Programming Uncertain <T>hings

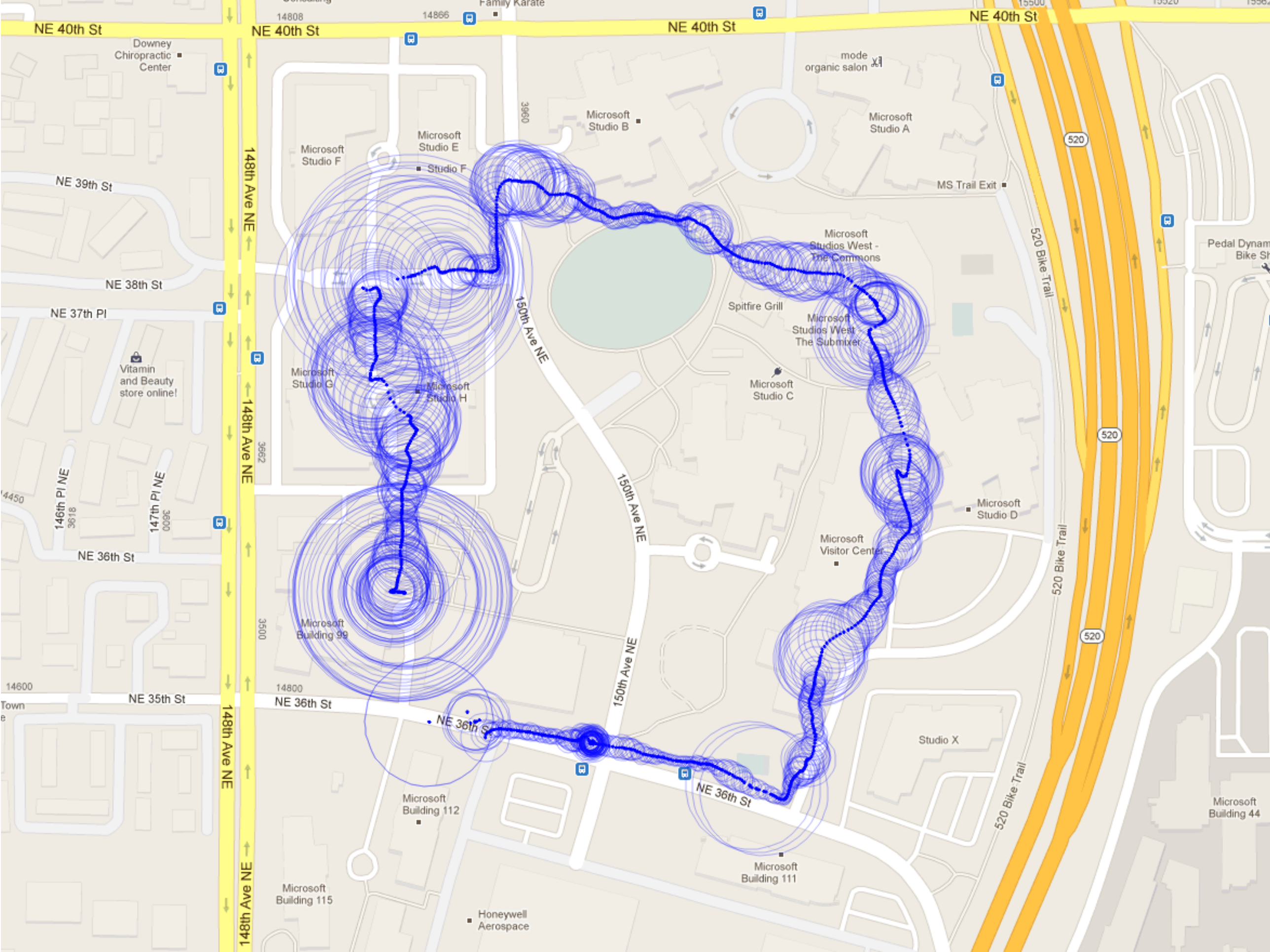
Kathryn S. McKinley Microsoft Research

Todd Mytkowicz Microsoft Research
James Bornholt University of Washington
Na Meng Virginia Tech
Adrian Sampson Microsoft Research, Cornell
Diman Zad Tootaghaj University of Pittsburgh

24 mph


NOKIA
runtastic
Home Map Split
Jake Van Damme
DURATION
00:23:43
DISTANCE
2.59
PACE
08:53
AVG. SPEED
6.5
ALTITUDE
10
CALORIES
354
Pause | Stop
59 mph

```
GeoCoordinate Location = GPS.Get();
```


NE 40th St
NE 40th St
NE 40th St
NE 40th St
14808
14866
Downey Chiropractic Center
Family Karate
mode organic salon
Microsoft Studio B
Microsoft Studio A
Microsoft Studio E
Microsoft Studio F
Studio F
3960
MS Trail Exit
520
520 Bike Trail
148th Ave NE
NE 39th St
NE 38th St
NE 37th Pl
Microsoft Studios West - The Commons
Spitfire Grill
Microsoft Studios West The Submixer
150th Ave NE
Microsoft Studio G
Microsoft Studio H
Microsoft Studio C
Vitamin and Beauty store online!
Pedal Dynam Bike Sh
146th Pl NE
147th Pl NE
NE 36th St
Microsoft Studio D
Microsoft Visitor Center
Microsoft Building 99
3500
14600
NE 35th St
14800
NE 36th St
Studio X
Microsoft Building 112
Microsoft Building 111
Microsoft Building 115
Honeywell Aerospace
Microsoft Building 44

```
GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
```

```
GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
double Dist =
                Distance(PrevLocn, Location);
double Speed = Dist / 5;
```

```
GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
double Dist =
                  Distance(PrevLocn, Location);
double Speed = Dist / 5;


Print(Speed);
```

59 mph


6.0mi
54:36
346kcals

sensors

Whales

Legs & Abdomen

Primates & Mya

Brazil

Fins & Whales

Sharks & Bunch

Prey & Species

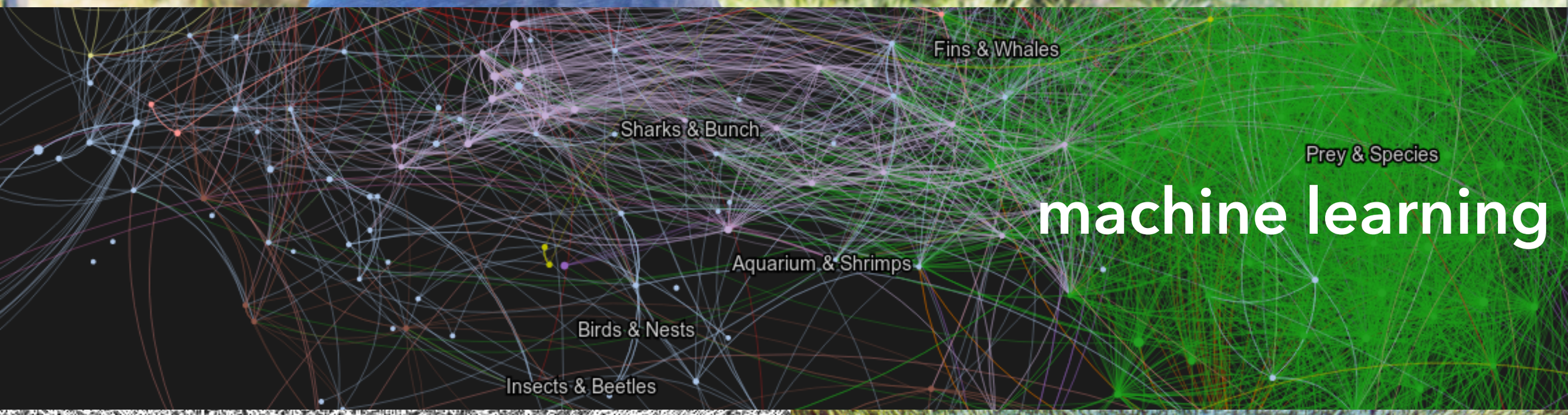
Aquarium & Shrimps

Birds & Nests

Insects & Beetles

machine learning

sensors



machine learning

approximate computing

Uncertain<*T*>

an abstraction for reasoning about noise

[ASPLOS'14]

exploiting context

improving accuracy with more language & inference

[rejected so far]

what does it mean?

probabilistic assertions

[PLDI'14]

Uncertain<*T*>

an abstraction for reasoning about noise

[ASPLOS'14]

exploiting context

improving accuracy with more language & inference

[rejected so far]

what does it mean?

probabilistic assertions

[PLDI'14]

Uncertain<*T*>

an abstraction for reasoning about noise

[ASPLOS'14]

exploiting context

improving accuracy with more language & inference

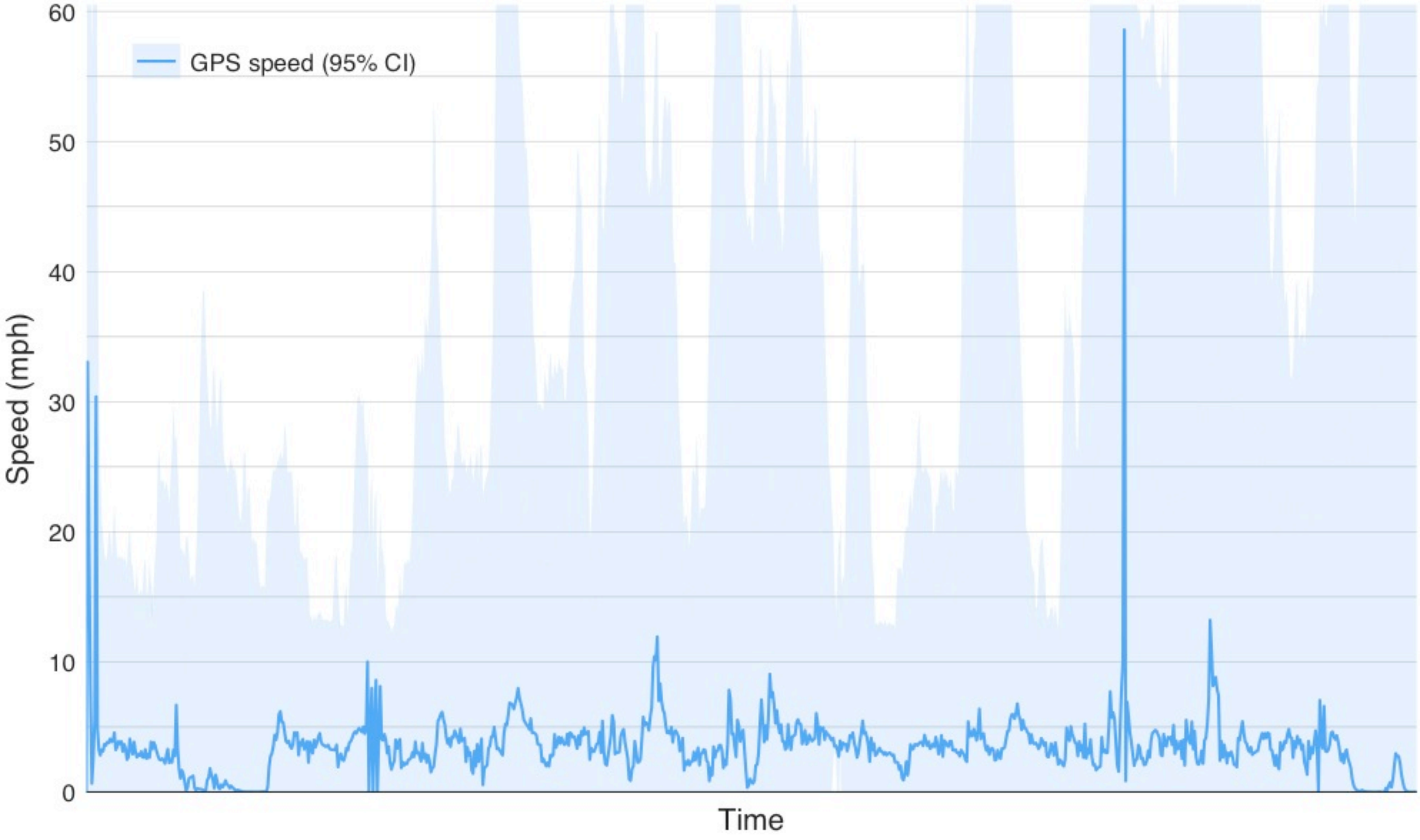
[rejected so far]

what does it mean?

probabilistic assertions

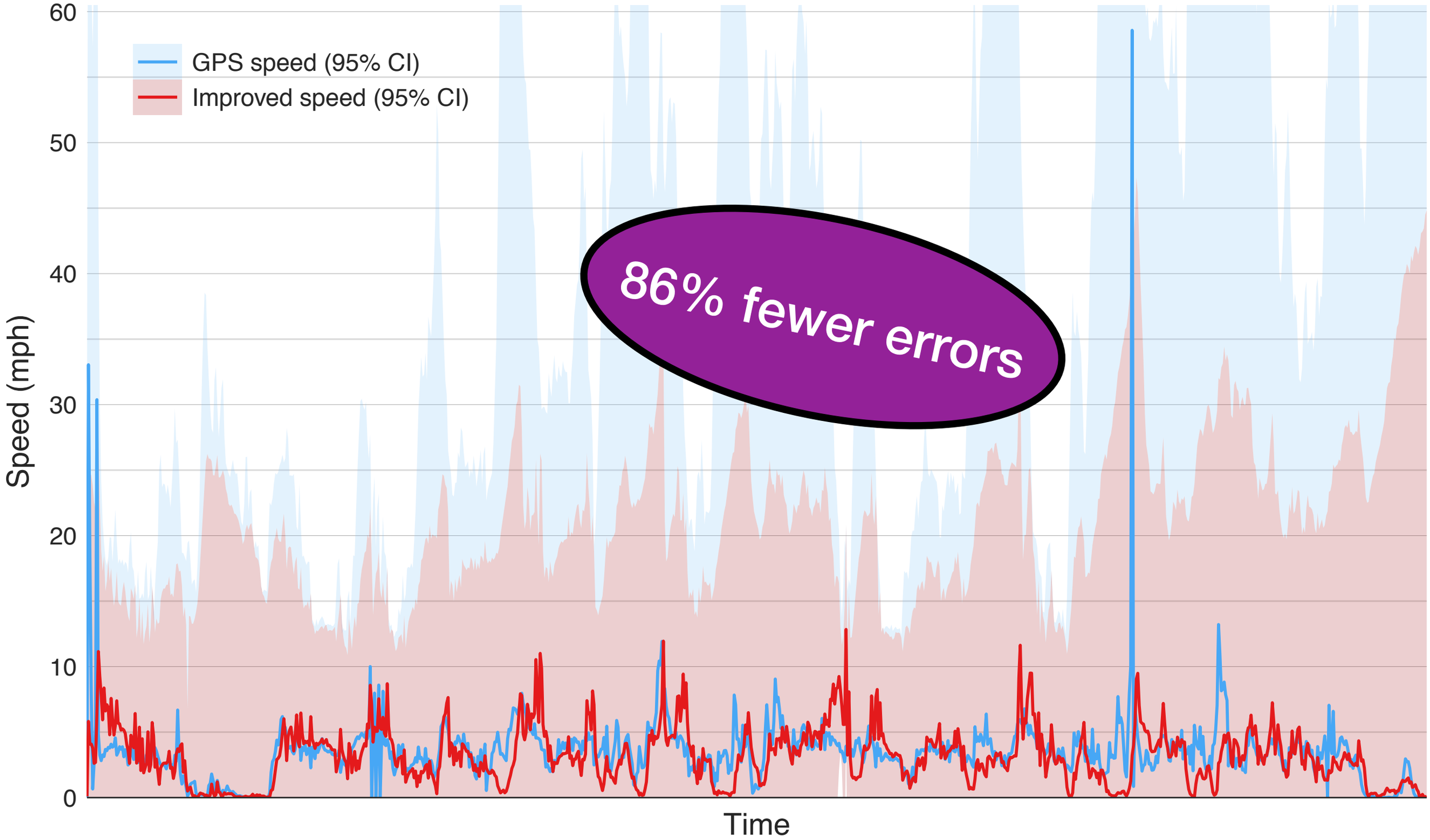
[PLDI'14]

```
GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
double Dist =
                  Distance(PrevLocn, Location);
double Speed = Dist / 5;


Print(Speed);
```


GPS speed (95% CI)
60
50
40
30
20
10
0
Speed (mph)
Time

```
GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
double Dist =
                  Distance(PrevLocn, Location);
double Speed = Dist / 5;


Print(Speed);
```

```
Uncertain<GeoCoordinate> PrevLocn = Get();
Sleep(5);
Uncertain<GeoCoordinate> Location = Get();
Uncertain<double> Dist =
                    Distance(PrevLocn, Location);
Uncertain<double> Speed = Dist / 5;


Print (Speed);  // Expected value at 95 CI
```


GPS speed (95% CI)
Improved speed (95% CI)
86% fewer errors
60
50
40
30
20
10
0
Speed (mph)
Time

Fitness application

```
Uncertain<GeoCoordinate> PrevLocn = Get();
Sleep(5);
Uncertain<GeoCoordinate> Location = Get();
Uncertain<double> Dist =
                    Distance(PrevLocn, Location);
Uncertain<double> Speed = Dist / 5;

if (Speed > 4)
     Alert(“Keep it up!”);
```

Hypothesis Test

more likely than

not Speed > 4

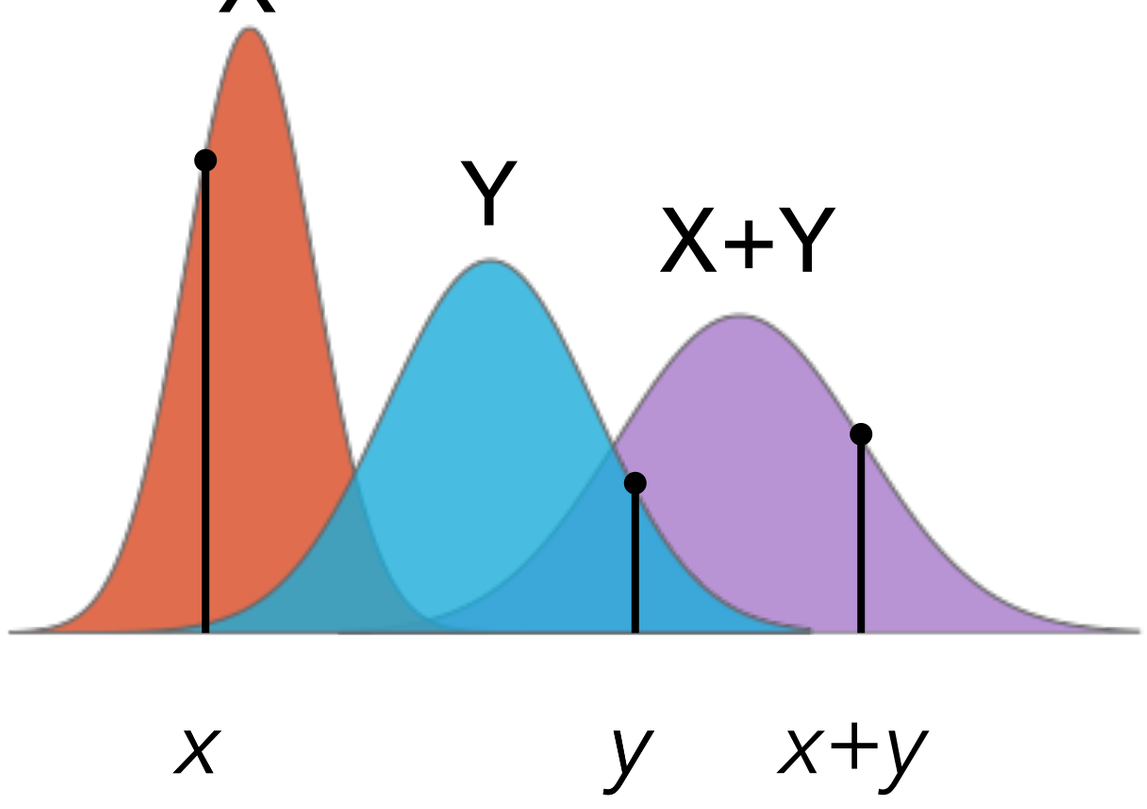
Semantics

Uncertain<T> encapsulates probability distributions and hides statistical complexity.

- Computing over random variables
- Evaluating expected value & conditionals

Computations

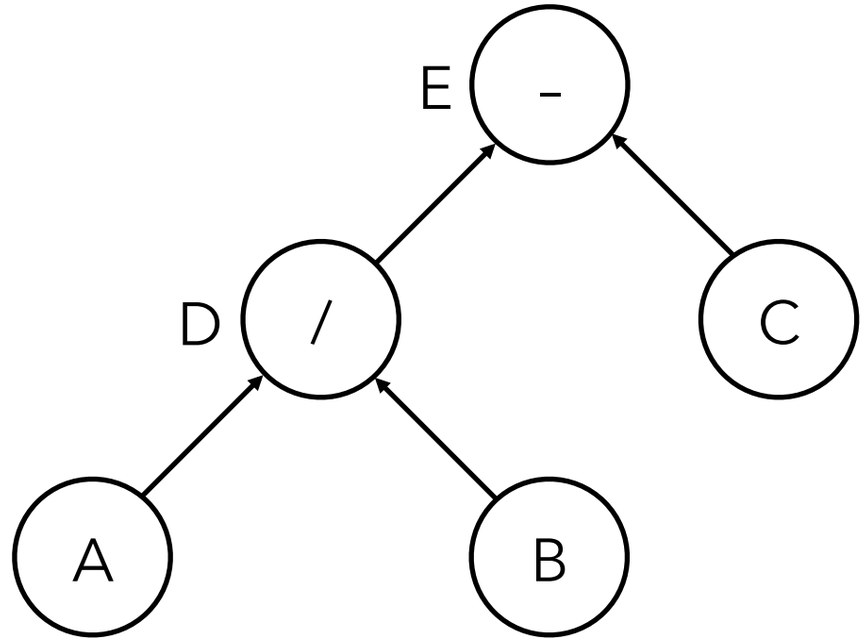
Represent distributions by random samples



Computations - lazy evaluation

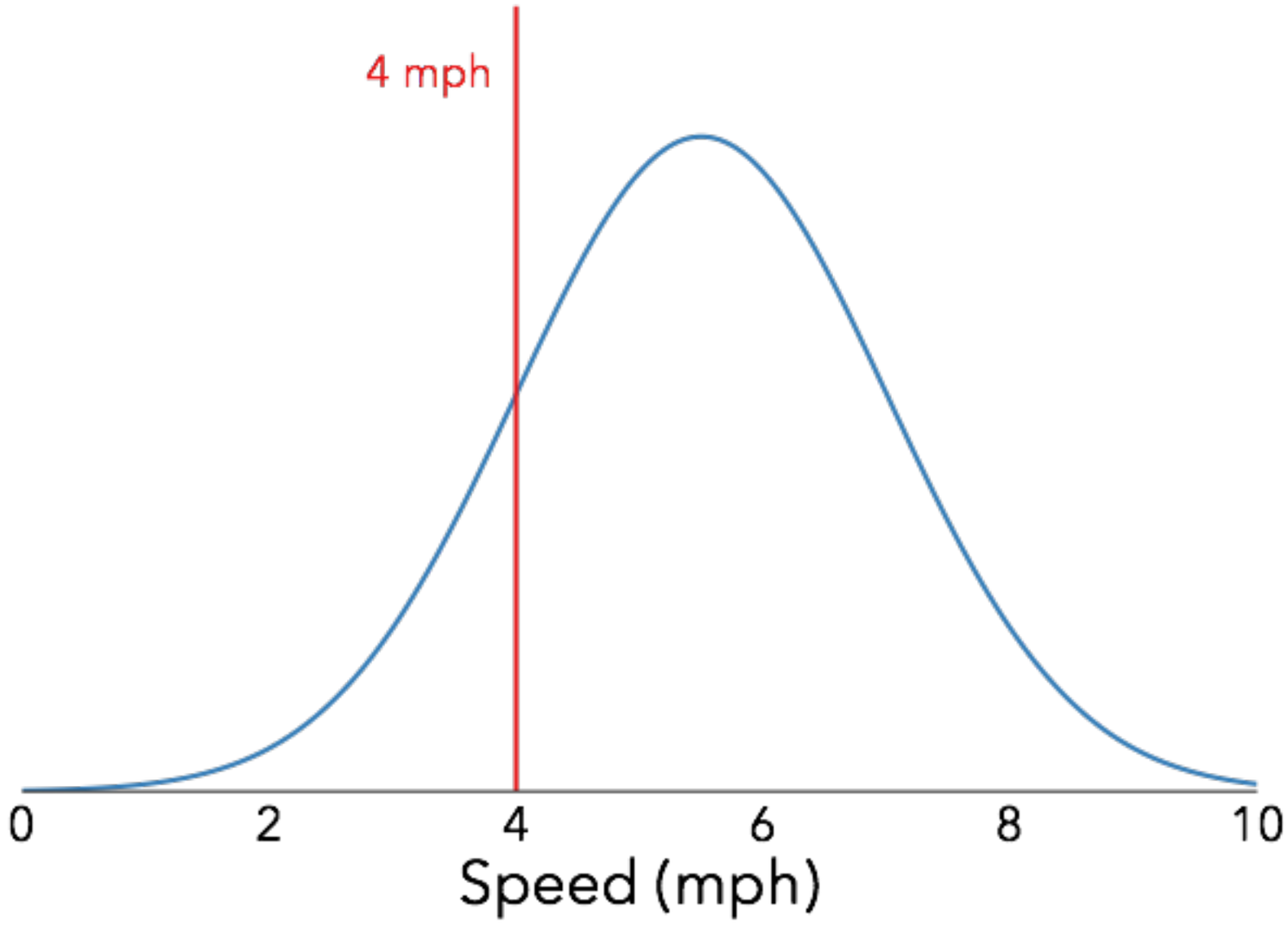
Operators build a Bayesian network rather than evaluating immediately.

D = A / B
E = D - C



Evaluating conditionals

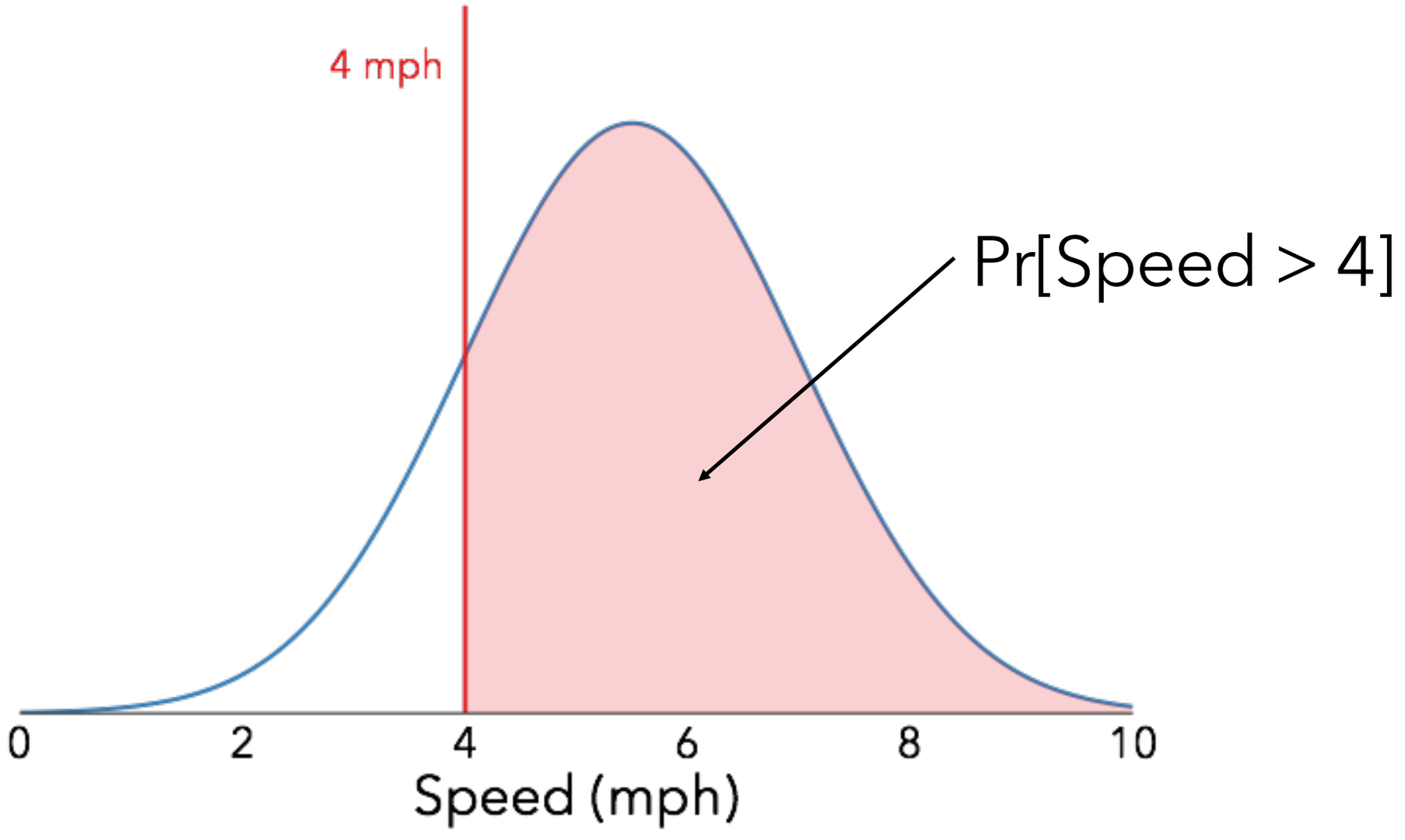
```
if (Speed > 4)
     Alert("Keep it up!");
```



Evaluating conditionals

```
if (Speed > 4)
     Alert("Keep it up!");
```

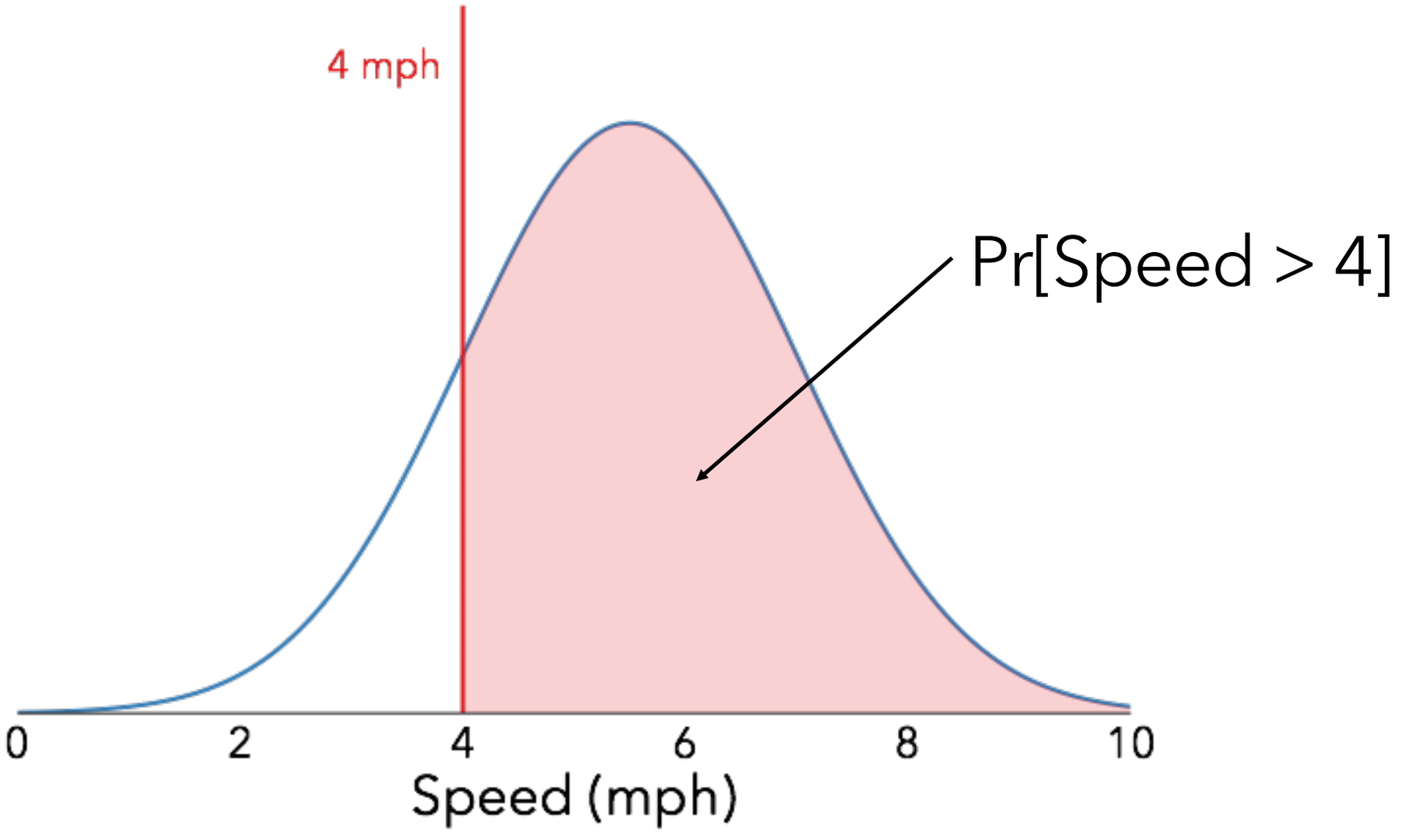
More likely than not that Speed > 4?



Evaluating conditionals

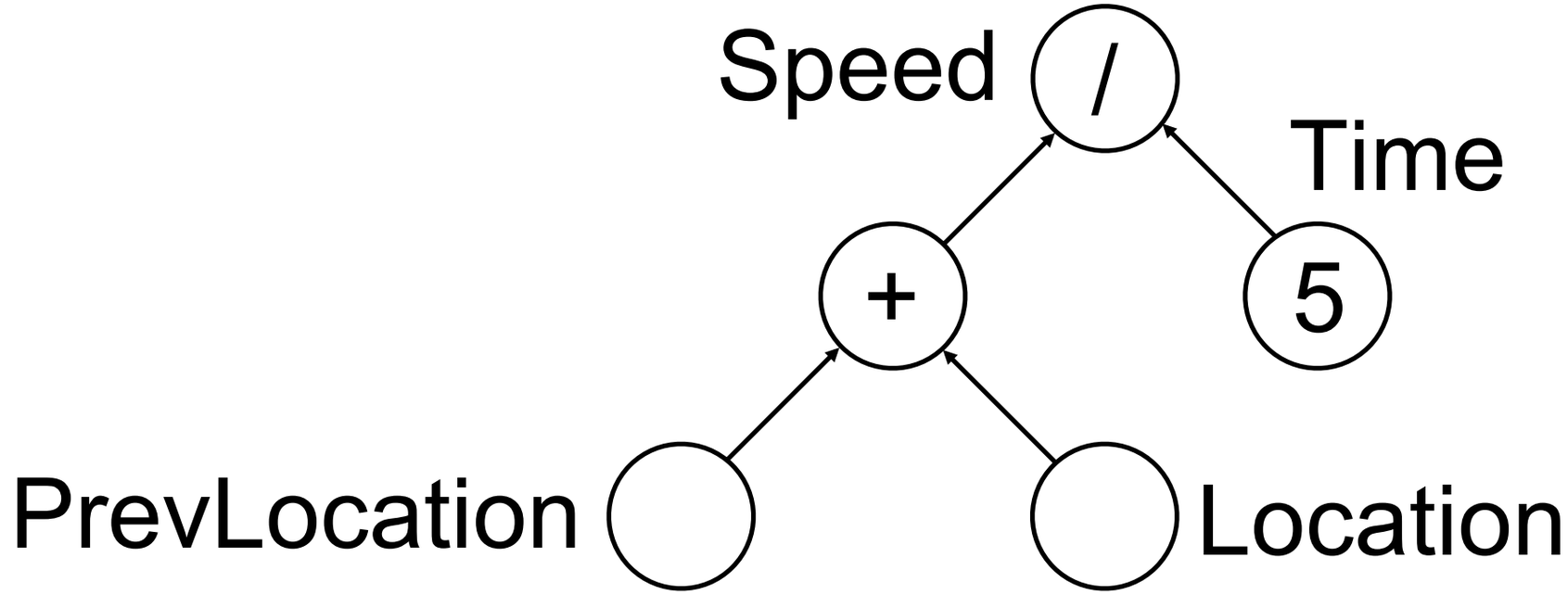
```
if ((Speed > 4).Pr(0.9))
    Alert("Keep it up!");
```

At least 90% likely that Speed > 4?



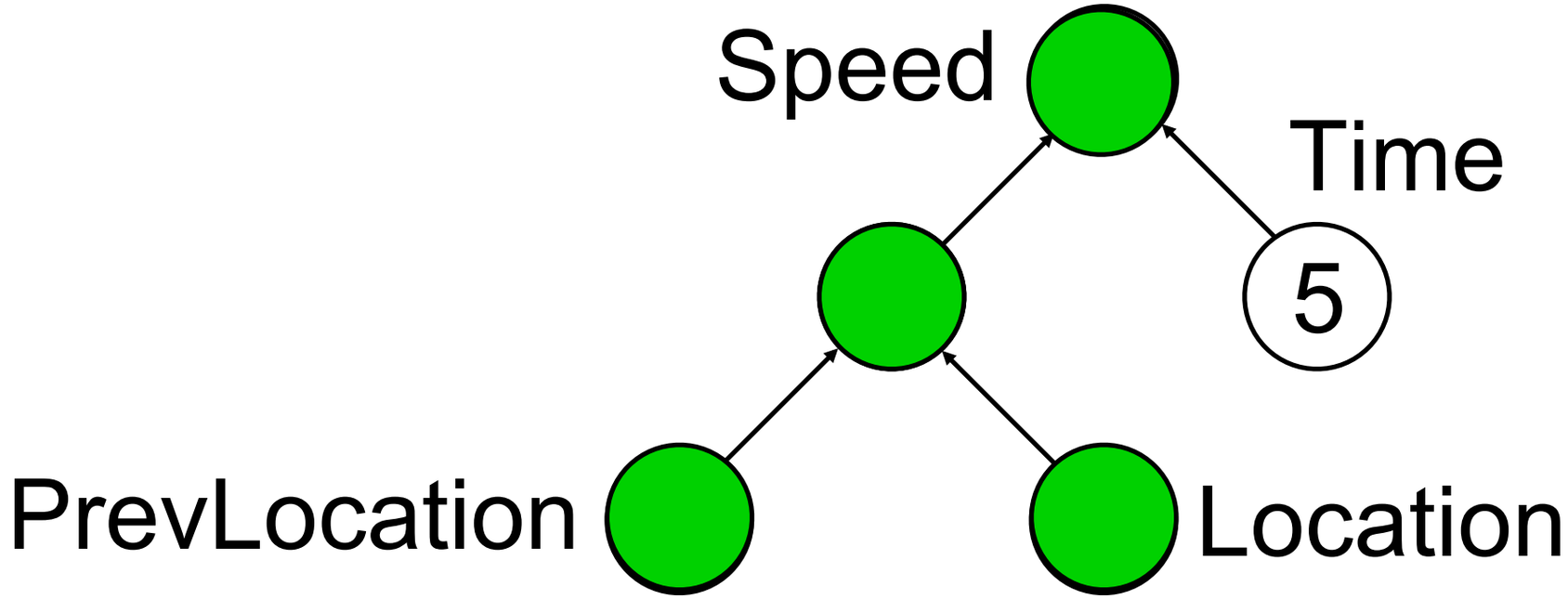
Sampling at runtime

```
if (Speed > 4)
     Alert("Keep it up!");
```



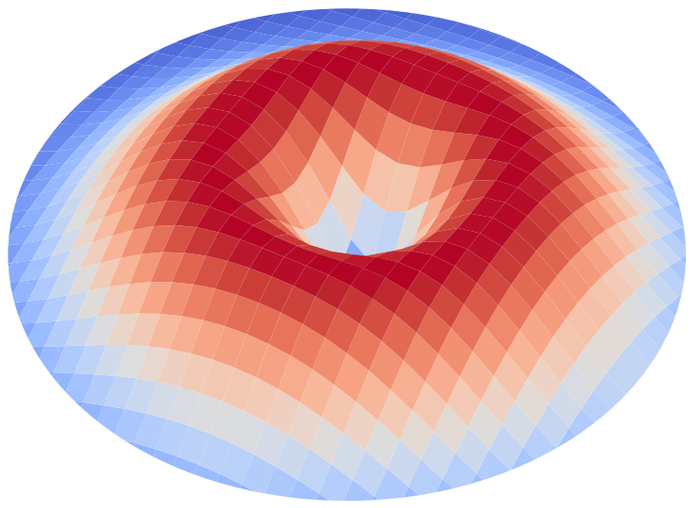
Sampling at runtime

```
if (Speed > 4)
     Alert("Keep it up!");
```



Sampling functions

Data source responsible for describing errors in estimate



3 D Raleigh Distribution
GPS error

Arbitrary
non-continuous functions

Sampling functions

Data source responsible for describing errors in estimate

```
GeoCoordinate Sample (GeoCoordinate location,
    double accuracy) {

    return SampleRayleigh(location, accuracy);
}
```

Identifying absurd data

```
GeoCoordinate PrevLocn = Get();
Sleep(5);
GeoCoordinate Location = Get();
double Dist =
                  Distance(PrevLocn, Location);
double Speed = Dist / 5;

if (Speed > 4)
     Alert("That’s crazy!");
```

Naïve 30 false positives

Identifying absurd data

```
Uncertain<GeoCoordinate> PrevLocn = Get();
Sleep(5);
Uncertain<GeoCoordinate> Location = Get();
Uncertain<double> Dist =
                    Distance(PrevLocn, Location);
Uncertain<double> Speed = Dist / 5;

if (Speed > 4)
     Alert("That's crazy!");
```

Naïve 30 false positives

50% 4 false positives

Identifying absurd data

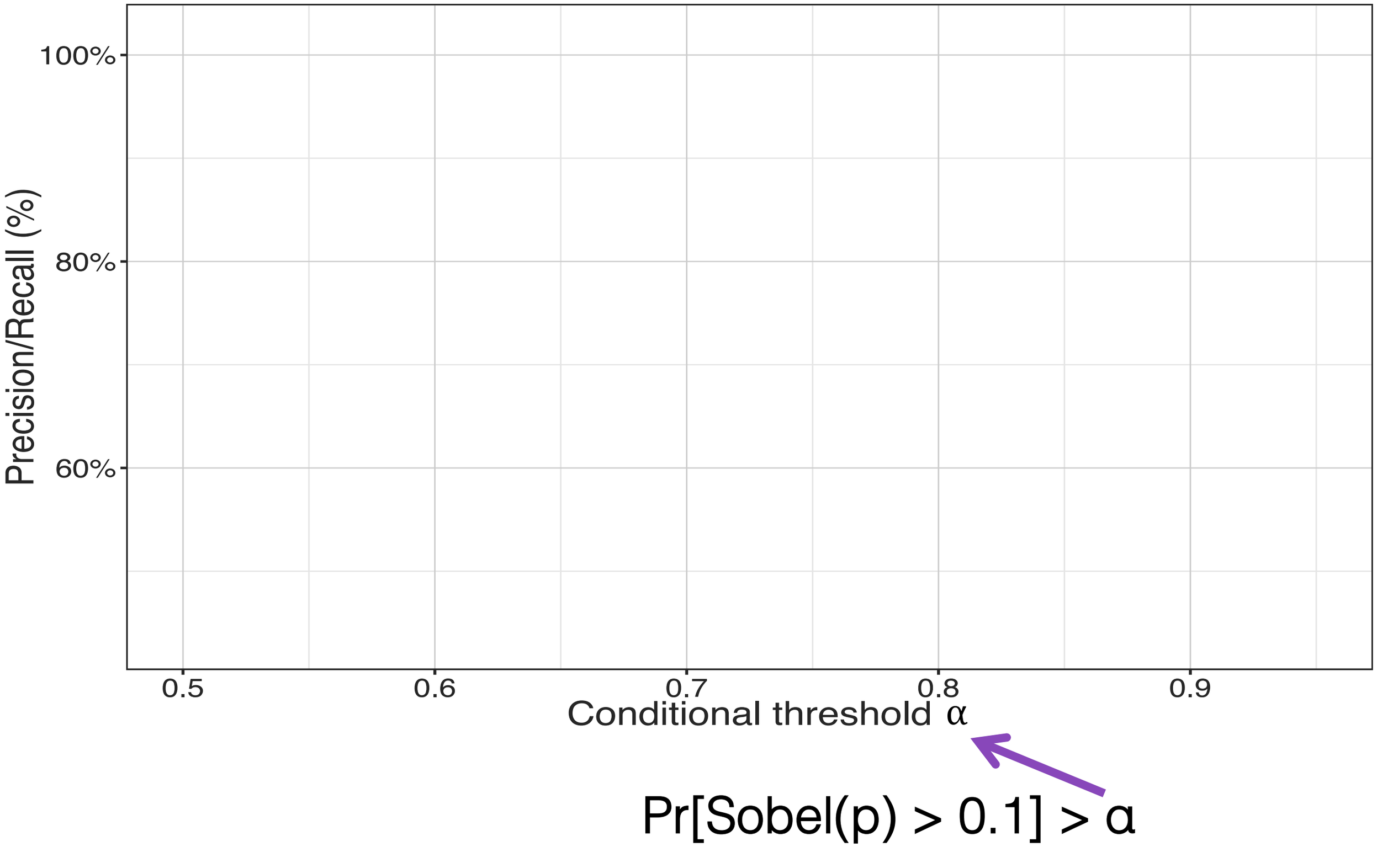
```
Uncertain<GeoCoordinate> PrevLocn = Get();
Sleep(5);
Uncertain<GeoCoordinate> Location = Get();
Uncertain<double> Dist =
                    Distance(LastLocn, Location);
Uncertain<double> Speed = Dist / 5;

if ((Speed > 4).Pr(0.9))
     Alert("That’s crazy!");
```

Naïve 30 false positives

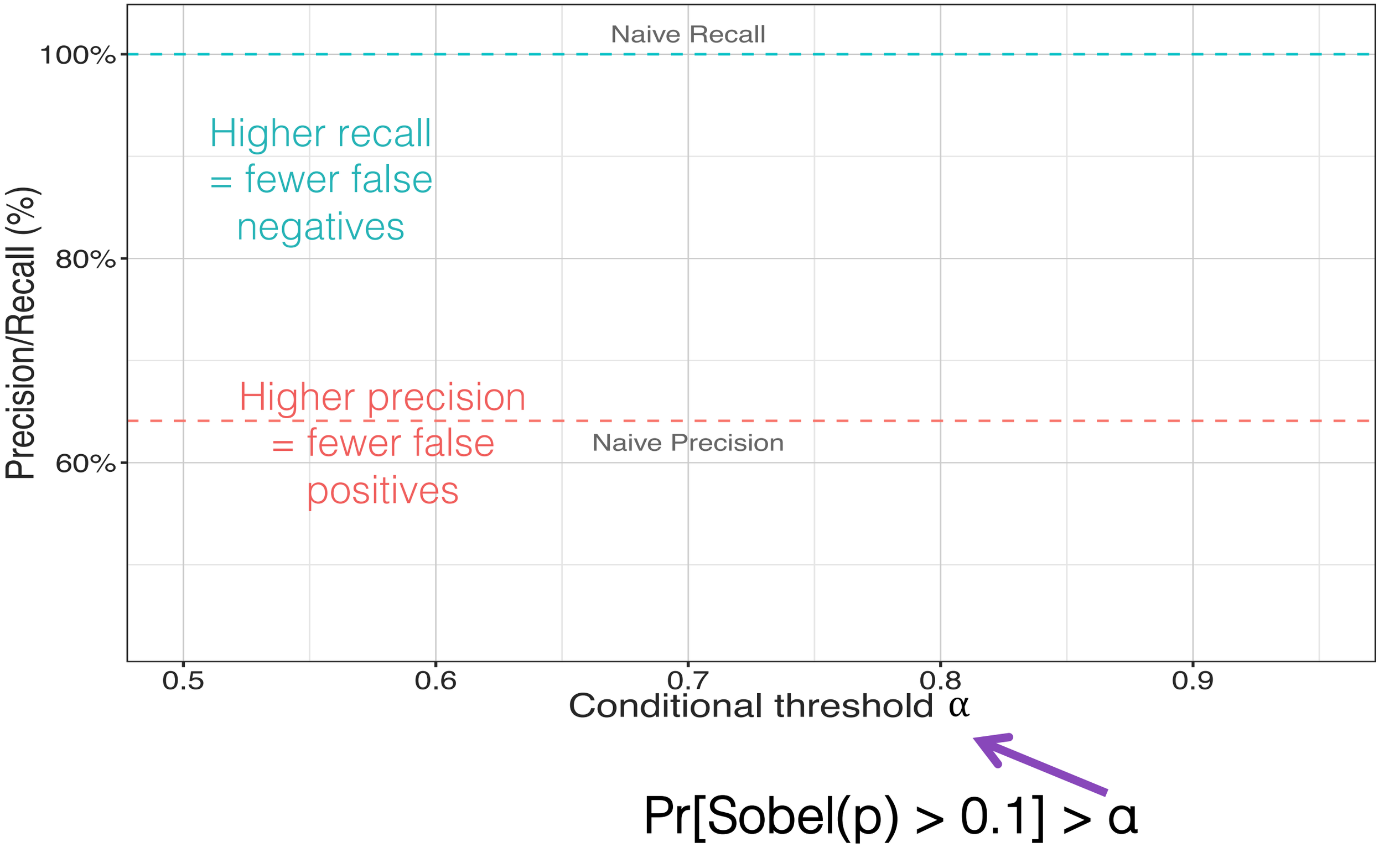
50% 4 false positives

90% none

Edge Detection

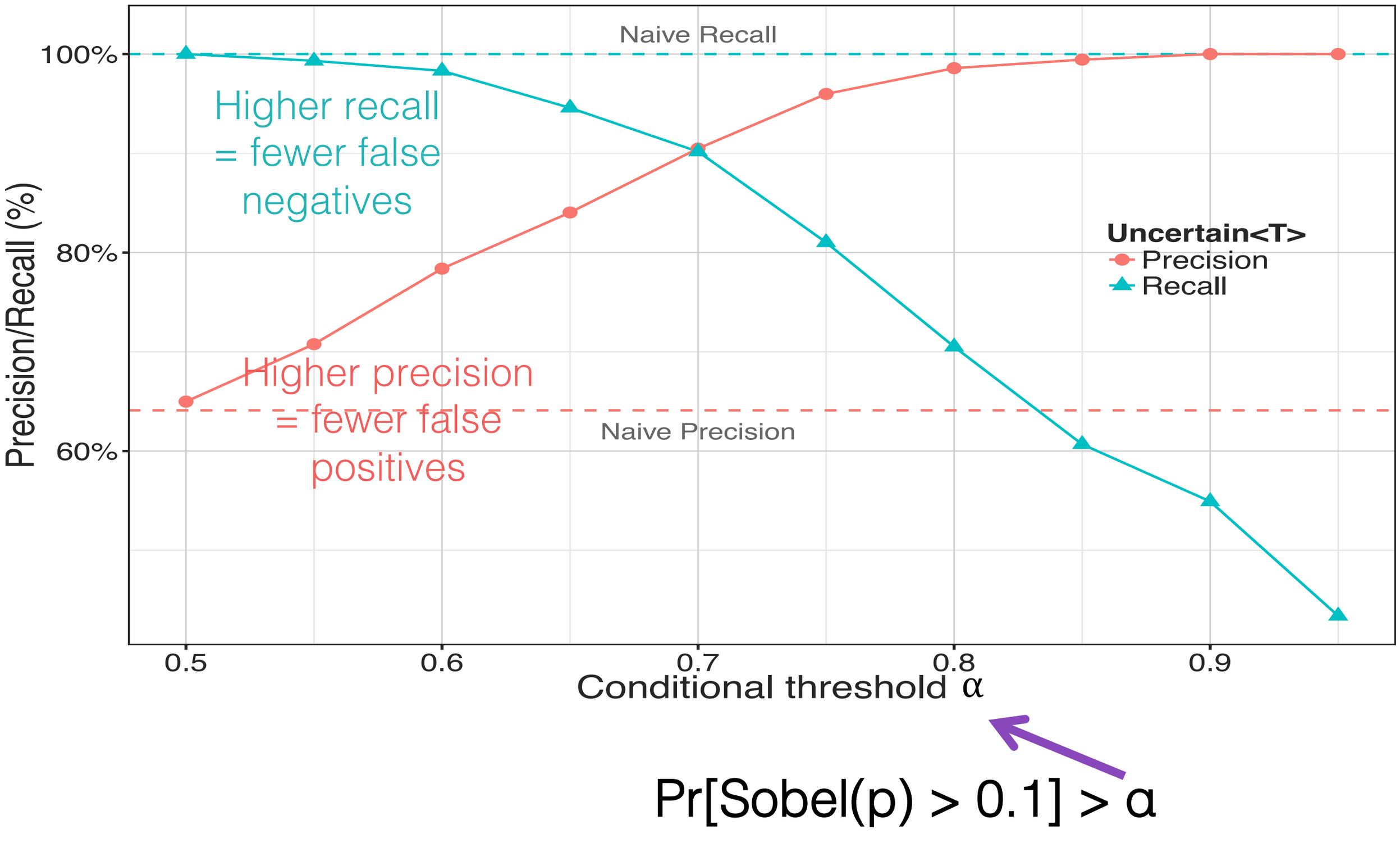


Edge Detection



Pr[Sobel(p) > 0.1] > α

Edge Detection



Pr[Sobel(p) > 0.1] > α

Uncertain<*T*>

an abstraction for reasoning about noise

[ASPLOS'14]

exploiting context

improving accuracy with more language & inference

[rejected so far]

what does it mean?

probabilistic assertions

[PLDI'14]

Context

AUSTIN RATINGS & REVIEWS LISTS BUZZ

Search Austin

Austin **restaurants**

Zagat Ratings

Food
0 or higher

Decor
0 or higher

Service
0 or higher

Cost
$134 or lower

1-15 OF 492 Sort by: **Relevance**

Uchi Restaurant
Japanese Bouldin Creek

FOOD	DECOR	SERVICE	COST
29	27	27	$73

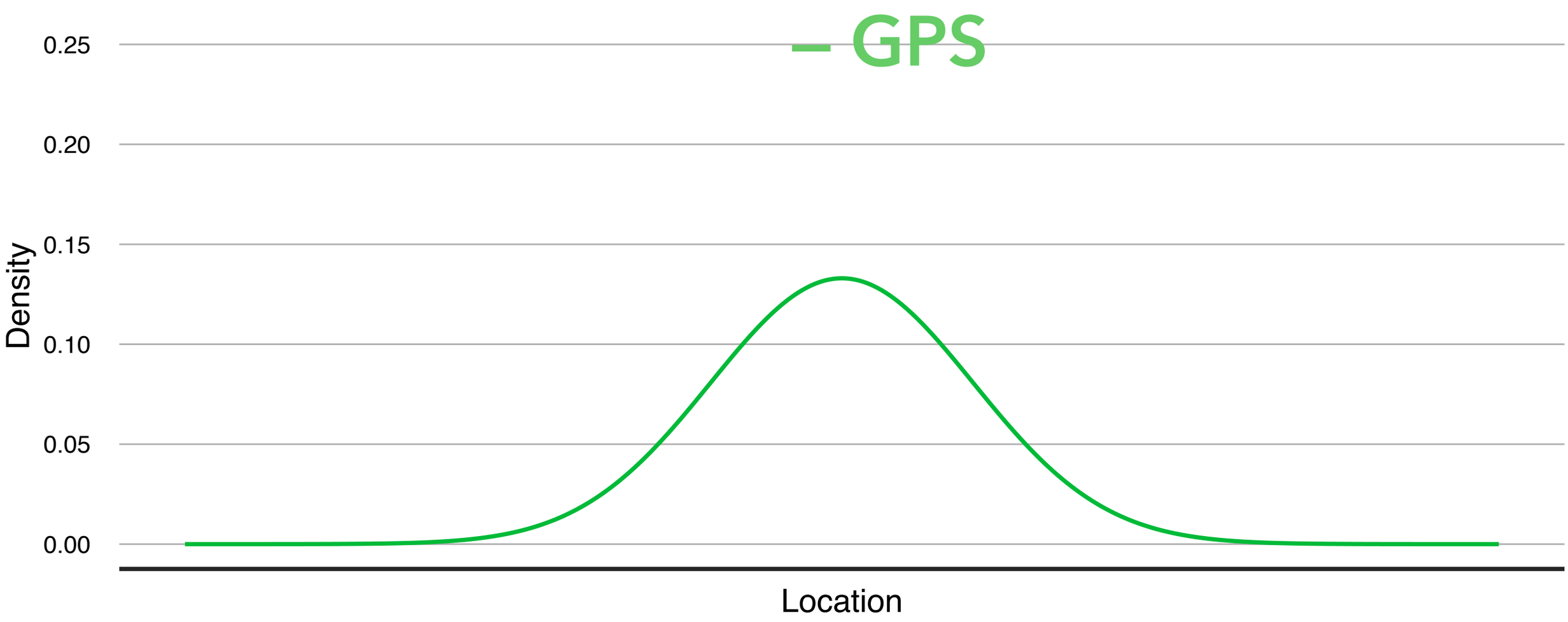
Franklin Barbecue
Barbecue East Austin

FOOD	DECOR	SERVICE	COST
28	20	23	$23

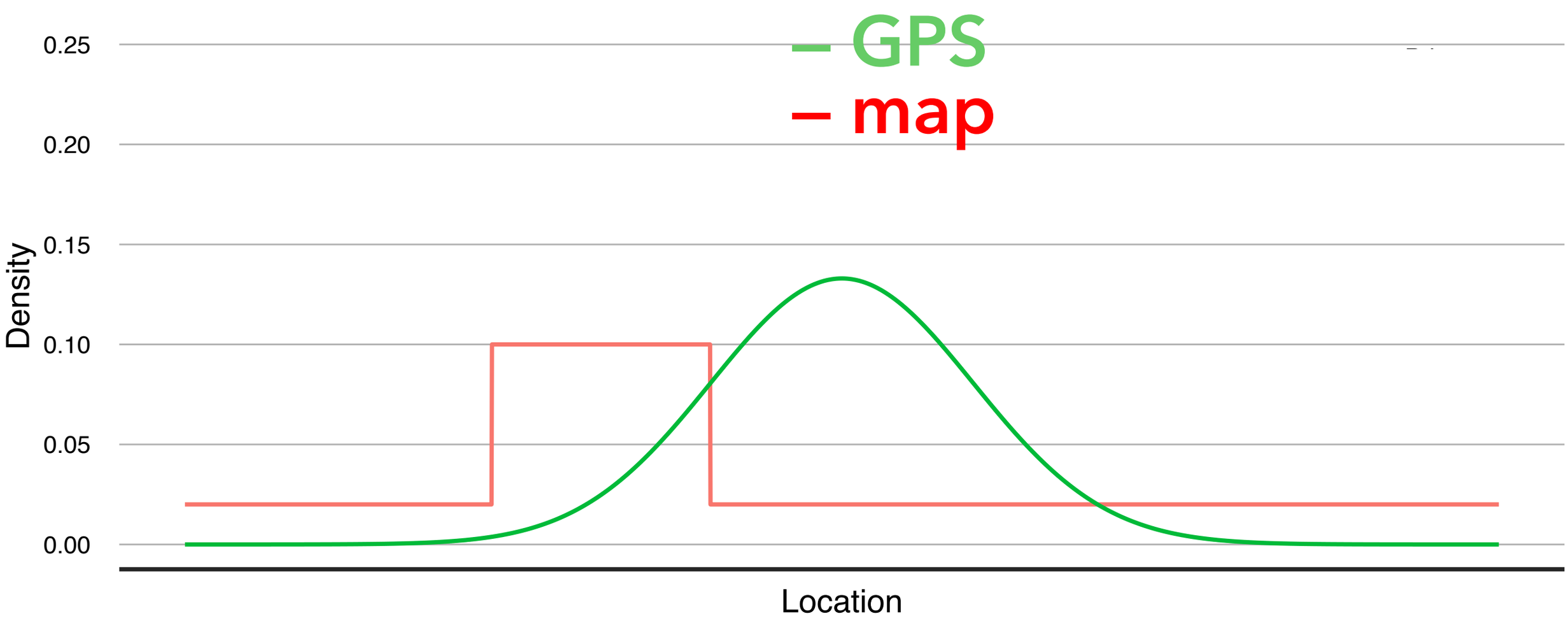
Uchiko
Japanese Rosedale

FOOD	DECOR	SERVICE	COST

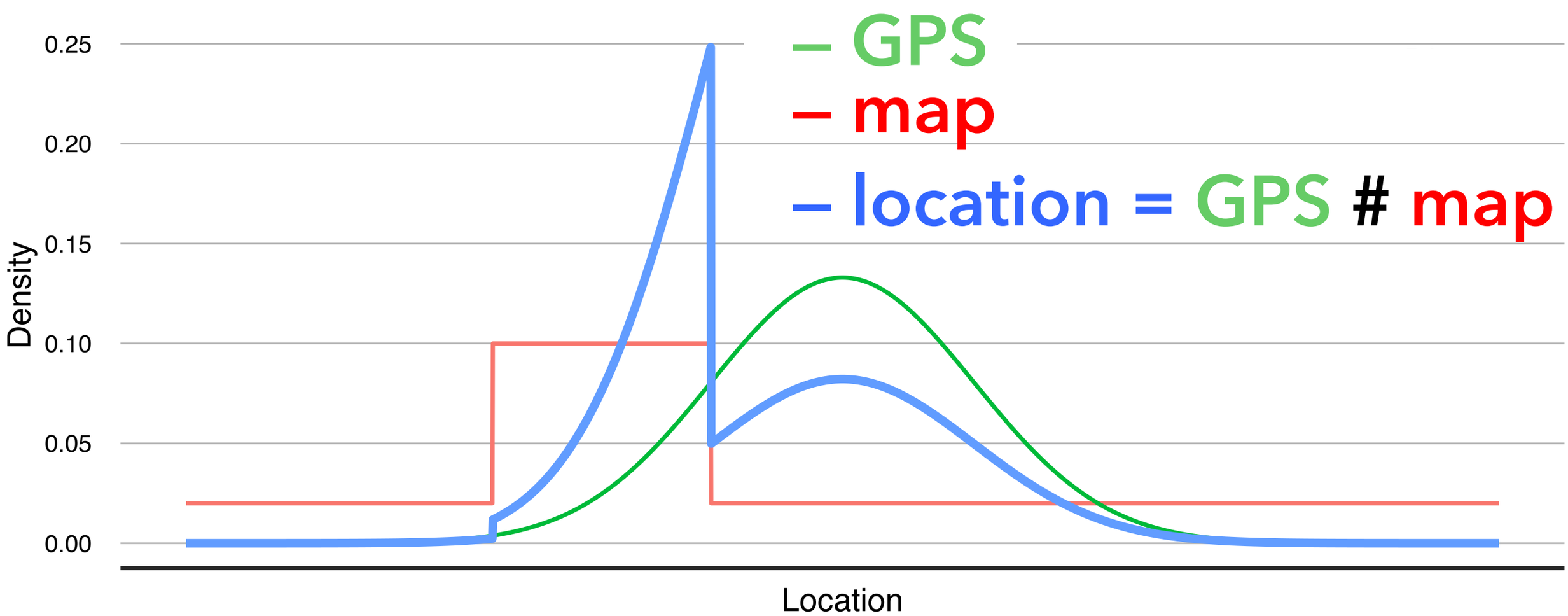
Adding domain knowledge



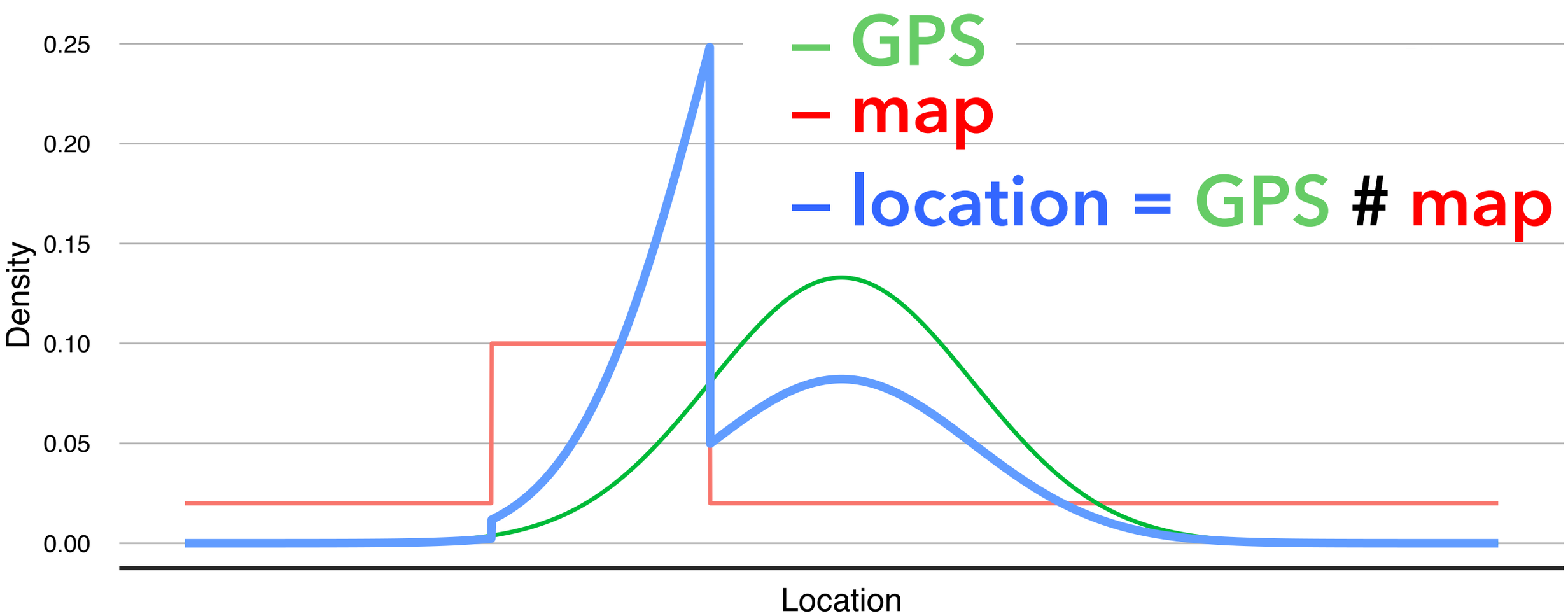
Adding domain knowledge



Adding domain knowledge



Adding domain knowledge



Bayes Rule

posterior likelihood prior

$$\Pr[H|E] = \Pr[E|H]\,\Pr[H] / \Pr[E]$$

Two constructs

<|

Building probability distributions

conditional probability

#

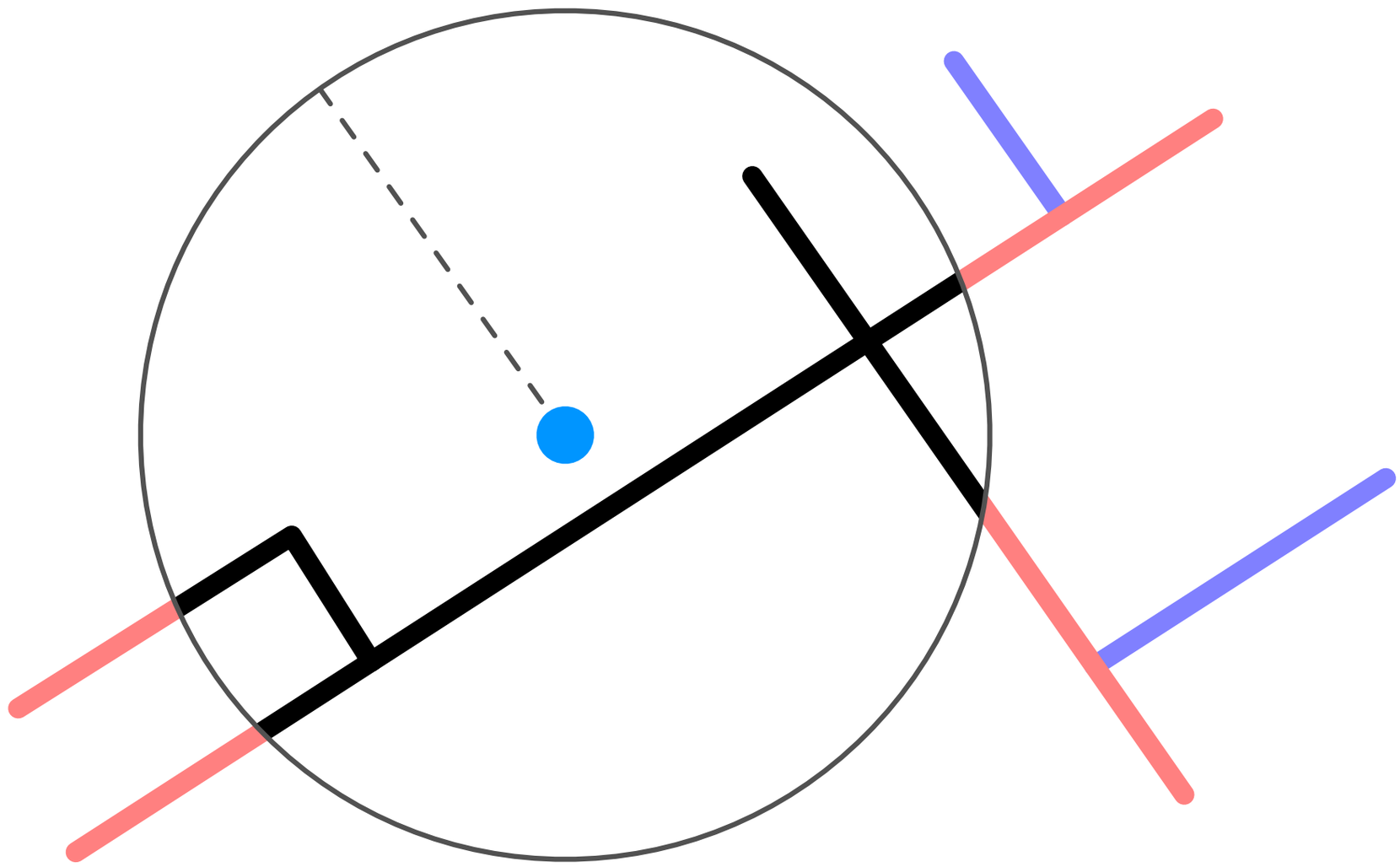
Composing context and estimates

Bayesian inference

Implementation a new *sequential likelihood reweighting* algorithm

GPS navigation – road snapping

```
// find relevant roads
Uncertain<Point> roadPrior = new uncertain<Point>(()=>
SamplePrior(location, accuracy, radiusFactor, weight))
```



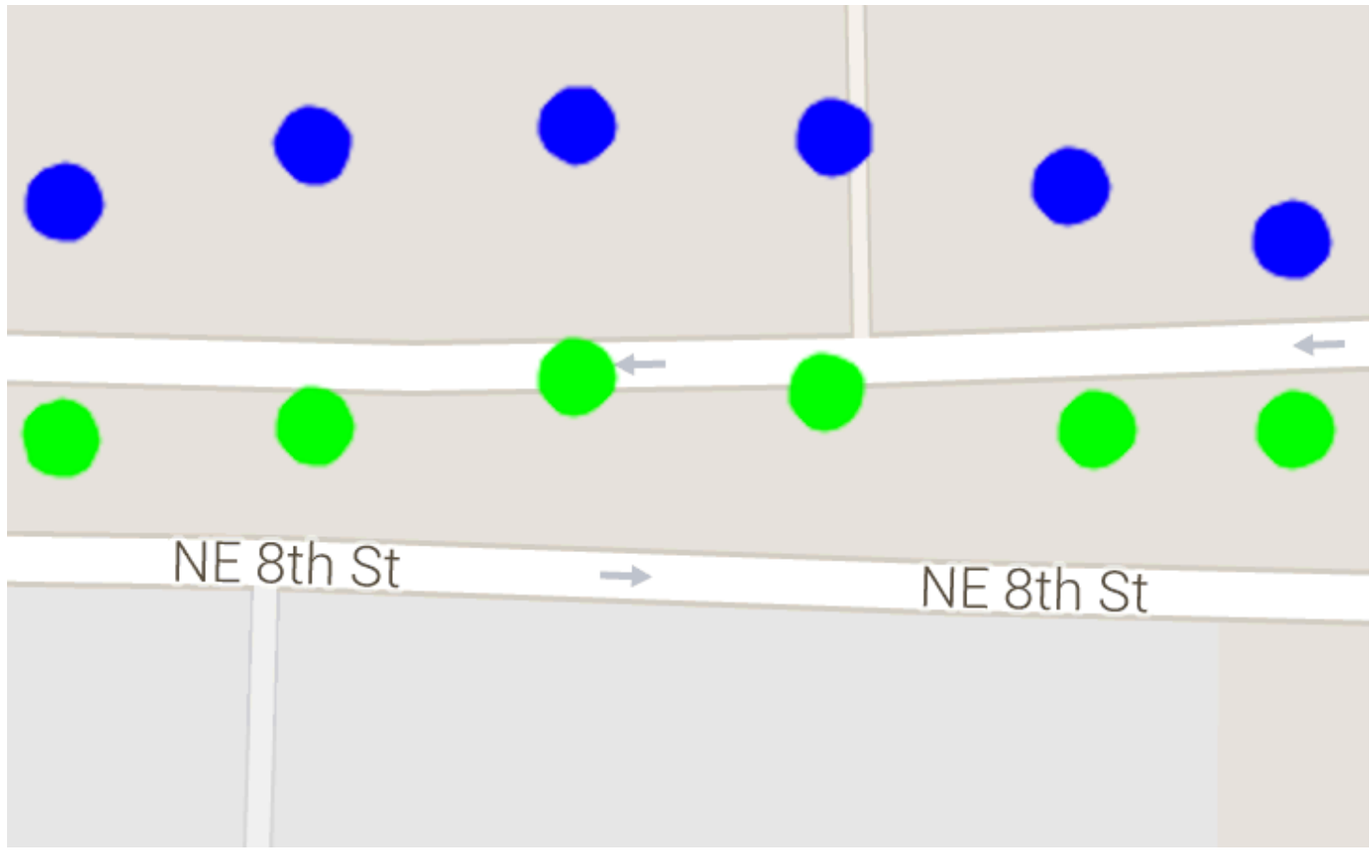
GPS navigation – road snapping

```
// find relevant roads
Uncertain<Point> roadPrior = new uncertain<Point>(()=>
SamplePrior(location, accuracy, radiusFactor, weight))

// improve location estimate
Uncertain<Point>
    NewLocation = GPSLikelihood # roadPrior
```

GPS navigation

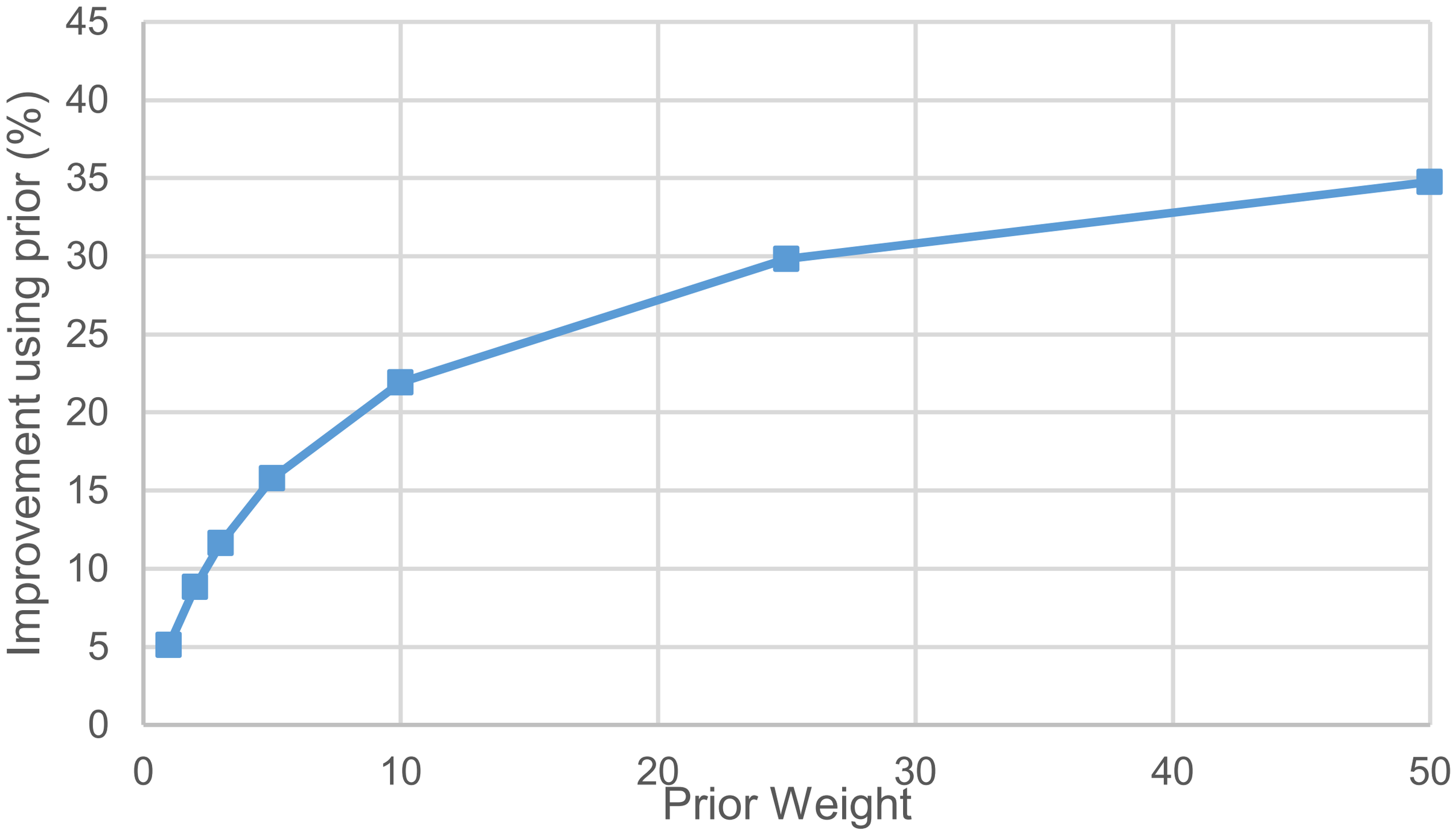
Driver is *likely* on a road!

Driving on a road (or not!)

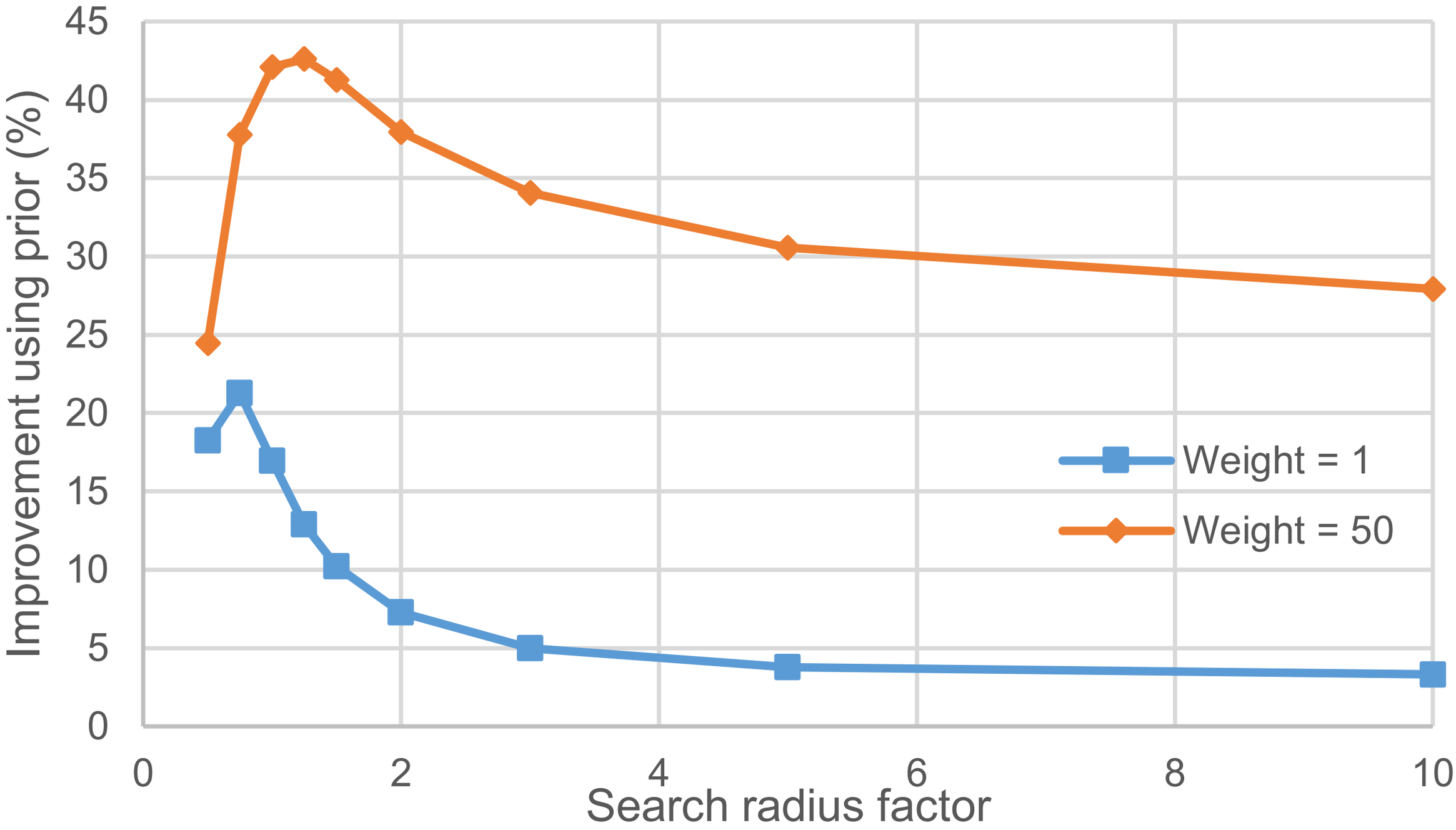


- GPS
- GPS + road snapping

Weighing the map prior

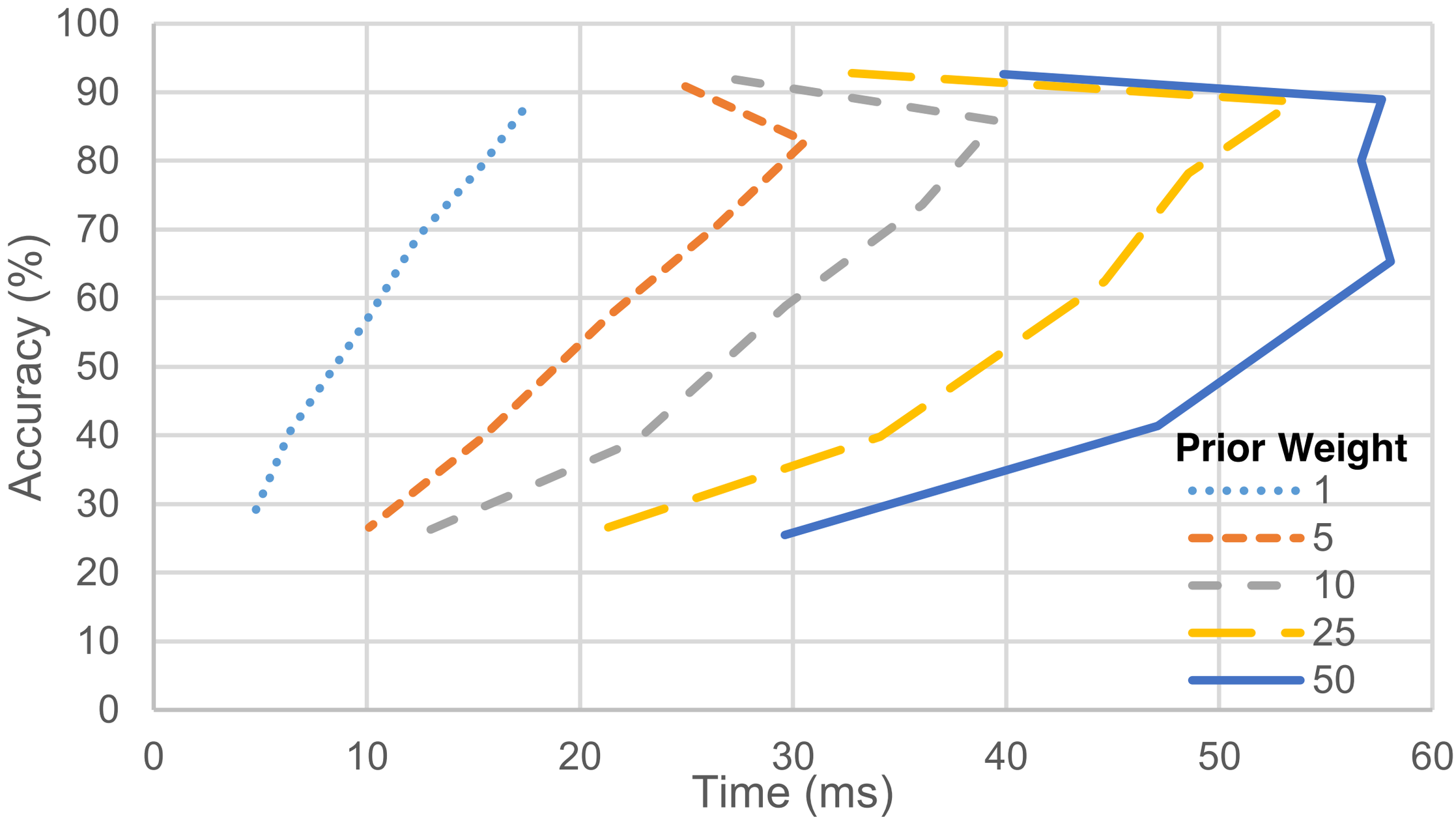


Map radius factor



Sequential likelihood reweighing

Time to sample as a function of prior strength



Uncertain<*T*>

an abstraction for reasoning about noise

[ASPLOS'14]

exploiting context

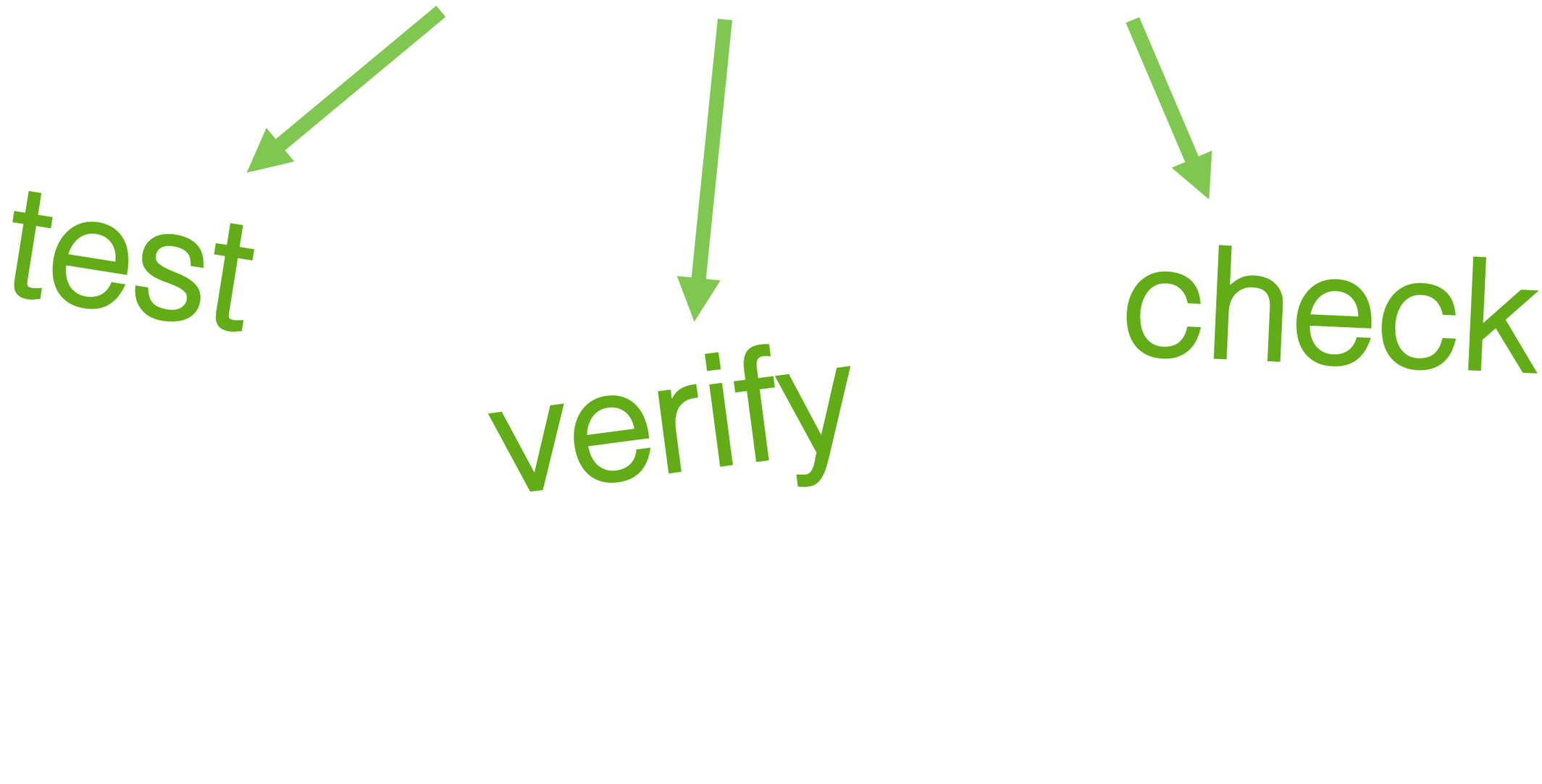
improving accuracy with more language & inference

[rejected so far]

what does it mean?

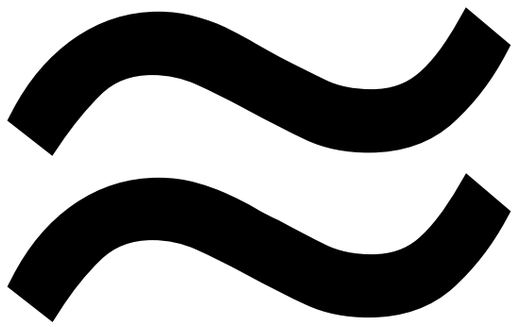
probabilistic assertions

[PLDI'14]


assert file != NULL
test
verify
check

assert *e*

e must hold on **every** execution



Approximate computing

The approximate image is close to the precise version

k-means clustering is likely to converge on unreliable hardware

`assert` *e*

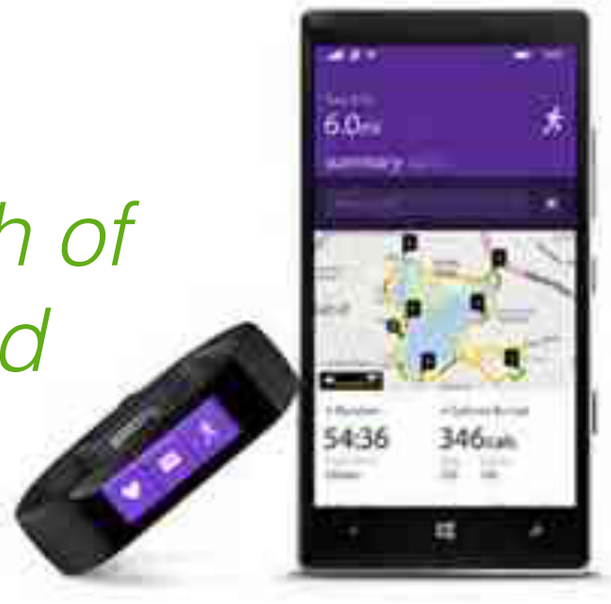
e must hold on **every** execution

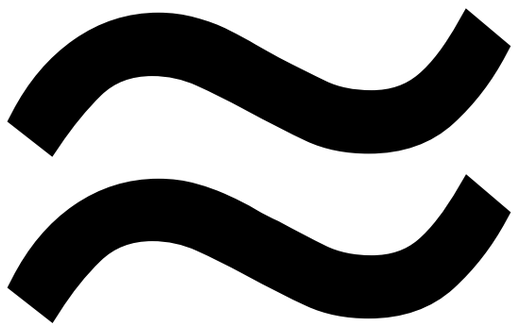
Obfuscation for privacy

obfuscated data is still useful in aggregate

Sensing

within 5 mph of actual speed





Approximate computing

The approximate image is close to the precise version

k-means clustering is likely to converge on unreliable hardware

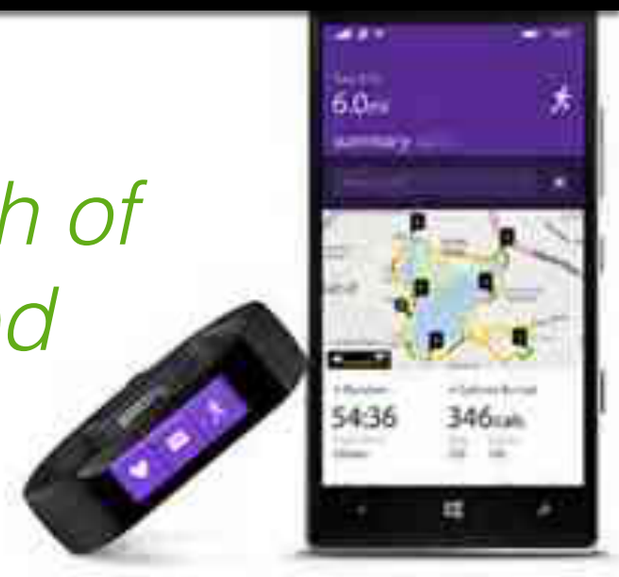
Traditional assertions are **insufficient** for programs with **probabilistic behavior**

Obfuscation for privacy

obfuscated data is still useful in aggregate

Sensing

within 5 mph of actual speed



Assertions are insufficient for data obfuscation

```
true_avg = average(salaries)
private_avg =
   average(obfuscate(salaries))
assert true_avg - private_avg
       <= 10,000
```

Assertions are insufficient for data obfuscation

```
true_avg = average(salaries)
private_avg =
   average(obfuscate(salaries))
assert true_avg - private_avg
       <= 10,000
```

probability distribution

Assertions

`assert` *e*

Probabilistic assertions

passert *e*, *p*, *c*

Probabilistic assertions

passert *e*, *p*, *c*

e must hold with probability *p*
at confidence *c*

Probabilistic assertion

```
true_avg = average(salaries)
private_avg =
   average(obfuscate(salaries))
passert (true_avg - private_avg
   <= 10,000), 90, 99
```

Verification

probabilistic program

```
float obfuscated(float n) {
  return n + gaussian(0.0, 1000.0);
}
float average_salary(float* salaries) {
  total = 0.0;
  for (int i = 0; i < COUNT; ++i)
    total += obfuscated(salaries[i]);
  avg = total / len(salaries);
  p_avg = ...;

passert e, p, c
```

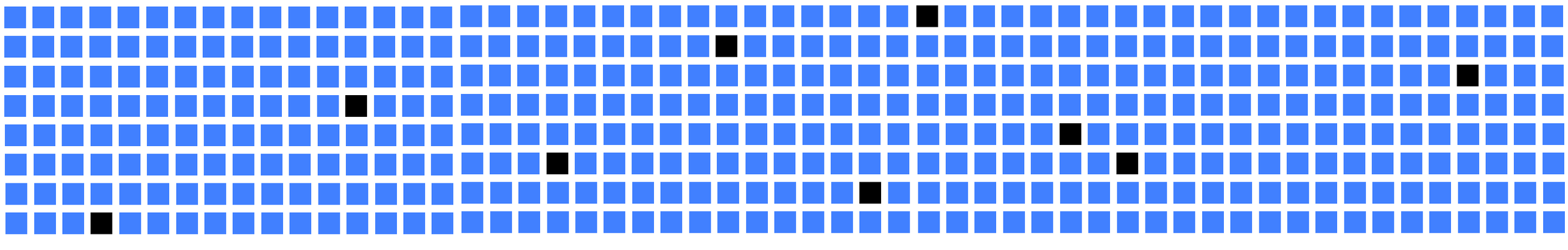
?

Naïve verification

probabilistic program

```
float obfuscated(float n) {
  return n + gaussian(0.0, 1000.0);
}
float average_salary(float* salaries) {
  total = 0.0;
  for (int i = 0; i < COUNT; ++i)
    total += obfuscated(salaries[i]);
  avg = total / len(salaries);
  p_avg = ...;

passert e, p, c
```

?



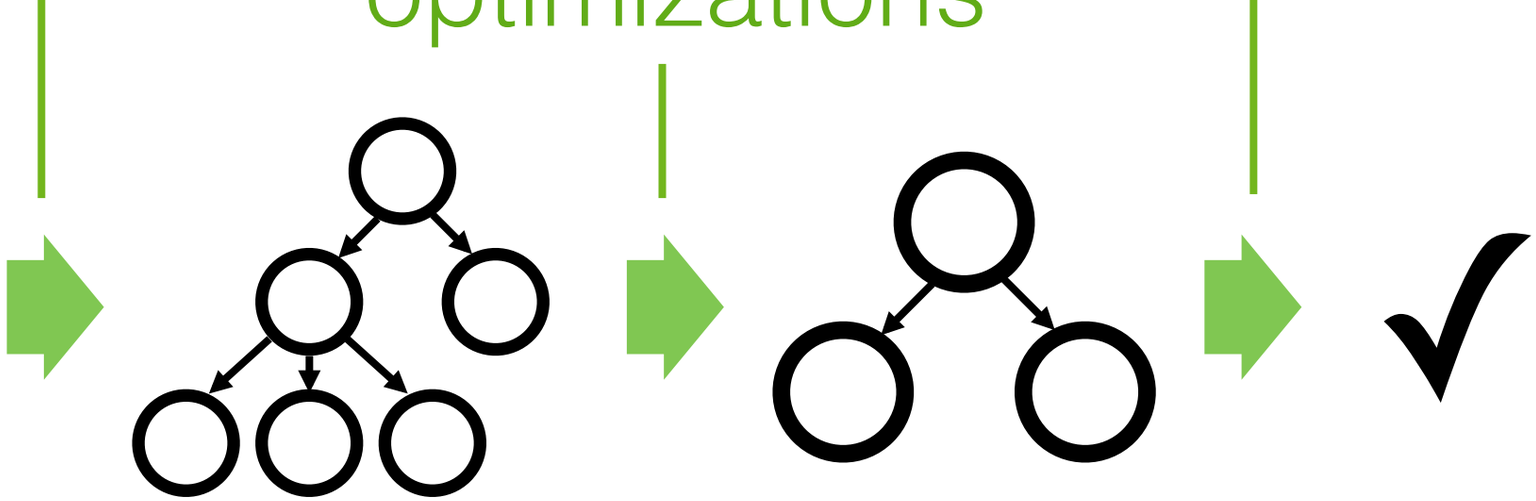
Efficient verification

```
float obfuscated(float n) {
  return n + gaussian(0.0, 1000.0);
}
float average_salary(float* salaries) {
  total = 0.0;
  for (int i = 0; i < COUNT; ++i)
    total += obfuscated(salaries[i]);
  avg = total / len(salaries);
  p_avg = ...;
passert e, p, c
}
```

distribution extraction
via symbolic execution

statistical
optimizations

verification

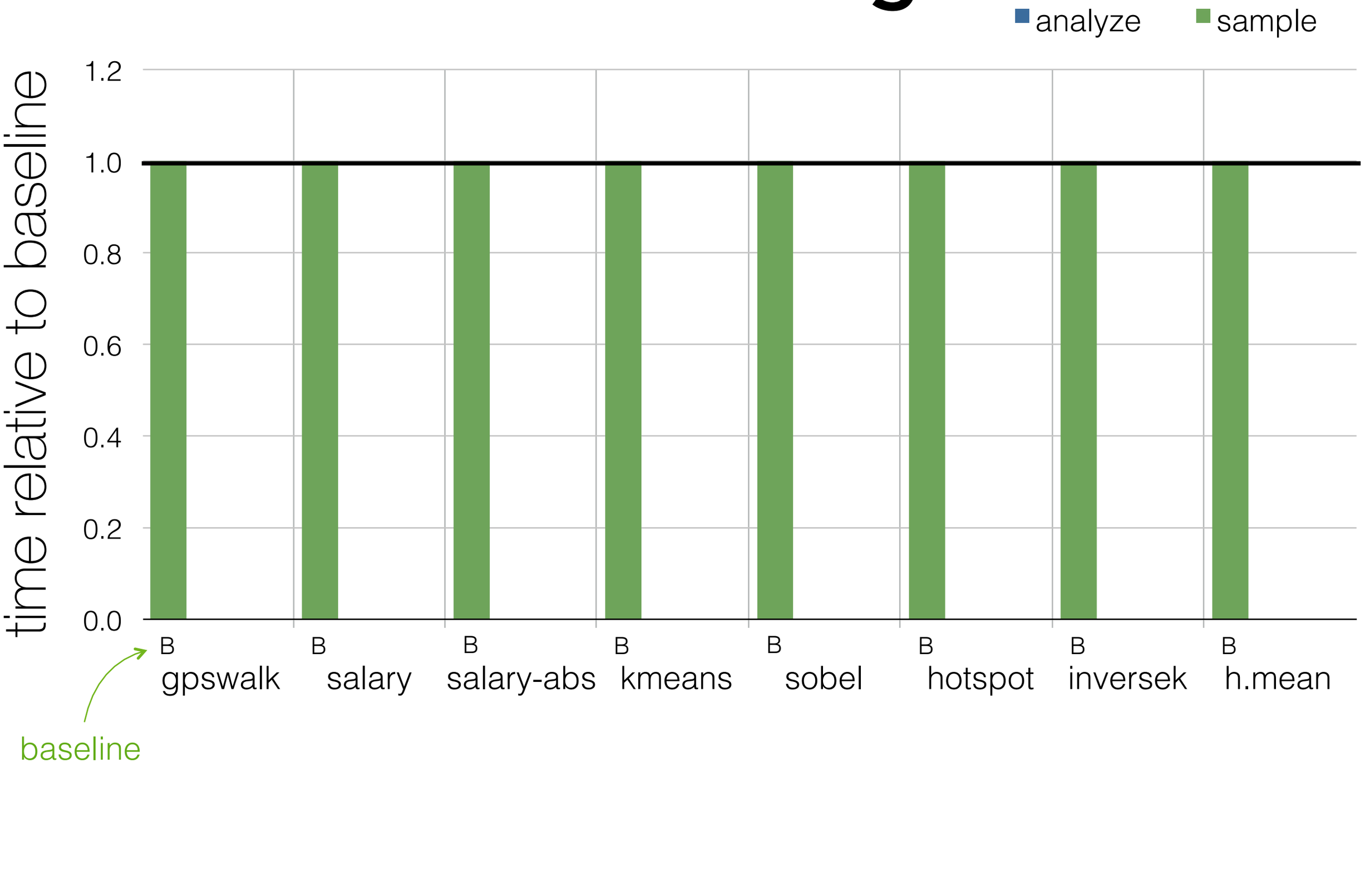


Bayesian network
IR

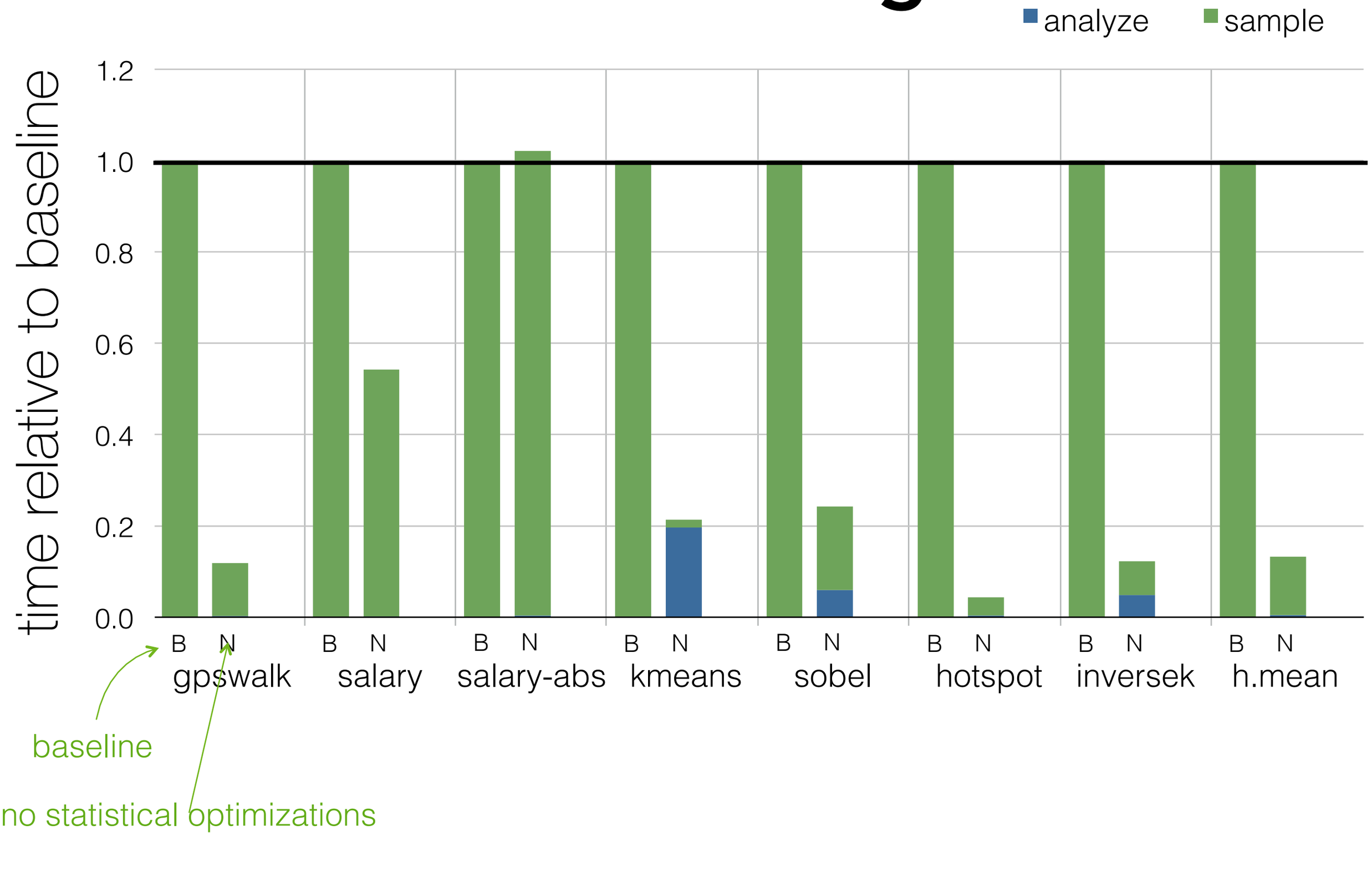
Evaluation

sensing	gpswalk
privacy	salary
	salary-abs
approximate computing	kmeans
	sobel
	hotspot
	inversek2j

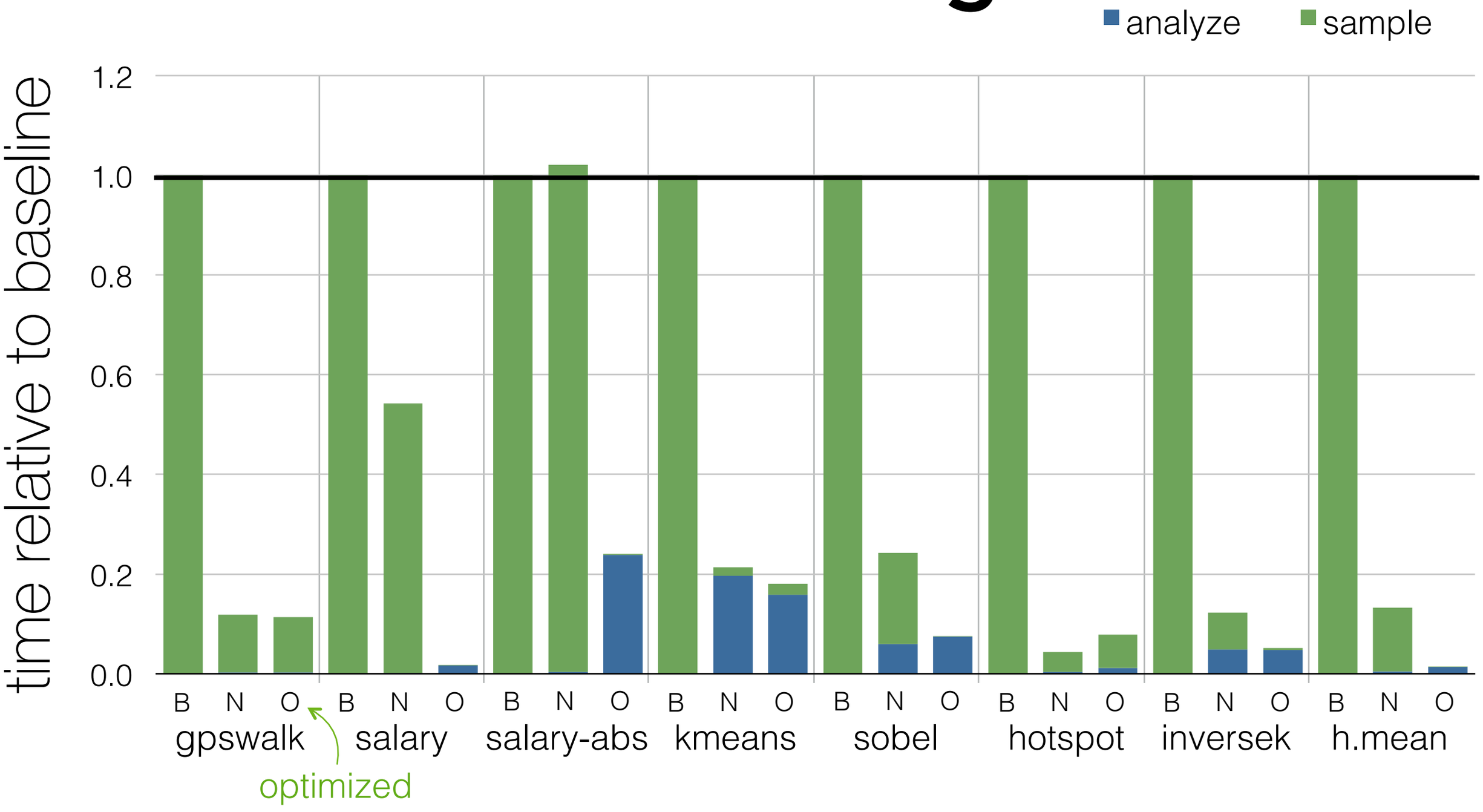
Time vs Stress Testing



Time vs Stress Testing



Time vs Stress Testing



- 24× faster than baseline verifier on average
- Mostly analysis time

Uncertain<*T*>

an abstraction for reasoning about noise

[ASPLOS'14]

exploiting context

improving accuracy with more language & inference

[rejected so far]

what does it mean?

probabilistic assertions

[PLDI'14]

1985 IEEE floating point

Computing is at a similar point in history for estimates

Thank you!

A Byte of My Story

A Byte of My Story

Mentors

ACM Fellow

Family

Congressional Testimony

Fail, learn, succeed, repeat

- Rejected job applications
  - 1984 (all), 1993 (8 of 11), 2011 (4 of 8)
- Failed PhD qualifying exam
- Rejected first three grant applications
- Rejected 3 times one of my most cited papers
- Rejected papers, grants, papers, …