GTP Benchmarks for Gradual Typing Performance

Ben Greenman

BROWN

THE UNIVERSITY OF UTAH

Benchmarks + Experiments are important.

Benchmarks + Experiments are important.

Must be:

- Relevant
- Rigorous
- Reproducible

Benchmarks + Experiments are important.

Must be:
- Relevant
- Rigorous
- Reproducible

Example:

the DaCapo benchmark suite

Benchmarks + Experiments are important.

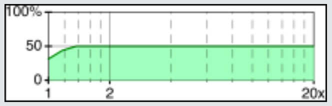
Must be:
- Relevant
- Rigorous
- Reproducible

Benchmarks + Experiments are important.

Must be:
- Relevant
- Rigorous
- Reproducible

benchmarks

measurement



visualization

How to encourage **domain-specific** benchmarks?

How to encourage **domain-specific** benchmarks?

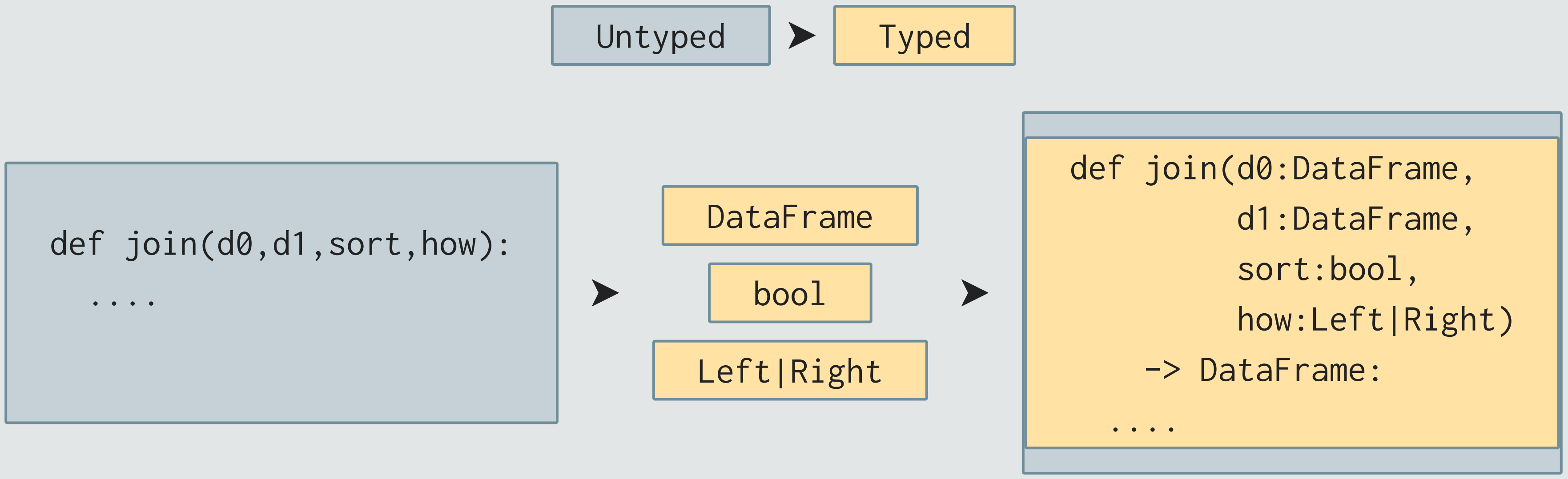
Main takeaway: **think like a practitioner**

GTP = Gradual Typing Performance

Gradual Typing

$$\begin{tikzcd} \texttt{Untyped} \arrow[r] & \texttt{Typed} \end{tikzcd}$$

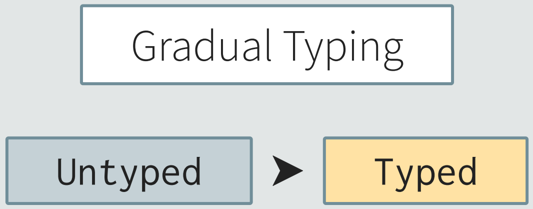
Gradual Typing



```
def join(d0,d1,sort,how):
  ....
```

```
def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

Types where useful ... and nowhere else!

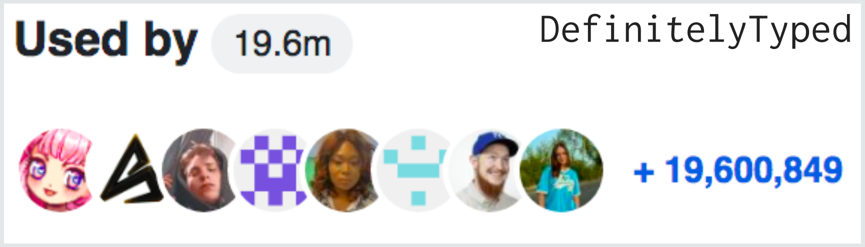

Gradual Typing
Untyped
Typed

Gradual Typing

Untyped ➤ Typed

JS ➤ TS

TypeScript is **JavaScript with syntax for types.**

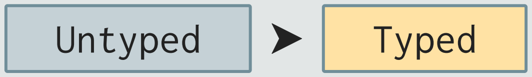


Used by 19.6m

DefinitelyTyped

+ 19,600,849

Gradual Typing Performance?



Gradual Typing Performance?

`Untyped` ➤ `Typed`

Run-time cost of sound types

Gradual Typing Performance?

Untyped ➤ Typed

Run-time cost of sound types

```
  def join(d0:DataFrame,
           d1:DataFrame,
           sort:bool,
           how:Left|Right)
      -> DataFrame:
    ....
```

Gradual Typing Performance?

Untyped ➤ Typed

Run-time cost of sound types

?? `join(x,y,z)` How to validate?

```
  def join(d0:DataFrame,
           d1:DataFrame,
           sort:bool,
           how:Left|Right)
      -> DataFrame:
    ....
```

Gradual Typing Performance?

Untyped ➤ Typed

Run-time cost of sound types

?? `join(x,y,z)` How to validate?

```
def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

TS (TypeScript does not validate)

Gradual Typing Performance?

Untyped ➤ Typed

Run-time cost of sound types

?? `join(x,y,z)` How to validate?

```
def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

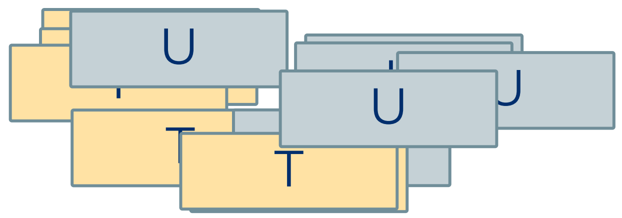
Gradual Typing Performance?

Untyped ➤ Typed

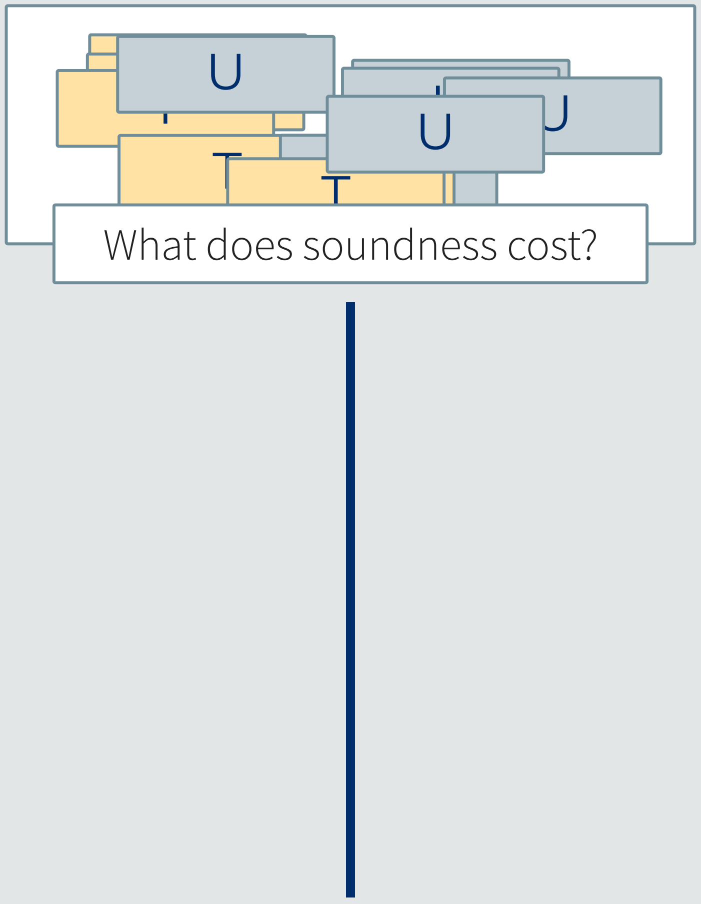
Run-time cost of sound types

?? `join(x,y,z)` How to validate?

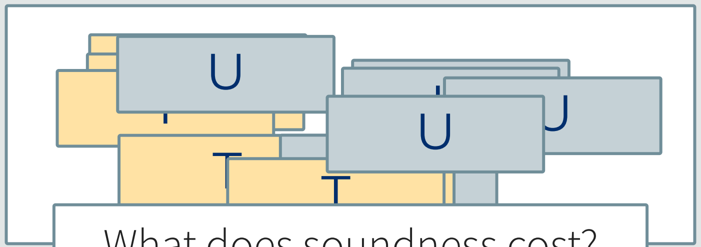
```
  def join(d0:DataFrame,
           d1:DataFrame,
           sort:bool,
           how:Left|Right)
      -> DataFrame:
    ....
```



Many interactions,
Maybe high costs



What does soundness cost?



What does soundness cost?

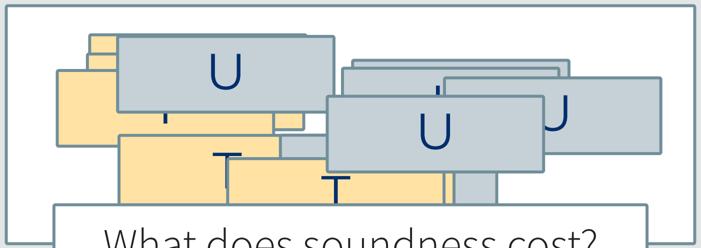
Typed Racket

+ object types, function types, ...

+ type-driven optimizer

Worst-case slowdown: **1.4x**

ecoop'15



What does soundness cost?

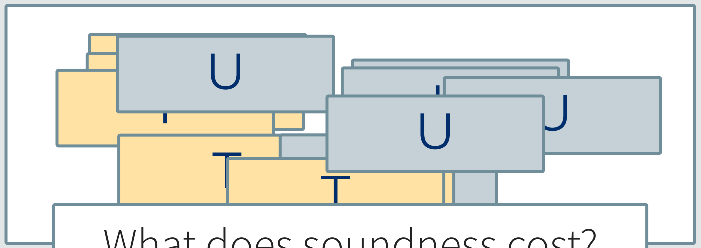
Typed Racket

+ object types, function types, ...

+ type-driven optimizer

Worst-case slowdown: **1.4x**

`ecoop'15`



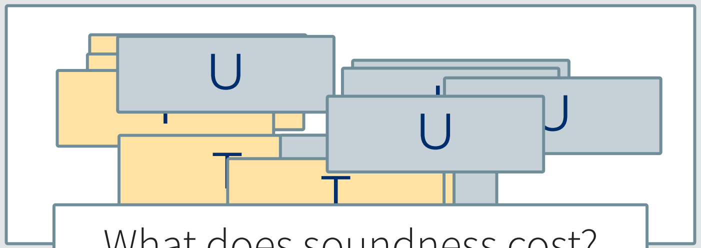
What does soundness cost?

Typed Racket
+ object types, function types, ...
+ type-driven optimizer

Worst-case slowdown: `1.4x`
`ecoop'15`

`2x` `30x` `12,000x`

(1ms to 12sec)



What does soundness cost?

Typed Racket

+ object types, function types, ...

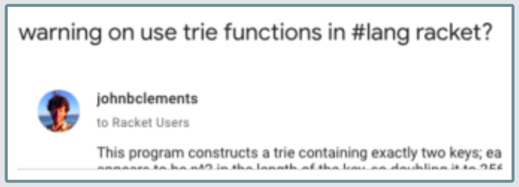
+ type-driven optimizer

Worst-case slowdown: `1.4x`

`ecoop'15`

`2x` `30x` `12,000x`

(1ms to 12sec)



warning on use trie functions in #lang racket?

johnbclements

to Racket Users

This program constructs a trie containing exactly two keys; ea

What does soundness cost?

Need a way to measure!

GTP Benchmarks

GTP Benchmarks

What to measure?

Cost of sound types

GTP Benchmarks

What to measure?	Cost of sound types
Which programs?	... Any

GTP Benchmarks

What to measure?	Cost of sound types
Which programs?	... Any
How fast is good enough?	???

GTP Benchmarks

What to measure?	Cost of sound types
Which programs?	... Any
How fast is good enough?	???
What is a benchmark?	???

GTP Benchmarks

What to measure?	Cost of sound types
Which programs?	... Any
How fast is good enough?	???
What is a benchmark?	???

Think like a practitioner

What is a gradual typing benchmark?

Untyped code?

```
def join(d0,d1,sort,how):
```

Not enough.

Typed code?

```
def join(d0:DataFrame, ...):
```

Not enough.

What is a gradual typing benchmark?

Untyped code?
```
def join(d0,d1,sort,how):
```
Not enough.

Typed code?
```
def join(d0:DataFrame, ...):
```
Not enough.

GT promise: can mix typed + untyped code

Need to measure **all configurations**

What is a gradual typing benchmark?

1. Start with a program

```
def join(d0,d1,sort,how):
  ....
```

What is a gradual typing benchmark?

1. Start with a program

```
def join(d0,d1,sort,how):
  ....
```

2. Add types

```
def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

What is a gradual typing benchmark?

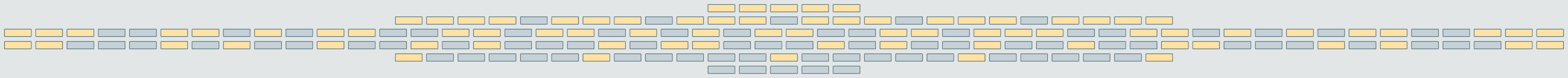
1. Start with a program

```
def join(d0,d1,sort,how):
  ....
```

2. Add types

```
def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
     -> DataFrame:
  ....
```

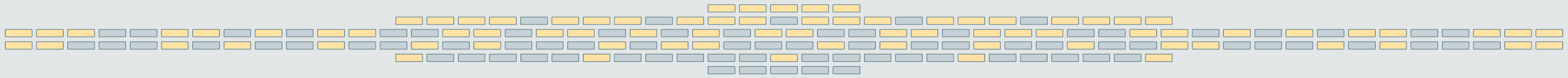
3. Explore all configurations



What is a gradual typing benchmark?

Explore by **module**

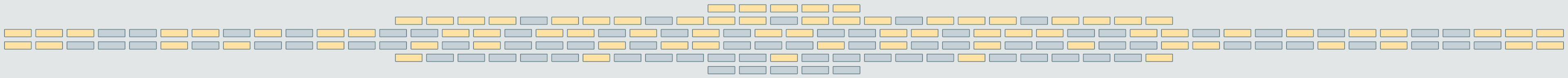
5 modules, 32 configurations



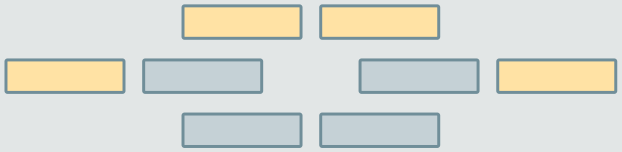
What is a gradual typing benchmark?

Explore by **module**

5 modules, 32 configurations



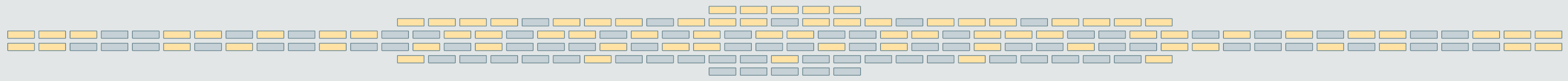
2 modules, 4 configurations



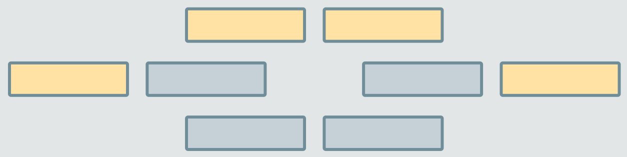
What is a gradual typing benchmark?

Explore by **module**

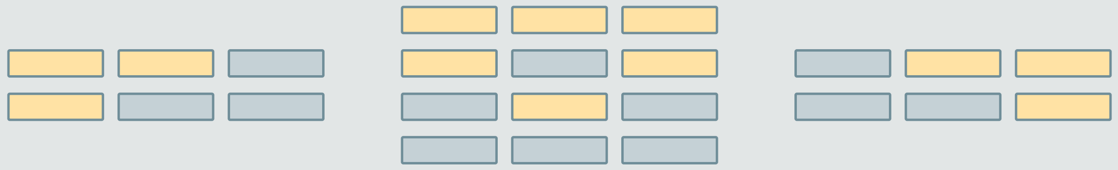
5 modules, 32 configurations



2 modules, 4 configurations



3 modules, 8 configurations



Where to find benchmarks?

Where to find benchmarks?

Wherever people share code

Where to find benchmarks?

Wherever people share code

Current status: 21 benchmarks, +40k configurations

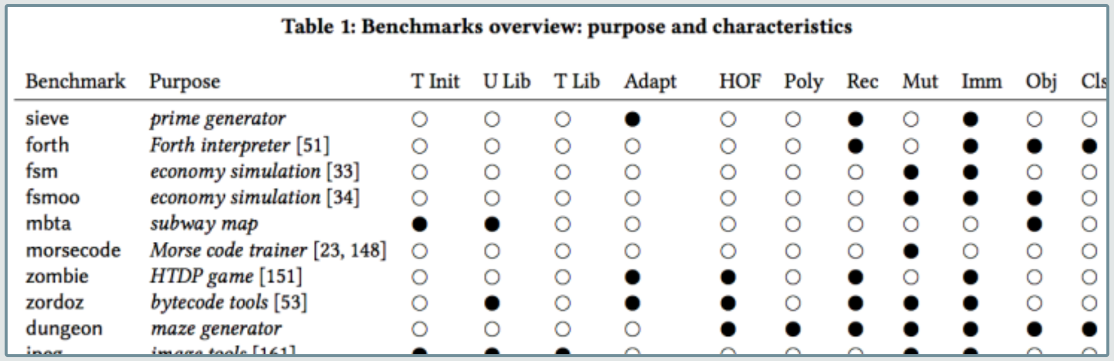
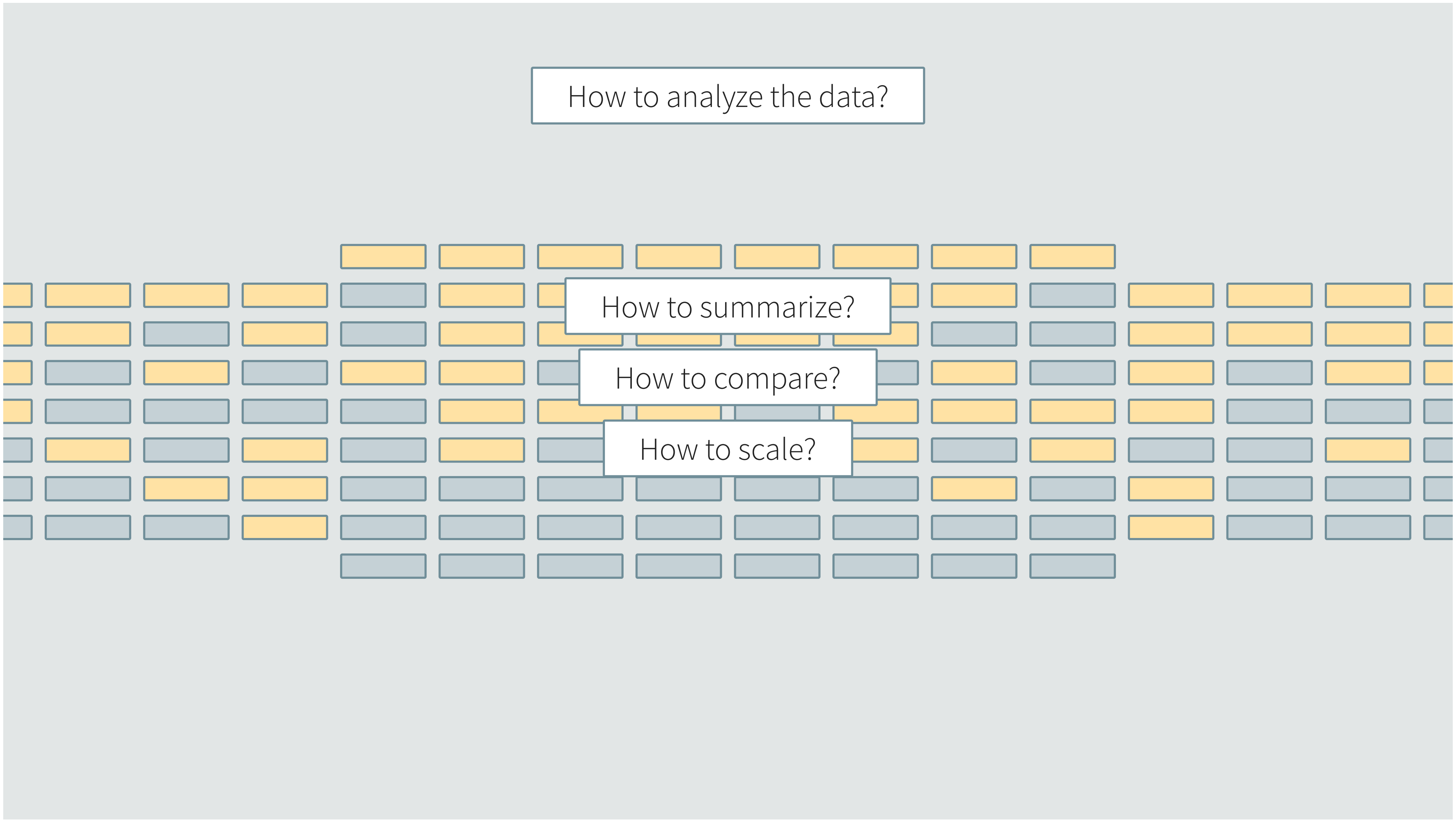
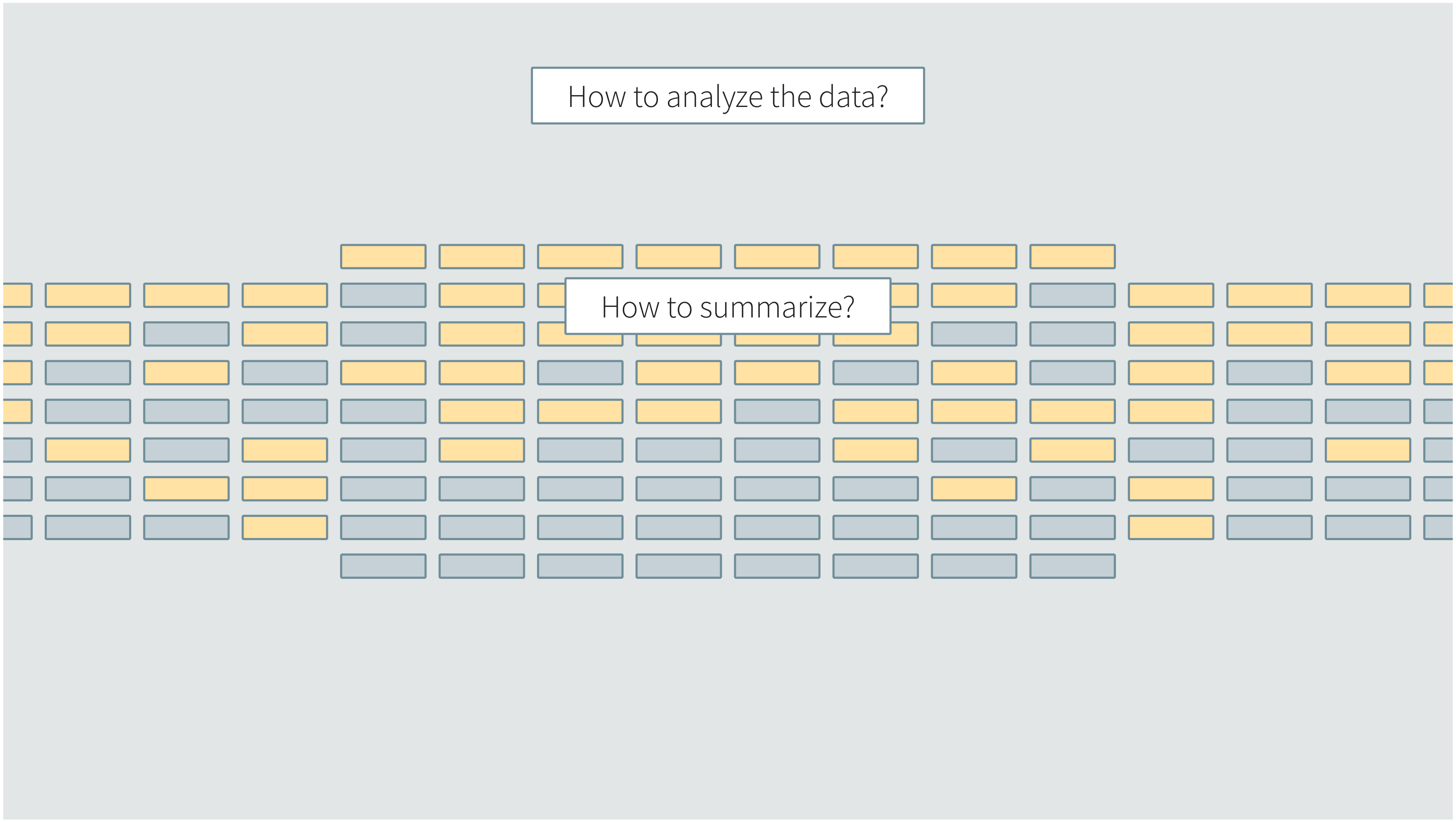


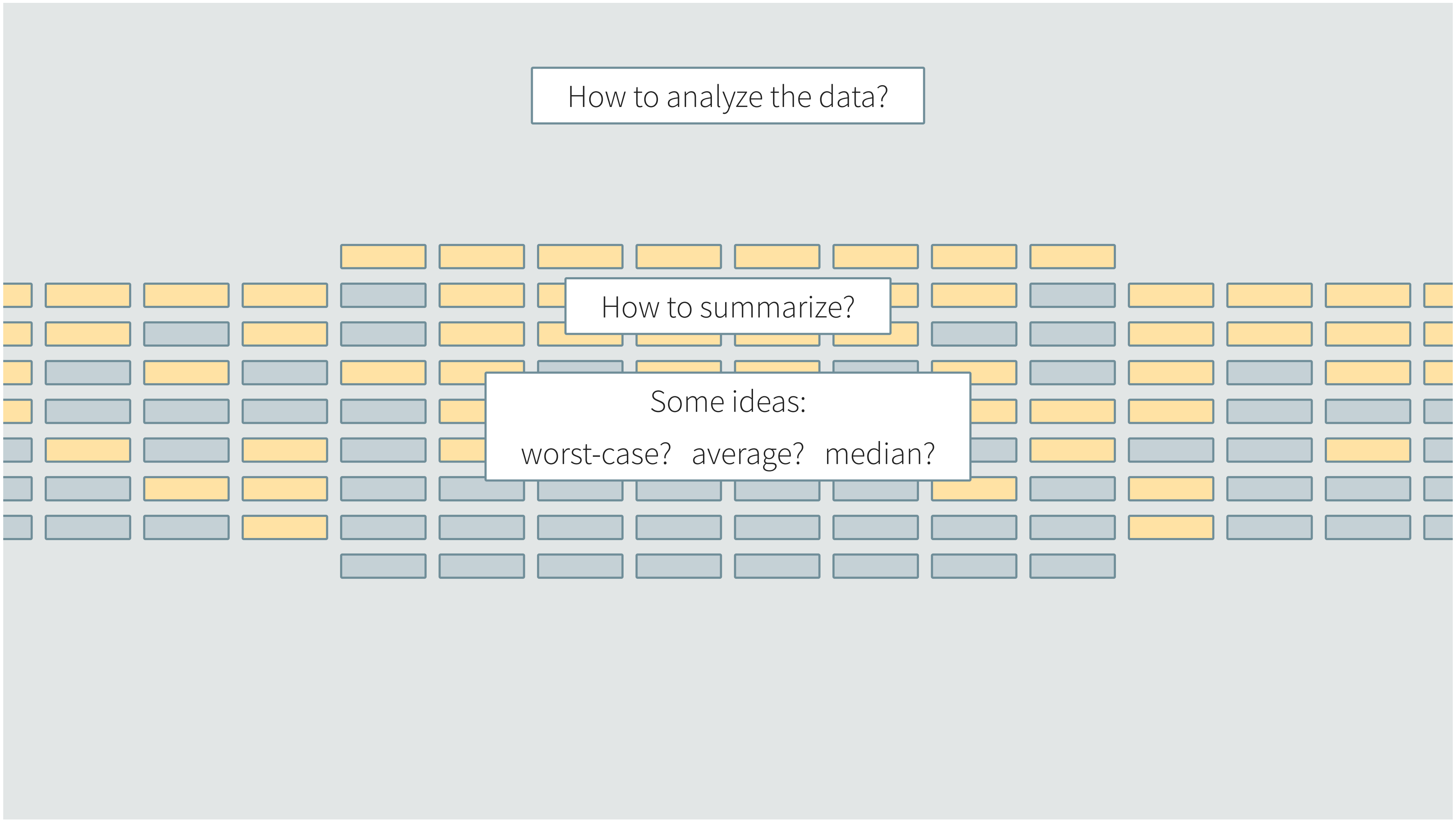
Table 1: Benchmarks overview: purpose and characteristics

Benchmark	Purpose	T Init	U Lib	T Lib	Adapt	HOF	Poly	Rec	Mut	Imm	Obj	Cls
sieve	*prime generator*	○	○	○	●	○	○	●	○	●	○	○
forth	*Forth interpreter* [51]	○	○	○	○	○	○	●	○	●	●	●
fsm	*economy simulation* [33]	○	○	○	○	○	○	○	●	●	○	○
fsmoo	*economy simulation* [34]	○	○	○	○	○	○	○	●	●	●	○
mbta	*subway map*	●	●	○	○	○	○	○	○	○	●	○
morsecode	*Morse code trainer* [23, 148]	○	○	○	○	○	○	○	●	○	○	○
zombie	*HTDP game* [151]	○	○	○	●	●	○	●	○	●	○	○
zordoz	*bytecode tools* [53]	○	●	○	●	●	○	●	●	●	○	○
dungeon	*maze generator*	○	○	○	○	●	●	●	●	●	●	●
jpeg	*image tools* [161]	●	●	●	○	○	○	○	●	●	○	○

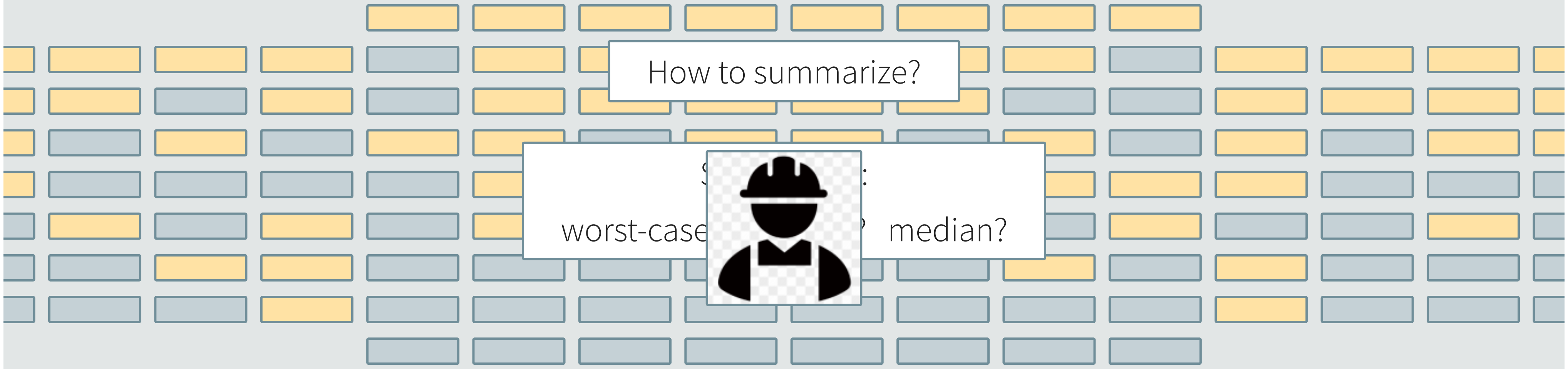
How to analyze the data?


How to analyze the data?
How to summarize?
How to compare?
How to scale?


How to analyze the data?
How to summarize?

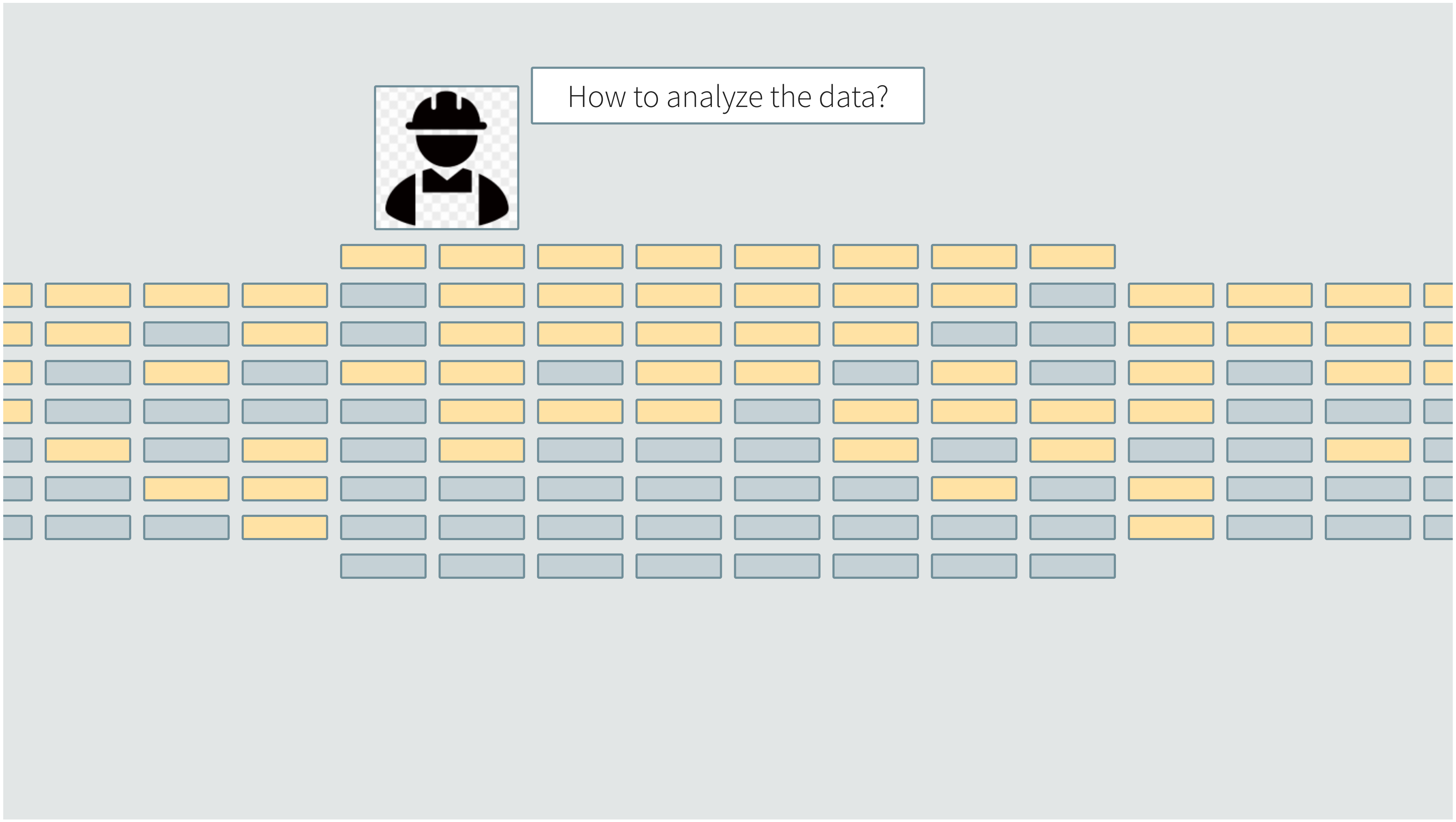

How to analyze the data?
How to summarize?
Some ideas:
worst-case? average? median?

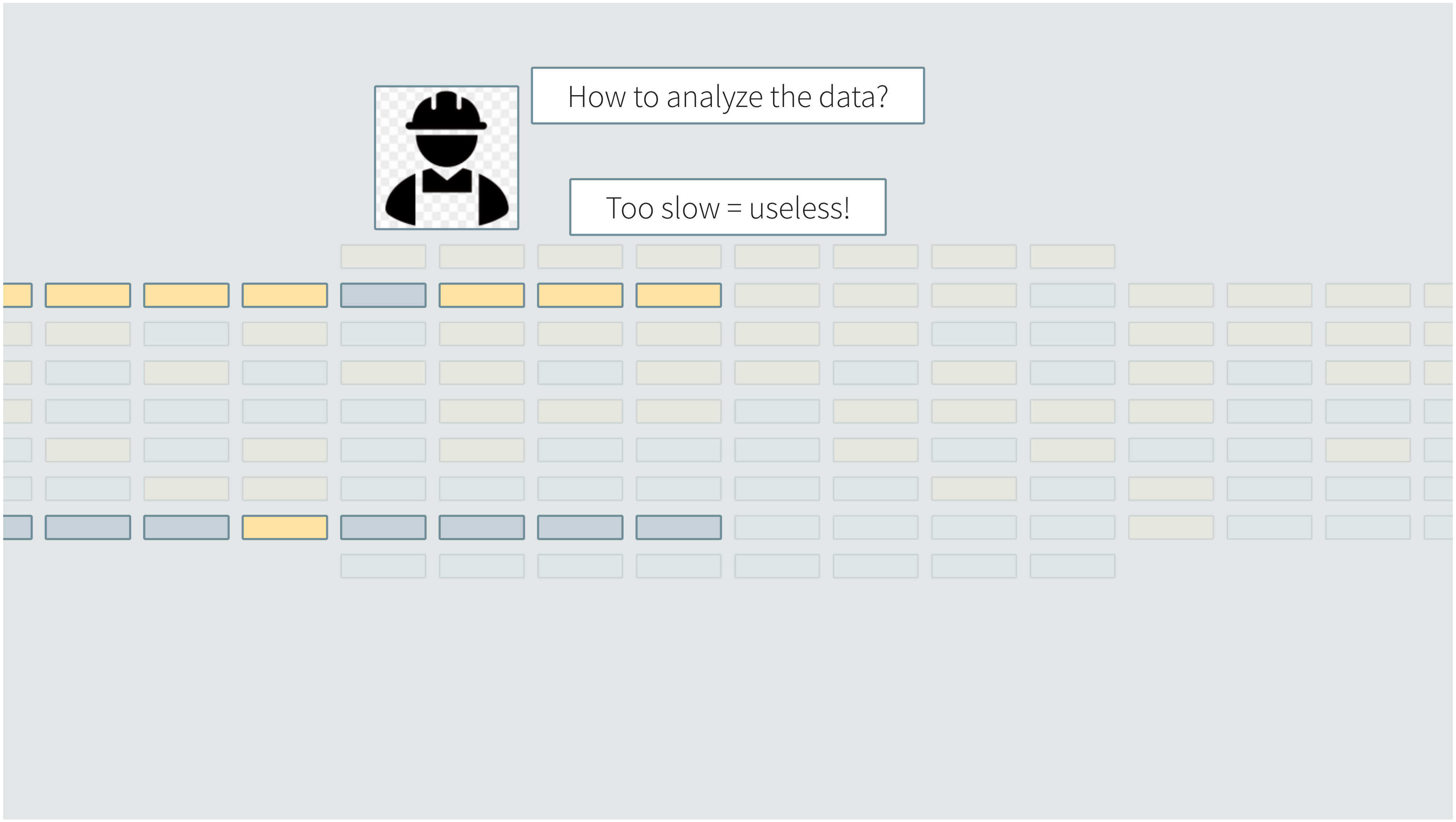
How to analyze the data?



How to summarize?

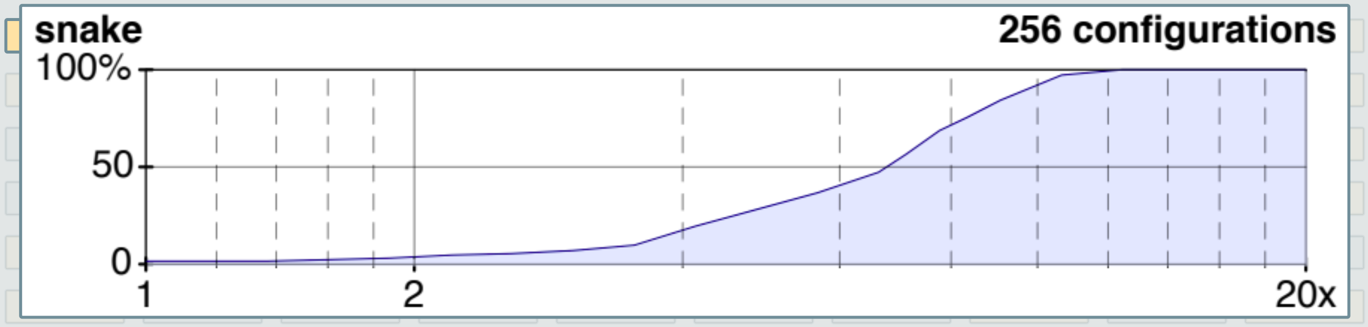
worst-case ? median?


How to analyze the data?


How to analyze the data?
Too slow = useless!

How to analyze the data?

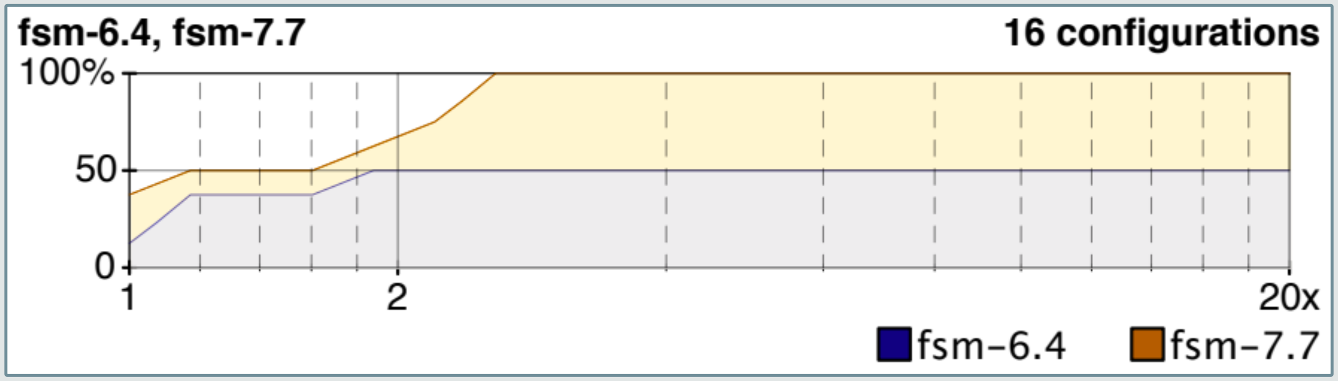
Too slow = useless!



x-axis = limit for "too slow" vs. untyped code (log scale)
y-axis = % usable configs.

How to compare

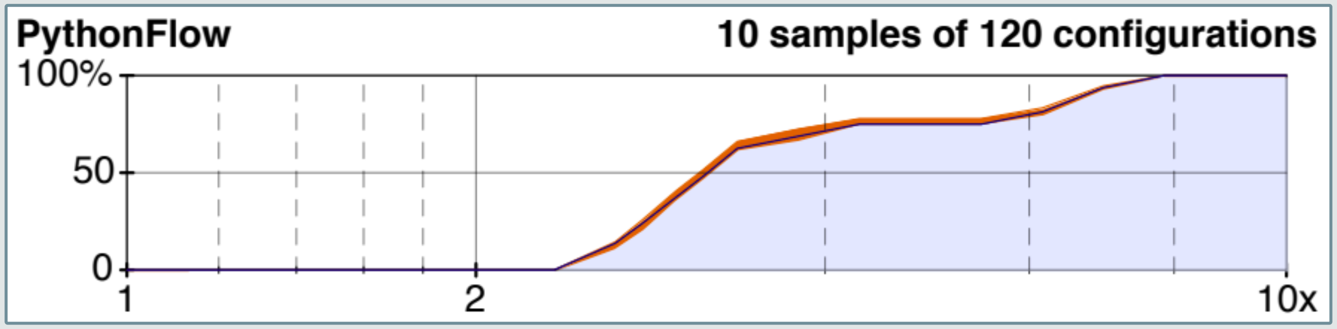
How to compare

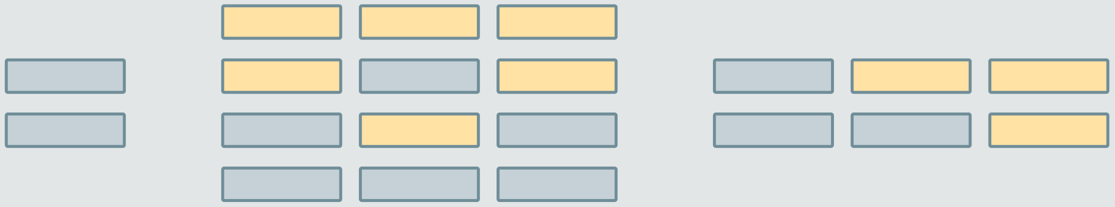


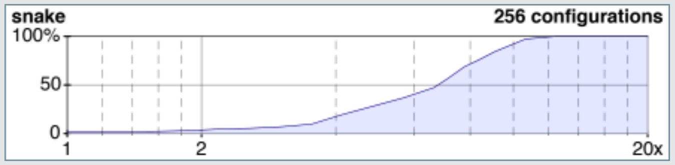
How to scale

How to scale

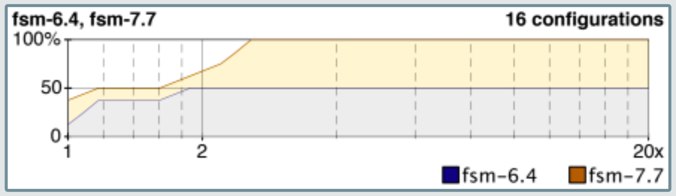
Linear-size random samples



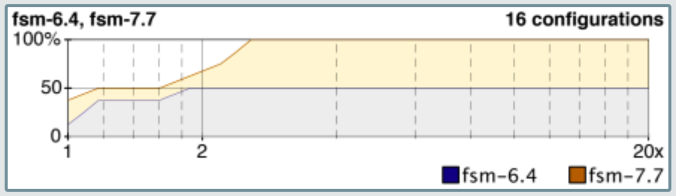


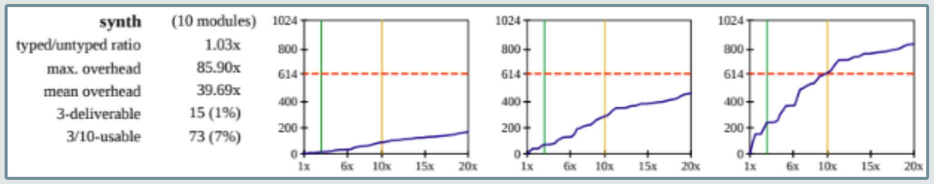

snake
256 configurations
100%
50
0
1
2
20x

Software for Measurement

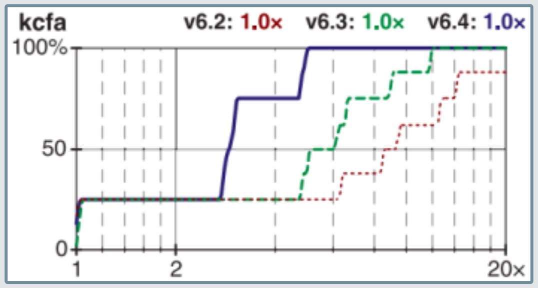


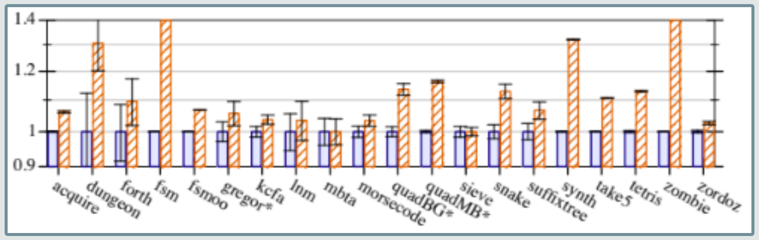
Software for Measurement





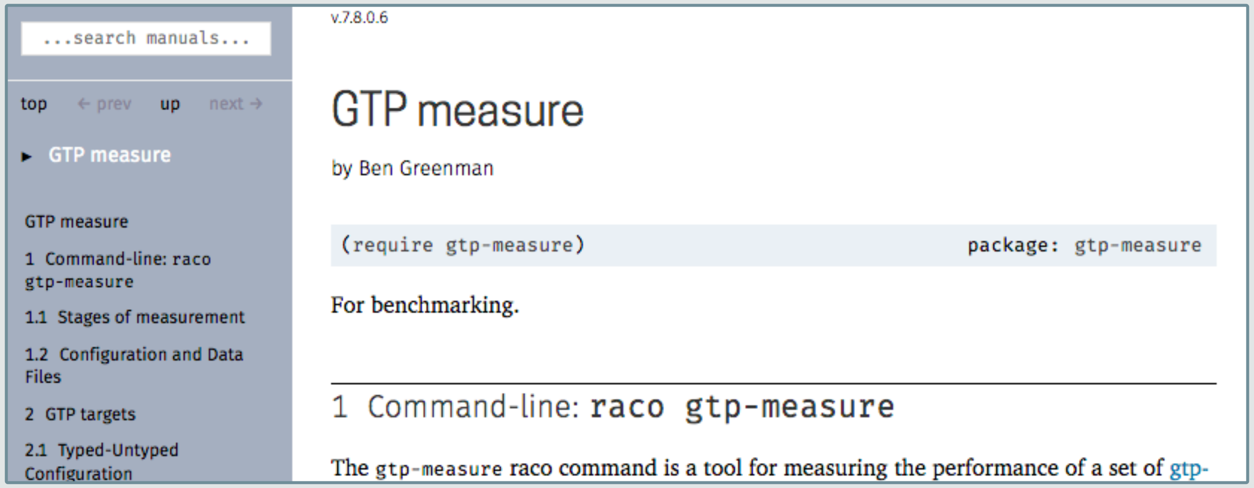
synth	(10 modules)
typed/untyped ratio	1.03x
max. overhead	85.90x
mean overhead	39.69x
3-deliverable	15 (1%)
3/10-usable	73 (7%)





Software for Measurement

Software for Measurement



...search manuals...

top ← prev up next →

▸ GTP measure

GTP measure

1 Command-line: raco gtp-measure

1.1 Stages of measurement

1.2 Configuration and Data Files

2 GTP targets

2.1 Typed-Untyped Configuration

v.7.8.0.6

GTP measure

by Ben Greenman

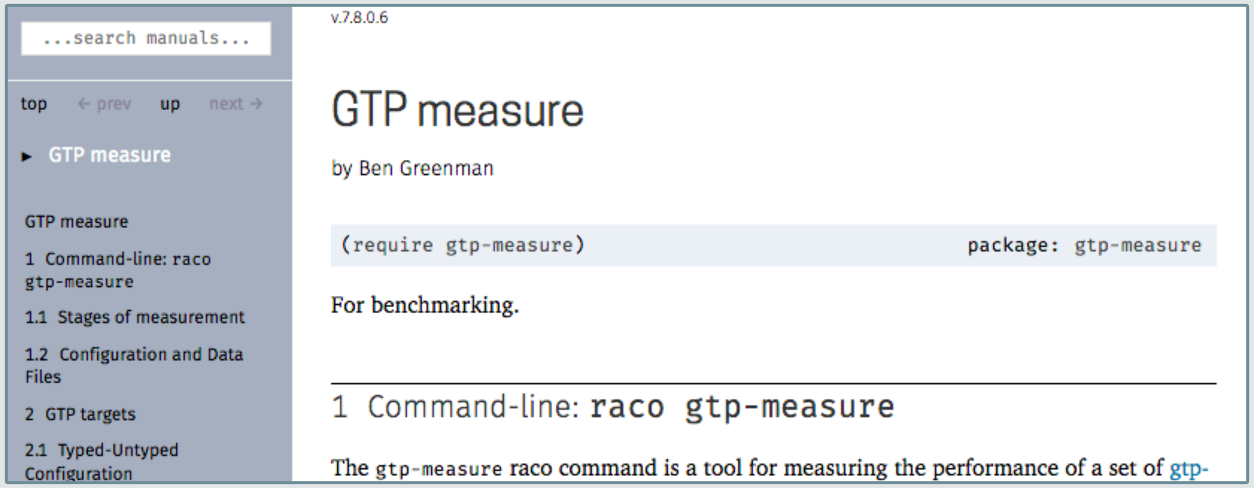
```
(require gtp-measure)                                    package: gtp-measure
```

For benchmarking.

1 Command-line: `raco gtp-measure`

The `gtp-measure` raco command is a tool for measuring the performance of a set of gtp-

Software for Measurement



...search manuals...

top ← prev up next →

▸ GTP measure

v.7.8.0.6

GTP measure

by Ben Greenman

```
(require gtp-measure)                package: gtp-measure
```

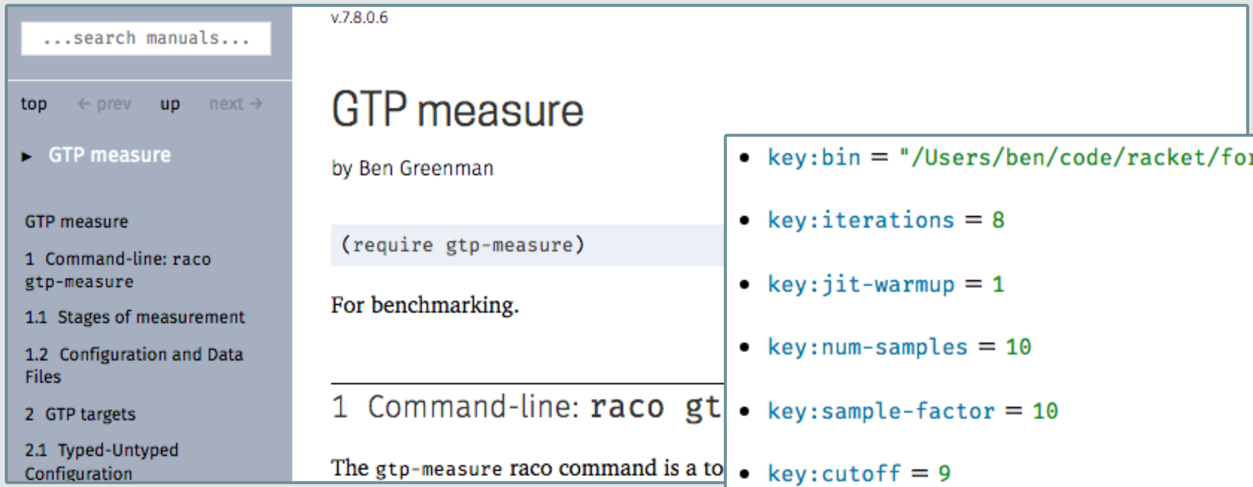
For benchmarking.

1 Command-line: `raco gtp-measure`

The `gtp-measure` raco command is a tool for measuring the performance of a set of gtp-

Interruptible!   Space-Efficient.   Configurable.

Software for Measurement



...search manuals...

top ← prev up next →

▸ GTP measure

GTP measure

1 Command-line: raco gtp-measure

1.1 Stages of measurement

1.2 Configuration and Data Files

2 GTP targets

2.1 Typed-Untyped Configuration

v.7.8.0.6

GTP measure

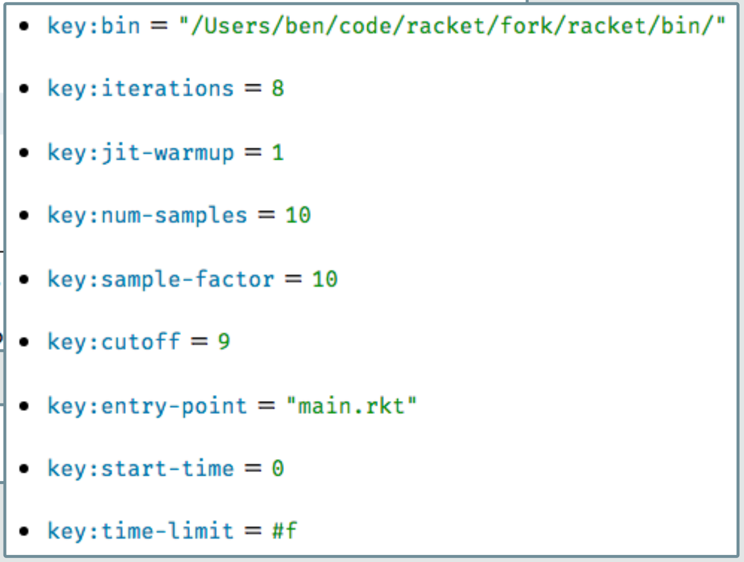
by Ben Greenman

```
(require gtp-measure)
```

For benchmarking.

1 Command-line: raco gt

The gtp-measure raco command is a to



- key:bin = "/Users/ben/code/racket/fork/racket/bin/"
- key:iterations = 8
- key:jit-warmup = 1
- key:num-samples = 10
- key:sample-factor = 10
- key:cutoff = 9
- key:entry-point = "main.rkt"
- key:start-time = 0
- key:time-limit = #f

Interruptible! Space-Efficie

Software for Measurement

Software for Measurement

Tiny DSL for experiments

```
#lang gtp-measure/manifest

#:config #hash(
  (bin . "/home/gtp/racket-8.8/bin/")
  (cutoff . 6)
  (num-samples . 10))

/home/gtp/benchmarks/morsecode
/home/gtp/benchmarks/take5
```

Software for Measurement

Software for Measurement

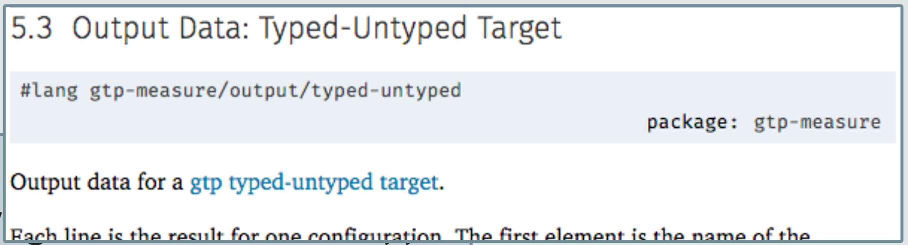
DSL for data

```
#lang gtp-measure/output/typed-untyped
("00000" ("cpu time: 566 real time: 567 gc time: 62" ....))
("00001" ("cpu time: 820 real time: 822 gc time: 46" ....))
("00010" ("cpu time: 561 real time: 562 gc time: 46" ....))
("00011" ("cpu time: 805 real time: 807 gc time: 47" ....))
....
```

Software for Measurement

DSL for data

```
#lang gtp-measure/output/typed-untyped
("00000" ("cpu time: 566 real time: 567
("00001" ("cpu time: 820 real time: 822 gc time: 46" ....))
("00010" ("cpu time: 561 real time: 562 gc time: 46" ....))
("00011" ("cpu time: 805 real time: 807 gc time: 47" ....))
....
```



5.3 Output Data: Typed-Untyped Target

```
#lang gtp-measure/output/typed-untyped
                                        package: gtp-measure
```

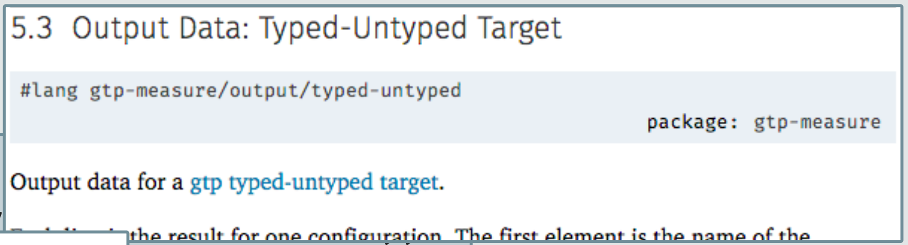
Output data for a gtp typed-untyped target.

Each line is the result for one configuration. The first element is the name of the

Software for Measurement

DSL for data

```
#lang gtp-measure/output/typed-untyped
("00000" ("cpu time: 566 real time: 567
("00001" ("cpu                          me: 46" ....))
("00010" ("cpu                          me: 46" ....))
("00011" ("cpu                          me: 47" ....))
....
```

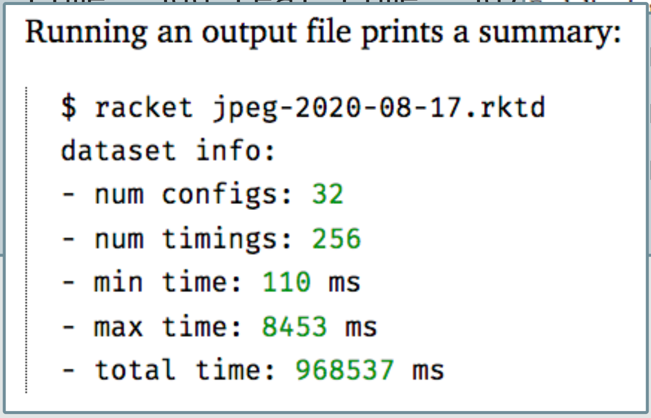


5.3 Output Data: Typed-Untyped Target

```
#lang gtp-measure/output/typed-untyped
                                              package: gtp-measure
```

Output data for a gtp typed-untyped target.

the result for one configuration. The first element is the name of the

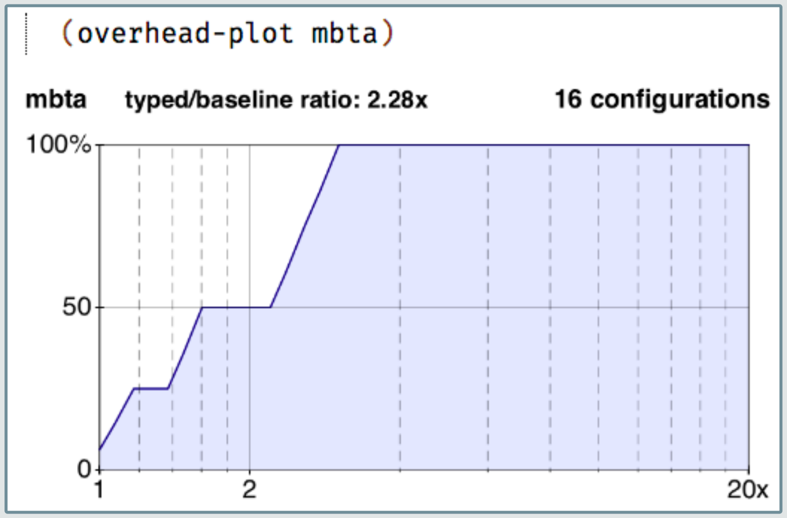


Running an output file prints a summary:

```
$ racket jpeg-2020-08-17.rktd
dataset info:
- num configs: 32
- num timings: 256
- min time: 110 ms
- max time: 8453 ms
- total time: 968537 ms
```

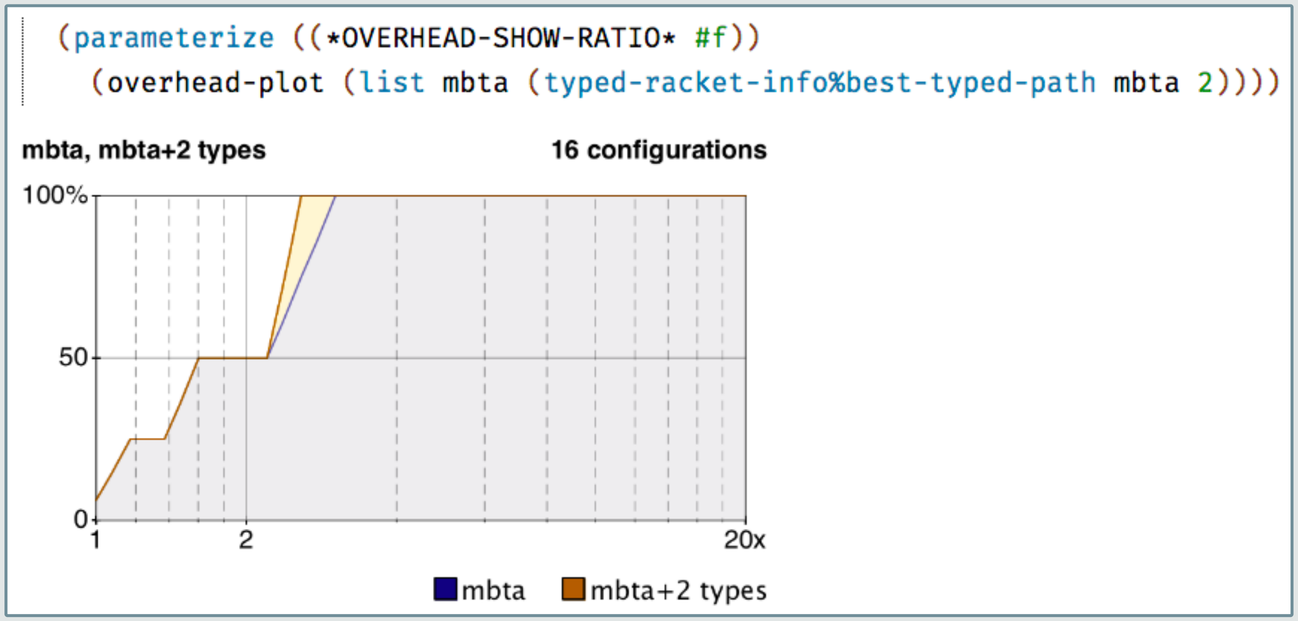
Software for Visualization

Software for Visualization



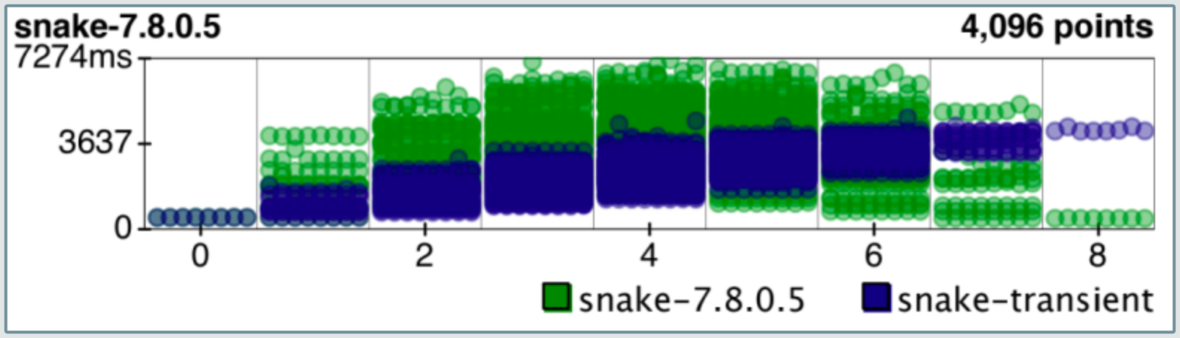
```
(overhead-plot mbta)
```

Software for Visualization

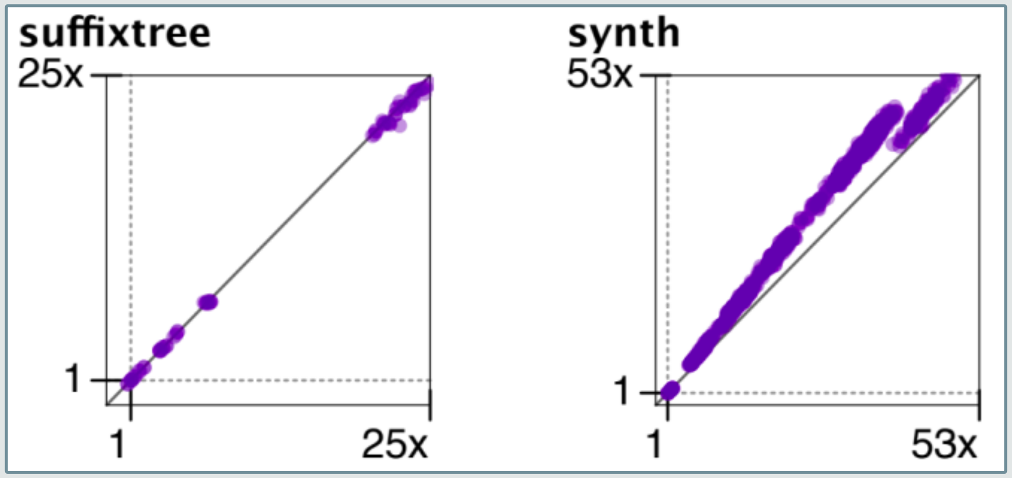


```
(parameterize ((*OVERHEAD-SHOW-RATIO* #f))
  (overhead-plot (list mbta (typed-racket-info%best-typed-path mbta 2))))
```

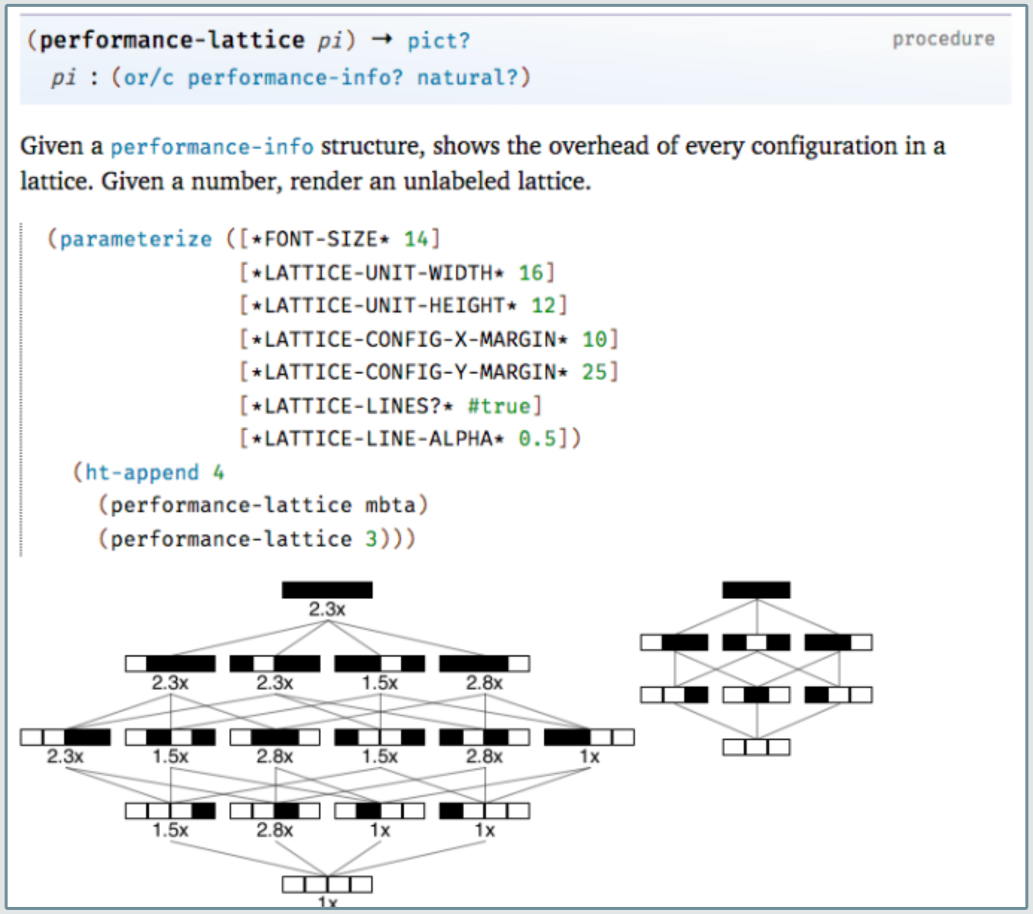
Software for Visualization



Software for Visualization



Software for Visualization



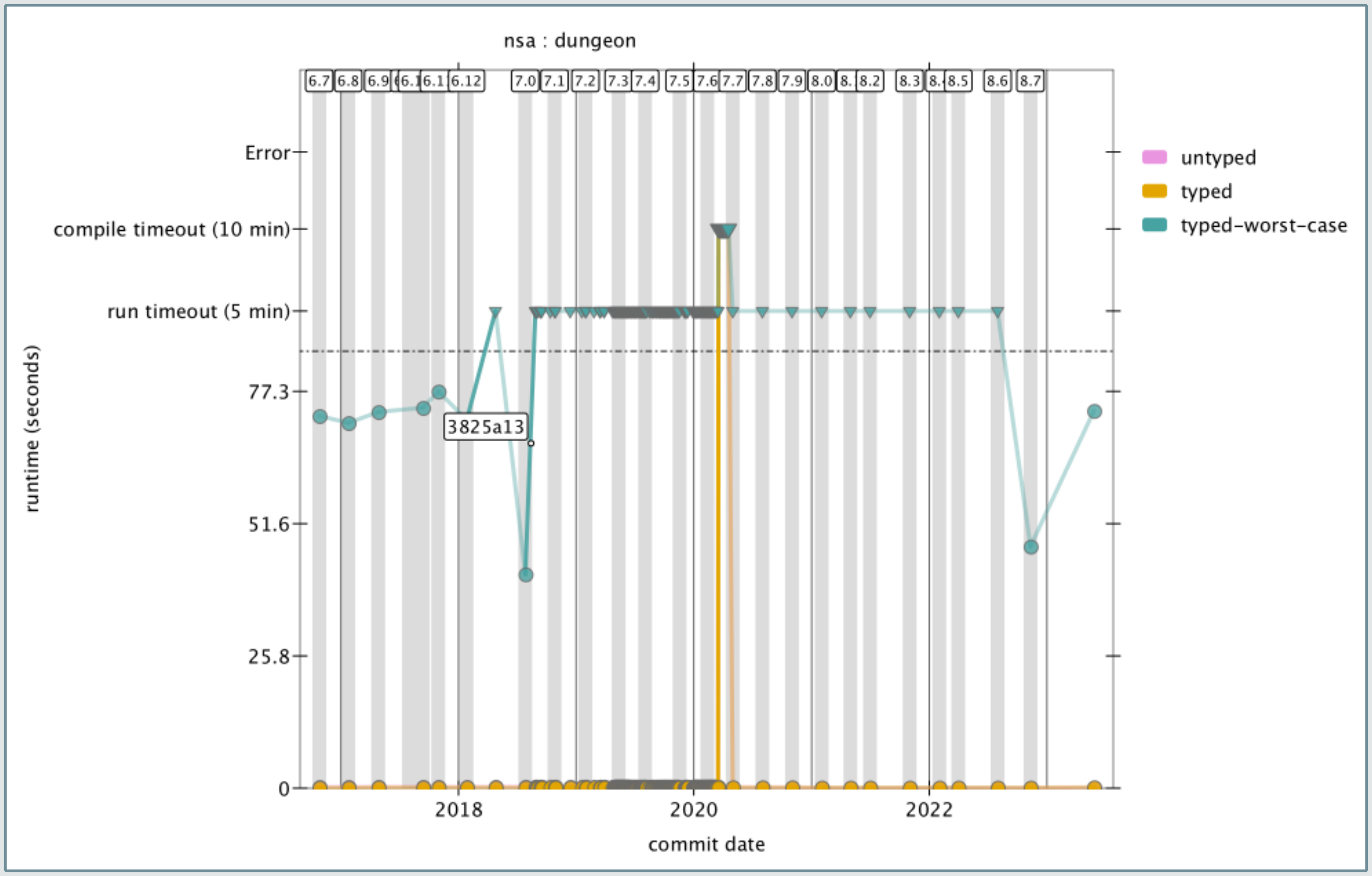
```
(performance-lattice pi) → pict?                                  procedure
  pi : (or/c performance-info? natural?)
```

Given a `performance-info` structure, shows the overhead of every configuration in a lattice. Given a number, render an unlabeled lattice.

```
(parameterize ([*FONT-SIZE* 14]
               [*LATTICE-UNIT-WIDTH* 16]
               [*LATTICE-UNIT-HEIGHT* 12]
               [*LATTICE-CONFIG-X-MARGIN* 10]
               [*LATTICE-CONFIG-Y-MARGIN* 25]
               [*LATTICE-LINES?* #true]
               [*LATTICE-LINE-ALPHA* 0.5])
  (ht-append 4
    (performance-lattice mbta)
    (performance-lattice 3)))
```

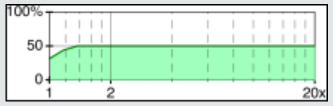
Continuous Testing

Continuous Testing



benchmarks

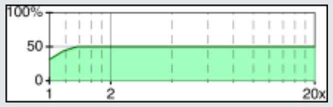
measurement



visualization

benchmarks

measurement

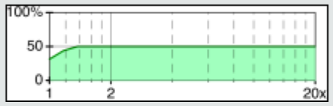


visualization

All 3 important ... but not to everyone

benchmarks

measurement



visualization

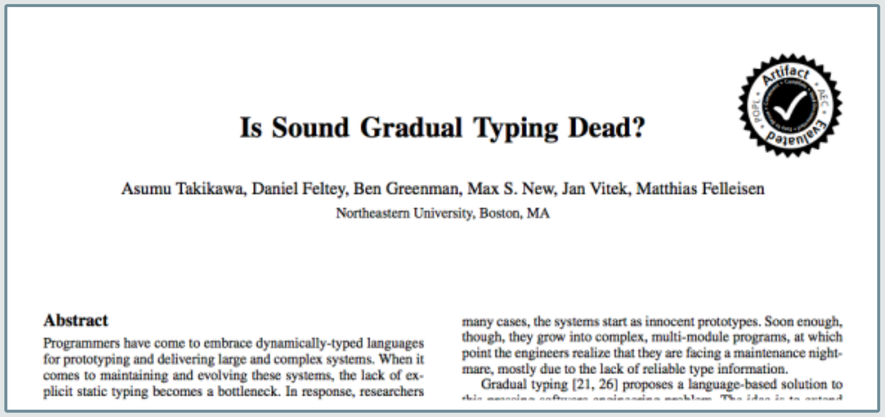
All 3 important ... but not to everyone

Lesson 2: loose coupling helps adoption

Still … low adoption

Still … low adoption

2014: few experiments,
~2 gradual configurations



Is Sound Gradual Typing Dead?

Asumu Takikawa, Daniel Feltey, Ben Greenman, Max S. New, Jan Vitek, Matthias Felleisen
Northeastern University, Boston, MA

Abstract

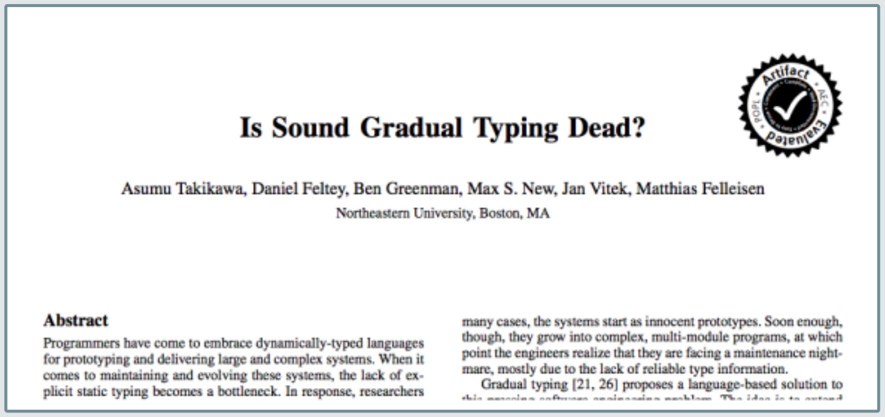
Programmers have come to embrace dynamically-typed languages for prototyping and delivering large and complex systems. When it comes to maintaining and evolving these systems, the lack of explicit static typing becomes a bottleneck. In response, researchers

many cases, the systems start as innocent prototypes. Soon enough, though, they grow into complex, multi-module programs, at which point the engineers realize that they are facing a maintenance nightmare, mostly due to the lack of reliable type information.

Gradual typing [21, 26] proposes a language-based solution to

Still ... low adoption

2014: few experiments,
~2 gradual configurations



Is Sound Gradual Typing Dead?

Asumu Takikawa, Daniel Feltey, Ben Greenman, Max S. New, Jan Vitek, Matthias Felleisen
Northeastern University, Boston, MA

Abstract

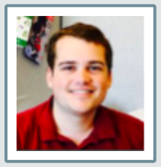
Programmers have come to embrace dynamically-typed languages for prototyping and delivering large and complex systems. When it comes to maintaining and evolving these systems, the lack of explicit static typing becomes a bottleneck. In response, researchers

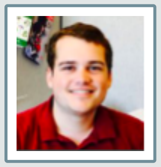
many cases, the systems start as innocent prototypes. Soon enough, though, they grow into complex, multi-module programs, at which point the engineers realize that they are facing a maintenance nightmare, mostly due to the lack of reliable type information.

Gradual typing [21, 26] proposes a language-based solution to

Lately: few experiments,
but thorough

Ok?





Thank You

Lessons

How to encourage **domain-specific** benchmarks?

Lessons

How to encourage **domain-specific** benchmarks?

Think like a practitioner

Lessons

How to encourage **domain-specific** benchmarks?

Think like a practitioner

Separate benchmarks from analysis tools

Lessons

How to encourage **domain-specific** benchmarks?

Think like a practitioner

Separate benchmarks from analysis tools

Borrow nodes

https://github.com/utahplt/gtp-benchmarks

https://github.com/utahplt/gtp-measure

https://github.com/utahplt/gtp-plot