“DISTRIBUTION TESTING?”
Property testing of probability distributions:
Property testing of probability distributions: sublinear,
Property testing of probability distributions: sublinear, approximate,
Property testing of probability distributions: sublinear, approximate, randomized
Property testing of probability distributions: sublinear, approximate, randomized algorithms that take random samples.
Property testing of probability distributions: sublinear, approximate, randomized algorithms that take random samples

- Big Dataset: too big
Property testing of probability distributions: sublinear, approximate, randomized algorithms that take random samples

- Big Dataset: too big
- Expensive access: pricey data
Property testing of probability distributions: sublinear, approximate, randomized algorithms that take random samples

- Big Dataset: too big
- Expensive access: pricey data
- “Model selection”: many options
Property testing of probability distributions: sublinear, approximate, randomized algorithms that take random samples

- Big Dataset: too big
- Expensive access: pricey data
- "Model selection": many options

Need to infer information – one bit – from the data: fast, or with very few samples.
(Property) Distribution Testing:
(Property) Distribution Testing:
(Property) Distribution Testing:

in an (egg)shell.
Known domain (here \([n] = \{1, \ldots, n\}\))
Property (or class) \(C \subseteq \Delta([n])\)
Independent samples from unknown \(D \in \Delta([n])\)
Distance parameter \(\varepsilon \in (0, 1]\)
Known domain (here \([\{1, \ldots, n\}\] = \([n]\))

Property (or class) \(C \subseteq \Delta([n])\)

Independent samples from unknown \(D \in \Delta([n])\)

Distance parameter \(\varepsilon \in (0, 1]\)

**Must decide:**

\[D \in C\]
Known domain (here $[n] = \{1, \ldots, n\}$)
Property (or class) $\mathcal{C} \subseteq \Delta([n])$
Independent samples from unknown $D \in \Delta([n])$
Distance parameter $\varepsilon \in (0, 1]$

Must decide:

$D \in \mathcal{C}$, or $\ell_1(D, \mathcal{C}) > \varepsilon$?
Known domain (here $[n] = \{1, \ldots, n\}$)

Property (or class) $\mathcal{C} \subseteq \Delta([n])$

Independent samples from unknown $D \in \Delta([n])$

Distance parameter $\varepsilon \in (0, 1]$

**Must decide:**

$$D \in \mathcal{C}, \text{ or } \ell_1(D, \mathcal{C}) > \varepsilon?$$

(and be correct on any $D$ with probability at least $2/3$)
Many individual results on specific properties:
Many individual results on specific properties:

- Uniformity [GR00, BFR+00, Pan08]

...but almost none on general frameworks.
Many *individual* results on *specific* properties:

- Uniformity [GR00, BFR°00, Pan08]
- Identity [BFF°01, VV14]
Many individual results on specific properties:

- Uniformity \([\text{GR00, BFR}^+00, \text{Pan08}]\)
- Identity \([\text{BFF}^+01, \text{VV14}]\)
- Equivalence \([\text{BFR}^+00, \text{Val11, CDVV14}]\)
Many *individual* results on *specific* properties:

- Uniformity $[\text{GR00, BFR}^+00, \text{Pan08}]$
- Identity $[\text{BFF}^+01, \text{VV14}]$
- Equivalence $[\text{BFR}^+00, \text{Val11, CDVV14}]$
- Independence $[\text{BFF}^+01, \text{LRR13}]$
Many individual results on specific properties:

- Uniformity
  - [GR00, BFR$^+$00, Pan08]
- Identity
  - [BFF$^+$01, VV14]
- Equivalence
  - [BFR$^+$00, Val11, CDVV14]
- Independence
  - [BFF$^+$01, LRR13]
- Monotonicity
  - [BKR04]
Many *individual* results on *specific* properties:

- Uniformity [GR00, BFR$^+$00, Pan08]
- Identity [BFF$^+$01, VV14]
- Equivalence [BFR$^+$00, Val11, CDVV14]
- Independence [BFF$^+$01, LRR13]
- Monotonicity [BKR04]
- Poisson Binomial Distributions [AD14]
Many *individual* results on *specific* properties:

- Uniformity [GR00, BFR\textsuperscript{+}00, Pan08]
- Identity [BFF\textsuperscript{+}01, VV14]
- Equivalence [BFR\textsuperscript{+}00, Val11, CDVV14]
- Independence [BFF\textsuperscript{+}01, LRR13]
- Monotonicity [BKR04]
- Poisson Binomial Distributions [AD14]
- and more…
Many *individual* results on *specific* properties:

- Uniformity [GR00, BFR⁺00, Pan08]
- Identity [BFF⁺01, VV14]
- Equivalence [BFR⁺00, Val11, CDVV14]
- Independence [BFF⁺01, LRR13]
- Monotonicity [BKR04]
- Poisson Binomial Distributions [AD14]
- and more…

...but *almost none* on general frameworks.*
Our focus

The property is a structured class $C$
Our focus

The property is a structured class $C$ (think “Binomial distributions”).
Our focus

The property is a **structured class** \( C \) (think “Binomial distributions”). We want methods that apply to many such classes at once.
Our focus

The property is a structured class $C$ (think “Binomial distributions”). We want methods that apply to many such classes at once.
Our focus

The property is a **structured class** $\mathcal{C}$ (think “Binomial distributions”). We want methods that apply to **many** such classes at once.
Theorem ([CDGR15])

There exists a generic algorithm that can test membership to any class that satisfies some structural criterion. (Moreover, for many such $\mathcal{C}$ this algorithm has near-optimal sample complexity.)
Theorem ([CDGR15])

There exists a *generic* algorithm that can test membership to any class that satisfies some *structural criterion*. (Moreover, for many such $\mathcal{C}$ this algorithm has near-optimal sample complexity.)

**Applications**

Monotonicity, unimodality, t-modality, log-concavity, convexity, histograms, piecewise-polynomials, monotone hazard rate, PBD, Binomials, and mixtures thereof.
Theorem ([CDGR15])

There exists a \textit{generic} algorithm that can test membership to \textbf{any} class that satisfies some \textit{structural criterion}. (Moreover, for many such $\mathcal{C}$ this algorithm has near-optimal sample complexity.)

\textbf{Applications}

Monotonicity, unimodality, $t$-modality, log-concavity, convexity, histograms, piecewise-polynomials, monotone hazard rate, PBD, Binomials, and mixtures thereof. \textit{(Better than snake oil!)}
Theorem ([CDGR15])

Any class $\mathcal{C}$ that can be (agnostically) learned efficiently is at least as hard to test as the hardest distribution it contains.
Theorem ([CDGR15])

Any class $\mathcal{C}$ that can be (agnostically) learned efficiently is at least as hard to test as the hardest distribution it contains.

Applications

Monotonicity, unimodality, t-modality, log-concavity, convexity, histograms, piecewise-polynomials, monotone hazard rate, PBD, Binomials.
Theorem ([CDGR15])

Any class $C$ that can be (agnostically) learned efficiently is at least as hard to test as the hardest distribution it contains.

Applications

Monotonicity, unimodality, t-modality, log-concavity, convexity, histograms, piecewise-polynomials, monotone hazard rate, PBD, Binomials. Also, k-SIIRVS.
Theorem ([CDGR15])

Any class $C$ that can be (agnostically) learned efficiently is at least as hard to test as the hardest distribution it contains.

Applications

Monotonicity, unimodality, t-modality, log-concavity, convexity, histograms, piecewise-polynomials, monotone hazard rate, PBD, Binomials. Also, k-SIIRVS.

(works for tolerant testing too.)
Theorem ([ADK15])

Any class $\mathcal{C}$ that can be learned efficiently \textit{in $\chi^2$ distance} can be tested with $O(\sqrt{n})$ samples.
Theorem ([ADK15])

Any class $\mathcal{C}$ that can be learned efficiently \textit{in $\chi^2$ distance} can be tested with $O(\sqrt{n})$ samples.

Applications

Monotonicity, unimodality, log-concavity, monotone hazard rate, independence.
Theorem ([ADK15])

Any class $C$ that can be learned efficiently in $\chi^2$ distance can be tested with $O(\sqrt{n})$ samples.

Applications

Monotonicity, unimodality, log-concavity, monotone hazard rate, independence. (Tight upper bounds!)
But... Why is it surprising?
But...

Why is it surprising?

· Only need to prove a **structural, existential** result about $C$!
But...

Why is it surprising?

- Only need to prove a structural, existential result about $C$!
- Learning and testing (in $\ell_1$) are unrelated for distributions.
But...

Why is it surprising?

- Only need to prove a structural, existential result about $C$!
- Learning and testing (in $\ell_1$) are unrelated for distributions.
- Testing-by-learning was seemingly ruled out... [VV11]
A UNIFIED APPROACH TO THINGS
Say $C$ is $(\gamma, L(\gamma))$-decomposable if any $D \in C$ is well-approximated by some piecewise-constant distribution on $L$ pieces $I_1, \ldots, I_L$:

1. $D(i) \in [(1 - \gamma), (1 + \gamma)] \cdot \frac{D(I)}{|I|}$ for all $i \in I$; or
2. $D(I) \leq \frac{\gamma}{L}$

for every $I$ among $I_1, \ldots, I_L$. 
Say $\mathcal{C}$ is $($\(\gamma, L(\gamma)\))-decomposable if any $D \in \mathcal{C}$ is well-approximated by some piecewise-constant distribution on $L$ pieces $I_1, \ldots, I_L$:

1. $D(i) \in [(1 - \gamma), (1 + \gamma)] \cdot \frac{D(I)}{|I|}$ for all $i \in I$; or
2. $D(I) \leq \frac{\gamma}{L}$

for every $I$ among $I_1, \ldots, I_L$.

I.e., each $D \in \mathcal{C}$ is piecewise flat, in a strong $\ell_2$-like sense.
Then...

Any \((\gamma, L(\gamma))\)-decomposable \(C\) can be tested by the same generic algorithm, with \(\tilde{O}(\frac{\sqrt{L(\epsilon)n}}{\epsilon^3} + \frac{L(\epsilon)}{\epsilon^2})\) samples.
Then...

Any \((\gamma, L(\gamma))\)-decomposable \(C\) can be tested by the same generic algorithm, with \(\tilde{O}(\frac{\sqrt{L(\varepsilon)n}}{\varepsilon^3} + \frac{L(\varepsilon)}{\varepsilon^2})\) samples.

Algorithm inspired from [BKR04]:
Then...

Any \((\gamma, L(\gamma))\)-decomposable \(C\) can be tested by the same generic algorithm, with \(\tilde{O}\left(\frac{\sqrt{L(\varepsilon)n}}{\varepsilon^3} + \frac{L(\varepsilon)}{\varepsilon^2}\right)\) samples.

Algorithm inspired from [BKR04]: decompose, learn, check.
Decompose: Attempt to recursively partition $[n]$ into $L$ intervals where $\ell_2(D, U) \leq \varepsilon/|I|$ or $D(I)$ small – should succeed if $D \in \mathcal{C}$ (by decomposability).
Decompose: Attempt to recursively partition \([n]\) into \(L\) intervals where \(\ell_2(D, U) \leq \varepsilon/|I|\) or \(D(I)\) small – should succeed if \(D \in \mathcal{C}\) (by decomposability).

Learn: Learn the “flattening” \(D'\) of \(D\) on this partition – if \(D \in \mathcal{C}\), then \(\ell_1(D, D')\) small.
**Decompose:** Attempt to recursively partition \([n]\) into \(L\) intervals where \(\ell_2(D, U) \leq \varepsilon/|I|\) or \(D(I)\) small – should succeed if \(D \in \mathcal{C}\) (by decomposability).

**Learn:** Learn the “flattening” \(D'\) of \(D\) on this partition – if \(D \in \mathcal{C}\), then \(\ell_1(D, D')\) small.

**Check:** Check offline that \(D'\) is close to \(\mathcal{C}\).
Decompose: Attempt to recursively partition $[n]$ into $L$ intervals where $\ell_2(D, U) \leq \varepsilon/|I|$ or $D(I)$ small – should succeed if $D \in \mathcal{C}$ (by decomposability).

Learn: Learn the “flattening” $D'$ of $D$ on this partition – if $D \in \mathcal{C}$, then $\ell_1(D, D')$ small.

Check: Check offline that $D'$ is close to $\mathcal{C}$. 

A few catches

Preliminary step: restrict to effective support.

Also… efficiency.

A few perks

Decomposability composes very well!
Decompose: Attempt to recursively partition \([n]\) into \(L\) intervals where \(\ell_2(D, U) \leq \varepsilon/|I|\) or \(D(I)\) small – should succeed if \(D \in \mathcal{C}\) (by decomposability).

Learn: Learn the “flattening” \(D'\) of \(D\) on this partition – if \(D \in \mathcal{C}\), then \(\ell_1(D, D')\) small.

Check: Check offline that \(D'\) is close to \(\mathcal{C}\).

A few catches

Preliminary step: restrict to effective support.
Decompose: Attempt to recursively partition \([n]\) into \(L\) intervals where \(\ell_2(D, U) \leq \varepsilon/|I|\) or \(D(I)\) small – should succeed if \(D \in \mathcal{C}\) (by decomposability).

Learn: Learn the “flattening” \(D'\) of \(D\) on this partition – if \(D \in \mathcal{C}\), then \(\ell_1(D, D')\) small.

Check: Check offline that \(D'\) is close to \(\mathcal{C}\).

A few catches

Preliminary step: restrict to effective support. Also... efficiency.

A few perks

Decomposability composes very well!
Theorem

Suppose $\mathcal{C}$ can be agnostically learned with sample complexity $q(\varepsilon, n)$ and contains a subclass $\mathcal{C}'$ that requires $t(\varepsilon, n) \gg q(\varepsilon, n)$ samples to be $\varepsilon$-tested. Then $\mathcal{C}$ requires $t(\varepsilon, n)$ samples to be $\varepsilon$-tested as well.
Theorem
Suppose $\mathcal{C}$ can be agnostically learned with sample complexity $q(\varepsilon, n)$ and contains a subclass $\mathcal{C}'$ that requires $t(\varepsilon, n) \gg q(\varepsilon, n)$ samples to be $\varepsilon$-tested. Then $\mathcal{C}$ requires $t(\varepsilon, n)$ samples to be $\varepsilon$-tested as well.

Proof.
Blackboard.
Theorem

Suppose $\mathcal{C}$ can be agnostically learned with sample complexity $q(\varepsilon, n)$ and contains a subclass $\mathcal{C}'$ that requires $t(\varepsilon, n) \gg q(\varepsilon, n)$ samples to be $\varepsilon$-tested. Then $\mathcal{C}$ requires $t(\varepsilon, n)$ samples to be $\varepsilon$-tested as well.

Proof.

Blackboard. Pictures of circles.
Theorem

Suppose $\mathcal{C}$ can be agnostically learned with sample complexity $q(\varepsilon, n)$ and contains a subclass $\mathcal{C}'$ that requires $t(\varepsilon, n) \gg q(\varepsilon, n)$ samples to be $\varepsilon$-tested. Then $\mathcal{C}$ requires $t(\varepsilon, n)$ samples to be $\varepsilon$-tested as well.

Proof.

Blackboard. Pictures of circles. $n$’s that look like $m$’s.
Theorem

Suppose $\mathcal{C}$ can be agnostically learned with sample complexity $q(\epsilon, n)$ and contains a subclass $\mathcal{C}'$ that requires $t(\epsilon, n) \gg q(\epsilon, n)$ samples to be $\epsilon$-tested. Then $\mathcal{C}$ requires $t(\epsilon, n)$ samples to be $\epsilon$-tested as well.

Proof.

Blackboard. Pictures of circles. $n$’s that look like $m$’s.

Combined with [VV14] and learning results from the literature, immediately implies many new or previous lower bounds. (Taking $\mathcal{C}' = \{U\}$ or $\{\text{Bin}(n, 1/2)\}$ often enough)
TESTING-BY-LEARNING
The usual argument for testing functions (or graphs)$^1$:

1. Learn $f$ as if $f \in C$, getting $\hat{f}$.
2. Check if $d(\hat{f}, C)$ is small.
3. Check if $d(\hat{f}, f)$ is small.

(Step 2 not even needed if the learning is proper.) If Step 1 is efficient, then so is the overall tester...

**Testing is no harder than learning!**

$^1$In Hamming distance.
The usual argument for testing functions (or graphs)$^1$: 

1. Learn $f$ as if $f \in \mathcal{C}$, getting $\hat{f}$.
2. Check if $d(\hat{f}, \mathcal{C})$ is small.
3. Check if $d(\hat{f}, f)$ is small.

(Step 2 not even needed if the learning is proper.) If Step 1 is efficient, then so is the overall tester...

Testing is no harder than learning!

but not for distributions.

---

$^1$In Hamming distance.
The usual argument for testing functions (or graphs)\(^1\):

1. Learn \(f\) as if \(f \in \mathcal{C}\), getting \(\hat{f}\).
2. Check if \(d(\hat{f}, \mathcal{C})\) is small.
3. Check if \(d(\hat{f}, f)\) is small.

(Step 2 not even needed if the learning is proper.) If Step 1 is efficient, then so is the overall tester...

Testing is no harder than learning!

but not for distributions. Step 3 is no longer easy for them! [VV11]

---

\(^1\)In Hamming distance.
So we hit a wall...

1. Learn $D$ (in $\ell_1$) as if $D \in C$, getting $\hat{D}$.
2. Check if $\ell_1(\hat{D}, C)$ is small.
3. Check if $\ell_1(\hat{D}, D)$ is small (or $\ell_1(\hat{D}, D)$ is big).  $\tilde{\Omega}(n)$ samples
So we hit a wall...

1. Learn $D$ (in $\ell_1$) as if $D \in \mathcal{C}$, getting $\hat{D}$.
2. Check if $\ell_1(\hat{D}, \mathcal{C})$ is small.
3. Check if $\ell_1(\hat{D}, D)$ is small (or $\ell_1(\hat{D}, D)$ is big). $\tilde{\Omega}(n)$ samples

[ADK15]'s idea: not breaking the wall. The wall is fine.
So we hit a wall...

1. Learn $D$ (in $\chi^2$) as if $D \in \mathcal{C}$, getting $\hat{D}$.

2. Check if $\ell_1(\hat{D}, \mathcal{C})$ is small.

3. Check if $\chi^2(\hat{D}, D)$ is small (or $\ell_1(\hat{D}, D)$ is big). $O(\sqrt{n}/\varepsilon^2)$ samples

[ADK15]’s idea: not breaking the wall. The wall is fine.
When can we?

When does it apply?

 APPLICATIONS Monotonicity, log-concavity, unimodality*, MHR, independence…

Perks and catches

It’s optimal!* But efficiency, as before, requires work.
When does it apply? Need an efficient $\chi^2$ learner for $C$. 
When does it apply? Need an efficient $\chi^2$ learner for $C$.

Applications

Monotonicity, log-concavity, unimodality*, MHR, independence...

Perks and catches

It’s optimal!* But efficiency, as before, requires work.
AND NOW...TESTING FLAT THINGS.
"Technically n, morally k."
“Technically n, morally k.”
How hard can it be to test that?
Previously:

\[ \tilde{O}(\sqrt{kn}/\epsilon^3) \text{ samples [ILR12, CDGR15], } \Omega(\sqrt{n}) \text{ [Pan08, ILR12]} \]
Previously:
\[ \tilde{O}(\sqrt{kn/\varepsilon^3}) \] samples [ILR12, CDGR15], \( \Omega(\sqrt{n}) \) [Pan08, ILR12]

**Theorem**

Testing k-histograms can be done (efficiently) with \( O\left(\frac{\sqrt{n}}{\varepsilon^2} \log k + \frac{k}{\varepsilon^3} \log^2 k\right) \) samples.
Previously:

\(\tilde{O}(\sqrt{kn}/\varepsilon^3)\) samples [ILR12, CDGR15], \(\Omega(\sqrt{n})\) [Pan08, ILR12]

**Theorem**

Testing \(k\)-histograms can be done (efficiently) with 

\[O\left(\frac{\sqrt{n}}{\varepsilon^2} \log k + \frac{k}{\varepsilon^3} \log^2 k\right)\] samples.

**Theorem**

Testing \(k\)-histograms requires 

\[\Omega\left(\frac{\sqrt{n}}{\varepsilon^2} + \frac{k}{\varepsilon \log k}\right)\] samples.
Previously:
\(\tilde{O}(\sqrt{kn}/\varepsilon^3)\) samples [ILR12, CDGR15], \(\Omega(\sqrt{n})\) [Pan08, ILR12]

Theorem

Testing \(k\)-histograms can be done (efficiently) with \(O\left(\frac{\sqrt{n}}{\varepsilon^2} \log k + \frac{k}{\varepsilon^3} \log^2 k\right)\) samples.

Theorem

Testing \(k\)-histograms requires \(\Omega\left(\frac{\sqrt{n}}{\varepsilon^2} + \frac{k}{\varepsilon \log k}\right)\) samples.

For \(k \gg \sqrt{n}\), first “natural property” provably harder than uniformity.
Idea:

Apply the “testing-by-learning” technique of [ADK15].
Idea:
Apply the “testing-by-learning” technique of [ADK15].

Problem
We know how to (optimally) learn k-histograms in $\ell_1$ and $\ell_2$; or, if the partition is known, in $\chi^2$. 
Idea:
Apply the “testing-by-learning” technique of [ADK15].

Problem
We know how to (optimally) learn k-histograms in $\ell_1$ and $\ell_2$; or, if the partition is known, in $\chi^2$. But we do not have a $\chi^2$ learner!
Idea:
Apply the “testing-by-learning” technique of [ADK15].

Problem
We know how to (optimally) learn k-histograms in $\ell_1$ and $\ell_2$; or, if the partition is known, in $\chi^2$. But we do not have a $\chi^2$ learner!

A solution
Do not actually “learn, then test.” Implicitly learn in $\chi^2$, then use testing to refine the learning.
Idea:
Apply the “testing-by-learning” technique of [ADK15].

Problem
We know how to (optimally) learn k-histograms in $\ell_1$ and $\ell_2$; or, if the partition is known, in $\chi^2$. But we do not have a $\chi^2$ learner!

A solution
Do not actually “learn, then test.” Implicitly learn in $\chi^2$, then use testing to refine the learning.

“Testing-by-(learning-by-testing)”
Idea:
Apply the “testing-by-learning” technique of [ADK15].

Problem
We know how to (optimally) learn k-histograms in $\ell_1$ and $\ell_2$; or, if the partition is known, in $\chi^2$. But we do not have a $\chi^2$ learner!

A solution
Do not actually “learn, then test.” Implicitly learn in $\chi^2$, then use testing to refine the learning.

“Testing-by-(learning-by-testing)”

(This is where the extra log k factor comes from.)
Idea:

“Use someone else’s work” (a.k.a reduction).
Idea:

“Use someone else’s work” (a.k.a reduction). Stronger type of lower bound known: [VV11], for estimating symmetric properties.
Idea:

“Use someone else’s work” (a.k.a reduction). Stronger type of lower bound known: [VV11], for estimating symmetric properties.

Problem

Being a k-histogram is not really really not a symmetric property.
Idea:

“Use someone else’s work” (a.k.a reduction). Stronger type of lower bound known: [VV11], for estimating symmetric properties.

Problem

Being a k-histogram is not really really not a symmetric property.

A solution

Symmetrize it by applying a random permutation!
Distribution $D$ on $[2k]$: support size $\leq k/2$ or $\geq 3k/2$?
Distribution $D$ on $[2k]$: support size $\leq k/2$ or $\geq 3k/2$? Hard by [VV11].
Distribution D on $[2k]$: support size $\leq k/2$ or $\geq 3k/2$? Hard by [VV11].

**Reduction:**

1. Embed D it in $[n]$, where $n = 1000k$;
Distribution D on $[2k]$: support size $\leq k/2$ or $\geq 3k/2$? Hard by [VV11].

**Reduction:**

1. Embed D it in $[n]$, where $n = 1000k$;
2. Randomly permute the support with an u.a.r. $\sigma \in S_n$;
Distribution $D$ on $[2k]$: support size $\leq k/2$ or $\geq 3k/2$? Hard by [VV11].

**Reduction:**

1. Embed $D$ in $[n]$, where $n = 1000k$;
2. Randomly permute the support with an u.a.r. $\sigma \in S_n$;
3. Use a tester for $k$-histograms on the resulting distribution $D'$:

   - If $\text{supp}(D) = k$, then $D'$ is a $k$-histogram with probability 1;
   - If $\text{supp}(D) = 3k$, then $D'$ is not a $\ell$-histogram for any $\ell < 1$ with probability $2/3$ (and $D' \Omega(1)$-far from any $k$-histogram).

**Upshot:** Can use a tester for $k$-histograms to solve the support size estimation problem! But this requires $\~\Omega(k)$ samples.
Distribution $D$ on $[2k]$: support size $\leq k/2$ or $\geq 3k/2$? Hard by [VV11].

**Reduction:**

1. Embed $D$ into $[n]$, where $n = 1000k$;
2. Randomly permute the support with a u.a.r. $\sigma \in S_n$;
3. Use a tester for $k$-histograms on the resulting distribution $D'$:
   - If $\text{supp}(D) \leq k/2$, then $D'$ is a $k$-histogram with probability 1;
   - If $\text{supp}(D) \geq 3k/2$, then $D'$ is not a $\ell$-histogram for any $\ell < 1.1k$ with probability $2/3$;
Distribution $D$ on $[2k]$: support size $\leq k/2$ or $\geq 3k/2$? Hard by [VV11].

Reduction:

1. Embed $D$ in $[n]$, where $n = 1000k$;
2. Randomly permute the support with an u.a.r. $\sigma \in S_n$;
3. Use a tester for $k$-histograms on the resulting distribution $D'$:
   - If $\text{supp}(D) \leq k/2$, then $D'$ is a $k$-histogram with probability 1;
   - If $\text{supp}(D) \geq 3k/2$, then $D'$ is not a $\ell$-histogram for any $\ell < 1.1k$ with probability $2/3$; (and $D' \Omega(1)$-far from any $k$-histogram)
Distribution D on \([2k]\): support size \(\leq k/2\) or \(\geq 3k/2\)? Hard by [VV11].

**Reduction:**

1. Embed D in \([n]\), where \(n = 1000k\);
2. Randomly permute the support with an u.a.r. \(\sigma \in S_n\);
3. Use a tester for \(k\)-histograms on the resulting distribution \(D'\):
   - If \(\text{supp}(D) \leq k/2\), then \(D'\) is a \(k\)-histogram with probability 1;
   - If \(\text{supp}(D) \geq 3k/2\), then \(D'\) is not a \(\ell\)-histogram for any \(\ell < 1.1k\) with probability 2/3; (and \(D' \Omega(1)\)-far from any \(k\)-histogram)

**Upshot**

Can use a tester for \(k\)-histograms to solve the support size estimation problem!
Distribution $D$ on $[2k]$: support size $\leq k/2$ or $\geq 3k/2$? Hard by [VV11].

**Reduction:**

1. Embed $D$ in $[n]$, where $n = 1000k$;
2. Randomly permute the support with an u.a.r. $\sigma \in S_n$;
3. Use a tester for $k$-histograms on the resulting distribution $D'$:
   - If $\text{supp}(D) \leq k/2$, then $D'$ is a $k$-histogram with probability 1;
   - If $\text{supp}(D) \geq 3k/2$, then $D'$ is not a $\ell$-histogram for any $\ell < 1.1k$ with probability $2/3$; (and $D' \Omega(1)$-far from any $k$-histogram)

**Upshot**

Can use a tester for $k$-histograms to solve the support size estimation problem! But this requires $\tilde{\Omega}(k)$ samples.
Questions?


Piotr Indyk, Reut Levi, and Ronitt Rubinfeld. 

Reut Levi, Dana Ron, and Ronitt Rubinfeld. 
Testing properties of collections of distributions. 

Liam Paninski. 
A coincidence-based test for uniformity given very sparsely sampled discrete data. 

Paul Valiant. 
Testing symmetric properties of distributions. 

Gregory Valiant and Paul Valiant. 
A CLT and tight lower bounds for estimating entropy. 
Electronic Colloquium on Computational Complexity (ECCC), 17:179, 2010.

Gregory Valiant and Paul Valiant. 
Estimating the unseen: A sublinear-sample canonical estimator of distributions. 
Electronic Colloquium on Computational Complexity (ECCC), 17:180, 2010.

Gregory Valiant and Paul Valiant. 
The power of linear estimators. 
See also [VV10a] and [VV10b].

Gregory Valiant and Paul Valiant. 
An automatic inequality prover and instance optimal identity testing. 