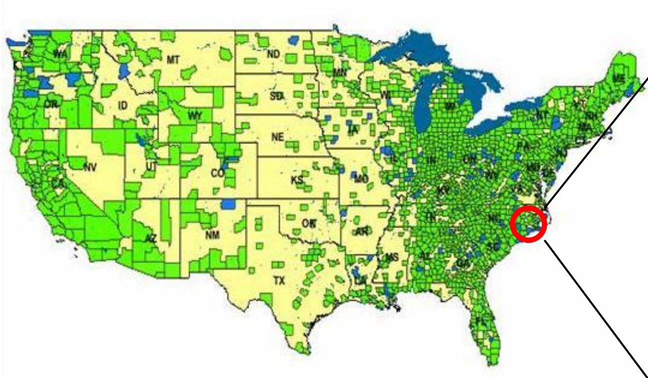
Scaling Up Event and Pattern Detection to Big Data

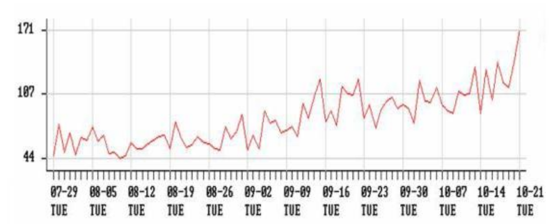
Daniel B. Neill
H.J. Heinz III College
Carnegie Mellon University
E-mail: neill@cs.cmu.edu

We gratefully acknowledge funding support from the National Science Foundation, grants IIS-0916345, IIS-0911032, and IIS-0953330.

Multivariate event detection



Spatial time series data from spatial locations s_i (e.g. zip codes)



Time series of counts $c_{i,m}^t$ for each zip code s_i for each data stream d_m.

Outbreak detection

d_1 = respiratory ED

d_2 = constitutional ED

d_3 = OTC cough/cold

d_4 = OTC anti-fever

(etc.)

Main goals:

Detect any emerging events.

Pinpoint the affected subset of locations and time duration.

Characterize the event by identifying the affected streams.

Compare hypotheses:

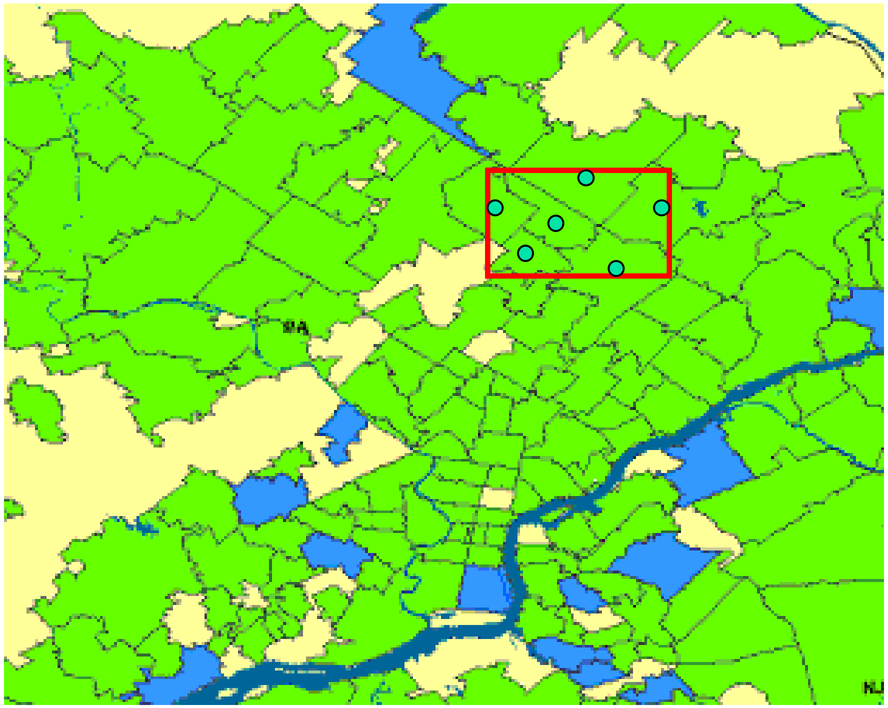
$H_1(D, S, W)$

D = subset of streams
S = subset of locations
W = time duration

vs. H_0: no events occurring

Expectation-based scan statistics

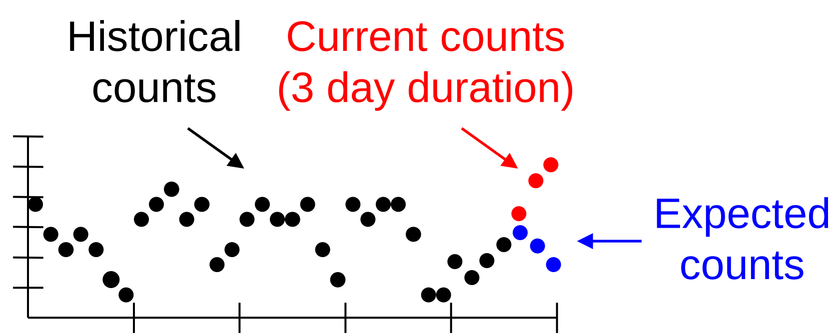
(Kulldorff, 1997; Neill and Moore, 2005)



We search for spatial regions (subsets of locations) where the recently observed counts for some subset of streams are significantly higher than expected.

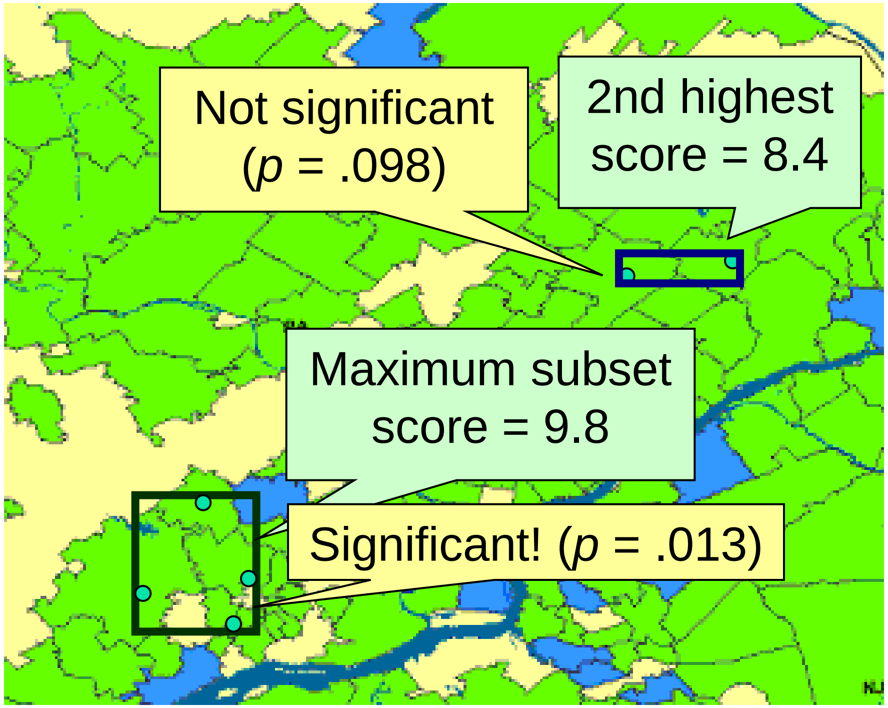
We perform **time series analysis** to compute expected counts (“baselines”) for each location and stream for each recent day.

We then compare the actual and expected counts for each subset (D, S, W) under consideration.



Expectation-based scan statistics

(Kulldorff, 1997; Neill and Moore, 2005)

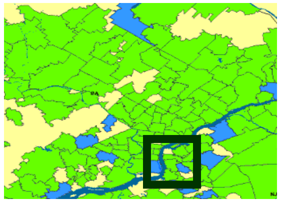


We find the subsets with highest values of a likelihood ratio statistic, and compute the *p*-value of each subset by randomization testing.

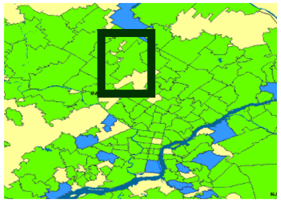
$$F(D,S,W) = \frac{\Pr(\text{Data} \mid H_1(D,S,W))}{\Pr(\text{Data} \mid H_0)}$$

<u>To compute p-value</u>
Compare subset score to maximum subset scores of simulated datasets under H_0.

$F_1^* = 2.4$

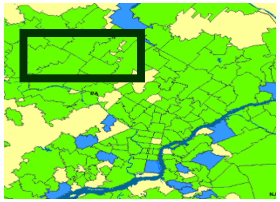


$F_2^* = 9.1$



...

$F_{999}^* = 7.0$



Which regions to search?

Typical approach: "spatial scan" (Kulldorff, 1997)

Each search region S is a **sub-region** of space.

- Choose some region shape (e.g. circles, rectangles) and consider all regions of that shape and varying size.
- Low power for true events that do not correspond well to the chosen set of search regions (e.g. irregular shapes).

Our approach: "subset scan" (Neill, 2012)

Each search region S is a **subset** of locations.

- Find the highest scoring subset, subject to some constraints (e.g. spatial proximity, connectivity).
- For multivariate, also optimize over subsets of streams.
- Exponentially many possible subsets, $O(2^N \times 2^M)$: computationally infeasible for naïve search.

Fast subset scan

- In certain cases, we can optimize F(S) over the exponentially many subsets of the data, while evaluating only O(N) rather than $O(2^N)$ subsets.
- Many commonly used scan statistics have the property of linear-time subset scanning:
  - Just sort the data records (or spatial locations, etc.) from highest to lowest priority according to some function…
  - … then search over groups consisting of the top-k highest priority records, for k = 1..N.

The highest scoring subset is **guaranteed** to be one of these!

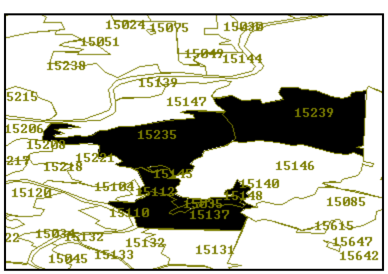
Sample result: we can find the **most anomalous** subset of Allegheny County zip codes in 0.03 sec vs. 10^{24} years.

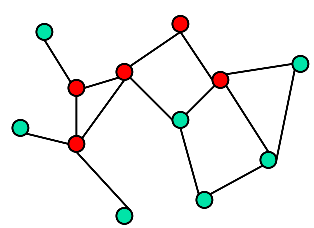
Constrained fast subset scanning

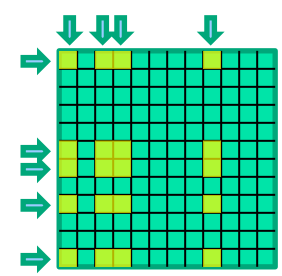
LTSS is a new and powerful tool for **exact** combinatorial optimization (as opposed to approximate techniques such as submodular function optimization). But it only solves the "best unconstrained subset" problem, and cannot be used directly for <u>constrained</u> optimization.

Much of our recent work has focused on how LTSS can be extended to the many real-world problems with (hard or soft) constraints on our search.

Proximity constraints	→	Fast spatial scan (irregular regions)
Multiple data streams	→	Fast multivariate scan
Connectivity constraints	→	Fast graph scan
Group self-similarity	→	Fast generalized subset scan







Fast subset scan with spatial proximity constraints

- Maximize a likelihood ratio statistic over all subsets of the “local neighborhoods” consisting of a center location s_i and its k-1 nearest neighbors, for a fixed neighborhood size k.
- For each local neighborhood, naïve search requires $O(2^k)$ time and is computationally infeasible for $k > 25$, but LTSS enables us to perform this search in $O(k)$ time.
- In Neill (2012), we show that this approach dramatically improves the timeliness and accuracy of outbreak detection for irregularly-shaped disease clusters.

Multivariate fast subset scan

(Neill, McFowland, and Zheng, 2013)

- The LTSS property allows us to efficiently optimize over subsets of spatial locations for a given subset of data streams.
- But it also allows us to efficiently optimize over subsets of **streams** for a given subset of **locations**…
- So we can jointly optimize over subsets of streams **and** locations by iterating between these two steps!
- For general pattern detection problems, a similar approach can be used to jointly optimize over subsets of data records and attributes in our Fast Generalized Subset Scan approach (McFowland et al., *JMLR*, 2013).

Incorporating soft constraints

(Speakman, Somanchi, McFowland, and Neill, 2014, submitted)

- So far we have talked about **hard** constraints (i.e., restrictions on the search space, ruling out some subsets).
- What about **soft** constraints?
  - We would like to search over all subsets, but reward more likely subsets and penalize those that are less likely.

For functions satisfying the **Additive Linear Time Subset Scanning** property, conditioning on the relative risk, *q,* allows the function to be written as an *additive* set function over the data elements s_i in *S*.

Expectation-based scan statistics in a one-parameter exponential family

$$F(S) = \max_{q>1} \log \frac{P(Data \mid H_1(S))}{P(Data \mid H_0)}$$

$$H_0 : x_i \sim Dist(\mu_i)$$

$$H_1 : x_i \sim Dist(q\mu_i)$$

Penalized Fast Subset Scanning

For functions satisfying the **Additive Linear Time Subset Scanning** property, conditioning on the relative risk, *q,* allows the function to be written as an *additive* set function over the data elements s_i in S.

Consequence #1: Extremely easy to maximize F(S) over subsets, for a given q, by including all "positive" elements and excluding "negative".

Consequence #2: Additional, element-specific penalty terms may be added to the scoring function while maintaining the additive property.

Expectation-based Poisson:

$$F(S) = \max_{q>1} \sum_{s_i \in S} x_i(\log q) + \mu_i(1-q)$$

Penalized Fast Subset Scanning

For functions satisfying the **Additive Linear Time Subset Scanning** property, conditioning on the relative risk, *q,* allows the function to be written as an *additive* set function over the data elements s_i in *S*.

<u>Consequence #1</u>: Extremely easy to maximize F(S) over subsets, for a given q, by including all "positive" elements and excluding "negative".

<u>Consequence #2</u>: Additional, element-specific penalty terms may be added to the scoring function while maintaining the additive property.

"Total Contribution" γ_i of record s_i for fixed risk, q

<u>Expectation-based Poisson:</u>

$$F_{penalized}(S) = \max_{q>1} \sum_{s_i \in S} \underbrace{[\, x_i(\log q) + \mu_i(1-q) + \boldsymbol{\Delta_i} \,]}$$

Penalized Fast Subset Scanning

For functions satisfying the **Additive Linear Time Subset Scanning** property, conditioning on the relative risk, *q,* allows the function to be written as an *additive* set function over the data elements s_i in *S*.

Consequence #1: Extremely easy to maximize F(S) over subsets, for a given q, by including all "positive" elements and excluding "negative".

Consequence #2: Additional, element-specific penalty terms may be added to the scoring function while maintaining the additive property.

How to optimize efficiently over all values of q, not just a given q???

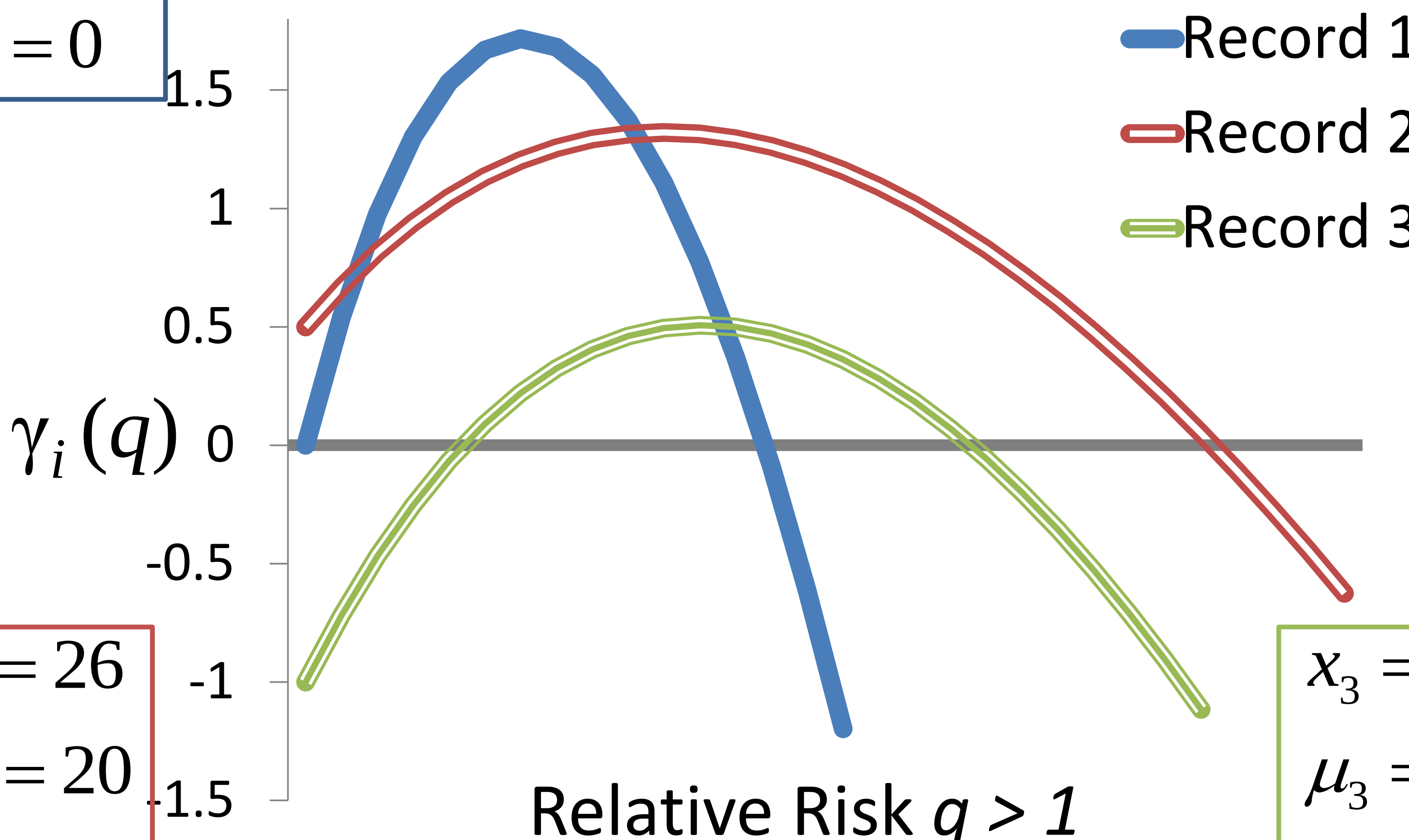
Theorem: the optimal subset $S^* = \arg\max_S F_{pen}(S)$ for a penalized expectation-based scan statistic satisfying the ALTSS property may be found by evaluating only O(N) of the 2^N subsets of data records.

“Proof by picture”

$x_1 = 130$

$\mu_1 = 110$

$\Delta_1 = 0$



$x_2 = 26$

$\mu_2 = 20$

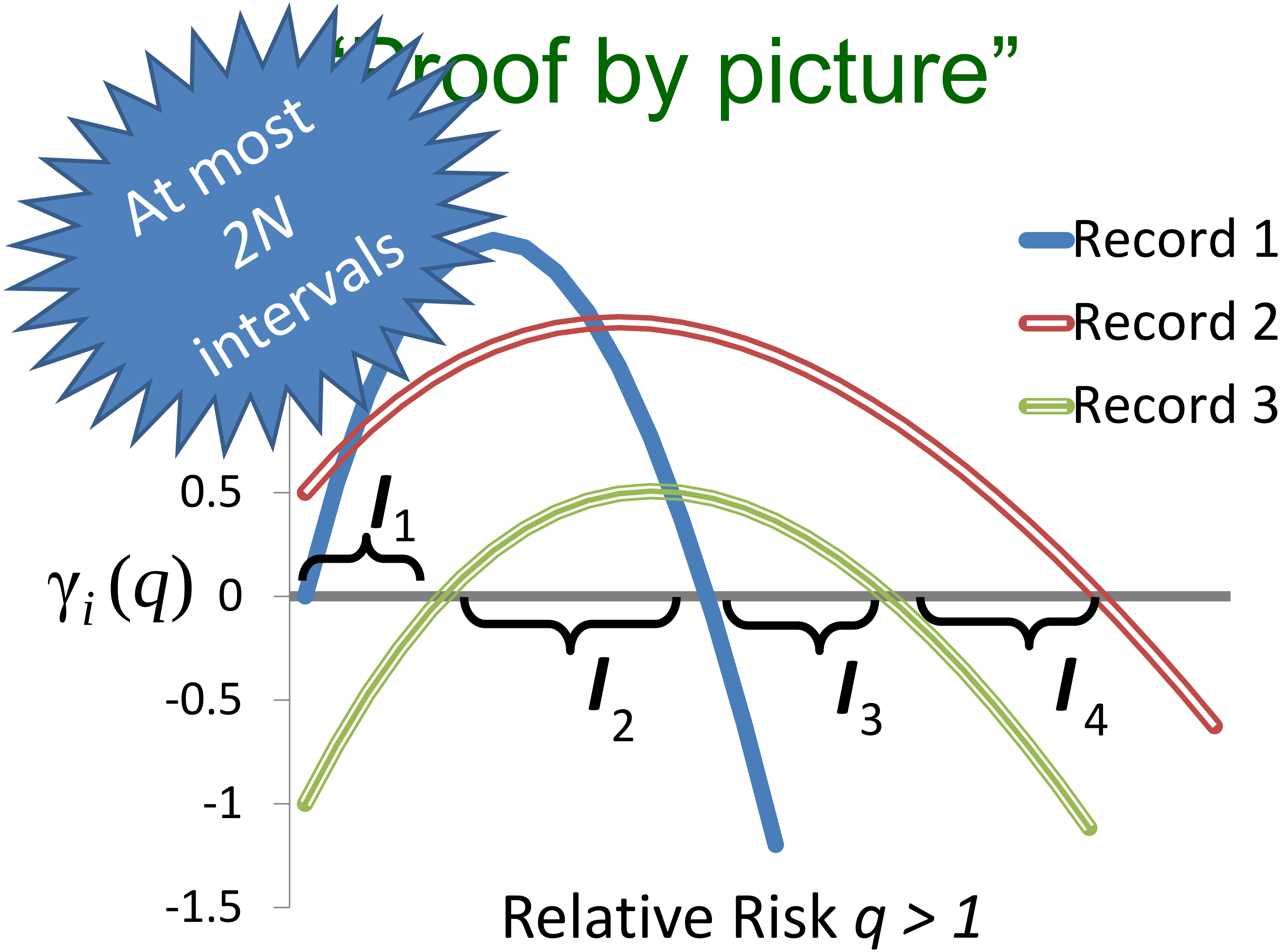
$\Delta_2 = 0.5$

$x_3 = 40$

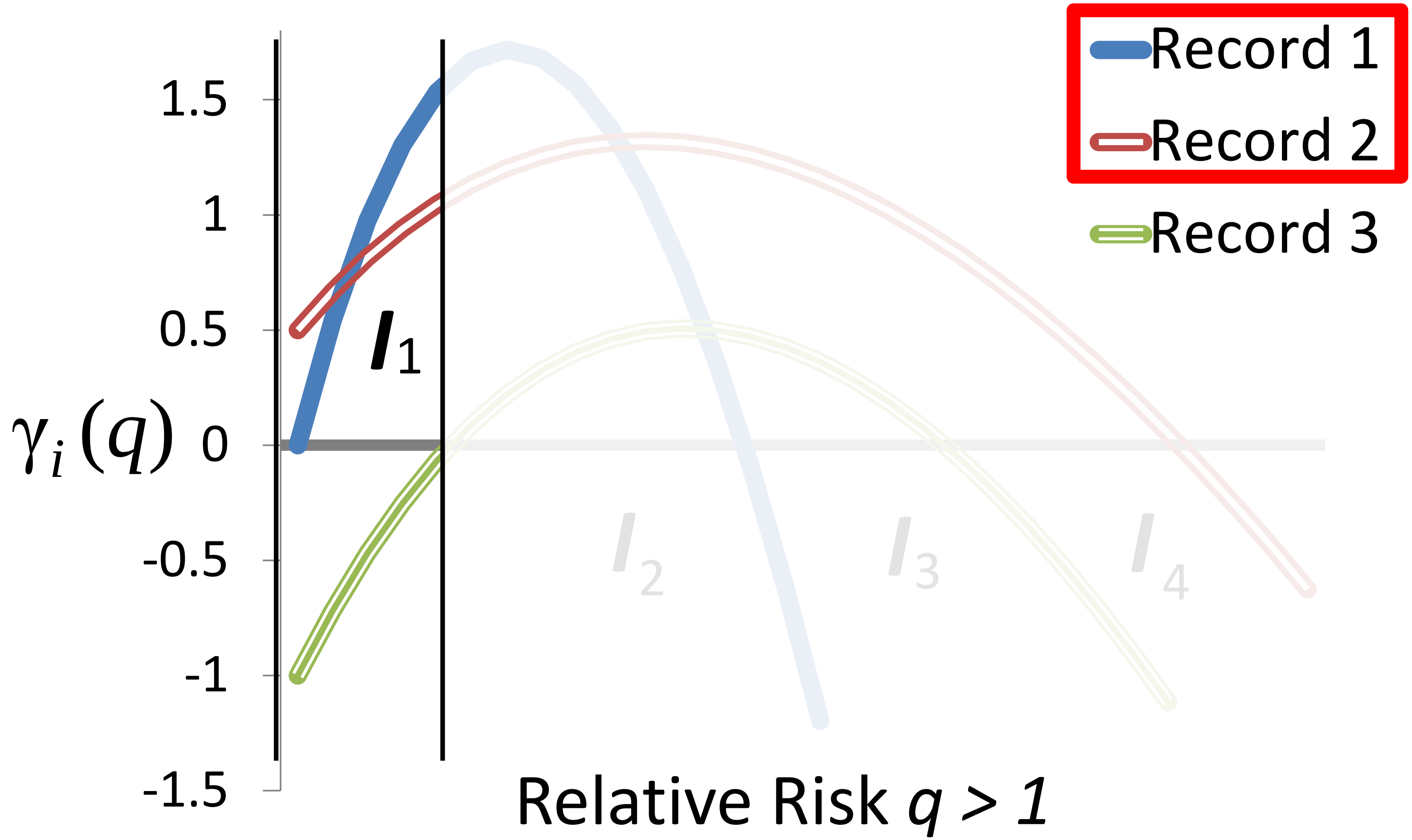
$\mu_3 = 30$

$\Delta_3 = -1$

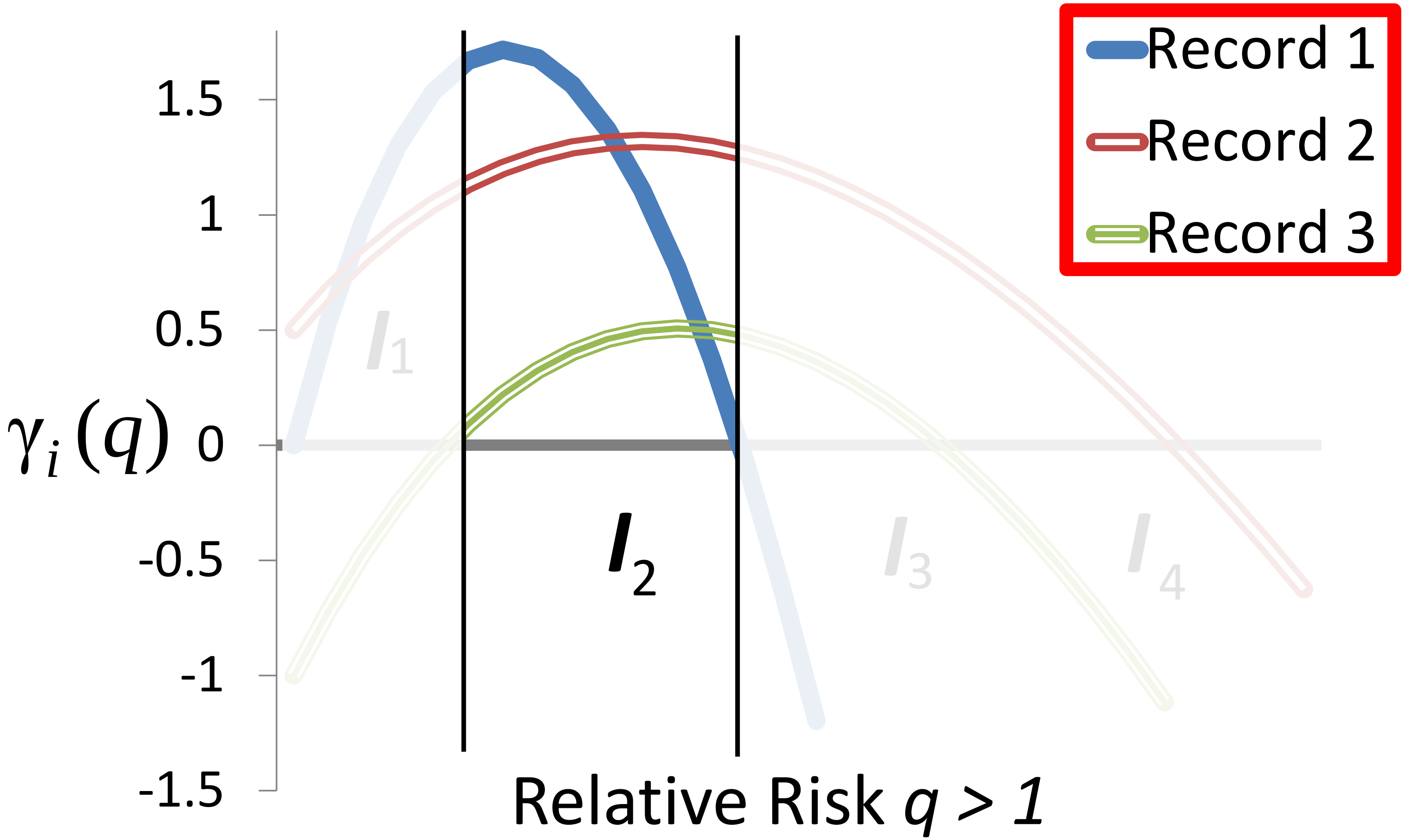
"Proof by picture"



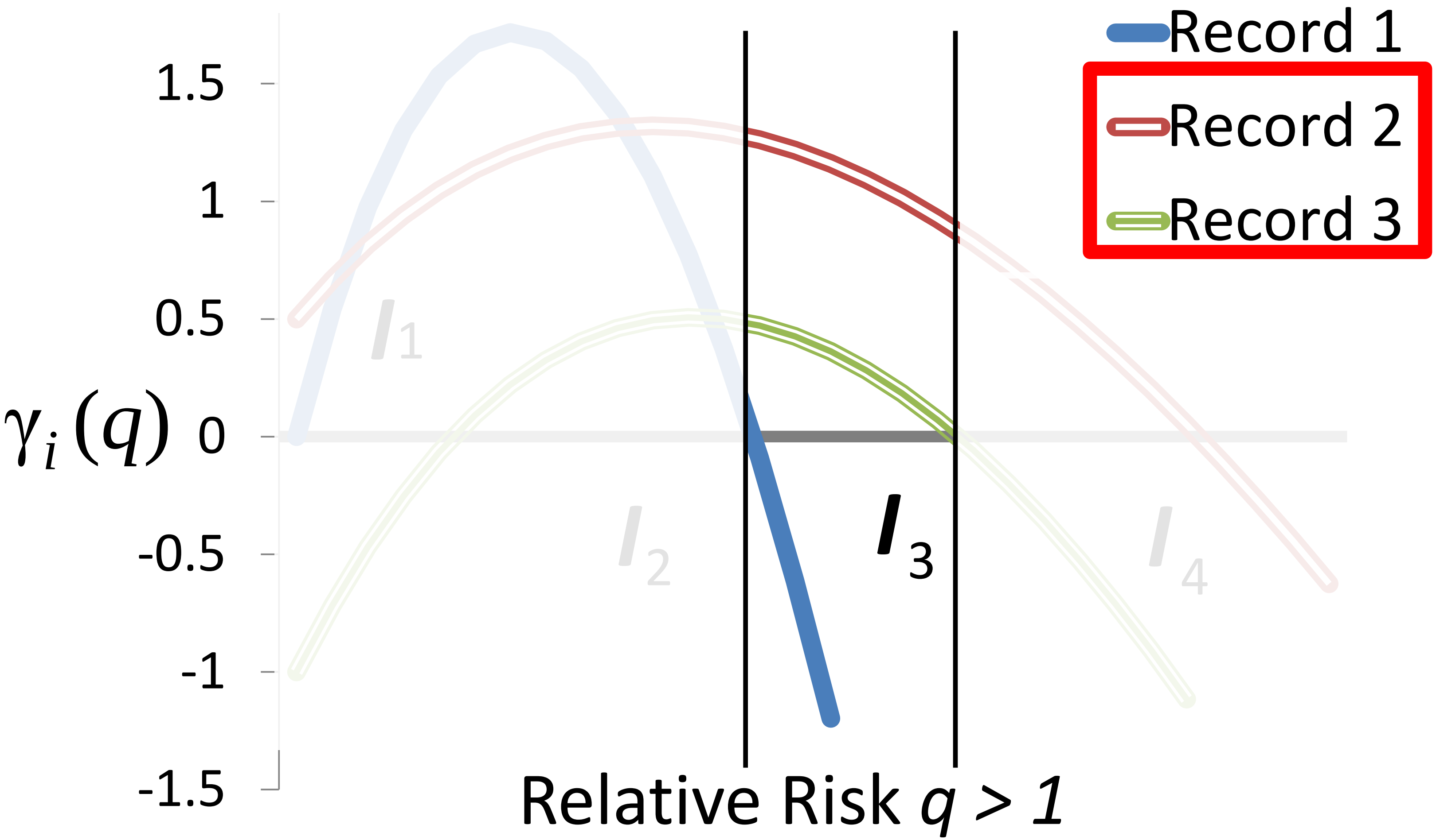
"Proof by picture"



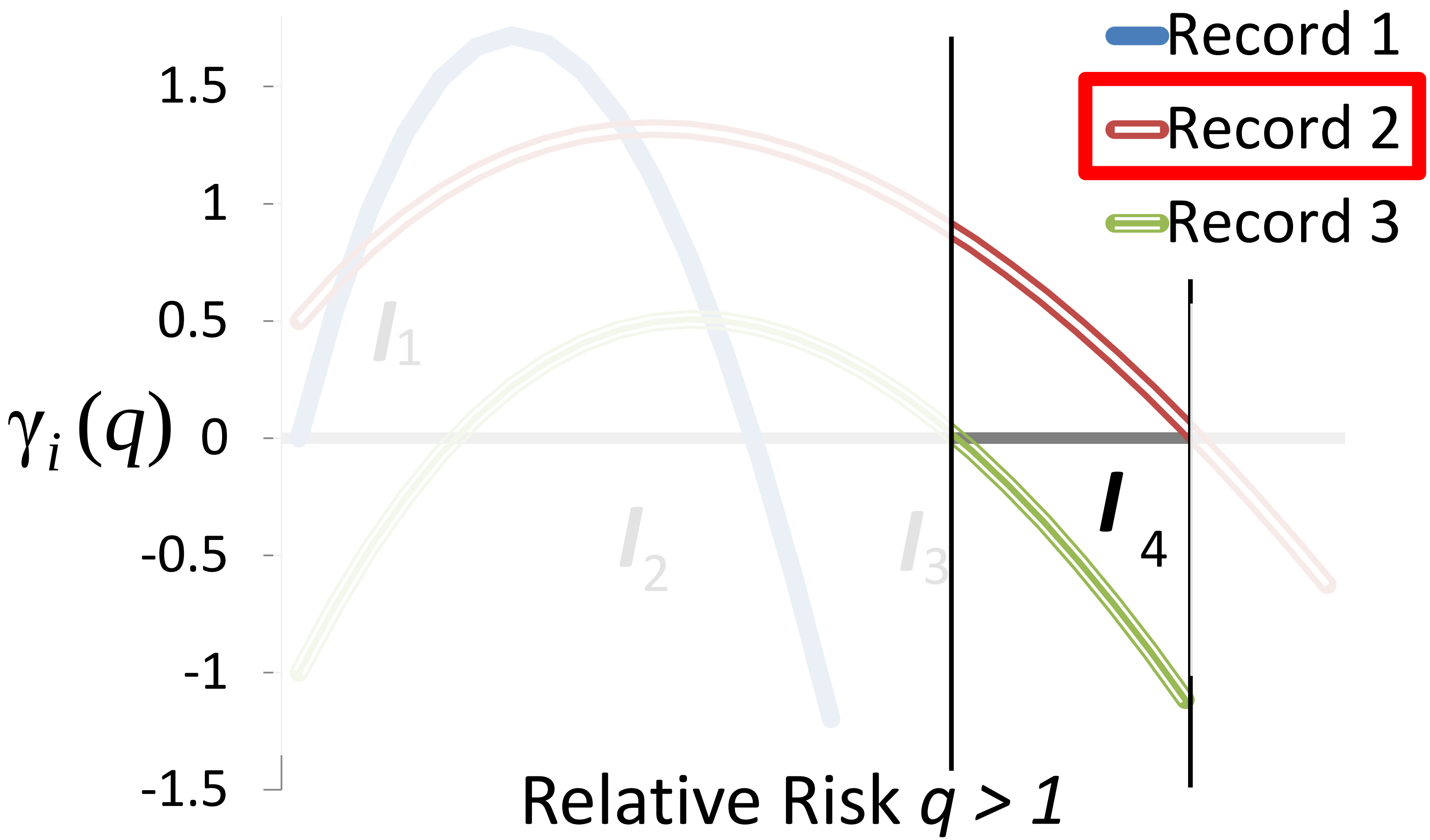
"Proof by picture"



"Proof by picture"



“Proof by picture”

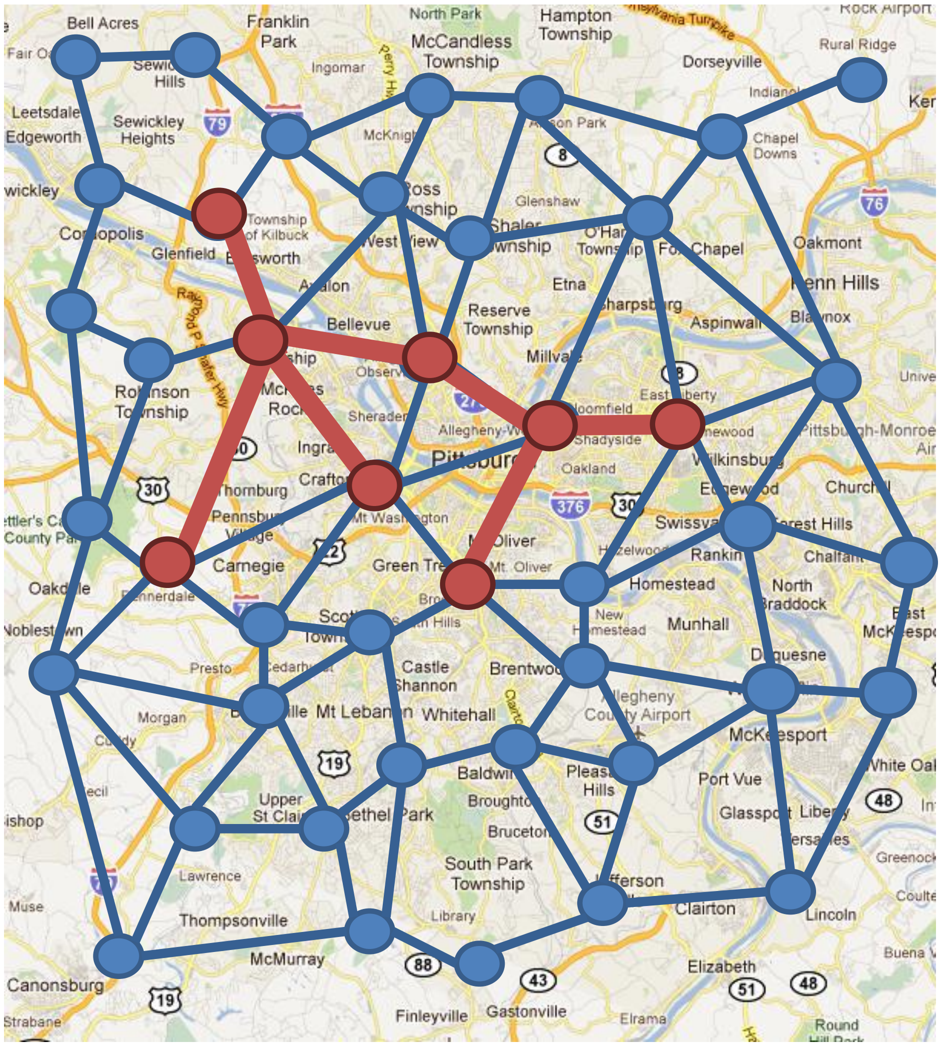


Penalized Fast Subset Scanning

Penalized Fast Subset Scanning is a general framework for scalable pattern detection with soft constraints.

- **Exactness**: The most anomalous (highest scoring) subset is guaranteed to be identified.
- **Efficiency**: Only *O(N)* subsets must be scanned in order to identify the most anomalous ***penalized*** subset in a dataset containing *N* elements.
- **Interpretability**: Soft constraints may be viewed as the prior log-odds for a given record to be included in the most anomalous penalized subset.

Detecting and Tracking Dynamic Patterns



Most subset scan methods have difficulty dealing with **dynamic** patterns, where the affected subset changes over time.

Optimizing each time step independently fails, as does neglecting event dynamics.

Our solution, Dynamic Subset Scan, uses soft constraints on **temporal consistency** to pass information between time steps.

Detecting and Tracking Dynamic Patterns

Dynamic Subset Scan algorithm

1) Identify subsets S_t independently for each time step t, using unpenalized fast subset scan.

2) Repeat until convergence:

   a) Choose a time step t.

   b) Compute Δ_i^t for each location s_i, given subsets S_{t-1} and S_{t+1}.

   c) Find new optimal subset S_t using penalized fast subset scan with the given Δ_i^t.

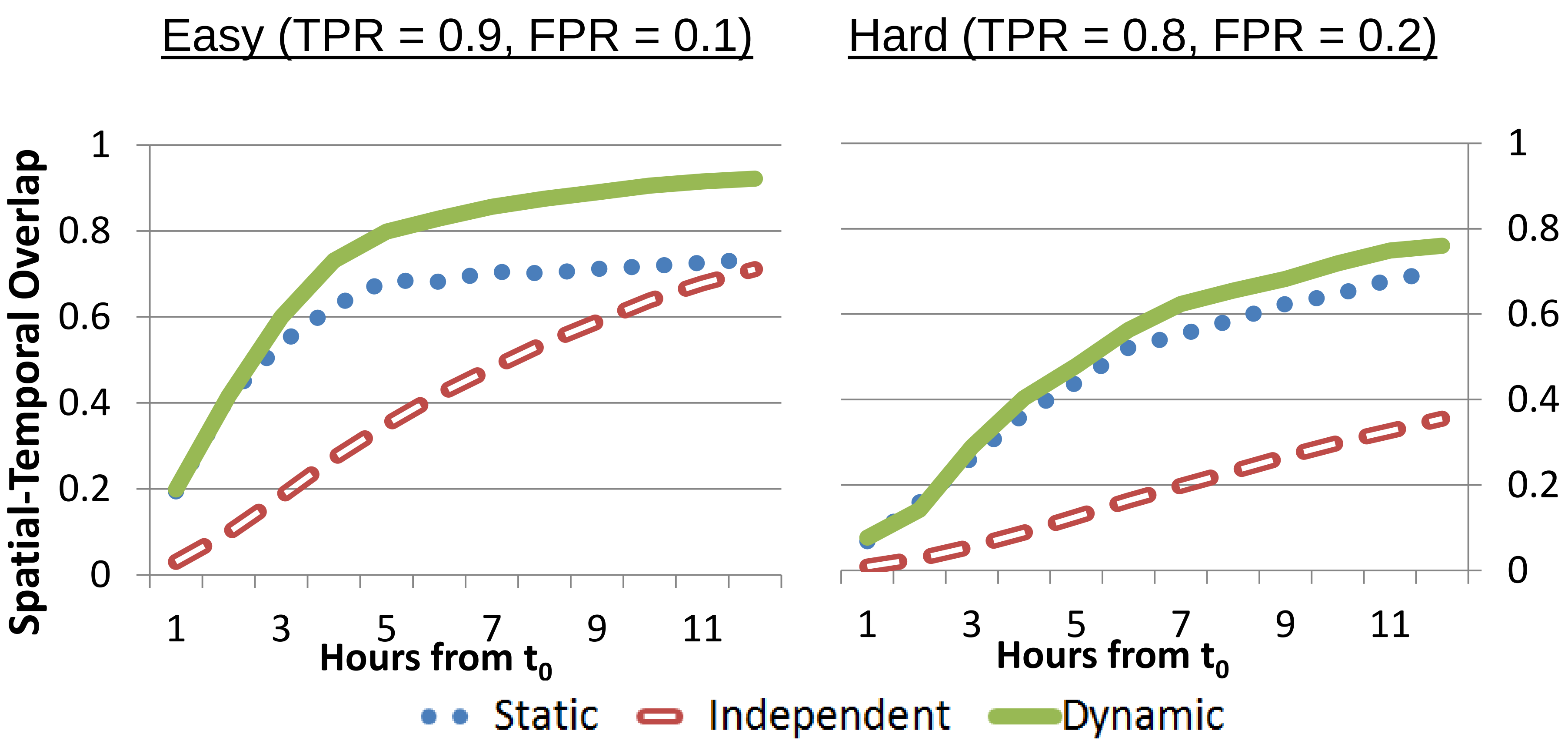
Generative model

$$\log\left(\frac{p_i^t}{1-p_i^t}\right) = \beta_0 + \beta_1 X_i^{t-1} + \beta_2 \frac{n_i^{t-1}}{k_i}$$

- $\log\left(\frac{p_i^t}{1-p_i^t}\right)$: Prior log-odds that location s_i affected on time step t.
- X_i^{t-1}: Equals 1 if location s_i affected on time step t-1.
- $\frac{n_i^{t-1}}{k_i}$: Fraction of neighbors affected on time step t-1.

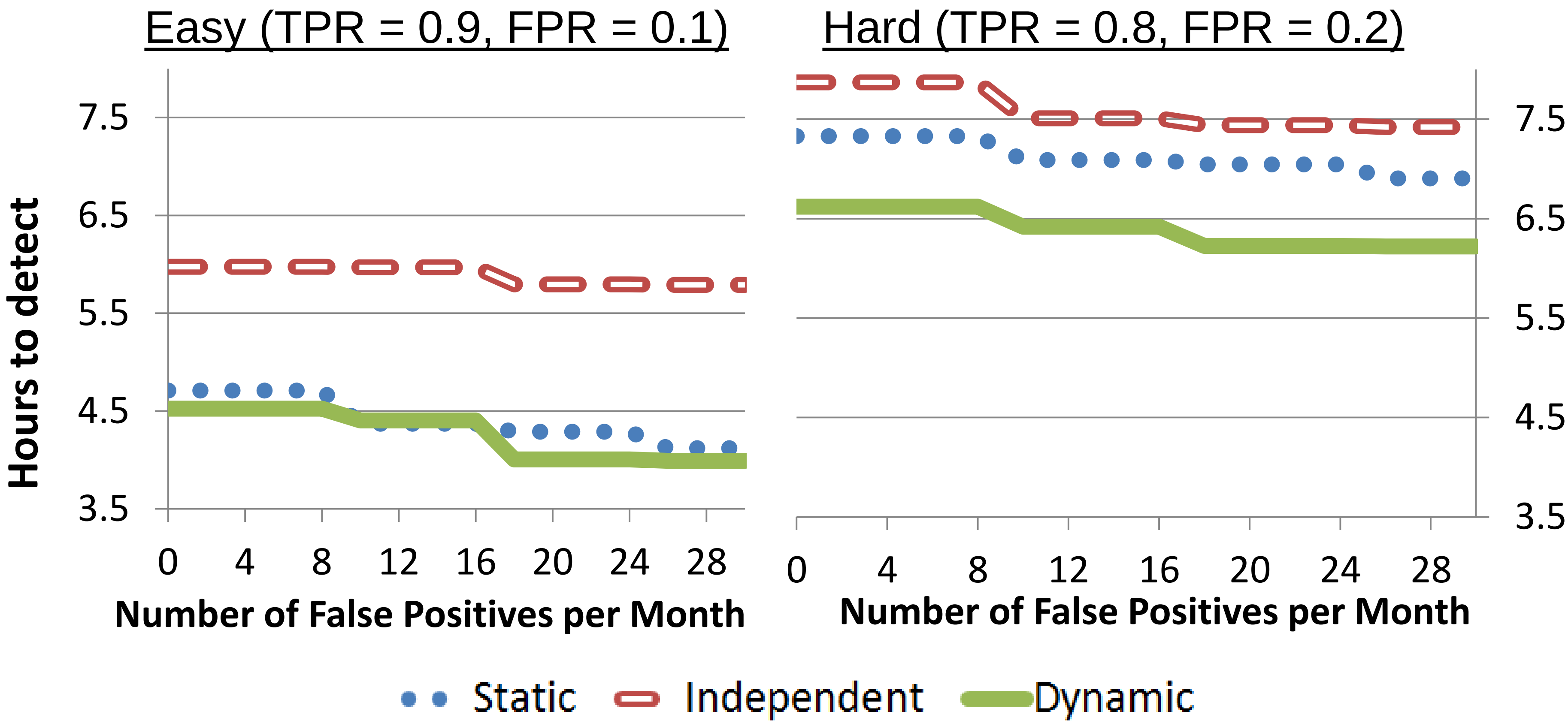
See our ICDM 2013 paper for more details!

Tracking Contaminant Plumes



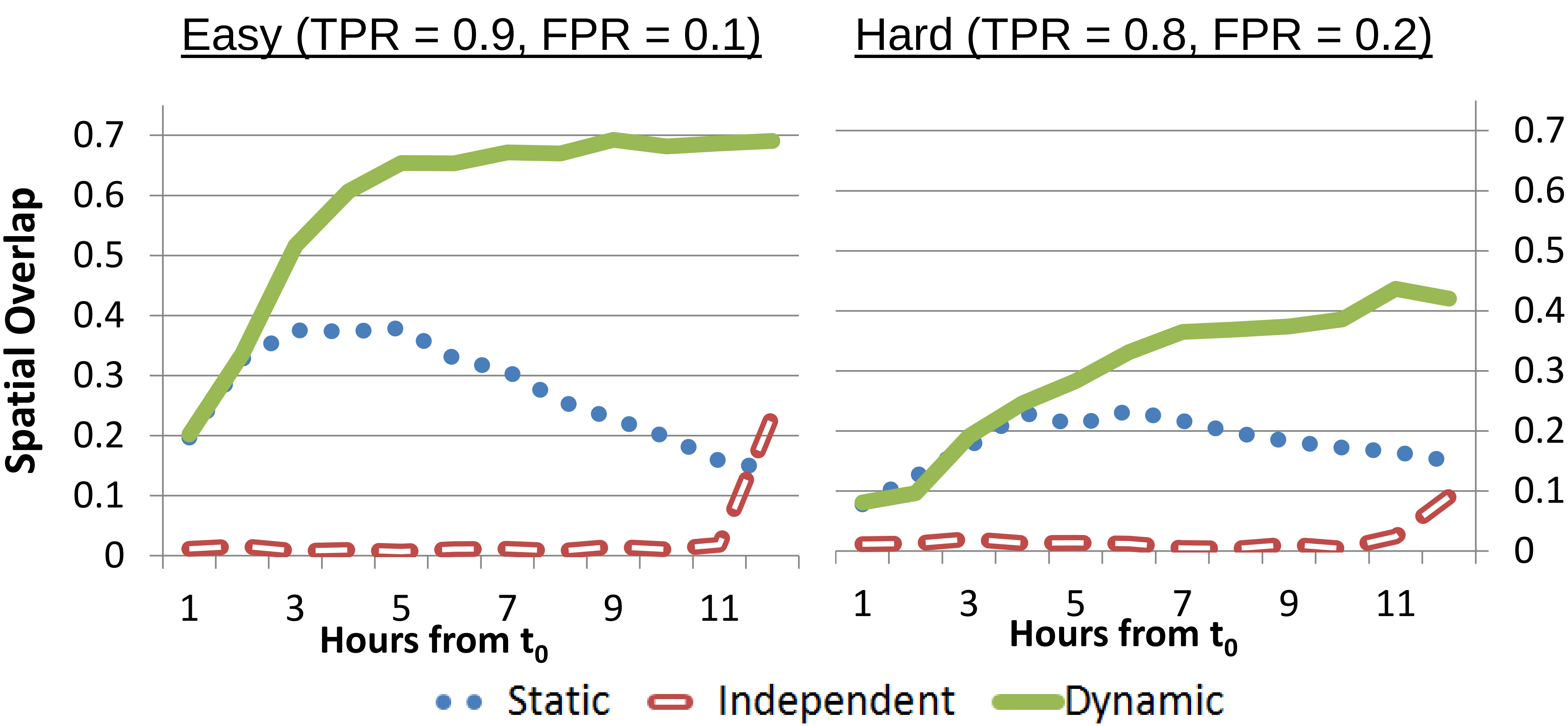
Dynamic Subset Scan improves event tracking, as measured by overlap coefficient between the true and detected regions.

Detecting Contaminant Plumes



Dynamic Subset Scan improves event detection, as measured by average number of hours needed to detect.

Source-Tracing Contaminant Plumes



Dynamic Subset Scan improves accuracy for locating the source of the event, as measured by overlap between true and detected regions.

Scaling up to even **bigger** data…

Currently the fast subset scan scales to datasets with **millions** of records.

Spatial constraints (FSS)
Similarity constraints (FGSS)
Soft constraints (PFSS)

But enforcing certain hard constraints (e.g., graph connectivity) dramatically impacts scalability.

GraphScan: 250 nodes
Additive Graphscan : 25K nodes

How to scale up to larger graphs with millions of nodes? ← ongoing EPD Lab research → How to scale up to datasets with billions or trillions of records?

Many possible answers!

Locality-Sensitive Hashing

Sampling

Sublinear-Time Algorithms

Problem Partitioning

Summarization

Parallelization

Randomization

Hierarchy

Idea #1: Massive parallelization

For example, what if we have a trillion records but a million processors?

Certain aspects of fast subset scan are **trivially parallelizable**:

- Randomization testing, to determine statistical significance.
- Scanning over many local neighborhoods (with proximity constraints).
- Scoring many subsets (but not exponentially many!).

For **unconstrained subset scan**, we have the necessary pieces:

- Parallel sorting (merge sort, sample sort): O(log N) with N processors.
- "Scan" (accumulate sums of top-k elements by priority): O(log N).

To incorporate **spatial proximity** or more general **similarity** constraints:

- **Locality-sensitive hashing** → neighborhoods of similar elements.

With more general constraints (e.g., graphs), we must develop new ways to partition the search space and merge solutions to sub-problems.

Idea #2: Incorporate hierarchy

Subsampling the raw data can miss a arbitrarily strong signal that affects a small enough proportion of the dataset.

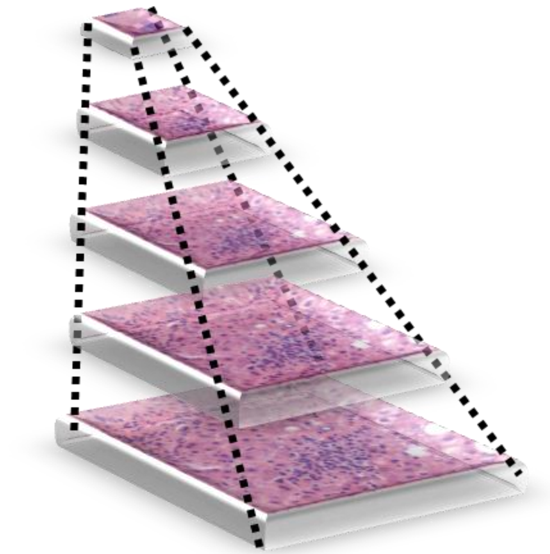
Possible solution: **summarization.**

Represent the data **hierarchically**, maintain summary statistics at each level of hierarchy, and search over coarse and fine resolutions.

<ins>Goal</ins>: find the most interesting subsets while only looking at a small fraction of the raw data.

<ins>Challenge 1</ins>: building the hierarchy may be expensive (though parallelizable).

<ins>Challenge 2</ins>: how to search the hierarchy, so that we are unlikely to miss small areas?

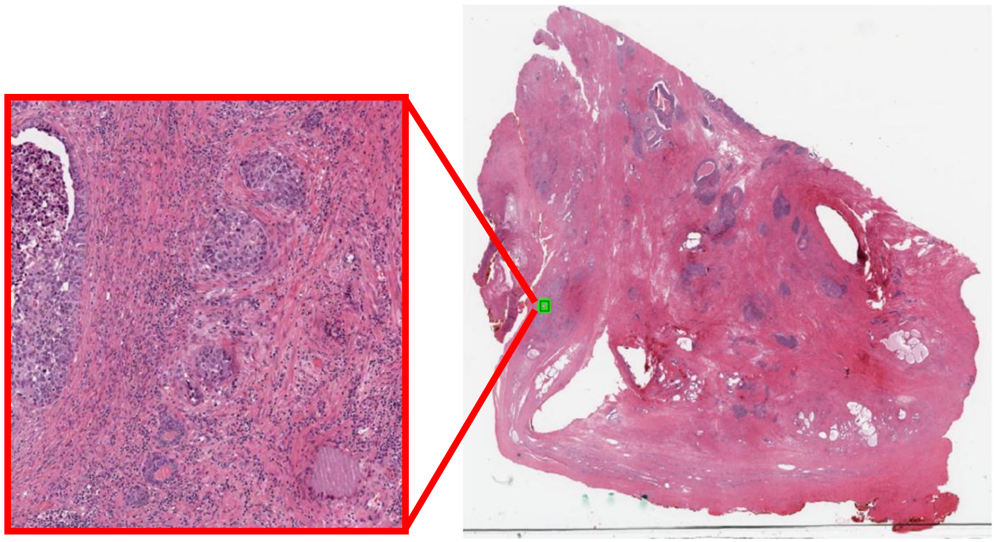


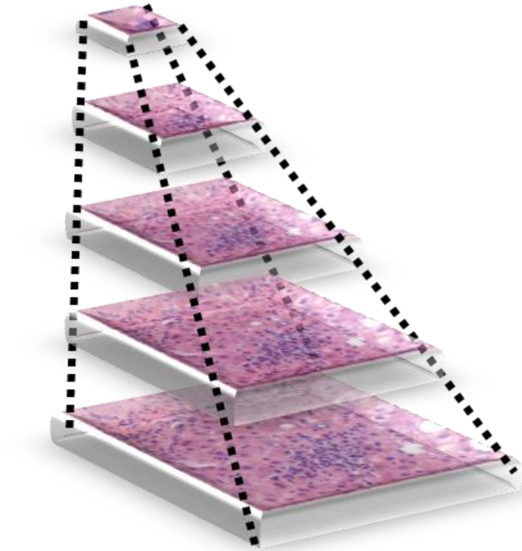
<ins>Example: image data</ins>
digital pathology slides, satellite images, etc.

Hierarchical Linear-Time Subset Scanning

(Somanchi & Neill, DMHI 2013)

Idea #2: Incorporate hierarchy





<u>Example: image data</u>
digital pathology slides, satellite images, etc.

HLTSS has been successfully applied to detect regions of interest in digital pathology slides, and works surprisingly well to detect prostate cancer!

Hierarchical Linear-Time Subset Scanning

(Somanchi & Neill, DMHI 2013)

Current application domains

Disease surveillance: Deployed systems in US, Canada, Sri Lanka, India.

In progress: deployments in Canada for monitoring hospital-acquired illness.

Crime prediction in Chicago:

Able to predict about 83% of "clustered" violent crimes and 57% of all violent crimes, with 15% false positive rate.

Many more applications:

- Illicit container shipments
- Clusters of water pipe breaks
- Spreading water contamination
- Network intrusion detection
- Economic growth "outbreaks"
- Patient care practices

Predicting civil unrest events using Twitter data:

By discovering anomalous subgraphs of nodes in the heterogeneous network formed by users, locations, nodes, tweets, etc., we can accurately predict events such as protests, strikes, and riots.

Interested?

More details on our web site:

http://epdlab.heinz.cmu.edu

Or e-mail me at:

neill@cs.cmu.edu