Like a Needle in a Haystack: Rates and Algorithms for Group Testing

Leonardo Baldassini

Joint work with Oliver Johnson, Matthew Aldridge and Karen Gunderson

19 December 2014

University of BRISTOL

Warm-up examples

- Molecular biology (DNA screening)
- Recommender systems

Warm-up examples

- Molecular biology (DNA screening)
- Recommender systems

- Spectrum sensing
- High-throughput screening techniques
- Network tomography
- Cryptography and cyber security

DNA screening

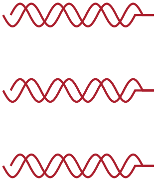
Problem

Find a specific, rare sequence of nucleotides among many DNA samples

DNA screening

Problem

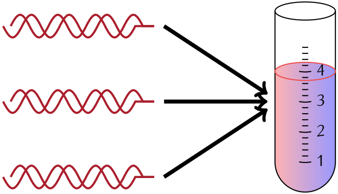
Find a specific, rare sequence of nucleotides among many DNA samples



DNA screening

Problem

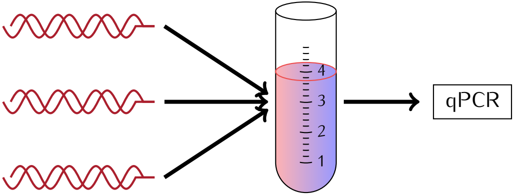
Find a specific, rare sequence of nucleotides among many DNA samples



DNA screening

Problem

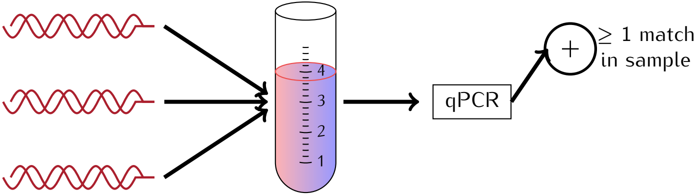
Find a specific, rare sequence of nucleotides among many DNA samples



DNA screening

Problem

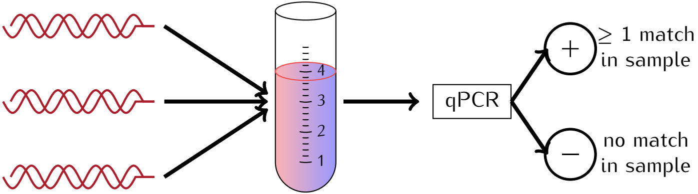
Find a specific, rare sequence of nucleotides among many DNA samples



DNA screening

Problem

Find a specific, rare sequence of nucleotides among many DNA samples



Recommender systems

Problem

Suggest songs to a user based on his past preferences.

Recommender systems

Problem

Suggest songs to a user based on his past preferences.

User selects
song

Recommender systems

Problem

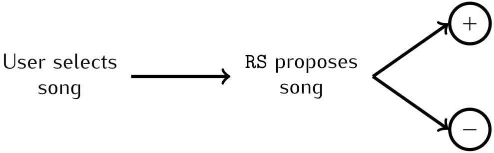
Suggest songs to a user based on his past preferences.

User selects song ⟶ RS proposes song

Recommender systems

Problem

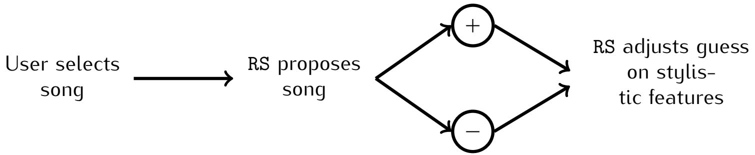
Suggest songs to a user based on his past preferences.



Recommender systems

Problem

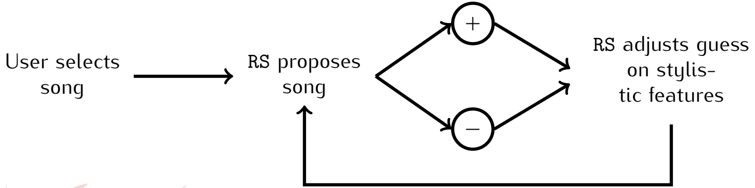
Suggest songs to a user based on his past preferences.



Recommender systems

Problem

Suggest songs to a user based on his past preferences.



What do they have in common?

Common features

What do they have in common?

COMMON FEATURES

- Search for *sparse* property

What do they have in common?

Common features

- Search for *sparse* property
- Property can be tested on groups

What do they have in common?

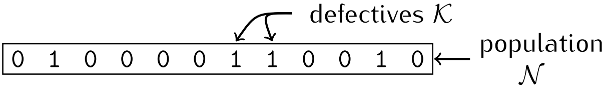
Common features

- Search for *sparse* property
- Property can be tested on groups

Group testing

Search methods to recover a sparse subset of items from a population that share a feature which can be detected on groups

Boolean group testing



Boolean group testing

defectives $\mathcal{K}$

population $\mathcal{N}$

0	1	0	0	0	0	1	1	0	0	1	0

0	0	0	0	0	0	1	1	0	0	1	0
0	0	0	0	1	0	1	0	1	1	0	0
0	1	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	1	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0

test design
X

Boolean group testing

defectives $\mathcal{K}$

population $\mathcal{N}$

0	1	0	0	0	0	1	1	0	0	1	0

test design X

0	0	0	0	0	0	1	1	0	0	1	0
0	0	0	0	1	0	1	0	1	1	0	0
0	1	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	1	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0

test oucomes $\mathbf{y}$

1
1
1
1
0
0
0
0
0
0
0

Boolean group testing

defectives $\mathcal{K}$

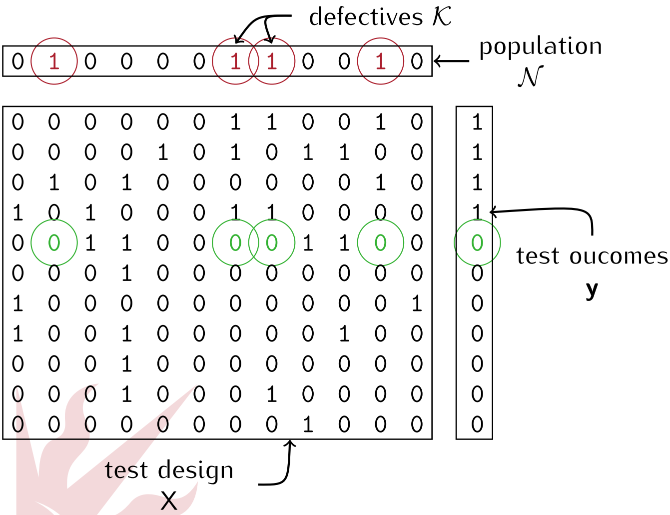
population $\mathcal{N}$

0	1	0	0	0	0	1	1	0	0	1	0

test design X / test oucomes **y**

												y
0	0	0	0	0	0	1	1	0	0	1	0	1
0	0	0	0	1	0	1	0	1	1	0	0	1
0	1	0	1	0	0	0	0	0	0	1	0	1
1	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	0	0	0	1	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0

Boolean group testing



Boolean group testing

defectives $\mathcal{K}$

population $\mathcal{N}$

0	1	0	0	0	0	1	1	0	0	1	0

test design X

0	0	0	0	0	0	1	1	0	0	1	0
0	0	0	0	1	0	1	0	1	1	0	0
0	1	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	1	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0

test oucomes **y**

1
1
1
1
0
0
0
0
0
0
0

$|\mathcal{N}| = N, |\mathcal{K}| = K$

$K = N^{1-\beta}, \ \beta > 0$

Boolean group testing

defectives $\mathcal{K}$

population $\mathcal{N}$

0	1	0	0	0	0	1	1	0	0	1	0

$|\mathcal{N}| = N, |\mathcal{K}| = K$

$K = N^{1-\beta}, \ \beta > 0$

test design $\mathbf{X}$

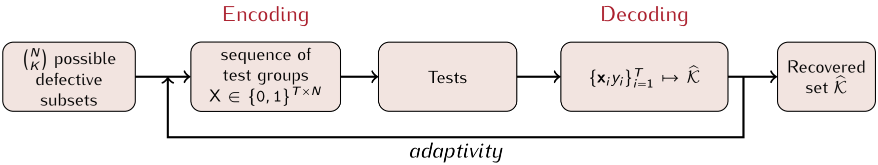
0	0	0	0	0	0	1	1	0	0	1	0
0	0	0	0	1	0	1	0	1	1	0	0
0	1	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	1	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0

test oucomes $\mathbf{y}$

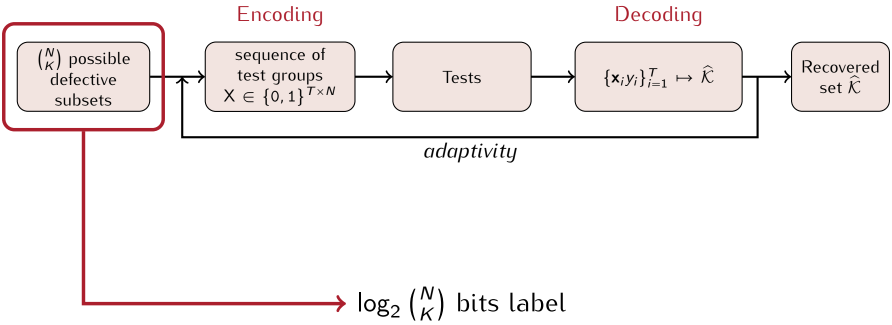
1
1
1
1
0
0
0
0
0
0
0

particularly good for non-adaptive algorithms

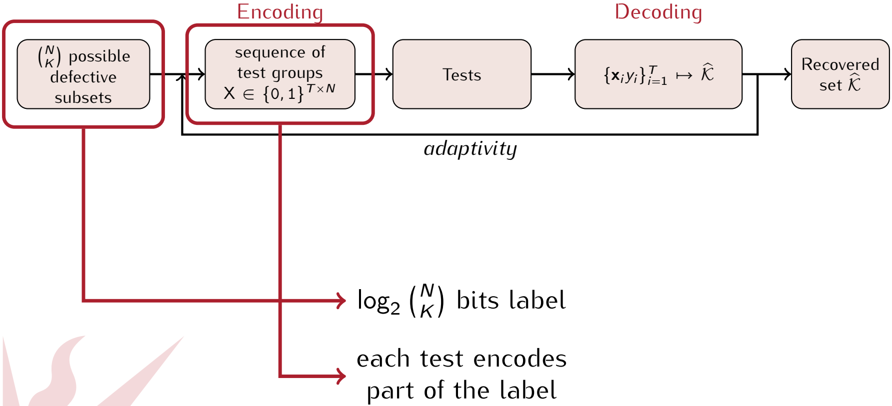
Anatomy of an algorithm



Anatomy of an algorithm



Anatomy of an algorithm



Rate of group testing (algorithms)

$$R_{\mathtt{A}} = \frac{\log_2 \binom{N}{K}}{T_{\mathtt{A}}} \text{ bits per test}$$

Rate of group testing (algorithms)

$$R_{\mathtt{A}} = \frac{\log_2 \binom{N}{K}}{T_{\mathtt{A}}} \text{ bits per test}$$

$R_{\mathtt{A}}$ measures how much we learn with each test, on average.

Rate of group testing (algorithms)

$$R_{\mathtt{A}} = \frac{\log_2 \binom{N}{K}}{T_{\mathtt{A}}} \text{ bits per test}$$

$R_{\mathtt{A}}$ measures how much we learn with each test, on average.
$R^*_{\mathtt{A}}(\beta)$ supremum of rates for algorithm A.

Rate of group testing (algorithms)

$$R_{\mathtt{A}} = \frac{\log_2 \binom{N}{K}}{T_{\mathtt{A}}} \text{ bits per test}$$

$R_{\mathtt{A}}$ measures how much we learn with each test, on average.
$R^*_{\mathtt{A}}(\beta)$ supremum of rates for algorithm A. (Yes, it's bounded).

Why do we like rates?

Universal upper bound for noiseless GT

Theorem (Aldridge, B., Johnson, 2013)

The probability of success of any algorithm $\mathcal{A}$ can be upper-bounded as

$$\mathbb{P}(success) \leq \frac{2^{T_{\mathcal{A}}}}{\binom{N}{K}} \ .$$

Two important consequences

$$\mathbb{P}(\text{success}) \leq \frac{2^{T_A}}{\binom{N}{K}}$$

Two important consequences

$$\mathbb{P}(\text{success}) \leq \frac{2^{T_A}}{\binom{N}{K}}$$

- Successful algorithms use $T \geq \log_2 \binom{N}{K}$ tests
  optimal algorithms use $T = c \log_2 \binom{N}{K}$ tests

Two important consequences

$$\mathbb{P}(\text{success}) \leq \frac{2^{T_{\mathtt{A}}}}{\binom{N}{K}}$$

- Successful algorithms use $T \geq \log_2 \binom{N}{K}$ tests
  optimal algorithms use $T = c \log_2 \binom{N}{K}$ tests
  - $R = \dfrac{\log_2 \binom{N}{K}}{T}$ “ranks” optimal algorithms

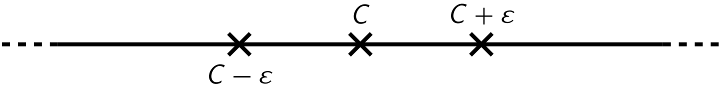
Two important consequences

$$\mathbb{P}(\text{success}) \le \frac{2^{T_{\mathtt{A}}}}{\binom{N}{K}}$$

- Successful algorithms use $T \geq \log_2 \binom{N}{K}$ tests
  optimal algorithms use $T = c \log_2 \binom{N}{K}$ tests
  - $R = \dfrac{\log_2 \binom{N}{K}}{T}$ "ranks" optimal algorithms
- Successful algorithms have $R^*_{\mathtt{A}}(\beta) \leq 1$

Capacity of GT

Capacity of GT



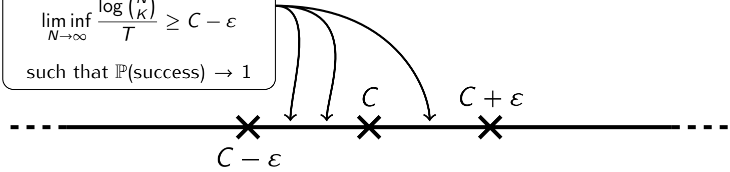
Capacity of GT

Direct part

There exists an algorithm with

$$\liminf_{N\to\infty} \frac{\log \binom{N}{K}}{T} \geq C - \varepsilon$$

such that $\mathbb{P}(\text{success}) \to 1$



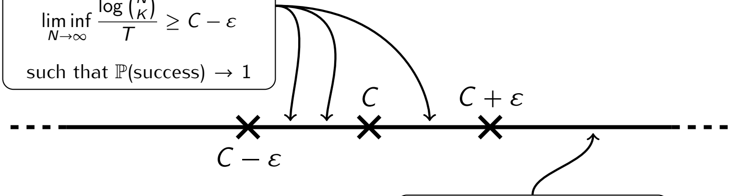
Capacity of GT

Direct part

There exists an algorithm with

$$\liminf_{N\to\infty} \frac{\log \binom{N}{K}}{T} \geq C - \varepsilon$$

such that $\mathbb{P}(\text{success}) \to 1$



Any algorithm with

$$\liminf_{N\to\infty} \frac{\log \binom{N}{K}}{T} \geq C + \varepsilon$$

has $\mathbb{P}(\text{success}) < 1 - \eta$, $\eta > 0$

Converse part

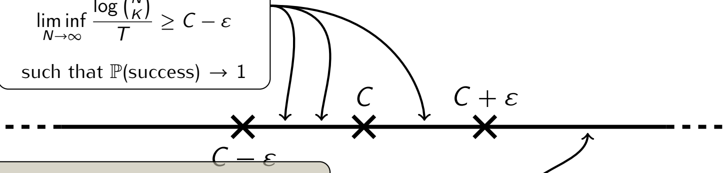
Capacity of GT

Direct part

There exists an algorithm with

$$\liminf_{N\to\infty} \frac{\log \binom{N}{K}}{T} \geq C - \varepsilon$$

such that $\mathbb{P}(\text{success}) \to 1$



- C is inherent in the GT model

Any algorithm with

$$\liminf_{N\to\infty} \frac{\log \binom{N}{K}}{T} \geq C + \varepsilon$$

has $\mathbb{P}(\text{success}) < 1 - \eta$, $\eta > 0$

Converse part

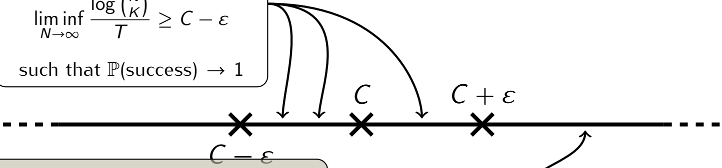
Capacity of GT

Direct part

There exists an algorithm with

$$\liminf_{N\to\infty} \frac{\log \binom{N}{K}}{T} \geq C - \varepsilon$$

such that $\mathbb{P}(\text{success}) \to 1$



- C is inherent in the GT model
- C measures the “hardness” of a GT problem

Any algorithm with

$$\liminf_{N\to\infty} \frac{\log \binom{N}{K}}{T} \geq C + \varepsilon$$

has $\mathbb{P}(\text{success}) < 1 - \eta$, $\eta > 0$

Converse part

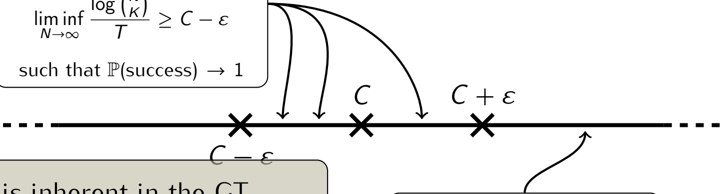
Capacity of GT

Direct part

There exists an algorithm with

$$\liminf_{N\to\infty} \frac{\log\binom{N}{K}}{T} \geq C - \varepsilon$$

such that $\mathbb{P}(\text{success}) \to 1$



- C is inherent in the GT model
- C measures the "hardness" of a GT problem
- C quantifies an efficiency/effectiveness trade-off

Any algorithm with

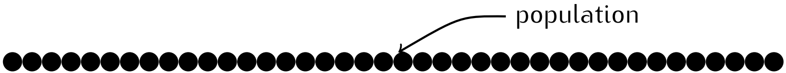
$$\liminf_{N\to\infty} \frac{\log\binom{N}{K}}{T} \geq C + \varepsilon$$

has $\mathbb{P}(\text{success}) < 1 - \eta$, $\eta > 0$

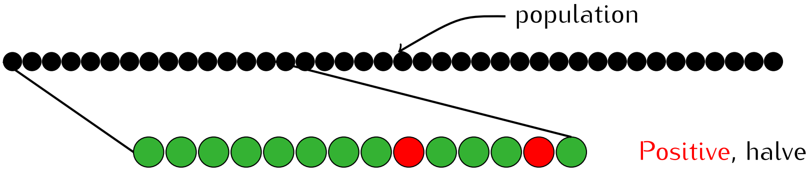
Converse part

Can we get to $R = 1$?

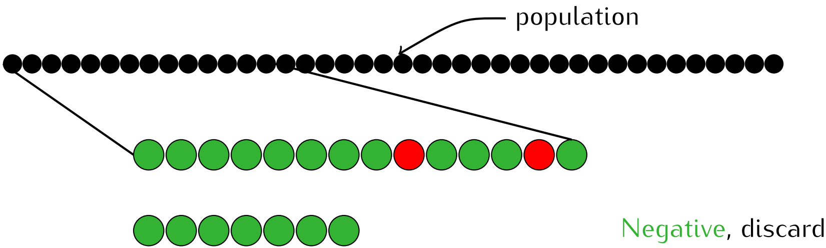
Can we get to $R = 1$?



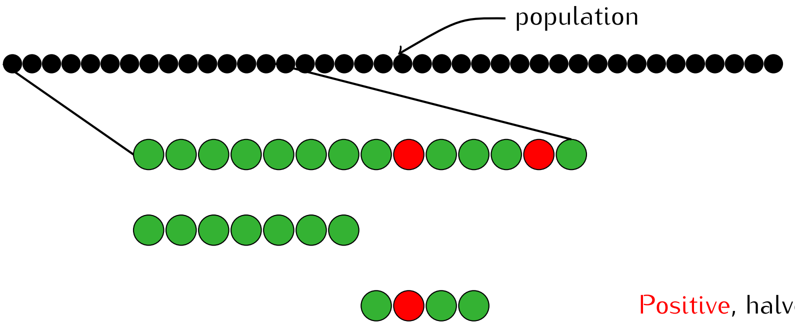
Can we get to $R = 1$?



Can we get to $R = 1$?

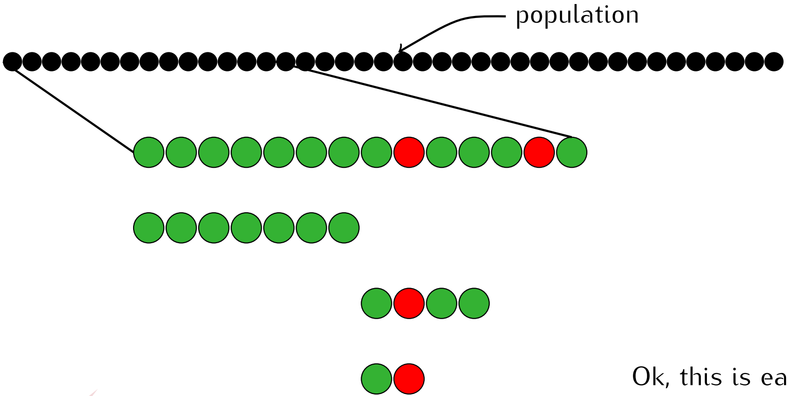


Can we get to $R = 1$?



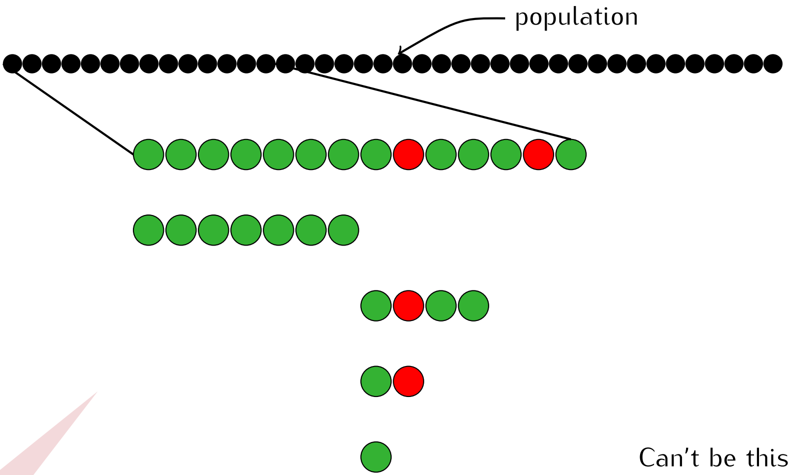
Positive, halve

Can we get to $R = 1$?



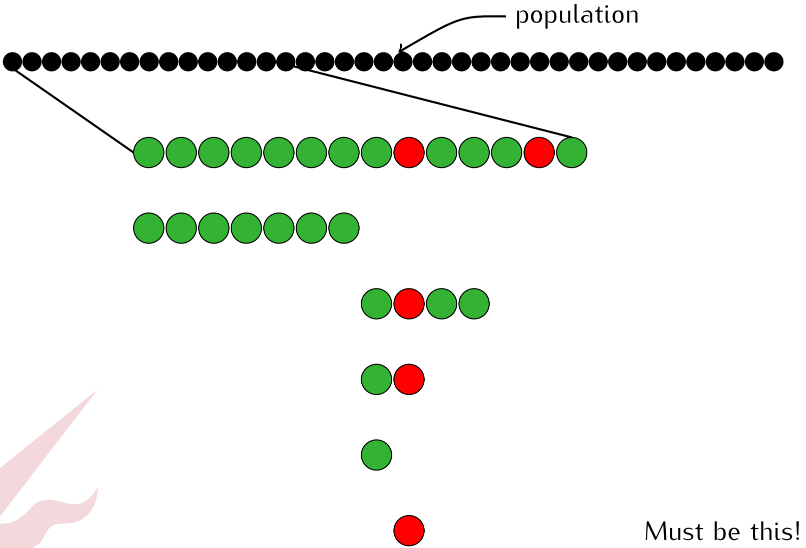
Ok, this is easy

Can we get to $R = 1$?

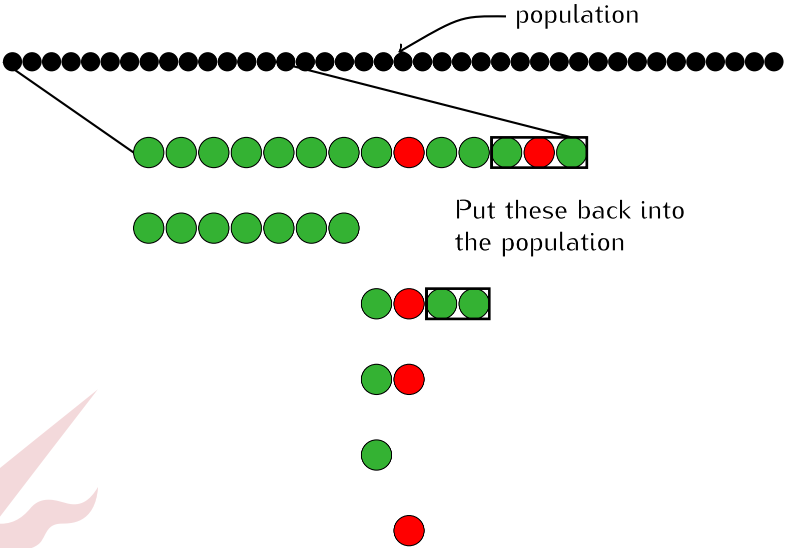


Can't be this...

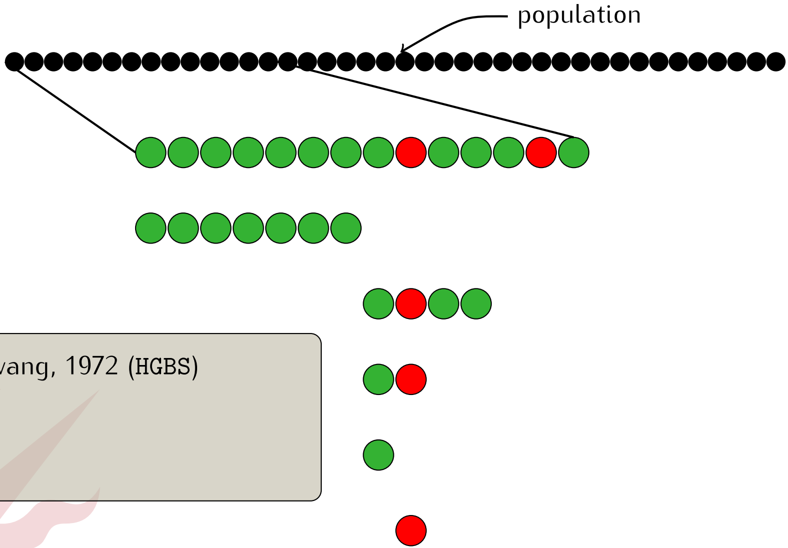
Can we get to $R = 1$?



Can we get to $R = 1$?

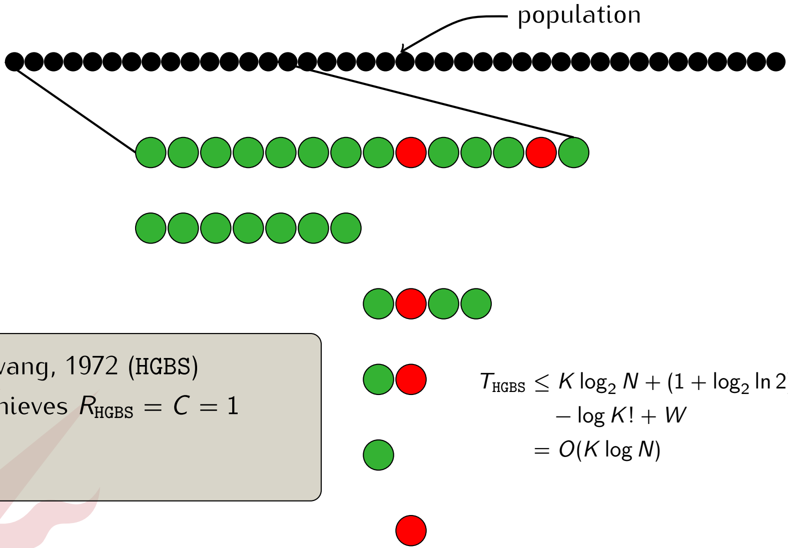


Can we get to $R = 1$?



- Hwang, 1972 (`HGBS`)

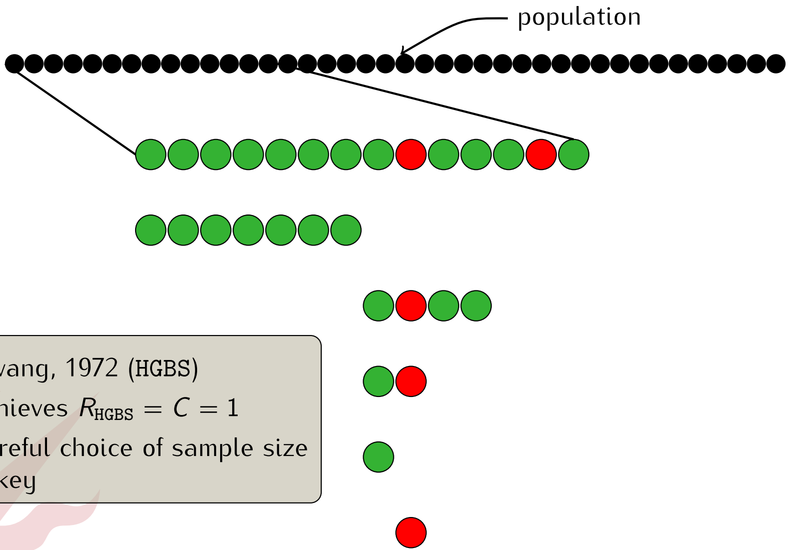
Can we get to $R = 1$?



- Hwang, 1972 (HGBS)
- Achieves $R_{\text{HGBS}} = C = 1$

$$\begin{aligned} T_{\text{HGBS}} &\leq K \log_2 N + (1 + \log_2 \ln 2)K \\ &\quad - \log K! + W \\ &= O(K \log N) \end{aligned}$$

Can we get to $R = 1$?



- Hwang, 1972 (HGBS)
- Achieves $R_{\texttt{HGBS}} = C = 1$
- Careful choice of sample size is key

Can we get there...non-adaptively?

Can we get there...non-adaptively?

- Maybe

Can we get there...non-adaptively?

- Maybe
- What we did:

Can we get there...non-adaptively?

- Maybe
- What we did:
  - Introduced and studied `DD` (Definite Defectives) and `SSS` (Smallest Satisfying Set)
  - ...hence Bernoulli sampling, $x_{ij} \sim \text{Bern}(p)$, $p = \frac{1}{K+1}$
  - Showed limitation of Bernoulli-based algorithms

DD: Definite Defectives

DD: Definite Defectives

0	1	0	0	0	0	1	1	0	0	1	0

0	0	1	0	0	0	1	1	0	0	1	0
0	0	0	0	1	0	1	0	1	1	0	0
1	1	0	1	0	1	0	0	0	0	1	0
1	0	1	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0	1	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	1	0	0	0

1
1
1
1
0
0
0
0
0
0
0

1. X Bernoulli design.

DD: Definite Defectives

0	1	0	0	0	0	1	1	0	0	1	0

0	0	1	0	0	0	1	1	0	0	1	0	1
0	0	0	0	1	0	1	0	1	1	0	0	1
1	1	0	1	0	1	0	0	0	0	1	0	1
1	0	1	0	0	0	1	0	0	0	0	0	1
0	0	1	1	0	0	0	0	1	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	1	0	0	0	0

1. X Bernoulli design.
2. Look at negative tests...

DD: Definite Defectives

0	1	0	0	0	0	1	1	0	0	1	0

0	0	1	0	0	0	1	1	0	0	1	0	1
0	0	0	0	1	0	1	0	1	1	0	0	1
1	1	0	1	0	1	0	0	0	0	1	0	1
1	0	1	0	0	0	1	0	0	0	0	0	1
0	0	1	1	0	0	0	0	1	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	1	0	0	0	0

1. X Bernoulli design.
2. Look at negative tests…
3. … classify items therein as non-def. …

DD: Definite Defectives

0	1	0	0	0	0	1	1	0	0	1	0

0	0		0	0	0	1	1	0	0	1	0
0	0	0	0	1	0	1	0			0	0
	1	0		0		0	0	0	0	1	0
	0		0	0	0	1	0	0	0	0	0

1
1
1
1
0
0
0
0
0
0
0

1. X Bernoulli design.
2. Look at negative tests...
3. ... classify items therein as non-def. ...
4. ... and remove them from X.

DD: Definite Defectives

0	1	0	0	0	0	1	1	0	0	1	0

0	0		0	0	0	1	1	0	0	1	0	1
0	0	0	0	1	0	1	0			0	0	1
	1	0		0		0	0	0	0	1	0	1
	0		0	0	0	1	0	0	0	0	0	1
												0
												0
												0
												0
												0
												0
												0

1. X Bernoulli design.
2. Look at negative tests...
3. ... classify items therein as non-def. ...
4. ... and remove them from X.
5. Look at positive tests with 1! unclassified item:

DD: Definite Defectives

0	1	0	0	0	0	1	1	0	0	1	0

0	0		0	0	0	1	1	0	0	1	0	1
0	0	0	0	1	0	1	0			0	0	1
	1	0		0		0	0	0	0	1	0	1
	0		0	0	0	1	0	0	0	0	0	1
												0
												0
												0
												0
												0
												0
												0

1. X Bernoulli design.
2. Look at negative tests. . .
3. . . . classify items therein as non-def. . . .
4. . . . and remove them from X.
5. Look at positive tests with 1! unclassified item:
6. That's definite defective!

DD: Definite Defectives

0	1	0	0	0	0	1	1	0	0	1	0

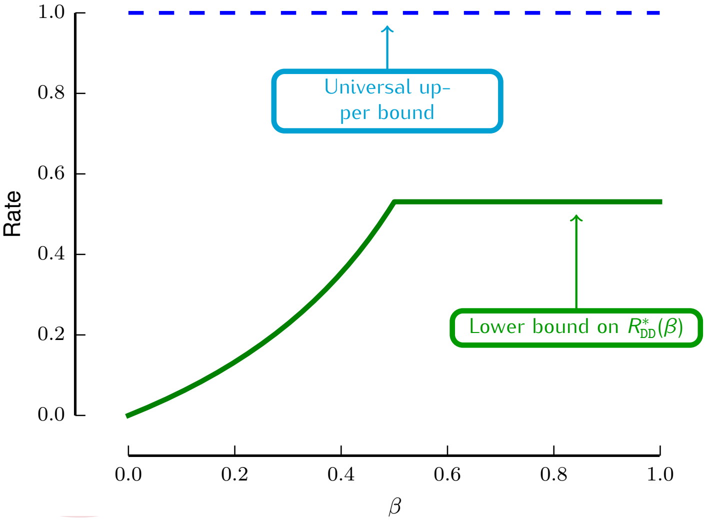
0	0		0	0	0	1	1	0	0	1	0	1
0	0	0	0	1	0	1	0			0	0	1
	1	0		0		0	0	0	0	1	0	1
	0		0	0	0	1	0	0	0	0	0	1
												0
												0
												0
												0
												0
												0
												0

1. X Bernoulli design.
2. Look at negative tests. . .
3. . . . classify items therein as non-def. . . .
4. . . . and remove them from X.
5. Look at positive tests with 1! unclassified item:
6. That's definite defective!

$$T_{\mathrm{DD}} \leq \max\{\beta, 1-\beta\}\, \mathrm{e} K \ln N$$

$$R^*_{\mathrm{DD}}(\beta) \geq \frac{1}{\mathrm{e}\ln 2} \min\left\{1, \frac{\beta}{1-\beta}\right\}$$

What $R^*_{\mathtt{DD}}(\beta)$ looks like



Computing $R^*_{\mathtt{DD}}(\beta)$: core of the proof - construction

Computing $R^*_{\mathtt{DD}}(\beta)$: core of the proof - construction

- $\mathcal{PD} =$ {items not in negative tests}

Computing $R^*_{\mathtt{DD}}(\beta)$: core of the proof - construction

- $\mathcal{PD} =$ {items not in negative tests}
- $|\mathcal{PD}| = K + G$, G *intruding non-defectives*

Computing $R^*_{\mathtt{DD}}(\beta)$: core of the proof - construction

- $\mathcal{PD} =$ {items not in negative tests}
- $|\mathcal{PD}| = K + G$, G *intruding non-defectives*

For every defective $i \in \mathcal{K}$:
$L_i = \#$ tests with i and no item from $\mathcal{PD}$

Computing $R^*_{\mathtt{DD}}(\beta)$: core of the proof - construction

- $\mathcal{PD} = \{\text{items not in negative tests}\}$
- $|\mathcal{PD}| = K + G$, G *intruding non-defectives*

For every defective $i \in \mathcal{K}$:
$L_i = \#$ tests with i and no item from $\mathcal{PD}$

$\mathbb{P}(\text{success}) = \mathbb{P}\left(\bigcap_{i\in\mathcal{K}}\{L_i \neq 0\}\right)$

Computing $R^*_{\mathrm{DD}}(\beta)$: core of the proof - construction

- $\mathcal{PD} = \{\text{items not in negative tests}\}$
- $|\mathcal{PD}| = K + G$, G *intruding non-defectives*

For every defective $i \in \mathcal{K}$:
$L_i = \#$ tests with i and no item from $\mathcal{PD}$

$\mathbb{P}(\text{success}) = \mathbb{P}\left(\bigcap_{i \in \mathcal{K}} \{L_i \neq 0\}\right)$

Bad news We can't control G

Computing $R^*_{\mathtt{DD}}(\beta)$: core of the proof - construction

- $\mathcal{PD} =$ {items not in negative tests}
- $|\mathcal{PD}| = K + G$, G *intruding non-defectives*

For every defective $i \in \mathcal{K}$: $L_i = \#$ tests with i and no item from $\mathcal{PD}$

$\mathbb{P}(\text{success}) = \mathbb{P}\left(\bigcap_{i \in \mathcal{K}} \{L_i \neq 0\}\right)$

Bad news We can't control G

Good news We can control $G \mid M_0$, $M_0 = \#$ negative tests.

Computing $R^*_{\mathrm{DD}}(\beta)$: core of the proof - construction

- $\mathcal{PD} = \{\text{items not in negative tests}\}$
- $|\mathcal{PD}| = K + G$, G *intruding non-defectives*

For every defective $i \in \mathcal{K}$:
$L_i = \#$ tests with i and no item from $\mathcal{PD}$

$\mathbb{P}(\text{success}) = \mathbb{P}\left(\bigcap_{i \in \mathcal{K}} \{L_i \neq 0\}\right)$

Bad news: We can't control G

Good news: We can control $G \mid M_0$, $M_0 = \#$ negative tests.

$$\mathbb{P}(\text{success}) = \sum_{m_0=0}^{T} \binom{T}{m_0} (1-p)^{Km_0} \left(1-(1-p)^K\right)^{T-m_0}$$
$$\times \sum_{g=0}^{N-K} \binom{N-K}{g} (1-p)^{gm_0} \left(1-(1-p)^{m_0}\right)^{N-K-g}$$
$$\times \, \mathbb{P}(\text{success} \mid M_0 = m_0, G = g)$$

Computing $R^*_{\mathrm{DD}}(\beta)$: core of the proof – construction

- $\mathcal{PD} = \{\text{items not in negative tests}\}$
- $|\mathcal{PD}| = K + G$, G *intruding non-defectives*

For every defective $i \in \mathcal{K}$:
L_i = # tests with i and no item from $\mathcal{PD}$

$\mathbb{P}(\text{success}) = \mathbb{P}\left(\bigcap_{i\in\mathcal{K}}\{L_i \neq 0\}\right)$

Bad news	We can't control G
Good news	We can control $G \mid M_0$, M_0 = # negative tests.

$$\mathbb{P}(\text{success}) = \sum_{m_0=0}^{T} \binom{T}{m_0} (1-p)^{Km_0} \left(1-(1-p)^K\right)^{T-m_0}$$

$$\times \sum_{g=0}^{N-K} \binom{N-K}{g} (1-p)^{gm_0} \left(1-(1-p)^{m_0}\right)^{N-K-g}$$

$$\times\, \mathbb{P}(\text{success} \mid M_0 = m_0, G = g)$$

there's a sum in here, too!

Computing $R^*_{\mathtt{DD}}(\beta)$: core of the proof - concentration

Computing $R^*_{\mathtt{DD}}(\beta)$: core of the proof - concentration

$$\sum_{g=0}^{N-K} \binom{N-K}{g} (1-p)^{g m_0} \left(1-(1-p)^{m_0}\right)^{N-K-g}$$
$$\times \mathbb{P}(\text{success} \mid M_0 = m_0, G = g)$$
$$\geq \max\{0, 1 - K \exp(\Theta(T, m_0))\}$$

Computing $R^*_{\mathtt{DD}}(\beta)$: core of the proof - concentration

$$\sum_{g=0}^{N-K} \binom{N-K}{g} (1-p)^{g m_0} \left(1-(1-p)^{m_0}\right)^{N-K-g} \times \mathbb{P}(\text{success} \mid M_0 = m_0, G = g) \geq \max\{0, 1 - K\exp(\Theta(T, m_0))\}$$

- $M_0 \in (\mathbb{E}M_0 - \varepsilon, \mathbb{E}M_0 + \varepsilon)$ w.h.p.

Computing $R^*_{\mathtt{DD}}(\beta)$: core of the proof - concentration

$$\sum_{g=0}^{N-K} \binom{N-K}{g} (1-p)^{g m_0} \left(1-(1-p)^{m_0}\right)^{N-K-g} \times \mathbb{P}(\text{success} \mid M_0 = m_0, G = g) \geq \max\{0, 1 - K \exp(\Theta(T, m_0))\}$$

- $M_0 \in (\mathbb{E}M_0 - \varepsilon, \mathbb{E}M_0 + \varepsilon)$ w.h.p.
- $\Theta(T, m_0) \leq -(\max\{\beta, 1-\beta\}) \ln N$ for M_0 close to $\mathbb{E}M_0$

Computing $R^*_{\mathrm{DD}}(\beta)$: core of the proof – concentration

$$\sum_{g=0}^{N-K} \binom{N-K}{g} (1-p)^{g m_0} \left(1-(1-p)^{m_0}\right)^{N-K-g} \times \mathbb{P}(\text{success} \mid M_0 = m_0, G = g) \geq \max\{0, 1 - K\exp(\Theta(T, m_0))\}$$

- $M_0 \in (\mathbb{E}M_0 - \varepsilon, \mathbb{E}M_0 + \varepsilon)$ w.h.p.
- $\Theta(T, m_0) \leq -(\max\{\beta, 1-\beta\}) \ln N$ for M_0 close to $\mathbb{E}M_0$

$$\mathbb{P}(\text{success}) \geq \mathbb{P}\left(T\left(1-p\right)^K - \varepsilon/\mathrm{e}\right) \leq M_0 \leq T\left((1-p)^K + \varepsilon/\mathrm{e}\right)\right)(1-N^{-\delta})$$

SSS: Smallest Satisfying Set

Group testing as LP:

SSS: Smallest Satisfying Set

Group testing as LP:

$$
\begin{aligned}
\text{minimize} \quad & \mathbf{1}^\top \mathbf{z} \\
\text{subject to} \quad & \mathbf{x}_t \cdot \mathbf{z} = 0 \quad \text{for } t \text{ with } y_t = 0 \\
& \mathbf{x}_t \cdot \mathbf{z} \geq 1 \quad \text{for } t \text{ with } y_t = 1 \\
& (\mathbf{1}^\top \mathbf{z} \geq K) \\
& \mathbf{z} \in \{0, 1\}^N
\end{aligned}
$$

SSS: Smallest Satisfying Set

Group testing as LP:

$$
\begin{aligned}
\text{minimize} \quad & \mathbf{1}^\top \mathbf{z} \\
\text{subject to} \quad & \mathbf{x}_t \cdot \mathbf{z} = 0 \quad \text{for } t \text{ with } y_t = 0 \\
& \mathbf{x}_t \cdot \mathbf{z} \geq 1 \quad \text{for } t \text{ with } y_t = 1 \\
& (\mathbf{1}^\top \mathbf{z} \geq K) \\
& \mathbf{z} \in \{0,1\}^N
\end{aligned}
$$

- SSS brute-force searches for a satisfying $\mathcal{K}$

SSS: Smallest Satisfying Set

Group testing as LP:

$$\begin{aligned} \text{minimize} \quad & \mathbf{1}^\top \mathbf{z} \\ \text{subject to} \quad & \mathbf{x}_t \cdot \mathbf{z} = 0 \quad \text{for } t \text{ with } y_t = 0 \\ & \mathbf{x}_t \cdot \mathbf{z} \geq 1 \quad \text{for } t \text{ with } y_t = 1 \\ & (\mathbf{1}^\top \mathbf{z} \geq K) \\ & \mathbf{z} \in \{0,1\}^N \end{aligned}$$

- SSS brute-force searches for a satisfying $\mathcal{K}$
- Best possible algorithm (sample-complexity-wise)

SSS: Smallest Satisfying Set

Group testing as LP:

$$\begin{aligned} \text{minimize} \quad & \mathbf{1}^\top \mathbf{z} \\ \text{subject to} \quad & \mathbf{x}_t \cdot \mathbf{z} = 0 \quad \text{for } t \text{ with } y_t = 0 \\ & \mathbf{x}_t \cdot \mathbf{z} \geq 1 \quad \text{for } t \text{ with } y_t = 1 \\ & (\mathbf{1}^\top \mathbf{z} \geq K) \\ & \mathbf{z} \in \{0, 1\}^N \end{aligned}$$

- SSS brute-force searches for a satisfying $\mathcal{K}$
- Best possible algorithm (sample-complexity-wise)
- Infeasible, but benchmark

SSS: Smallest Satisfying Set

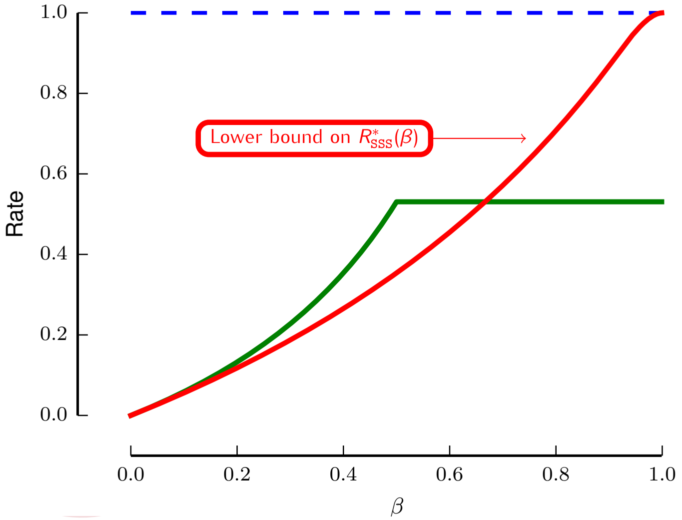
Group testing as LP:

$$\begin{aligned} \text{minimize} \quad & \mathbf{1}^\top \mathbf{z} \\ \text{subject to} \quad & \mathbf{x}_t \cdot \mathbf{z} = 0 \quad \text{for } t \text{ with } y_t = 0 \\ & \mathbf{x}_t \cdot \mathbf{z} \geq 1 \quad \text{for } t \text{ with } y_t = 1 \\ & (\mathbf{1}^\top \mathbf{z} \geq K) \\ & \mathbf{z} \in \{0,1\}^N \end{aligned}$$

- SSS brute-force searches for a satisfying $\mathcal{K}$
- Best possible algorithm (sample-complexity-wise)
- Infeasible, but benchmark

$$R^*_{\text{SSS}}(\beta) \geq \frac{1}{\ln 2} \max_{\alpha \in [\ln 2, 1]} \min \left\{ 2\alpha e^{-\alpha} \frac{\beta}{2-\beta}, -\ln\left(1 - 2e^{-\alpha} + 2e^{-2\alpha}\right) \right\} .$$

What $R^*_{\text{SSS}}(\beta)$ looks like



Separability

SSS may fail if more than one subset satisfies constraints

Separability

SSS may fail if more than one subset satisfies constraints ⟶ Question: Can we enforce uniqueness in X ?

Separability

SSS may fail if more than one subset satisfies constraints $\longrightarrow$

Question:
Can we enforce uniqueness in X ?

Definition

X is K-separable if, for any K-subsets $\mathcal{L}, \mathcal{M} \subset \{1, \ldots, N\}$, it is

$$\bigvee_{i \in \mathcal{M}} \mathbf{x}^{(i)} \neq \bigvee_{j \in \mathcal{L}} \mathbf{x}^{(j)}$$

Separability

SSS may fail if more than one subset satisfies constraints ⟶ Question: Can we enforce uniqueness in X ?

Definition

X is K-separable if, for any K-subsets $\mathcal{L}, \mathcal{M} \subset \{1, \ldots, N\}$, it is

$$\bigvee_{i \in \mathcal{M}} \mathbf{x}^{(i)} \neq \bigvee_{j \in \mathcal{L}} \mathbf{x}^{(j)}$$

Notice that $\mathbf{y} = \bigvee_{\mathcal{K}} \mathbf{x}^{(i)}$.

Almost separability

Building separable X requires
$T = \Omega(K^2 \log N)$ tests

Almost separability

Building separable X requires
$T = \Omega(K^2 \log N)$ tests

$\longrightarrow$

Idea:
What if we relax separability?

Almost separability

Building separable X requires
$T = \Omega(K^2 \log N)$ tests ⟶

Idea:
What if we relax separability?

Definition

X is ε-almost K-separable if there are at most $\varepsilon\binom{N}{K}$ K-subsets that break separability.

Almost separability

Building separable X requires $T = \Omega(K^2 \log N)$ tests ⟶ Idea: What if we relax separability?

Definition

X is ε-almost K-separable if there are at most $\varepsilon\binom{N}{K}$ K-subsets that break separability.

- Almost-separable matrices exist
- Need $T = O(K \log N)$ tests to get one
- A Bernoulli test design is almost-separable w.h.p. (via concentration)

Family-wise upper bound via SSS

Family-wise upper bound via SSS

Theorem (Aldridge, B., Johnson, 2014)

Consider SSS *using* T_{SSS} *tests. Then,*

$$\mathbb{P}(\text{success}) \to 1 \Rightarrow T_{\text{SSS}} > \frac{(1-\beta)\mathrm{e}\ln 2}{\beta} \log_2 \binom{N}{K} ,$$

and, if the necessary condition is violated,

$$T_{\text{SSS}} \leq \frac{(1-\beta)\mathrm{e}\ln 2}{\beta} \log_2 \binom{N}{K} \Rightarrow \mathbb{P}(\text{success}) \leq \frac{2}{3} .$$

Family-wise upper bound via SSS

Theorem (Aldridge, B., Johnson, 2014)

Consider SSS *using* T_{SSS} *tests. Then,*

$$\mathbb{P}(\text{success}) \to 1 \Rightarrow T_{\text{SSS}} > \frac{(1-\beta)e \ln 2}{\beta} \log_2 \binom{N}{K} ,$$

and, if the necessary condition is violated,

$$T_{\text{SSS}} \leq \frac{(1-\beta)e \ln 2}{\beta} \log_2 \binom{N}{K} \Rightarrow \mathbb{P}(\text{success}) \leq \frac{2}{3} .$$

- "Weak converse"

Family-wise upper bound via SSS

Theorem (Aldridge, B., Johnson, 2014)

Consider SSS *using* T_{SSS} *tests. Then,*

$$\mathbb{P}(\text{success}) \to 1 \Rightarrow T_{\text{SSS}} > \frac{(1-\beta)e\ln 2}{\beta} \log_2 \binom{N}{K} ,$$

and, if the necessary condition is violated,

$$T_{\text{SSS}} \leq \frac{(1-\beta)e\ln 2}{\beta} \log_2 \binom{N}{K} \Rightarrow \mathbb{P}(\text{success}) \leq \frac{2}{3} .$$

- "Weak converse"
- Applies to *all* Bernoulli-based algorithms

Family-wise upper bound via SSS

Theorem (Aldridge, B., Johnson, 2014)

Consider SSS *using* T_{SSS} *tests. Then,*

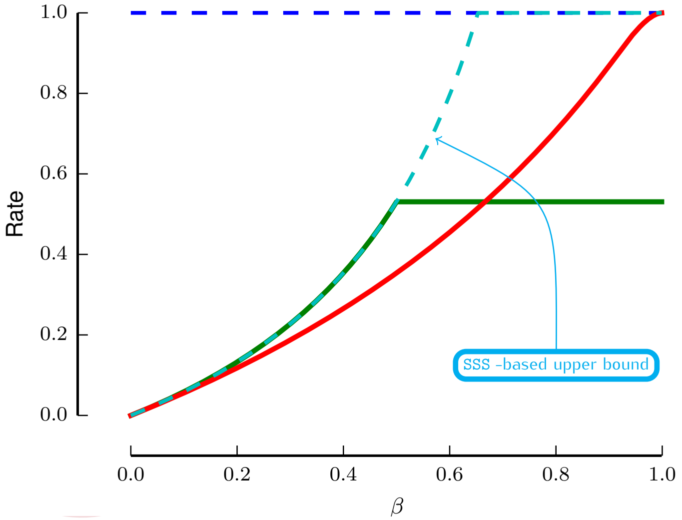
$$\mathbb{P}(\text{success}) \to 1 \Rightarrow T_{\text{SSS}} > \frac{(1-\beta)e \ln 2}{\beta} \log_2 \binom{N}{K} ,$$

and, if the necessary condition is violated,

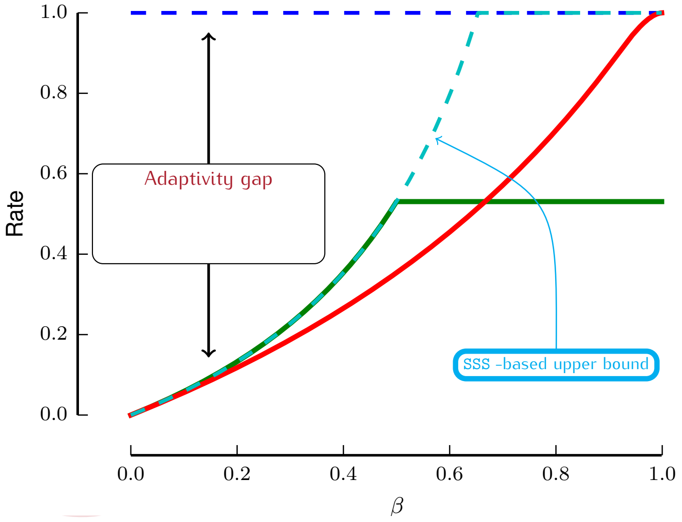
$$T_{\text{SSS}} \leq \frac{(1-\beta)e \ln 2}{\beta} \log_2 \binom{N}{K} \Rightarrow \mathbb{P}(\text{success}) \leq \frac{2}{3} .$$

- "Weak converse"
- Applies to *all* Bernoulli-based algorithms
- $R^*_{\text{SSS}}(\beta) \leq \min \left\{ 1, \frac{\beta}{(1-\beta)e \ln 2} \right\}$

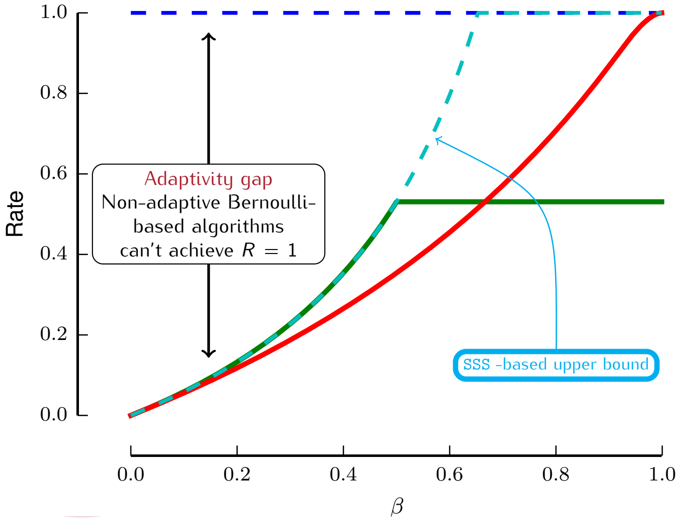
Adaptivity gap



Adaptivity gap



Adaptivity gap



The moral

The moral

- Group testing

The moral

- Group testing
  - Collective detection of sparse properties
  - Information-poor tests

The moral

- Group testing
  - Collective detection of sparse properties
  - Information-poor tests
- Rates and capacities

The moral

- Group testing
  - Collective detection of sparse properties
  - Information-poor tests
- Rates and capacities
  - Granular information
  - Capacities bound performance of algorithms

The moral

- Group testing
  - Collective detection of sparse properties
  - Information-poor tests
- Rates and capacities
  - Granular information
  - Capacities bound performance of algorithms
- SSS: bound the performance of *any* non-adaptive algorithm based on Bernoulli sampling

The moral

- Group testing
  - Collective detection of sparse properties
  - Information-poor tests
- Rates and capacities
  - Granular information
  - Capacities bound performance of algorithms
- `SSS`: bound the performance of *any* non-adaptive algorithm based on Bernoulli sampling
- `DD`: order-optimal, rate-optimal (for dense problems)

The moral

- Group testing
  - Collective detection of sparse properties
  - Information-poor tests
- Rates and capacities
  - Granular information
  - Capacities bound performance of algorithms
- `SSS`: bound the performance of *any* non-adaptive algorithm based on Bernoulli sampling
- `DD`: order-optimal, rate-optimal (for dense problems)
- Future work: noise models, applications, non-identical GT, non-independent GT, collateral open questions...