarrow\$ \mathsf{Uniform}[[N]]$. Form the tuple $S \stackrel{\mathsf{def}}{=} (x, z_1, \ldots, z_s)$.
3. $\mathsf{T} \to \mathsf{P}^{\mathcal{D}}$: Send the permuted tuple $S' \stackrel{\mathsf{def}}{=} \{S_{\pi(1)} \ldots, S_{\pi(s+1)}\}$, where π is a permutation of $[s+1]$ chosen uniformly at random.
4. $\mathsf{T} \leftarrow \mathsf{P}^{\mathcal{D}}$: Return $\widetilde{y} \leftarrow\$ \{i \in [s+1] : S'_i \in \mathrm{Supp}(\mathcal{D})\}$, an index chosen uniformly at random among all those for which the corresponding element of S' is in the support of $\mathcal{D}$.
5. If $\widetilde{y} \neq \pi(1)$ (i.e., $\widetilde{y}$ does not correspond to the index of x) output Reject, otherwise output Accept.

Figure 5: Second Proof System For Support Size Range For Distributions With Large Support

Claim 4.3: [Detecting Large Support] The proof system given in Fig. 5 has the following guarantees. On input access to an $(1+\alpha)$-flat distribution $\mathcal{D}$, along with an element $x \in \mathrm{Supp}(\mathcal{D})$ and integers $B < B' \leq N$ such that $0 \leq \alpha \leq \frac{3}{2}\frac{B'-B}{B}$, it has communication complexity $O(N/B')$, sample complexity 1 (and query complexity 0), and decides between (1) $|\mathrm{Supp}(\mathcal{D})| \leq B$ and (2) $|\mathrm{Supp}(\mathcal{D})| \geq B'$ with completeness and soundness errors at most p'_{Reject} and q'_{Accept}, explicit values such that

$$1 - q'_{\mathsf{Accept}} > p'_{\mathsf{Reject}} + \Omega\left(\left(\frac{B'-B}{B'}\right)^2 \frac{B}{B'}\right).$$

Moreover, it has round complexity 1.

Proof. The communication complexity follows from our choice of $s = O(N/B')$, and the sample complexity from the draw of x (only call to the oracle for $\mathcal{D}$).

Turning to correctness, we first define the following three disjoint events:

- $\mathcal{E}_0$: none of the s uniformly drawn elements $z_1, \ldots z_s$ falls in the support of the distribution; that is, $|\{z_1, \ldots, z_s\} \cap \mathrm{Supp}(\mathcal{D})| = 0$.
- $\mathcal{E}_1$: exactly one of the s uniformly drawn elements $z_1, \ldots z_s$ falls in the support of the distribution; that is, $|\{z_1, \ldots, z_s\} \cap \mathrm{Supp}(\mathcal{D})| = 1$.
- $\mathcal{E}_{\geq 2}$: at least two of the s uniformly drawn elements $z_1, \ldots z_s$ falls in the support of the distribution; that is, $|\{z_1, \ldots, z_s\} \cap \mathrm{Supp}(\mathcal{D})| \geq 2$.

The intuition is that a prover (honest or not) is able to send back the index of the verifier's true sample x whenever $\mathcal{E}_0$ holds; under $\mathcal{E}_1$, while a prover cannot always send the index of x, there is still a strategy for them to guess it (among two alternatives) with probability $\approx 1/2$. Under $\mathcal{E}_{\geq 2}$, there is not much we can say, but fortunately this is a very unlikely event in the completeness case.

With this in hand, we analyse the completeness and soundness claims. For convenience, set

$$\lambda \stackrel{\mathsf{def}}{=} \frac{B}{B'} < 1$$

and observe that our assumption on α implies $\alpha \leq \frac{3}{2} \cdot \frac{1-\lambda}{\lambda}$.

Completeness: Assume $|\mathrm{Supp}(\mathcal{D})| \leq B$. As discussed above, if $\mathcal{E}_0$ holds then the honest prover succeeds, as it can uniquely identify x in S'. If $\mathcal{E}_1$ holds (in which case $|S \cap \mathrm{Supp}(\mathcal{D})| = 2$), then the honest prover can still succeed in returning the index of x with probability $1/2$, by returning one of the two possible indices uniformly at random. If $\mathcal{E}_{\geq 2}$ holds, while there is still a positive probability of guessing the index of x correctly, we can (conservatively) neglect it for the analysis and count $\mathcal{E}_{\geq 2}$ as a failure. This leads to the following bound for the rejection probability, where we use the fact that $|\{z_1, \ldots, z_s\} \cap \mathrm{Supp}(\mathcal{D})|$ is a Binomial random variable

with parameters s and $\frac{|\mathrm{Supp}(\mathcal{D})|}{N} \leq \frac{B}{N}$:

$$\begin{aligned}
\Pr\left[\mathsf{T} \text{ outputs } \mathsf{Reject}\right] &\leq 0 \cdot \Pr\left[\mathcal{E}_0\right] + \frac{1}{2} \cdot \Pr\left[\mathcal{E}_1\right] + 1 \cdot \Pr\left[\mathcal{E}_{\geq 2}\right] \\
&\leq \frac{1}{2} \cdot s\frac{B}{N}\left(1 - \frac{B}{N}\right)^{s-1} + \Pr\left[\mathcal{E}_{\geq 2}\right] \\
&= \frac{c\lambda}{2}\left(1 - \frac{c\lambda}{s}\right)^{s-1} + \Pr\left[\mathcal{E}_{\geq 2}\right] \\
&\leq \frac{c\lambda}{2}\frac{e^{-c\lambda}}{1 - \frac{c\lambda}{s}} + \binom{s}{2}\left(\frac{B}{N}\right)^2 \\
&\leq \frac{c\lambda}{2}\frac{1}{1 - \frac{c\lambda}{2}} + \frac{(c\lambda)^2}{2} \\
&\leq \frac{c\lambda}{2} + (c\lambda)^2 \leq \frac{c\lambda}{2} + c^2\lambda = p'_{\mathsf{Reject}},
\end{aligned}$$

by our setting of $s = c\frac{N}{B'}$, and crudely bounding $\Pr\left[\mathcal{E}_{\geq 2}\right] \leq \binom{s}{2}\left(\frac{B}{N}\right)^2 \leq \frac{s^2}{2}\left(\frac{c\lambda}{s}\right)^2$ before concluding with $\frac{1}{1-x} \leq 1 + 2x$ (which holds for $x \in [0, 1/2]$). Importantly, we here implicitly used the fact that $\Pr[\mathcal{E}_1] = \Pr_{X \sim \mathrm{Bin}(s,p)}\left[X = 1\right] = sp(1-p)^{s-1}$ is non-decreasing in the Binomial parameter p, as $\frac{B}{N}$ is only an upper bound on this parameter. One can check that this is true as long as $p \leq \frac{1}{s+1}$, which is satisfied for us, as, given our setting of s and c, we have

$$p \leq \frac{B}{N} \leq \frac{B}{B'} \leq 9\frac{N}{B' - B} = \frac{1}{2s} \leq \frac{1}{s+1}.$$

Remark 2 Note that the above analysis critically saves (almost) a factor 2 over a naive union bound, which using the above notation would have given $\leq c\lambda$. Instead, we get (almost) $\leq \frac{c\lambda}{2}$. This is critical, as for sequential repetition, we need $p'_{\mathsf{Reject}} < q'_{\mathsf{Reject}} \approx \frac{1}{2}(1 - e^{-c})$, which would have been impossible if we didn't save this factor 2, given that $\lambda \approx 1 - \tau$ can be very close to 1 in the settings we use these protocols in.

Soundness: Assume $|\mathrm{Supp}\left(\mathcal{D}\right)| \geq B'$. If $|S \cap \mathrm{Supp}\left(\mathcal{D}\right)| \geq 2$ (i.e., if $\bar{\mathcal{E}}_0 = \mathcal{E}_1 \cup \mathcal{E}_{\geq 2}$ holds), we claim that the prover can only return the index of x (and thus make the tester accept) with probability at most $\frac{1+\alpha}{2+\alpha} = \frac{1}{2} + O(\alpha)$. A dishonest prover maximizes its chance of returning the index of x by choosing the index with the highest conditional probability of being $\pi(1)$ given the permuted tuple S'. By Bayes' rule, this conditional probability of an index is proportional to the probability mass of the corresponding element under $\mathcal{D}$. Let z_i be another element in $\mathrm{Supp}\left(\mathcal{D}\right)$. Then since the distribution is $(1+\alpha)$-flat, the dishonest prover chooses the correct index of x with probability at most

$$\frac{\mathcal{D}[x]}{\mathcal{D}[x] + \mathcal{D}[z_i]} \leq \frac{1+\alpha}{2+\alpha}.$$

Of course, if $|S \cap \mathrm{Supp}\left(\mathcal{D}\right)| = 1$ (i.e., if $\mathcal{E}_0$ holds) then the prover can always identify x and

return the correct index. Hence, we have

$$\begin{aligned}\Pr[\mathsf{T} \text{ outputs } \mathsf{Accept}] &= \Pr[\text{Prover guesses } x \text{ from } S] \Pr\left[\bar{\mathcal{E}}_0\right] + 1 \cdot \Pr[\mathcal{E}_0] \\ &\leq \frac{1+\alpha}{2+\alpha} \Pr\left[\bar{\mathcal{E}}_0\right] + \Pr[\mathcal{E}_0] \\ &= \frac{1+\alpha}{2+\alpha} + \frac{1}{2+\alpha} \Pr[\mathcal{E}_0] \\ &\leq \frac{1+\alpha}{2+\alpha} + \frac{1}{2+\alpha}\left(1 - \frac{B'}{N}\right)^s \\ &\leq \frac{1+\alpha}{2+\alpha} + \frac{1}{2+\alpha} e^{-c} = q'_{\mathsf{Accept}},\end{aligned}$$

the last inequality from our choice of $s = c\frac{N}{B'}$. Equivalently,

$$\Pr[\mathsf{T} \text{ outputs } \mathsf{Reject}] \geq 1 - q'_{\mathsf{Accept}} = \frac{1}{2+\alpha}\left(1 - e^{-c}\right) \geq \frac{1}{2} \cdot \frac{1 - e^{-c}}{1 + \frac{3}{4} \cdot \frac{1-\lambda}{\lambda}} = q'_{\mathsf{Reject}},$$

where the last inequality follows by the assumption that $\alpha \leq \frac{3}{2} \cdot \frac{B'-B}{B}$. Now, by our setting of $c = \frac{B'-B}{18B'} = \frac{1-\lambda}{18}$, we have

$$4\frac{1 - e^{-c}}{(1+4c)c} - 3 \geq 1 - 18c = \lambda$$

where the first inequality follows from convexity of the function $f : x \mapsto 4\frac{1-e^{-x}}{(1+4x)x} - 3$ (extended at 0 by continuity), above its tangent $x \mapsto 1 - 18x$ at 0. Reorganizing and recalling the expressions of $p'_{\mathsf{Reject}}, q'_{\mathsf{Reject}}$, is equivalent to

$$q'_{\mathsf{Reject}} \geq p'_{\mathsf{Reject}} + c^2\lambda,$$

concluding the proof since $c^2\lambda = \Theta\left(\left(\frac{B'-B}{B'}\right)^2 \frac{B}{B'}\right)$.

□

4.2 Proofs For Testing Support Size For Distributions With Small Support

The techniques described above to failed to give us sub-linear communication when dealing with distributions with small supports. To deal with this case, we describe a non-interactive proof system with sublinear communication, that decides the support size decision problem.

Small Support Size Protocol

Common Input: Description of domain $[N]$, parameters A', A, B, B' where $B \leq \sqrt{N}$. Soundness parameters δ_s and completeness parameters δ_c.
Verifier Input: `PCond` access to a $(1+\alpha)$-flat distribution $\mathcal{D} \in \Delta([N])$.
Prover Input: Full description of $\mathcal{D}$.

Set $s_1 = \left\lceil \frac{\log(1/\delta_s)}{\log(A/A')} \right\rceil$, and $s_2 = \left\lceil \frac{\log(1/\delta_s)}{\log((1+\alpha)B'/B)} \right\rceil$

Prover sends a set $\widetilde{\mathrm{Supp}(\mathcal{D})}$ of size at most B (and at least A), which it claims is the support of $\mathcal{D}$. If the set size is outside this range, the verifier immediately outputs Reject.

Test 1: "Detect Small Support"

Sample $z_1, \ldots, z_{s_1} \overset{\text{i.i.d}}{\leftarrow\$} \widetilde{\text{Supp}}(\mathcal{D})$ uniformly, and $x \leftarrow\$ \mathcal{D}$. Check if $z_j \in \text{Supp}(\mathcal{D})$, for all $j \in [s_1]$, using the IsInSupport algorithm algorithm with error parameter $\frac{\delta_c}{s_1}$ and reference point x. Output Reject if *any* invocation outputs reject.

Test 2: "Detect Large Support"

Verifier samples $x_1, \ldots, x_{s_2} \overset{\text{i.i.d}}{\leftarrow\$} \mathcal{D}$ and outputs Reject Test 2 if any of the samples are not in $\widetilde{\text{Supp}}(\mathcal{D})$.

Final Output: The verifier outputs Accept if and only if it accepts both tests above.

Figure 6: Proof System For Support Size Range For Distributions With Small Support

Lemma 4.4: [Support Size Difference For Small Support Distributions] Fix parameters for completeness error $\delta_c \in (0,1)$ and soundness error $\delta_s \in (0,1)$. Let $N \in \mathbb{N}$ be the domain size and $\alpha \in [0,1]$. Fix integer parameters $A' < A < B < B'$. Given PCond access to an $(1+\alpha)$-flat distribution $\mathcal{D} \in \Delta([N])$, the an interactive proof system described in Figure 6 decides the following promise problem with communication complexity $O(B)$.

$$\Pi_{\text{Yes}} \overset{\text{def}}{=} \{\mathcal{D} \in \Delta([N]) : A \le |\text{Supp}(\mathcal{D})| \le B \wedge \mathcal{D} \text{ is } (1+\alpha)\text{-flat}\} \tag{6}$$

$$\Pi_{\text{No}} \overset{\text{def}}{=} \{\mathcal{D} \in \Delta([N]) : (|\text{Supp}(\mathcal{D})| \le A' \vee |\text{Supp}(\mathcal{D})| \ge B') \wedge \mathcal{D} \text{ is } (1+\alpha)\text{-flat}\} \tag{7}$$

The sample complexity is $O\left(\frac{\log(1/\delta_s)}{\log(B'/B)}\right)$ and the query complexity is $\widetilde{O}\left(\frac{\log(1/\delta_s)\log(1/\delta_c)}{\log(A/A')}\right)$.

Proof. The communication complexity is immediate from the cost of sending the purported support $\widetilde{\text{Supp}}(\mathcal{D})$ (of size at most B). The sample and query complexities are, respectively, $O(s_2)$ and $O(s_1 \log \frac{s_1}{\delta_c})$ (from the s_1 calls to the IsInSupport algorithm with error parameter δ_c/s_1); and the claimed bounds follow from our choice of s_1, s_2 and (for the latter) the query complexity of Lemma 2.12. We now turn to correctness, arguing completeness and soundness separately.

Completeness: If the prover follows the protocol honestly, then Test 2 always passes. The probability that Test 1 fails is equal to the probability that the IsInSupport algorithm fails, which, due to our assumption of near flatness and our choice of s_1, is at most δ_c by Lemma 2.12, along with a union bound over s_1 tests.

Soundness: Suppose now that $|\text{Supp}(\mathcal{D})| \notin (A', B')$. We distinguish two cases, showing that **Test 1** will reject with probability at least $1 - \delta_s$ if $|\text{Supp}(\mathcal{D})| \le A'$, and **Test 2** will reject with probability at least $1 - \delta_s$ if $|\text{Supp}(\mathcal{D})| \ge B'$.

- Suppose that $|\text{Supp}(\mathcal{D})| \le A'$. We then have that, for each $1 \le i \le s_1$, $\Pr[z_i \in \text{Supp}(\mathcal{D})] \le \frac{A'}{A}$. The tester accepts Test 1 if *every* $z_i \in \widetilde{\text{Supp}}(\mathcal{D})$ is found to be the support of $\mathcal{D}$. Thus, with

s_1 independent samples,

$$\Pr[\mathsf{T} \text{ outputs } \mathsf{Accept}] = \Pr[\forall i,\ (z_i \in \mathrm{Supp}(\mathcal{D}) \wedge \texttt{IsInSupport algorithm returns Yes})]$$
$$\leq \Pr[\forall i,\ z_i \in \mathrm{Supp}(\mathcal{D})] \leq \left(\frac{A'}{A}\right)^{s_1},$$

Which is at most δ_s by our choice of $s_1 \geq \frac{\log(1/\delta_s)}{\log(A/A')}$.

- Suppose now that $|\mathrm{Supp}(\mathcal{D})| \geq B'$. The tester accepts Test 2 if every $x_1, \ldots, x_{s_2}$ is in $\widetilde{\mathrm{Supp}(\mathcal{D})}$. Here we are checking if that cheating prover lied by claiming that $\left|\widetilde{\mathrm{Supp}(\mathcal{D})}\right|$ is much smaller than the true support size. Intuitively, the best chance for the prover to get away with this is to use $\left|\widetilde{\mathrm{Supp}(\mathcal{D})}\right| = B$, when $|\mathrm{Supp}(\mathcal{D})| = B'$ (as this is the smallest possible discrepancy between the cheating prover, and the correct answer), and then put the heaviest items of $\mathcal{D}$ in $\widetilde{\mathrm{Supp}(\mathcal{D})}$. This way, when the honest verifier samples $x \leftarrow\!\!\$\ \mathcal{D}$, they are more likely to find that x is in $\widetilde{\mathrm{Supp}(\mathcal{D})}$. The winning probability of this optimal cheating prover is maximised when the items in $|\widetilde{\mathrm{Supp}(\mathcal{D})}|$ is heaviest. However, as the distribution is $(1+\alpha)$-flat, there is no such thing as an outright heavy or light element. This means that the provers strategy of declaring only B elements as support is bound to miss out on items that are still highly likely to show up as samples.

  More formally, observe first that since $\mathcal{D}$ is $(1+\alpha)$-flat, we have, for every $x \in \mathrm{Supp}(\mathcal{D})$,[11]

$$\frac{1}{(1+\alpha)\,|\mathrm{Supp}(\mathcal{D})|} \leq \mathcal{D}[x] \leq \frac{1+\alpha}{|\mathrm{Supp}(\mathcal{D})|}$$

  The probability that $x \leftarrow\!\!\$\ \mathcal{D}$ belongs to $\widetilde{\mathrm{Supp}(\mathcal{D})}$ is then

$$\begin{aligned}
\Pr_{x \leftarrow\!\!\$\, \mathcal{D}}\left[x \in \widetilde{\mathrm{Supp}(\mathcal{D})}\right] = \mathcal{D}(\widetilde{\mathrm{Supp}(\mathcal{D})}) &= \sum_{y \in \widetilde{\mathrm{Supp}(\mathcal{D})} \cap \mathrm{Supp}(\mathcal{D})} \mathcal{D}[y] \\
&\leq \left|\widetilde{\mathrm{Supp}(\mathcal{D})} \cap \mathrm{Supp}(\mathcal{D})\right| \cdot \frac{1+\alpha}{|\mathrm{Supp}(\mathcal{D})|} \\
&\leq (1+\alpha)\frac{\left|\widetilde{\mathrm{Supp}(\mathcal{D})}\right|}{|\mathrm{Supp}(\mathcal{D})|} \\
&\leq (1+\alpha)\frac{B}{B'},
\end{aligned}$$

  and so the probability that all s_2 draws $x_1 \ldots, x_{s_2} \overset{\text{i.i.d}}{\leftarrow\!\!\$}\ \mathcal{D}$ fall in $\widetilde{\mathrm{Supp}(\mathcal{D})}$ is at most

$$\begin{aligned}
\Pr[\mathsf{T} \text{ outputs } \mathsf{Accept}] &= \Pr_{x_1,\ldots,x_{s_2} \leftarrow\!\!\$\, \mathcal{D}}\left[\forall i \in [s_2],\ x_i \in \widetilde{\mathrm{Supp}(\mathcal{D})}\right] \\
&\leq \left((1+\alpha)\frac{B}{B'}\right)^{s_2} \leq \delta_s,
\end{aligned}$$

  the last inequality from our choice of $s_2 \geq \frac{\log(1/\delta_s)}{\log((1+\alpha)B'/B)}$.

This concludes the proof. □

[11] Indeed, $1 = \sum_{x \in \mathrm{Supp}(\mathcal{D})} \mathcal{D}[x] \leq |\mathrm{Supp}(\mathcal{D})| \cdot \max_{x \in \mathrm{Supp}(\mathcal{D})} \mathcal{D}[x] \leq |\mathrm{Supp}(\mathcal{D})| \cdot (1+\alpha) \min_{x \in \mathrm{Supp}(\mathcal{D})} \mathcal{D}[x]$, and similarly $1 \geq |\mathrm{Supp}(\mathcal{D})| \cdot \frac{\max_{x \in \mathrm{Supp}(\mathcal{D})}}{1+\alpha} \mathcal{D}[x]$, from which

$$\frac{1}{(1+\alpha)\,|\mathrm{Supp}(\mathcal{D})|} \leq \min_{x \in \mathrm{Supp}(\mathcal{D})} \mathcal{D}[x] \leq \max_{x \in \mathrm{Supp}(\mathcal{D})} \mathcal{D}[x] \leq \frac{1+\alpha}{|\mathrm{Supp}(\mathcal{D})|}.$$

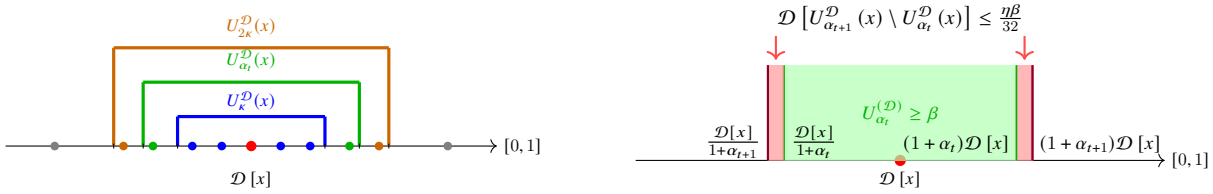
4.3 Learning The Neighbourhood Of A "Relevant" Domain Element

Definition 4.5: Given a parameter $\kappa \in (0,1]$ and point $y \in [N]$, let the *κ-neighbourhood of x under $\mathcal{D}$* be defined as

$$U_\kappa^{\mathcal{D}}(x) = \left\{ y \in [N] : \frac{1}{1+\kappa}\mathcal{D}[x] \leq \mathcal{D}[y] \leq (1+\kappa)\mathcal{D}[x] \right\}$$

that is, the set of points whose probability is within an $1+\kappa$ factor of the probability of x.

Later in the main protocol, the prover will claim that certain bucket indices of the (N,τ)-histogram of $\mathcal{D}$ have large mass. If the prover is honest, then buckets with large mass must also contain domain elements with heavy neighbourhoods (see claim 5.2), where the definition of neighbourhood is given in Definition 4.5. In an ideal world, we would like to verify this exact claim that the a domain element has a heavy κ-neighbourhood. However, due to boundary conditions and the inherent randomness of sampling, we are only able to verify a fuzzy version of the above claim. More specifically, using a result by Canonne et al. [2015, Lemma 3] we show that, if the prover claims that the κ-neighbourhood (shown in blue in Figure 7a) of x is heavy, we estimate the $\alpha_t \in [\kappa, 2\kappa]$ neighbourhood (shown in green in Figure 7a) of x, up to a multiplicative factor of η. This fuzziness is unavoidable, but we later show that this fuzzy approximation suffices to catch a cheating prover. A distinctive feature of the α_t-neighbourhood above is that apart from being heavy, it also has what we refer to as a "moat" (as shown in Figure 7b)- by which we mean that most of the neighbourhood mass is concentrated away from the boundary. This moat proves to be useful, when using the Compare Algorithm to check whether a sample $z \leftarrow\!\$\ \mathcal{D}$ is in $U_{\alpha_t}^{\mathcal{D}}(x)$ or not. The idea is that the `Compare` algorithm inherently has some randomness we cannot avoid, and thus a sample that is actually in the moat but not in $U_{\alpha_t}^{\mathcal{D}}(x)$ might get incorrectly misclassified as being in the α_t-neighbourhood. When the probability mass of the moat is small, the chances of sampling an element in the moat are slim, and therefore we are unlikely to err. See Sampling From Neighbourhoods With Moats Lemma for details.



(a) A fuzzy approximation of the the provers claim.

(b) A neighbourhood with a moat

Figure 7: The prover claims that the mass of the κ-neighbourhood of x is at least as large as β. The Estimate Neighbourhood algorithm of Canonne et al. [2015] allows us to estimate α_t neighbourhood of x up to a multiplicative factor of η, where $\alpha_t \in [\kappa, 2\kappa]$. Furthermore, α_t is such that while $\mathcal{D}\left[U_{\alpha_t}^{\mathcal{D}}(x)\right] > \beta$, there is a "moat" around the $U_{\alpha_t}^{\mathcal{D}}(x)$ such that $\mathcal{D}\left[U_{\alpha_{t+1}}^{\mathcal{D}}(x) \setminus U_{\alpha_t}^{\mathcal{D}}(x)\right] \leq \frac{\eta\beta}{32}$. For any sample $z \leftarrow\!\$\ \mathcal{D}$, if we wish to use the Compare Algorithm to check whether $z \in U_{\alpha_t}^{\mathcal{D}}(x)$, this "moat" will act as insurance against the randomness of the `PCond` oracle, and therefore ensure our check is accurate with high probability.

Lemma 4.6: [The `EstimateNeighborhood` procedure of [Canonne et al., 2015, Lemma 3]] Given `PCond` access to $\mathcal{D} \in \Delta([N])$, there exists an algorithm `EstimateNeighborhood` which, on input $\kappa, \beta, \eta, \delta \in (0, 1/2])$ as well as a domain element $x \in [N]$, has the following guarantees. Letting $T = 128/(\eta\beta\delta)$, and defining, for $t \in [T+1]$,

$$\alpha_t = \kappa + (t-1)\kappa/T \in [\kappa, 2\kappa)\,,$$

the algorithm outputs a pair $(\hat{w}, t) \in [0,1] \times [T]$, such that t is uniformly distributed in $[T]$, and:

1. Completeness: if $\mathcal{D}[U^{\mathcal{D}}_{\alpha_t}(x)] \geq \beta$, then

$$\frac{\hat{w}}{\mathcal{D}[U^{\mathcal{D}}_{\alpha_t}(x)]} \in \left[\frac{1}{1+\eta}, 1+\eta\right] \quad \text{and} \quad \mathcal{D}\left[U^{\mathcal{D}}_{\alpha_{t+1}}(x) \setminus U^{\mathcal{D}}_{\alpha_t}(x)\right] \leq \frac{\eta\beta}{32} \tag{8}$$

   simultaneously with probability at least $1-\delta$.

2. Soundness: if $\mathcal{D}[U^{\mathcal{D}}_{\alpha_t}(x)] < \beta$, then

$$\hat{w} \leq (1+\eta)\beta \quad \text{and} \quad \mathcal{D}\left[U^{\mathcal{D}}_{\alpha_{t+1}}(x) \setminus U^{\mathcal{D}}_{\alpha_t}(x)\right] \leq \frac{\eta\beta}{32} \tag{9}$$

   simultaneously with probability at least $1-\delta$.

The query complexity of the algorithm is $\widetilde{O}\left(\frac{1}{\kappa^2\eta^4\beta^3\delta^2}\right)$.

Proof. This is a consequence of the guarantees from [Canonne et al., 2015, Lemma 3] (slightly simplified, and replacing η by $\eta/2$ to have $[1/(1+\eta), 1+\eta]$ instead of $[1-\eta, 1+\eta]$ in the completeness guarantee). □

Remark 3 Note that, in the above statement, we could weaken the conditions of completeness and soundness to $\mathcal{D}[U^{\mathcal{D}}_{\kappa}(x)] \geq \beta$ and $\mathcal{D}[U^{\mathcal{D}}_{2\kappa}(x)] < \beta$, respectively, as $\mathcal{D}[U^{\mathcal{D}}_{\kappa}(x)] \leq \mathcal{D}[U^{\mathcal{D}}_{\alpha_t}(x)] \leq \mathcal{D}[U^{\mathcal{D}}_{2\kappa}(x)]$ for all $t \in [T]$.

Sampling from the neighbourhood. Foreshadowing ahead, as a sub-routine of our main protocol, we will invoke the support size protocol on the re-normalised distribution $\mathcal{D}'$, which is obtained by restricting the domain of $\mathcal{D}$ to the set $U^{\mathcal{D}}_{\alpha_t}(x)$, and re-normalising $\mathcal{D}\left[U^{\mathcal{D}}_{\alpha_t}(x)\right]$. In order to run the support size protocol as prescribed, it is important that the verifier be able to sample from $\mathcal{D}'$. We now explain how after running `EstimateNeighbourhood` to obtain a tuple $(\hat{w}, t)$ corresponding neighbourhood $U_t \stackrel{\text{def}}{=} U^{\mathcal{D}}_{\alpha_t}(x)$ and an estimate $\hat{w}$ of its probability mass, we can obtain *samples* from $\mathcal{D}'$.

Lemma 4.7: [Sampling From Neighbourhoods With Moats] Given parameters $\kappa, \beta, \delta, \eta \in (0, 1/2)$ and domain element $x \in [N]$. Setting $T = 128/(\eta\beta\delta)$ and let

$$(\hat{w}, t) \leftarrow\!\$\ \texttt{EstimateNeighbourhood}(x; \kappa, \beta, \eta, \delta) \quad \text{and} \quad \alpha_t = \kappa + (t-1)\kappa/T \in [\kappa, 2\kappa).$$

If $\mathcal{D}[U^{\mathcal{D}}_{\alpha_t}(x)] \geq \beta$ and $\mathcal{D}[U^{\mathcal{D}}_{\alpha_{t+1}}(x) \setminus U^{\mathcal{D}}_{\alpha_t}(x)] \leq \frac{\eta\beta}{32}$ [a], then there exists a sampling algorithm with sample complexity $O\left(\frac{\log(1/\eta)}{\beta}\right)$ and query complexity $\widetilde{O}\left(\frac{1}{\beta^3\eta^2\delta^2\kappa^2}\right)$ that returns:

- $y \leftarrow\!\$\ \mathcal{D}'$ with probability at least $1 - \eta\log(1/\eta)$, where $\mathcal{D}'[y] \stackrel{\texttt{def}}{=} \frac{\mathcal{D}[y]}{\sum_{y' \in U^{\mathcal{D}}_{\alpha_t}(x)} \mathcal{D}[y']}$, or
- $\perp$ otherwise.

[a] These are the guarantees obtained with high probability when $(\hat{w}, t)$ is returned by `EstimateNeighbourhood` invoked on x with parameters $\kappa, \beta, \eta, \delta$.

Proof. Suppose that $(\hat{w}, t) \in [0, 1] \times [T]$ is such that $\mathcal{D}[U^{\mathcal{D}}_{\alpha_t}(x)] \geq \beta$ and $\mathcal{D}[U^{\mathcal{D}}_{\alpha_{t+1}}(x) \setminus U^{\mathcal{D}}_{\alpha_t}(x)] \leq \frac{\eta\beta}{32}$ both hold. That is, the neighbourhood is large enough (mass at least β), and it is surrounded by a "moat" (of probability mass at most $O(\eta\beta)$). The algorithm proceeds as follows, where we set $\delta' = c \cdot \eta\log(1/\eta)$ for a suitably small constant $c > 0$.

- Sample $s = \left\lceil \frac{\log(3/\delta')}{\log \frac{1}{1-\beta}} \right\rceil = O(\log(1/\delta')/\beta)$ points $x_1, \ldots, x_s \stackrel{\text{i.i.d}}{\leftarrow\!\$}\ \mathcal{D}$. Note that (1) with probability at least $1 - \frac{\delta'}{3}$ at least one x_i will be in $U_t \stackrel{\texttt{def}}{=} U^{\mathcal{D}}_{\alpha_t}(x)$, and (2) (by a union bound) with probability at least $1 - \frac{s\eta\beta}{32} = 1 - O(\eta\log(1/\delta'))$, no x_i will be in the "moat" $U^{\|}_t \stackrel{\texttt{def}}{=} U^{\mathcal{D}}_{\alpha_{t+1}}(x) \setminus U^{\mathcal{D}}_{\alpha_t}(x)$.
- For each x_i, `Compare` on x, x_i with failure probability $\frac{\delta'}{3s}$, maximum ratio bound $K = 2$, and accuracy parameter $\gamma = \Theta\left(\frac{\kappa}{T}\right)$ to obtain an estimate of $\frac{\mathcal{D}[x_i]}{\mathcal{D}[x]}$ accurate to within an additive γ (or detect that the ratio is not even in $[1/2, 2]$). By a union bound over all s calls to `Compare`, with probability at least $1 - \frac{\delta'}{3}$ all estimates are indeed this accurate. By our choice of γ, this means that we can then decide which of the x_i's are in U_t, and only (possibly) make a mistake on those which are in the moat $U^{\|}_t$. (This has query complexity $s \cdot O\left(\frac{\log(s/\delta')}{\gamma^2}\right) = O\left(\frac{\log(1/\delta')\cdot\log(s/\delta')}{\beta\cdot(\kappa/T)^2}\right) = \widetilde{O}\left(\frac{1}{\beta^3\eta^2\delta^2\kappa^2}\right)$.)
- Return any element x_i detected to be in U_t uniformly at random, and denote it with y. Return $\perp$ if there is none.

By a union bound, except with probability at most

$$\frac{\delta'}{3} + \frac{\delta'}{3} + \frac{s\eta\beta}{32} = \frac{2}{3}\delta' + O\left(\eta\log\frac{1}{\delta'}\right) \leq \delta'$$

(the last inequality for $\frac{s\eta\beta}{32} \leq \frac{\delta'}{3}$, which is satisfied as $\delta' = \Omega(\eta\log(1/\eta))$), then we have simultaneously (1) at least one x_i in U_t, (2) none in the moat $U^{\|}_t$, and (3) a correct decision on all elements (since the only errors would be for elements in $U^{\|}$, and there aren't any by (2)). That is, with probability at least $1 - \delta'$, the above procedure returns an element y sampled from $\mathcal{D}'$. □

5 Interactive Proofs For Label Invariant Properties

Given parameter τ', in this section, we describe our main result, an interactive proof system for certifying the τ-approximate histogram, where $\tau \stackrel{\text{def}}{=} 0.01\tau'$. Note that the histogram is constructed at a finer granularity than the proximity τ' parameter warrants, as $\tau < \tau'$, but this only makes testing for label invariant properties with proximity parameter τ' easier.

The Approximate Single Algorithm

Verifier Input:

- Domain $[N]$, soundness and completeness parameters $\delta' \in (0,1)$, proximity parameter $\tau \in (0, 0.1)$ and `PCond` access to $\mathcal{D}$
- Prover claims: $\widetilde{\text{Bucket}_\tau(y^\star)} = j^\star$ and $\widetilde{\mathcal{D}}\left[U^{\mathcal{D}}_{\tau/3}(y^\star)\right] \geq \frac{\tau}{30L_\tau}$

Verifier Output: Either Reject or Accept of above claim.

Parameters For Invoking Sub-routines

- Set $\delta_c = \delta_s = \delta'/2$.
- Set $\kappa = \tau/3$, $\beta = \frac{\tau}{30L_\tau}$.
- Set $\eta = \min\left(\frac{\tau}{10}, \frac{\log s_1}{100 s_1 \delta'}, \frac{\log s_2}{100 s_2 \delta'}\right) = \widetilde{\Omega}\left(\tau^4/\delta'\right)$ where s_1, s_2 are sample complexities from Figures 6 and 3 with $B'/B = A/A' = 1+\tau$.

Algorithm Body

Step 1: Estimate Neighborhood

Run $(\hat{w}, t) \leftarrow \texttt{EstimateNeighbourhood}(y^\star; \kappa, \eta, \beta, \delta'/2)$
If $\hat{w} \leq (1+\eta)\beta$ **then return** Reject.
Otherwise, set $T = 128/(\eta\beta\delta)$ and $\alpha \stackrel{\text{def}}{=} \kappa + (t-1)\kappa/T \in [\kappa, 2\kappa)$.

Step 2: Compute Bounds

Set:

$$A' = \frac{\hat{w}N}{\tau(1+\tau)^{j^\star+1}(1+\eta)(1+\alpha)}, \quad A = \frac{\hat{w}N}{\tau(1+\tau)^{j^\star}(1+\eta)(1+\alpha)}$$

$$B = \frac{\hat{w}(1+\eta)(1+\alpha)N}{\tau(1+\tau)^{j^\star-1}}, \quad B' = \frac{\hat{w}(1+\eta)(1+\alpha)N}{\tau(1+\tau)^{j^\star-2}}$$

Define $\mathcal{D}' \in \Delta\left(U^{\mathcal{D}}_{\alpha}(y^\star)\right)$ where $\mathcal{D}'[x] = \frac{\mathcal{D}[x]}{Z}$ with $Z = \sum_{x' \in U^{\mathcal{D}}_{\alpha}(y^\star)} \mathcal{D}[x']$.

Step 3: Support Size Interactive Protocols

If $B \in O\left(\sqrt{N}\right)$ **then**
Run Figure 6 with parameters A', A, B, B', $\delta_c = \delta_s = \delta'/2$.

Else if $A \in \omega\left(\sqrt{N}\right)$ **then**

Run Figure 3 with parameters A', A, B, B', $\delta_c = \delta_s = \delta'/2$.

In both cases sample from $(1+\alpha)$-flat distribution $\mathcal{D}'$ using Lemma 4.7 with $\kappa = \tau/3$.
If either algorithm outputs Reject **then return** Reject.

Return $\widetilde{\mathcal{D}}[y^\star] = \frac{\tau(1+\tau)^{j^\star}}{N}$ and Accept.

Figure 8: Approximating the probability mass of a single domain item.

Recall, for any distribution $\mathcal{D} \in \Delta([N])$ and domain element $z \in [N]$, we use $\text{Bucket}_\tau(z) \in [L_\tau]$ to denote the bucket index of the (N,τ)-histogram of $\mathcal{D}$ that z belongs to.

Theorem 5.1: [Approximate Single] Fix $\tau > 0$ and $\mathcal{D} \in \Delta([N])$ for some large enough $N \in \mathbb{N}$. Let L_τ denote the number of buckets of the (N,τ)-histogram of $\mathcal{D}$, The protocol starts with the provers claim that for domain element $y^\star \in [N]$, its bucket index is $\widetilde{\text{Bucket}_\tau}(y^\star) = j^\star$, and additionally that $y^\star$ has a heavy $\tau/3$-neighbourhood i.e. $\widetilde{\mathcal{D}}\left[U^{\mathcal{D}}_{\tau/3}(y^\star)\right] \geq \frac{\tau}{30L_\tau}$. For every $\delta' \in (0, 1/2)$, the proof system described by Figure 8, causes the verifier to output Accept or Reject based on the following properties:

1. **Completeness:** If the prover is honest, then

$$\left(\widetilde{\text{Bucket}_\tau}(y^\star) = \text{Bucket}_\tau(y^\star)\right) \wedge \left(\mathcal{D}\left[U^{\mathcal{D}}_{\tau/3}(y^\star)\right] = \widetilde{\mathcal{D}}\left[U^{\mathcal{D}}_{\tau/3}(y^\star)\right] \geq \frac{\tau}{30L_\tau}\right)$$

   As a result $\Pr[\mathsf{T} \text{ outputs Reject}] \leq \delta'$

2. **Soundness:** For any cheating prover strategy if

$$\left(\left|\widetilde{\text{Bucket}_\tau}(y^\star) - \text{Bucket}_\tau(y^\star)\right| \geq 2\right) \vee \left(\mathcal{D}\left[U^{\mathcal{D}}_{2\tau/3}(y^\star)\right] \leq \frac{\tau}{30L_\tau}\right)$$

   thenPr$[\mathsf{T} \text{ outputs Accept}] \leq \delta'$

If the verifier outputs Accept, then it accepts that $\widetilde{\mathcal{D}}[y^\star] = \frac{\tau(1+\tau)^{j^\star}}{N}$ as the approximate probability mass of $y^\star$ under $\mathcal{D}$.

Proof. Set $T = 128/(\eta\beta\delta)$. Re-stating the parameters defined in Figure 8, we have $\delta_c = \delta_s = \delta'/2$, $\kappa = \tau/3$, $\beta = \frac{\tau}{30L_\tau}$, and $\eta = \min\left(\frac{\tau}{10}, \frac{\log s_1}{100 s_1 \delta'}, \frac{\log s_2}{100 s_2 \delta'}\right) = \widetilde{\Omega}\left(\tau^4/\delta'\right)$ where s_1, s_2 are sample complexities from Figures 6 and 3 with $B'/B = A/A' = 1+\tau$. Let

$$(\hat{w}, t) \leftarrow\!\$\ \texttt{EstimateNeighborhood}(y^\star; \tau/3, \eta, \beta, \delta'/2)$$

and $\alpha \stackrel{\text{def}}{=} \kappa + (t-1)\kappa/T \in [\tau/3, 2\tau/3]$.

Completeness: As the prover is honest, we have $j^\star = \text{Bucket}_\tau(y^\star)$, and therefore,

$$\frac{\widetilde{\mathcal{D}}[y^\star]}{\mathcal{D}[y^\star]} \in \left[\frac{1}{1+\tau}, 1+\tau\right] \tag{10}$$

where $\widetilde{\mathcal{D}}[y^\star] = \frac{\tau(1+\tau)^{j^\star}}{N}$. Additionally, we also have $\mathcal{D}\left[U_\alpha^{\mathcal{D}}(y^\star)\right] \geq \mathcal{D}\left[U_{\tau/3}^{\mathcal{D}}(y^\star)\right] \geq \beta \stackrel{\text{def}}{=} \frac{\tau}{30L_\tau}$.

As we are upper bounding the completeness error, we assume that any time $\hat{w}$ is *not* a good approximation of $\mathcal{D}\left[U_\alpha^{\mathcal{D}}(y^\star)\right]$, the worst happens, the verifier rejects the honest provers claim. $\hat{w}$ is a bad approximation with probability at most $\delta'/2$. Finally, assuming that the verifier does not reject the honest provers claim, then this restricts the size of $U_\alpha^{\mathcal{D}}(y^\star)$ More specifically, from the provers claim $j^\star$ is the bucket of $y^\star$, we get an upper and lower bound on $\left|U_\alpha^{\mathcal{D}}(y^\star)\right|$. More specifically

$$\left|U_\alpha^{\mathcal{D}}(y^\star)\right| \leq \frac{\mathcal{D}\left[U_\alpha^{\mathcal{D}}(y^\star)\right]}{\min_{z\in U_\alpha^{\mathcal{D}}(y^\star)}\mathcal{D}[z]} \tag{11}$$

$$\leq \frac{\mathcal{D}\left[U_\alpha^{\mathcal{D}}(y^\star)\right](1+\alpha)}{\mathcal{D}[y^\star]} \tag{12}$$

$$\leq \frac{\mathcal{D}\left[U_\alpha^{\mathcal{D}}(y^\star)\right](1+\alpha)N}{\tau(1+\tau)^{j^\star-1}} \tag{13}$$

$$\leq \frac{\hat{w}(1+\eta)(1+\alpha)N}{\tau(1+\tau)^{j^\star-1}} \tag{14}$$

where Equation (12) comes from the definition of a α-neighbourhood,(13) comes from the limits of bucket index $j^\star$, and (14) comes from the approximation guarantees of the The `EstimateNeighborhood` procedure of [Canonne et al., 2015, Lemma 3]. A lower bound on the size of $U_\alpha^{\mathcal{D}}(y^\star)$ can similarly defined, giving us

$$\left|U_\alpha^{\mathcal{D}}(y^\star)\right| \leq \frac{(1+\alpha)N\ \mathcal{D}\left[U_\alpha^{\mathcal{D}}(y^\star)\right]}{\tau(1+\tau)^{j^\star-1}} \leq \frac{N\ \hat{w}(1+\eta)(1+\alpha)}{\tau(1+\tau)^{j^\star-1}} \stackrel{\text{def}}{=} B \tag{15}$$

$$\left|U_\alpha^{\mathcal{D}}(y^\star)\right| \geq \frac{N\ \mathcal{D}\left[U_\alpha^{\mathcal{D}}(y^\star)\right]}{(1+\alpha)\tau(1+\tau)^{j^\star}} \geq \frac{\hat{w}N}{\tau(1+\tau)^{j^\star}(1+\eta)(1+\alpha)} \stackrel{\text{def}}{=} A \tag{16}$$

Define $\mathcal{D}' \in \Delta\left(U_\alpha^{\mathcal{D}}(y^\star)\right)$ such that for all $x \in U_\alpha^{\mathcal{D}}(y^\star)$, $\mathcal{D}'[x] = \frac{\mathcal{D}[x]}{Z}$, where $Z \stackrel{\text{def}}{=} \sum_{x'\in U_\alpha^{\mathcal{D}}(y^\star)}\mathcal{D}[x']$. That is $\mathcal{D}'$ is $\mathcal{D}$ restricted and re-normalised on $U_\alpha^{\mathcal{D}}(y^\star)$. As the verifier has not yet rejected, from the guarantees of the Lemma 4.6 we are in the regime where $\mathcal{D}\left[U_\alpha^{\mathcal{D}}(y^\star)\right] \geq \beta$ and the mass of the "moat" is at most $\frac{\eta\beta}{32}$. This implies that we can invoke the Sampling From Neighbourhoods With Moats to sample from $\mathcal{D}'$. More specifically, to obtain one sample $z \hookleftarrow \mathcal{D}'$ we need to draw $\widetilde{O}\left(\frac{\log 1/\eta}{\beta}\right)$ samples from $\mathcal{D}$, and make $\widetilde{O}\left(\frac{1}{\beta^3\eta^2\delta'^2\kappa^2}\right)$ `PCond` queries, and with probability $1-\eta\log(1/\eta)$ we succeed in obtaining this sample. We want to to invoke the support size protocol with A, A' and B, B' as defined in Figure 8. In total we need to draw $O\left((1/\tau)^4\log\frac{1}{\delta'}\right)$ samples from $\mathcal{D}'$.

The probability that the support size protocol rejects is at most $\delta'/2$. Thus, the total completeness error for this protocol is δ'.

Soundness: Soundness follows pretty similarly. Assume that the prover lies about by more than 1 bucket, then

$$\frac{\widetilde{\mathcal{D}}[y^\star]}{\mathcal{D}[y^\star]} \notin \left[\frac{1}{(1+\tau)^2}, (1+\tau)^2\right] \tag{17}$$

Let β be as defined in Figure 8. If the prover *also* lied about the mass of the neighbourhood around $y^\star$, and $\mathcal{D}\left[U_{2\tau/3}^{\mathcal{D}}(y^\star)\right] \leq \beta$, then the probability that the prover does not output Reject is at most $\delta'/2$. Assuming, $\mathcal{D}\left[U_{2\tau/3}^{\mathcal{D}}(y^\star)\right] \geq \beta$, the the probability that $\hat{w}$ is not a good approximation

of $\mathcal{D}\left[U_{\alpha}^{\mathcal{D}}(y^{\star})\right]$ is at most $\delta/'2$. By the worst case assumption, the verifier accepts the provers claim with probability at most $\delta'/2$. Assuming that $\hat{w}$ is a good approximation, and $\mathcal{D}\left[U_{2\tau/3}^{\mathcal{D}}(y^{\star})\right] \geq \beta$, once again, by a similar analysis, this implies that $\left|U_{\alpha}^{\mathcal{D}}(y^{\star})\right| \geq B'$ or $\left|U_{\alpha}^{\mathcal{D}}(y^{\star})\right| \leq A'$, where

$$B' = \frac{\hat{w}(1+\eta)(1+\alpha)N}{\tau(1+\tau)^{j^{\star}-2}} \tag{18}$$

$$A' = \frac{\hat{w}N}{\tau(1+\tau)^{j^{\star}+1}(1+\eta)(1+\alpha)} \tag{19}$$

$\mathcal{D}'$ is $(1+\alpha)$-flat and sample-able, so we get from soundness of the support size protocol, the prover accepts with probability at most $\delta'/2$.

Complexity: For `EstimateNeighbourhood` the verifier makes $\widetilde{O}\left(\frac{L_{\tau}^{3}}{\tau^{5}\eta^{4}\delta'^{2}}\right) = \widetilde{O}\left(\frac{(\log N)^{3}}{\tau^{24}\delta'^{2}}\right)$ queries and samples. For `SupportSize` we draw at worst $O\left((1/\tau)^{4}\log\frac{1}{\delta'}\right)$ samples (from the distribution restricted to the neighborhood), and make at worst $\widetilde{O}\left((1/\tau)^{4}\log\frac{1}{\delta'}\right)$ `PCond` queries. Substituting in the sample complexity of generating a single sample from the neighborhood (Lemma 4.7) This corresponds to $\widetilde{O}\left(L_{\tau}\cdot(1/\tau)^{5}\log\frac{1}{\delta'}\right) = \widetilde{O}\left(\log N\cdot(1/\tau)^{6}\log\frac{1}{\delta'}\right)$ samples from the true distribution, and $\widetilde{O}\left(L_{\tau}^{3}\cdot(1/\tau)^{21}\cdot\frac{1}{\delta'^{2}}\right) = \widetilde{O}\left((\log N)^{3}\cdot(1/\tau)^{24}\cdot\frac{1}{\delta'^{2}}\right)$ `PCond` queries. The communication complexity is at worst $\widetilde{O}\left(\sqrt{N}\,\mathrm{poly}(1/\tau)\right)$. The round complexity is $O\left((1/\tau)^{4}\cdot\log\frac{1}{\delta'}\right)$.

□

Remark 4 In the case that the prover claims $\widetilde{\mathrm{Bucket}_{\tau}}(y^{\star}) \leq 1$ or $\widetilde{\mathrm{Bucket}_{\tau}}(y^{\star}) \geq L_{\tau}-1$, then we are saved from doing 2 sided tests, and can just check one side for support size decision problem. Of course if a bucket index large enough, and has large enough mass, then the verifier could simply find all the points in the bucket by mere sampling. We do not explicitly spell out this easy case, as its covered by the protocol above.

5.1 Succinct Proof Systems For Any Label Invariant Property

Claim 5.2: Let B be a bucket of the (N,τ)-histogram of a distribution $\mathcal{D}$ with probability mass p. Then, there exists a point $y^{\star} \in B$ such that $\mathcal{D}\left[U_{\tau/3}^{\mathcal{D}}(y^{\star})\right] \geq p/3$, where the neighborhood $U_{\tau/3}^{\mathcal{D}}\left(y_{j}^{\star}\right)$ is as defined in Definition 4.5.

Proof. For any $x \in B$, define $t(x) \stackrel{\text{def}}{=} -\log \mathcal{D}[x]$. From the definition of a (N,τ)-histogram we have that $\Delta \stackrel{\text{def}}{=} \max_{x\in B} t(x) - \min_{x\in B} t(x) \leq \log(1+\tau)$. Define $\delta = \log(1+\tau/3)$. Let $S = x_1,\ldots,x_{|B|}$ denote the ordered set of the points in B in ascending order of their probability mass (ties broken arbitrarily). Now we greedily partition S into disjoint clusters of mass at most 2δ. As the total mass of S is p, the maximum number of clusters in S, denoted by M is at most $\lceil\frac{\Delta}{2\delta}\rceil+1$. By the averaging argument, there must be one cluster with mass at least $\frac{p}{M}$. Let $y^{\star}$ denote the median point in this cluster. This implies that $\mathcal{D}\left[U_{\alpha}^{\mathcal{D}}(y^{\star})\right] \geq \frac{p}{M}$ (as the total mass of the cluster is 2δ and $y^{\star}$ is the median). To get the desired lower bound, we upper bound $M \leq \lceil\frac{\Delta}{2\delta}\rceil+1 \leq 2$, where the last inequality comes from the fact that $\Delta = \log(1+\tau) \leq 3\log(1+\tau/3) = 3\delta$, giving us $M \leq 3$.

□

Proof System For Verifiable Succint Histograms

Common Inputs: $\tau' \in (0, 0.1)$. Verifier has `PCond` access to $\mathcal{D}$. Prover knows $\mathcal{D}$ in clear.

Define $\tau \stackrel{\text{def}}{=} 0.01\tau'$, $\gamma = \frac{\tau}{s(1+\tau)}$, $s \stackrel{\text{def}}{=} \frac{10L_\tau}{\tau}\log(10L_\tau)$.

1. **Verifier ← Prover**: For every "relevant" bucket, i.e bucket indices $j \in [L_\tau]$ such that $\mathcal{D}\left[B_j^{(\mathcal{D},\tau)}\right] \geq \frac{\tau}{10L_\tau}$, send the verifier $y_j \in B_j^{(\mathcal{D},\tau)}$ such that $\mathcal{D}\left[U^{\mathcal{D}}_{2\tau/3}(y_k)\right] \geq \frac{\tau}{30L_\tau}$, along with its bucket index $\widetilde{\text{Bucket}_\tau(y_j)}$. That is the prover is asked to declare all the relevant bucket indices j, and a point y_j in each relevant bucket with a heavy neighborhood. Prover claims that $\widetilde{K} \subseteq [L_\tau]$ are the relevant buckets, and sends $y_1, \ldots, y_{|\widetilde{K}|}$ as representative samples for each bucket.
2. **Verifier ↔ Prover**: For each $j \in \widetilde{K}$, Invoke the proof system specified by Approximate Single with input y_j, parameters $\tau/3$, and $\delta = \frac{1}{20|\widetilde{K}|}$, and output Reject if the proof system rejects *any* of the tests. Otherwise, accept the claim $\widetilde{\mathcal{D}}\left[y_j\right] = \frac{\tau(1+\tau)^j}{N}$ for all $j \in \widetilde{K}$.
3. **Verifier → Prover:** Send set $S = (x_1, \ldots, x_s) \stackrel{\text{i.i.d}}{\hookleftarrow} \mathcal{D}$.
4. **Verifier ← Prover** Send $\{\widetilde{\text{Bucket}_\tau(x_1)}, \ldots, \widetilde{\text{Bucket}_\tau(x_s)}\}$,
5. **Verifier Computation:** Compute for all $j \in [L_\tau]$. $\widetilde{p_j} \stackrel{\text{def}}{=} \frac{1}{|S|}\left|\left\{x \in S : \widetilde{\text{Bucket}_\tau(x)} = j\right\}\right|$
6. **Verifier Computation**: For each $x \in S$, let $y^\star(x) \in \{y_1, \ldots, y_{|\widetilde{K}|}\}$ be the coreresponding y_j such that $\widetilde{\text{Bucket}_\tau(y^\star(x))} = \widetilde{\text{Bucket}_\tau(x)} = j$. If no just $y^\star(x)$ exists for even one x, the verifier outputs Reject. The verifier also rejects if the prover declares multiple $y^\star$'s for a single x. Otherwise

for $x \in S$ **do**
  Compute $\alpha_{xy^\star(x)} \leftarrow \texttt{Compare}\left(\{x_i\}, \{y^\star(x_i)\}, \gamma, \frac{1}{20s}, K = 2\right)$

If `Compare` outputs HIGH or LOW, for any x, output Reject.

Define $\widetilde{\mathcal{D}}[x] \stackrel{\text{def}}{=} \frac{\tau(1+\tau)^{\widetilde{\text{Bucket}_\tau(x)}}}{N}$ for all $x \in S$

if $\sum_{x\in S}\left|\widetilde{\mathcal{D}}\left[y^\star\right]\alpha_{xy^\star(x)} - \widetilde{\mathcal{D}}[x]\right| \geq 2\tau' - (2\tau+\tau^2) - \frac{\tau}{1+\tau}$ **then**
  Output Reject and halt
else
  Output $\{\widetilde{p_j}\}_{j\in[L_\tau]}$ as the learned histogram

Figure 9: A General Proof System To Obtain Verifiable (N, τ)-Histograms for any distribution $\mathcal{D} \in \Delta([N])$

Theorem 5.3: [Public-Coin Proof System For Succinct Histograms] Fix $N \in \mathbb{N}$, $\tau' \in (0, 0.1)$ and $\mathcal{D} \in \Delta([N])$. Define $\tau \stackrel{\text{def}}{=} 0.1\tau'$. There exists an interactive protocol Π between an honest verifier T, and an omniscient untrusted prover $\mathsf{P}^{\mathcal{D}}$, described in Figure 9, where the verifier has `PCond` access to $\mathcal{D}$, such that at the end of the interaction the verifier either outputs a (N, τ) histogram $\{\widetilde{p_j}\}_{j \in [L_\tau]}$ or outputs Reject, with the following constraints:

1. **Completeness:** If the prover follows the protocol as prescribed, then

$$\Pr\left[\text{out}\left[\Pi\left(\mathsf{P}^{\mathcal{D}}, \mathsf{T}^{(\mathcal{D})}; \tau', N\right)\right] = \mathsf{Accept} \wedge \mathtt{RL}\left(\mathcal{D}, \{\widetilde{p_j}\}_{j \in [L_\tau]}\right) \leq 0.1\tau'\right] \geq 2/3$$

2. **Soundness:** For any cheating strategy $\widetilde{\mathsf{P}}^{\mathcal{D}}$, we have

$$\Pr\left[\text{out}\left[\Pi\left(\widetilde{\mathsf{P}}^{\mathcal{D}}, \mathsf{T}^{(\mathcal{D})}; \tau', N\right)\right] \neq \mathsf{Reject} \wedge \mathtt{RL}\left(\mathcal{D}, \{\widetilde{p_j}\}_{j \in [L_\tau]}\right) \geq \tau'\right] \leq 1/3$$

1. **Sample and Query Complexity**: $\widetilde{O}\left(\frac{(\log N)^4}{\text{poly}(\tau)}\right)$
2. **Communication Complexity** $\widetilde{O}\left(\sqrt{N}\,\text{poly}(1/\tau)\right)$
3. **Round Complexity**: $\widetilde{O}\left(\log N \cdot \text{poly}(1/\tau)\right)$

Proof. Let $K \subseteq L_\tau$ denote the indices of the "relevant" buckets of the (N, τ)-histogram of $\mathcal{D}$. By Claim 5.2, for each relevant bucket $j \in K$ there exists some $y_j \in B_j^{(\mathcal{D},\tau)}$ such that $\mathcal{D}\left[U^{\mathcal{D}}_{\tau/3}\left(y_j\right)\right] \geq \frac{\tau}{30L}$.

Completeness: As the prover is honest, the histogram that the verifier outputs is simply the histogram one would learn if they could sample from the histogram itself. The folklore learning theorem says it is possible to learn an arbitrary distribution over domain of size L_τ up to ϵ error in total variation distance with probability δ takes $O\left(\frac{L_\tau + \log(1/\delta)}{\epsilon^2}\right)$. Setting $\delta = 1/20$, $\epsilon = 0.1\tau'$, and from our definition of s, we have with probability at most $1/20$ we fail to learn a good approximation of the histogram, with room to spare. Thus, with probability at least $1 - 1/20$ we have that $\mathtt{RL}\left(\mathcal{D}, \{\widetilde{p_j}\}_{j \in [L_\tau]}\right) \leq 0.1\tau'$.

Next we show that the probability that the prover outputs Reject is upper bounded by 1/3. Based on Claim 5.2, the honest prover is able to find a good y_j for all $j \in K$ where $\widetilde{K} = K$. We also have the guarantee that for all $j \in K$ (as the prover is honest about the bucket index of y_j)

$$r_j \stackrel{\text{def}}{=} \frac{\widetilde{\mathcal{D}}\left[y_j\right]}{\mathcal{D}\left[y_j\right]} \in \left[\frac{1}{1+\tau}, 1+\tau\right] \tag{20}$$

For every $j \in K$, the verifier runs `Approximate Single` with $y^\star = y_j$ and $\delta' = \frac{1}{20L}$. Therefore, the probability that the verifier accidentally rejects any of the claims about $\widetilde{\mathcal{D}}\left[y_j\right]$ is at most 1/20 (the union bound over the probability that `Approximate Single` fails).

Let S be the set of samples as described in Figure 9, with $s \stackrel{\text{def}}{=} |S|$. For each $x \in S$, let $\widetilde{\text{Bucket}_\tau}(x) \in [L_\tau]$ denote the provers claim about the bucket index of x. If for any x, $\widetilde{\text{Bucket}_\tau}(x) \notin \widetilde{K}$, the verifier outputs Reject immediately. If the prover honestly declares all relevant buckets, this event happens when at least 1 out of the s samples from $\mathcal{D}$ are in "non-relevant" buckets, which happens with probability at most $\tau/10 < 1/20$.

Set $\delta = 1/20$ and $\gamma = \frac{\tau}{s(1+\tau)}$. As we did not reject in the previous step, for each $x \in S$, there is a corresponding y_k such that $k = \widetilde{\text{Bucket}_\tau}(x)$. This implies, with only $O\left(\frac{K \log 2s/\delta}{\gamma^2}\right)$ PCond queries, the Compare Algorithm guarantees, for every $x \in S$, where $K = 2$.

$$\Pr_{\alpha_{xy_k} \leftarrow \$ \text{Compare}(x, y_k; \gamma, \delta/s)} \left[\left|\alpha_{xy_k} - \frac{\mathcal{D}[x]}{\mathcal{D}[y_k]}\right| \geq \gamma\right] \leq \frac{\delta}{s} \tag{21}$$

Let $\widetilde{\mathcal{D}} = \arg\min_{D' \leftarrow \{\widetilde{p_j}\}_{j \in [L_\tau]}} d_{\text{TV}}(\mathcal{D}, D')$, where we used the notation that $D' \leftarrow \{\widetilde{p_j}\}_{j \in [L_\tau]}$ to denote that the (N, τ)-histogram of $\mathcal{D}'$ is $\{\widetilde{p_j}\}_{j \in [L_\tau]}$. In what follows, for any $x \in S$, we use $y^\star(x)$ to denote the corresponding y_k such that $\widetilde{\text{Bucket}_\tau}(x) = k$, $r(x)$ to be the corresponding ratio r_k. Thus, we have with probability $1 - \delta/s$, for any given x,

$$\begin{aligned}
\left|\widetilde{\mathcal{D}}\left[y^\star(x)\right]\alpha_{xy^\star} - \widetilde{\mathcal{D}}[x]\right| &= \left|\widetilde{\mathcal{D}}\left[y^\star(x)\right]\alpha_{xy^\star(x)} - \widetilde{\mathcal{D}}\left[y^\star(x)\right]\frac{\mathcal{D}[x]}{\mathcal{D}[y^\star(x)]} + \widetilde{\mathcal{D}}\left[y^\star(x)\right]\frac{\mathcal{D}[x]}{\mathcal{D}[y^\star(x)]} - \widetilde{\mathcal{D}}[x]\right| \\
&\leq \widetilde{\mathcal{D}}\left[y^\star(x)\right]\left|\alpha_{xy^\star(x)} - \frac{\mathcal{D}[x]}{\mathcal{D}[y^\star(x)]}\right| + \left|\widetilde{\mathcal{D}}\left[y^\star(x)\right]\frac{\mathcal{D}[x]}{\mathcal{D}[y^\star(x)]} - \widetilde{\mathcal{D}}[x]\right| \\
&\leq \widetilde{\mathcal{D}}\left[y^\star(x)\right]\gamma + \left|\mathcal{D}[x]r(x) - \widetilde{\mathcal{D}}[x]\right| \\
&= \widetilde{\mathcal{D}}\left[y^\star(x)\right]\gamma + \left|\mathcal{D}[x](r(x) - 1) + \mathcal{D}[x] - \widetilde{\mathcal{D}}[x]\right| \\
&\leq \gamma + \mathcal{D}[x]\,|r(x) - 1| + \left|\mathcal{D}[x] - \widetilde{\mathcal{D}}[x]\right| \\
&\leq \gamma + \mathcal{D}[x] \cdot \tau + \left|\mathcal{D}[x] - \widetilde{\mathcal{D}}[x]\right|
\end{aligned}$$

where for the last inequality we used $|r(x) - 1| \leq \max\left\{1 - \frac{1}{1+\tau}, 1 + \tau - 1\right\} \leq \tau$. Summing over all $x \in S$, and taking the union bound, with probability at least $1 - 1/20$ we have

$$\begin{aligned}
\sum_{x \in S}\left|\widetilde{\mathcal{D}}\left[y^\star(x)\right]\alpha_{xy^\star(x)} - \widetilde{\mathcal{D}}[x]\right| &\leq s\gamma + |(r-1)|\sum_{x \in S}\mathcal{D}[x] + \sum_{x \in S}\left|\mathcal{D}[x] - \widetilde{\mathcal{D}}[x]\right| \\
&\leq 3\tau \\
&< 2\tau' - (2\tau + \tau^2) - \frac{\tau}{1+\tau}.
\end{aligned}$$

Of course there is a chance that the prover does everything honestly, and the verifier accepts but the verifier got very unlucky, and the samples in S are a bad empirical estimator for $\{p_j\}_{j \in [L_\tau]}$. With how we set s, this happens with probability at most $1/20$. This completes the proof for completeness with error $4/20 < 1/3$.

Soundness: For any $y^\star \in \{y_1, \ldots, y_k\}$, if the prover lies about $\widetilde{\mathcal{D}}[y^\star]$ such that $\frac{\widetilde{\mathcal{D}}[y^\star]}{\mathcal{D}[y^\star]} \notin [\frac{1}{(1+\tau)^2}, (1+\tau)^2]$, based on the soundness of Theorem 5.1, $\Pr[\mathsf{T} \text{ outputs } \mathsf{Accept}] \leq 1/20L$. We require the verifier to not reject *any* of the tests, so by a union bound, the probability that the verifier accepts given the prover grossly lied about any y_j's bucket index, is at most $1/20$ with plenty of room to spare.

Conditioning on the event that that the verifier did not reject the claim about the mass of any $y^\star \in \{y_1, \ldots, y_k\}$, we have with probability at least $1 - 1/20$, for all $y^\star \in \{y_1, \ldots, y_k\}$,

$$r \stackrel{\text{def}}{=} \frac{\widetilde{\mathcal{D}}[y^\star]}{\mathcal{D}[y^\star]} \in [\frac{1}{(1+\tau)^2}, (1+\tau)^2]$$

There is a chance that the prover might not declare every relevant bucket of the distribution. The prover gets away with this if none of the verifiers samples are from said relevant bucket. However as

$s \stackrel{\text{def}}{=} \frac{10L_\tau}{\tau}\log(10L_\tau)$, using a simple tail bound argument that $L\left(1-\frac{\tau}{10L_\tau}\right)^s \le 1/20$, it is easy to see that with probability at least 1 - 1/10, every relevant bucket will be represented in the samples. For any $x \in S$, once again we use $y^\star(x)$ to denote the corresponding y_k such that $\widetilde{\text{Bucket}}_\tau(x) = k$. We have $|\delta_x| \le \gamma$ for all $x \in S$, where $\delta_x \stackrel{\text{def}}{=} \alpha_{xy^\star} - \frac{\mathcal{D}[x]}{\mathcal{D}[y^\star]}$, (Safe to assume `COMPARE` did not output High or Low, or we would reject immediately), with probability at most 1/20. With a bit of algebra we get

$$\widetilde{\mathcal{D}}\left[y^\star(x)\right]\alpha_{xy^\star(x)} = r(x)\mathcal{D}[x] + \widetilde{\mathcal{D}}\left[y^\star(x)\right]\delta_x \tag{22}$$

Thus,

$$\begin{aligned}
|\widetilde{\mathcal{D}}\left[y^\star(x)\right]\alpha_{xy^\star(x)} - \widetilde{\mathcal{D}}[x]| &= |(\mathcal{D}[x] - \widetilde{\mathcal{D}}[x]) + (r(x)-1)\mathcal{D}[x] + \widetilde{\mathcal{D}}\left[y^\star(x)\right]\delta_x| && (23)\\
&\ge |\mathcal{D}[x] - \widetilde{\mathcal{D}}[x]| - |r(x)-1|\,\mathcal{D}[x] - \widetilde{\mathcal{D}}\left[y^\star(x)\right]|\delta_x| && (24)
\end{aligned}$$

Where (23) follows from Equation (22) and adding $\mathcal{D}[x] - \mathcal{D}[x] = 0$ to the equation, and Inequality (24) comes from the reverse-triangle inequality. Summing over $x \in S$ gives, with probability at least $1 - 1/20$,

$$\begin{aligned}
\sum_{x\in S}\left|\widetilde{\mathcal{D}}\left[y^\star(x)\right]\alpha_{xy^\star(x)} - \widetilde{\mathcal{D}}[x]\right| &\ge \sum_{x\in S}\left|\mathcal{D}[x] - \widetilde{\mathcal{D}}[x]\right| - \left|r(x)-1\right|\sum_{x\in S}\mathcal{D}[x] - \widetilde{\mathcal{D}}\left[y^\star(x)\right]\sum_{x\in S}\left|\delta_x\right| && (25)\\
&\ge \sum_{x\in S}\left|\mathcal{D}[x] - \widetilde{\mathcal{D}}[x]\right| - |r(x)-1| - \widetilde{\mathcal{D}}\left[y^\star(x)\right]s\gamma && (26)\\
&\ge \sum_{x\in S}\left|\mathcal{D}[x] - \widetilde{\mathcal{D}}[x]\right| - (2\tau+\tau^2) - \widetilde{\mathcal{D}}\left[y^\star(x)\right]s\gamma && (27)\\
&\ge 2\tau' - (2\tau+\tau^2) - \widetilde{\mathcal{D}}\left[y^\star(x)\right]s\gamma && (28)\\
&\ge 2\tau' - (2\tau+\tau^2) - \frac{\tau}{1+\tau} && (29)
\end{aligned}$$

Inequality (26) comes from $\sum_{x\in S}\mathcal{D}[x] \le 1$ and $\sum_{x\in S}|\delta_x| \le s\gamma$. Inequality (27) comes from the assumption on r

$$|r(x)-1| \le \max\left\{(1+\tau)^2 - 1,\ 1 - \frac{1}{(1+\tau)^2}\right\} = 2\tau+\tau^2.$$

Inequality (28) is a direct consequence of Lemma 2.7. Indeed, observe that $\sum_{x\in S}\left|\mathcal{D}[x] - \widetilde{\mathcal{D}}[x]\right| = \text{EMD}\left(\{p_j\}_{j\in[L_\tau]}, \{\widetilde{p_j}\}_{j\in[L_\tau]}\right)$ denotes the minimal probability mass of dirt one must move from $\{p_j\}_{j\in[L_\tau]}$ to construct $\{\widetilde{p_j}\}_{j\in[L_\tau]}$ as $\widetilde{\mathcal{D}} = \arg\min_{D'\leftarrow\{\widetilde{p_j}\}_{j\in[L_\tau]}} d_{\text{TV}}(\mathcal{D}, D')$.

Complexity: The verifier needs to run `ApproximateSingle` at most L_τ times. This give us a query and sample complexity of $\widetilde{O}\left(\frac{(\log N)^4}{\text{poly}(\tau)}\right)$. As every thing is done by sequential repetition, the final round complexity is $\widetilde{O}(\log N \cdot \text{poly}(1/\tau))$. As the communication of each `ApproximateSingle` is at most $\widetilde{O}\left(\sqrt{N}\right)$, by repeating $L_\tau = O(\log N)$ times, we get a $\widetilde{O}\left(\sqrt{N}\right)$ communication complexity. □

6 Conclusion And Open Problems

We have shown that *any* label-invariant property of a distribution can be efficiently certified using $\text{poly}(\log N)$ pairwise conditional queries, and with sublinear communication. In this section, we outline some open problems and future directions.

1. Is it possible to obtain *constant* (or, as a perhaps more achievable goal, depending only on τ and not on N) query complexity for verifying any label-invariant property with sub-linear communication? This question applies to the stronger `Cond` model as well.

2. Although we achieve $\text{poly}(\log N)$ complexity, there is still a gap between the identity tester that makes $O\left(\sqrt{\log N}\right)$ queries, and our verifier which makes $O\left(\log^4 N\right)$ queries. Can we close this gap, or is this is an inherent artefact of having less information, given the requirement of sub-linear communication?

3. Our protocol currently requires $\text{poly}(\log N)$ rounds of interaction. The main culprit here is sequential composition. Are these protocols composable in parallel? This would immediately give us constant-round communication. [Herman and Rothblum, 2022] give a 2-round interactive proof. Given that our support size protocol requires amplification that depends on the parameter τ, we cannot hope to achieve universally constant round complexity using just the techniques described in this paper.

4. Currently, our analysis requires the prover to have full (exact) knowledge of the underlying probability distribution. Herman and Rothblum [2023] introduce "doubly-efficient" proofs, where the prover itself only needs to learn the distribution up to small error. Can we tighten the analysis, or modify the protocol to accommodate this weaker version of the prover? We conjecture this is possible by tightening the analysis of our protocols.

5. Very little is known about the minimum amount of information a prover must communicate for efficient verification in both the sample and conditional access models. As a concrete question, can the $O(\sqrt{N})$ communication be improved for the broad class of label-invariant properties?

References

J. Acharya, C. L. Canonne, and G. Kamath. A chasm between identity and equivalence testing with conditional queries. In N. Garg, K. Jansen, A. Rao, and J. D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, volume 40 of *LIPIcs*, pages 449–466. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICS.APPROX-RANDOM.2015.449. URL https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2015.449.

A. Arun, S. Setty, and J. Thaler. Jolt: Snarks for virtual machines via lookups. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–33. Springer, 2024. URL https://eprint.iacr.org/2023/1217.

T. Batu and C. L. Canonne. Generalized uniformity testing. In *FOCS*, pages 880–889. IEEE Computer Society, 2017. URL https://arxiv.org/abs/1708.04696.

T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. In *41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000)*, pages 259–269. IEEE Comput. Soc. Press, Los Alamitos, CA, 2000. doi:10.1109/SFCS.2000.892113. URL https://doi.org/10.1109/SFCS.2000.892113.

T. Batu, S. Dasgupta, R. Kumar, and R. Rubinfeld. The complexity of approximating entropy. In J. H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 678–687. ACM, 2002. doi:10.1145/509907.510005. URL https://doi.org/10.1145/509907.510005.

I. Berman, R. D. Rothblum, and V. Vaikuntanathan. Zero-knowledge proofs of proximity. In A. R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 19:1–19:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.ITCS.2018.19. URL https://doi.org/10.4230/LIPIcs.ITCS.2018.19.

R. Bhattacharyya and S. Chakraborty. Property testing of joint distributions using conditional samples. *ACM Trans. Comput. Theory*, 10(4):16:1–16:20, 2018. doi:10.1145/3241377. URL https://doi.org/10.1145/3241377.

A. Blanca, Z. Chen, D. Stefankovic, and E. Vigoda. Complexity of high-dimensional identity testing with coordinate conditional sampling. *ACM Trans. Algorithms*, 21(1):7:1–7:58, 2025. doi:10.1145/3686799. URL https://doi.org/10.1145/3686799.

C. L. Canonne. *A Survey on Distribution Testing: Your Data is Big. But is it Blue?* Number 9 in Graduate Surveys. Theory of Computing Library, 2020. doi:10.4086/toc.gs.2020.009. URL http://www.theoryofcomputing.org/library.html.

C. L. Canonne. Topics and techniques in distribution testing: A biased but representative sample. *Foundations and Trends® in Communications and Information Theory*, 19(6):1032–1198, 2022. ISSN 1567-2190. doi:10.1561/0100000114. URL http://dx.doi.org/10.1561/0100000114.

C. L. Canonne and R. Rubinfeld. Testing probability distributions underlying aggregated data. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 283–295. Springer, 2014. doi:10.1007/978-3-662-43948-7_24. URL https://doi.org/10.1007/978-3-662-43948-7_24.

C. L. Canonne, D. Ron, and R. A. Servedio. Testing equivalence between distributions using conditional samples. In *ACM–SIAM Symposium on Discrete Algorithms (SODA)*, 2014. doi:http://dx.doi.org/10.1137/1.9781611973402.87. URL http://www.cs.columbia.edu/~rocco/papers/soda14cond.html.

C. L. Canonne, D. Ron, and R. A. Servedio. Testing probability distributions using conditional samples. *SIAM Journal on Computing*, 44(3):540–616, 2015. URL https://epubs.siam.org/doi/10.1137/130945508.

C. L. Canonne, X. Chen, G. Kamath, A. Levi, and E. Waingarten. Random restrictions of high dimensional distributions and uniformity testing with subcube conditioning. In D. Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 321–336. SIAM, 2021. doi:10.1137/1.9781611976465.21. URL https://doi.org/10.1137/1.9781611976465.21.

M. C. Caro, J. Eisert, M. Hinsche, M. Ioannou, A. Nietner, and R. Sweke. Interactive proofs for verifying (quantum) learning and testing. *CoRR*, 2024a. doi:10.48550/ARXIV.2410.23969. URL https://doi.org/10.48550/arXiv.2410.23969.

M. C. Caro, M. Hinsche, M. Ioannou, A. Nietner, and R. Sweke. Classical verification of quantum learning. In V. Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPIcs*, pages 24:1–24:23.

Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024b. doi:10.4230/LIPICS.ITCS.2024.24. URL https://doi.org/10.4230/LIPIcs.ITCS.2024.24.

D. Chakrabarty, X. Chen, S. Ristic, C. Seshadhri, and E. Waingarten. Monotonicity testing of high-dimensional distributions with subcube conditioning. In M. Koucký and N. Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 1019–1030. ACM, 2025. doi:10.1145/3717823.3718297. URL https://doi.org/10.1145/3717823.3718297.

D. Chakraborty, G. Kumar, and K. S. Meel. Support size estimation: The power of conditioning. In J. Leroux, S. Lombardy, and D. Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023, August 28 to September 1, 2023, Bordeaux, France*, volume 272 of *LIPIcs*, pages 33:1–33:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.MFCS.2023.33. URL https://doi.org/10.4230/LIPIcs.MFCS.2023.33.

D. Chakraborty, S. Chakraborty, and G. Kumar. Tight lower bound on equivalence testing in conditional sampling model. In D. P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4371–4394. SIAM, 2024. doi:10.1137/1.9781611977912.153. URL https://doi.org/10.1137/1.9781611977912.153.

S. Chakraborty, E. Fischer, Y. Goldhirsh, and A. Matsliah. On the power of conditional samples in distribution testing. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 561–580, 2013. URL https://arxiv.org/abs/1210.8338.

X. Chen, R. Jayaram, A. Levi, and E. Waingarten. Learning and testing junta distributions with sub cube conditioning. In M. Belkin and S. Kpotufe, editors, *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pages 1060–1113. PMLR, 2021. URL http://proceedings.mlr.press/v134/chen21b.html.

A. Chiesa and T. Gur. Proofs of proximity for distribution testing. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. URL https://drops.dagstuhl.de/storage/00lipics/lipics-vol094-itcs2018/LIPIcs.ITCS.2018.53/LIPIcs.ITCS.2018.53.pdf.

I. Diakonikolas, D. M. Kane, and A. Stewart. Sharp bounds for generalized uniformity testing. In *NeurIPS*, pages 6204–6213, 2018. URL https://arxiv.org/abs/1709.02087.

M. Falahatgar, A. Jafarpour, A. Orlitsky, V. Pichapati, and A. T. Suresh. Faster algorithms for testing under conditional sampling. In *Conference on Learning Theory*, pages 607–636. PMLR, 2015. URL https://arxiv.org/abs/1504.04103.

R. Ferreira Pinto Jr. and N. Harms. Testing support size more efficiently than learning histograms. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, STOC '25, pages 995–1006, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715105. doi:10.1145/3717823.3718134. URL https://doi.org/10.1145/3717823.3718134.

O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

O. Goldreich and D. Ron. On testing expansion in bounded-degree graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation: In*

Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman, pages 68–75. Springer, 2011. URL https://link.springer.com/chapter/10.1007/978-3-642-22670-0_9.

O. Goldreich and D. Ron. On the relation between the relative earth mover distance and the variation distance (an exposition). In *Computational Complexity and Property Testing: On the Interplay Between Randomness and Computation*, pages 141–151. Springer, 2020. URL https://link.springer.com/book/10.1007/978-3-030-43662-9.

O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, July 1998. URL https://dl.acm.org/doi/10.1145/285055.285060.

S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, page 291âĂŞ304, New York, NY, USA, 1985. Association for Computing Machinery. ISBN 0897911512. doi:10.1145/22145.22178. URL https://doi.org/10.1145/22145.22178.

S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4), Sept. 2015. ISSN 0004–5411. doi:10.1145/2699436. URL https://doi.org/10.1145/2699436.

S. Goldwasser, G. N. Rothblum, J. Shafer, and A. Yehudayoff. Interactive proofs for verifying machine learning. In J. R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 41:1–41:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ITCS.2021.41. URL https://doi.org/10.4230/LIPIcs.ITCS.2021.41.

S. Guha, A. McGregor, and S. Venkatasubramanian. Sublinear estimation of entropy and information distances. *ACM Trans. Algorithms*, 5(4):35:1–35:16, 2009. doi:10.1145/1597036.1597038. URL https://doi.org/10.1145/1597036.1597038.

T. Gur, M. M. Jahanara, M. M. Khodabandeh, N. Rajgopal, B. Salamatian, and I. Shinkar. On the power of interactive proofs for learning. In B. Mohar, I. Shinkar, and R. O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1063–1070. ACM, 2024. doi:10.1145/3618260.3649784. URL https://doi.org/10.1145/3618260.3649784.

T. Herman. Public coin interactive proofs for label-invariant distribution properties. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2024. URL https://drops.dagstuhl.de/storage/00lipics/lipics-vol317-approx-random2024/LIPIcs.APPROX-RANDOM.2024.72/LIPIcs.APPROX-RANDOM.2024.72.pdf.

T. Herman and G. Rothblum. Doubley-efficient interactive proofs for distribution properties. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 743–751. IEEE, 2023. URL https://eccc.weizmann.ac.il/report/2023/161/download.

T. Herman and G. N. Rothblum. Verifying the unseen: interactive proofs for label-invariant distribution properties. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1208–1219, 2022. URL https://dl.acm.org/doi/10.1145/3519935.3519987.

T. Herman and G. N. Rothblum. Interactive proofs for general distribution properties. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*, pages 528–538. IEEE, 2024. doi:10.1109/FOCS61266.2024.00041. URL https://doi.org/10.1109/FOCS61266.2024.00041.

T. Herman and G. N. Rothblum. How to verify any (reasonable) distribution property: Computationally sound argument systems for distributions. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=GfXMTAJaxZ.

G. Kamath and C. Tzamos. Anaconda: A non-adaptive conditional sampling algorithm for distribution testing. In T. M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 679–693. SIAM, 2019. doi:10.1137/1.9781611975482.43. URL https://doi.org/10.1137/1.9781611975482.43.

G. Kumar, K. S. Meel, and Y. Pote. Tolerant testing of high-dimensional samplers with subcube conditioning. *CoRR*, abs/2308.04264, 2023. doi:10.48550/ARXIV.2308.04264. URL https://doi.org/10.48550/arXiv.2308.04264.

S. Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, Oct. 2000. ISSN 0097-5397. doi:10.1137/S0097539795284959. URL https://doi.org/10.1137/S0097539795284959.

S. Mutreja and J. Shafer. PAC verification of statistical algorithms. In G. Neu and L. Rosasco, editors, *The Thirty Sixth Annual Conference on Learning Theory, COLT 2023, 12-15 July 2023, Bangalore, India*, volume 195 of *Proceedings of Machine Learning Research*, pages 5021–5043. PMLR, 2023. URL https://proceedings.mlr.press/v195/mutreja23a.html.

S. Narayanan. On tolerant distribution testing in the conditional sampling model. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '21, pages 357–373, USA, 2021. Society for Industrial and Applied Mathematics. ISBN 9781611976465. URL https://dl.acm.org/doi/10.5555/3458064.3458087.

K. Onak and X. Sun. Probability-revealing samples. In *AISTATS*, volume 84 of *Proceedings of Machine Learning Research*, pages 2018–2026. PMLR, 2018.

L. Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Trans. Inf. Theor.*, 54(10):4750âĂŞ–4755, Oct. 2008. ISSN 0018-9448. doi:10.1109/TIT.2008.928987. URL https://doi.org/10.1109/TIT.2008.928987.

S. Raskhodnikova, D. Ron, A. Shpilka, and A. D. Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM J. Comput.*, 39(3):813–842, 2009. URL https://cs-people.bu.edu/sofya/pubs/dss-sicomp.pdf.

G. N. Rothblum, S. Vadhan, and A. Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 793–802, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320290. doi:10.1145/2488608.2488709. URL https://doi.org/10.1145/2488608.2488709.

R. Rubinfeld. Taming big probability distributions. *XRDS: Crossroads, The ACM Magazine for Students*, 19(1):24, sep 2012. doi:10.1145/2331042.2331052. URL http://dx.doi.org/10.1145/2331042.2331052.

R. Rubinfeld and R. A. Servedio. Testing monotone high-dimensional distributions. In H. N. Gabow and R. Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 147–156. ACM, 2005. doi:10.1145/1060590.1060613. URL https://doi.org/10.1145/1060590.1060613.

R. Rubinfeld and M. Sudan. Robust characterizations of polynomials withapplications to program testing. *SIAM J. Comput.*, 25(2):252–271, Feb. 1996. ISSN 0097-5397. doi:10.1137/S0097539793255151. URL https://doi.org/10.1137/S0097539793255151.

G. Valiant and P. Valiant. Estimating the unseen: an n/log(n)-sample estimator for entropy and support size, shown optimal via new clts. In *STOC*, pages 685–694. ACM, 2011. URL https://dl.acm.org/doi/10.1145/1993636.1993727.

G. Valiant and P. Valiant. Estimating the unseen: improved estimators for entropy and other properties. *Journal of the ACM (JACM)*, 64(6):1–41, 2017. URL https://dl.acm.org/doi/10.1145/3125643.

P. Valiant. Testing symmetric properties of distributions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 383–392, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580470. doi:10.1145/1374376.1374432. URL https://doi.org/10.1145/1374376.1374432.

Y. Wu and P. Yang. Chebyshev polynomials, moment matching, and optimal estimation of the unseen. *Ann. Statist.*, 47(2):857–883, 2019. ISSN 0090-5364. doi:10.1214/17-AOS1665. URL https://doi.org/10.1214/17-AOS1665.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il