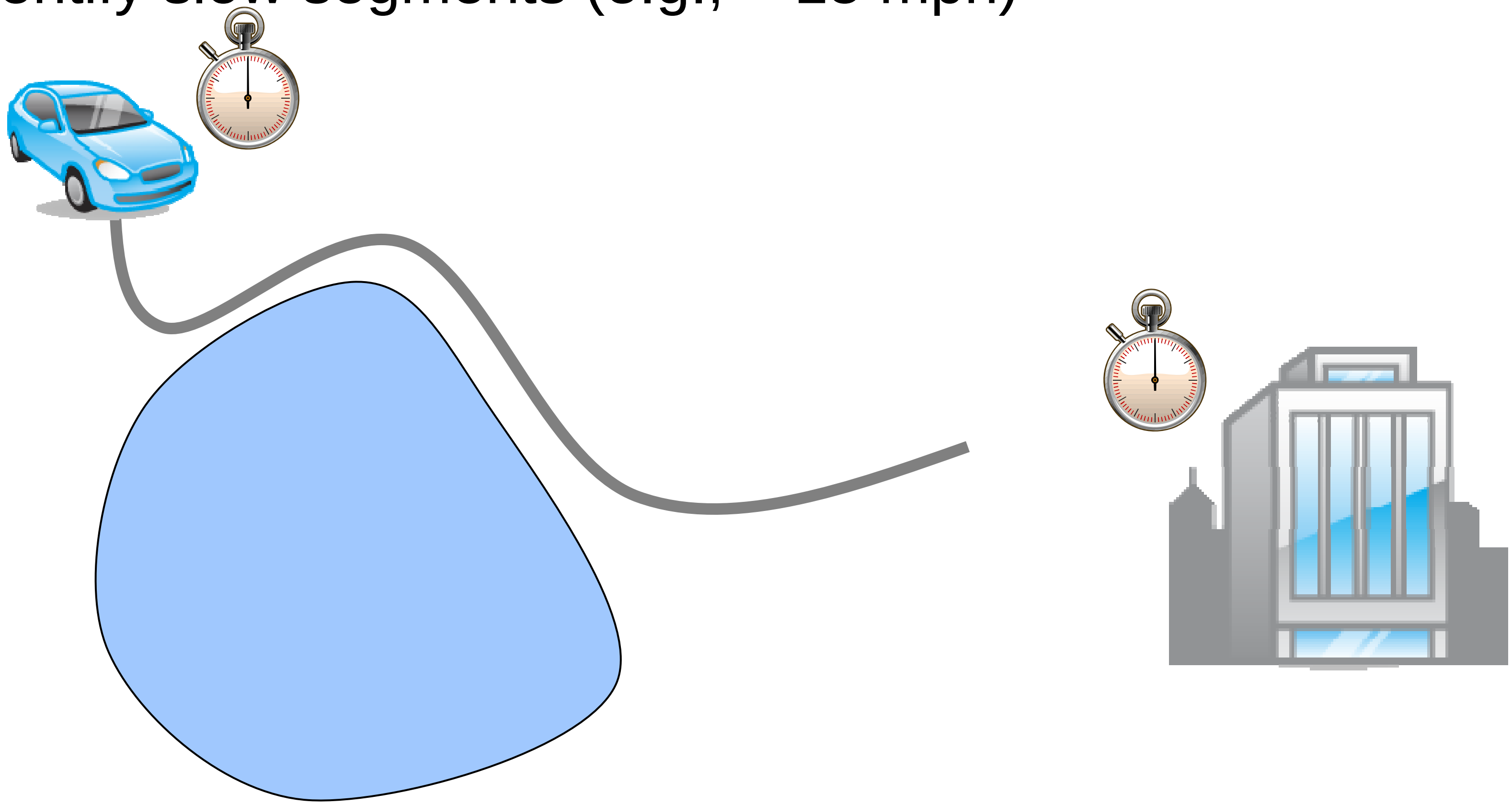
We have it easy, but do we have it right?

Amer Diwan

University of Colorado at Boulder

Research model in experimental sciences:
(i) Find bottlenecks

Identify slow segments (e.g., < 15 mph)

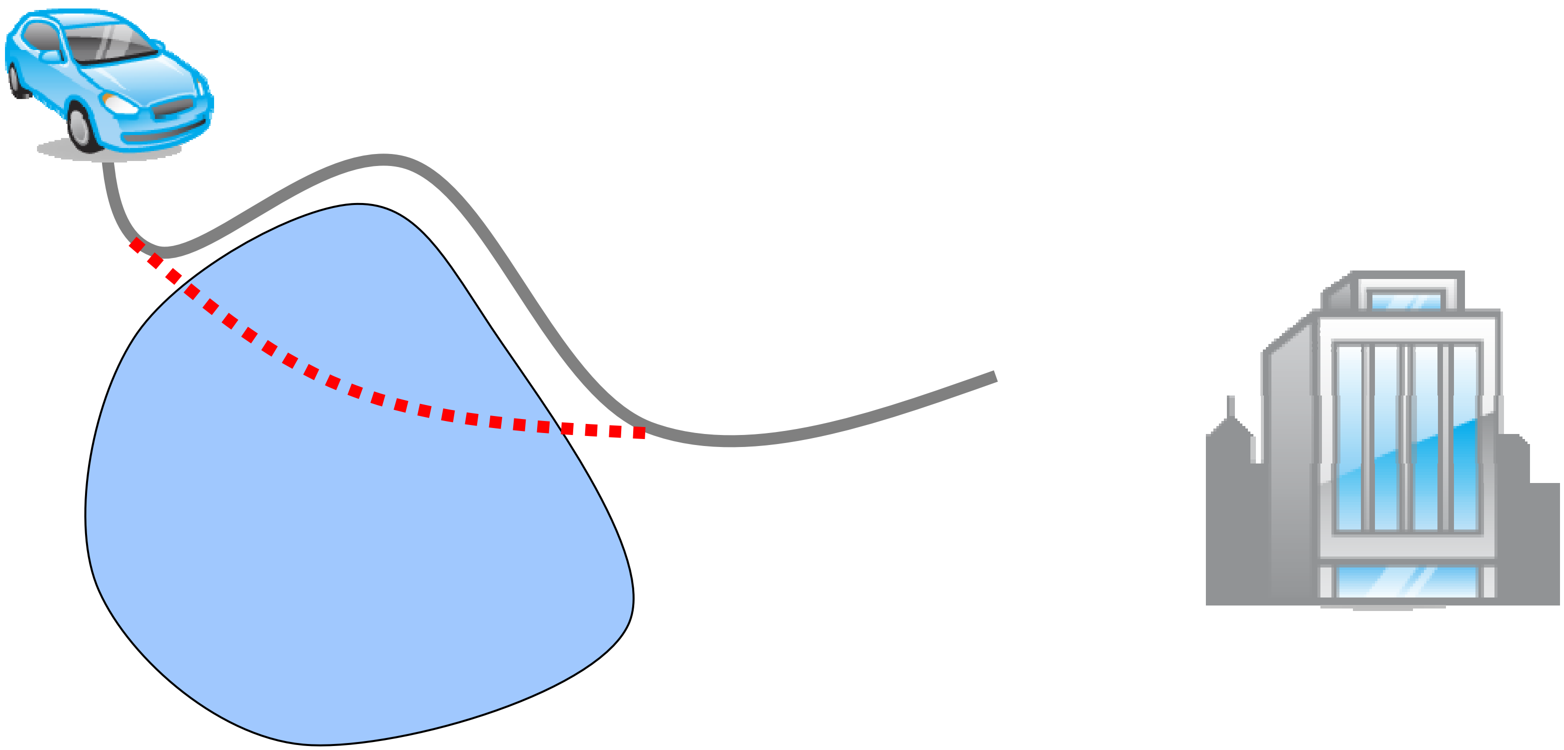


Repeat this survey for a number of drivers

Research model in experimental sciences:

(ii) Act on the data

Build a bridge

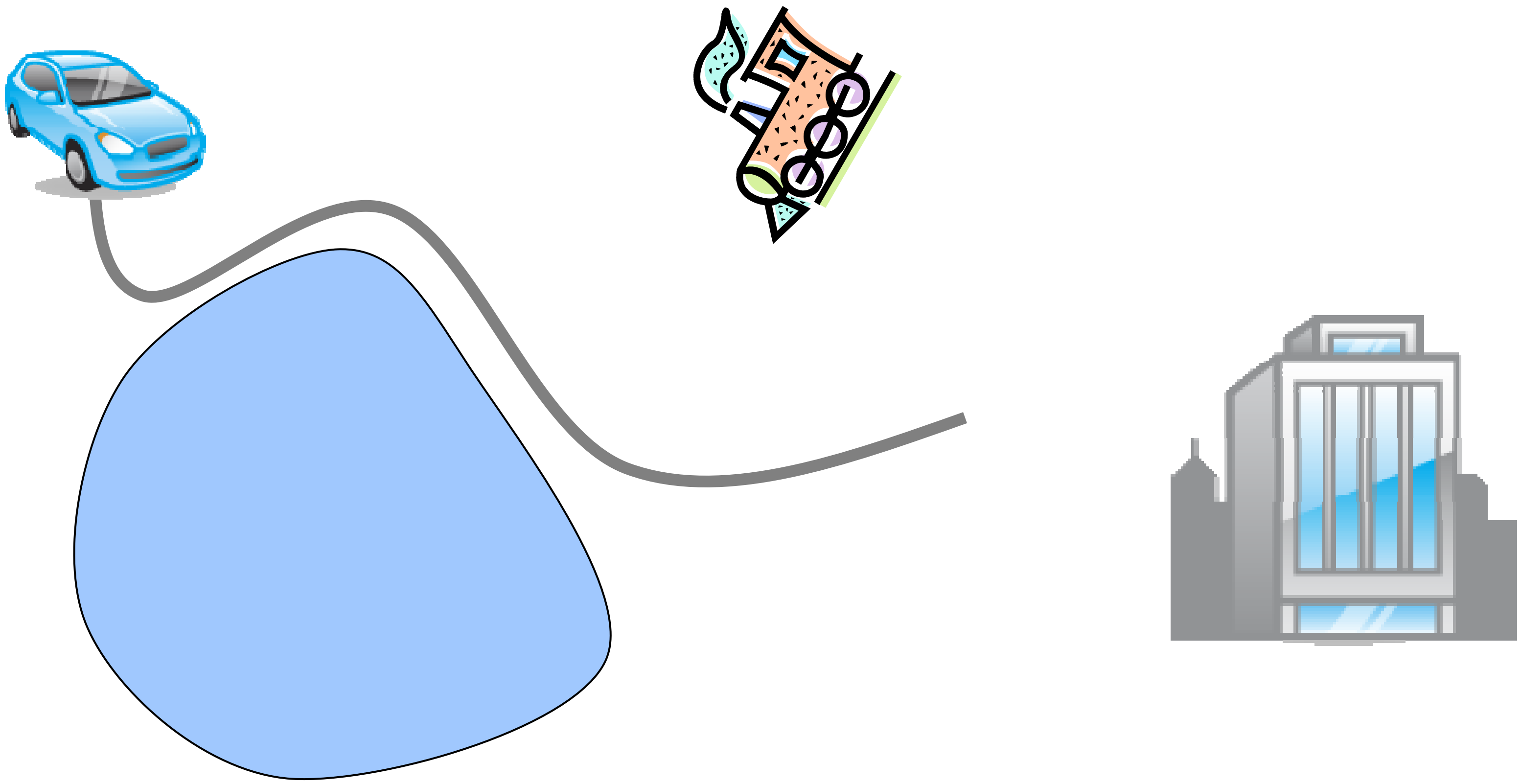


But what if the data is wrong?

A bridge that no one needs!

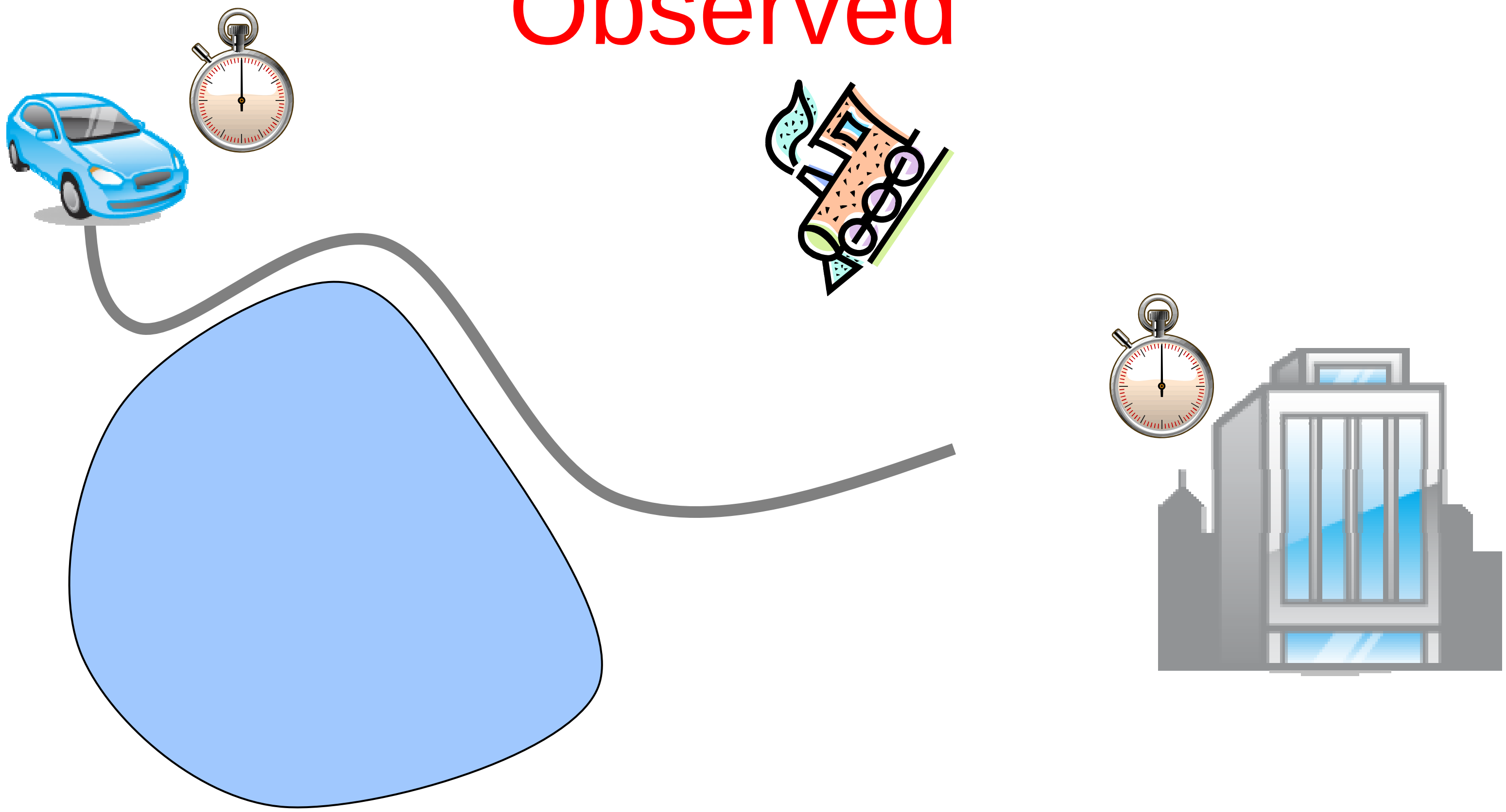
How could the data be wrong?

Reality



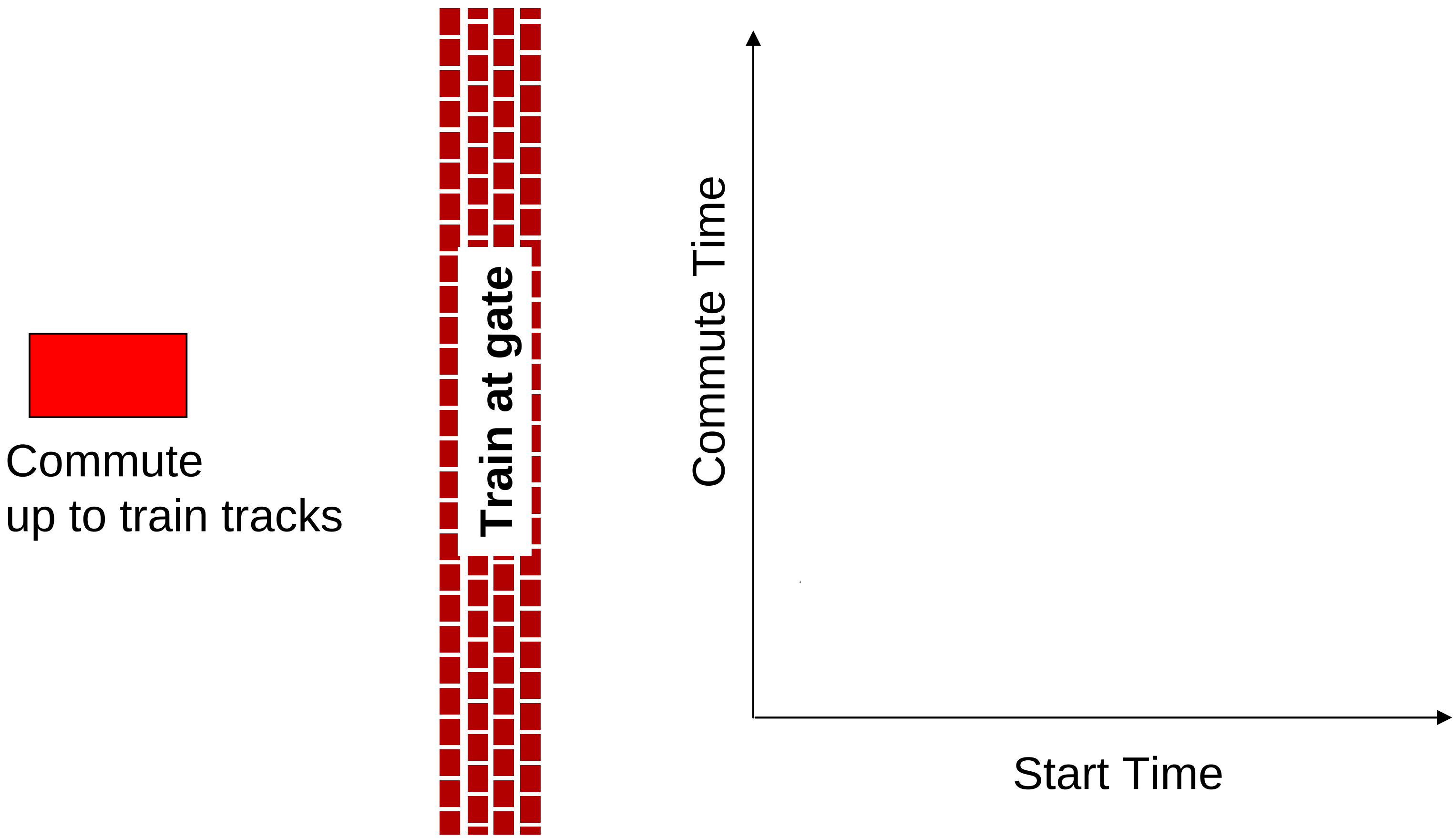
How could the data be wrong?

Observed

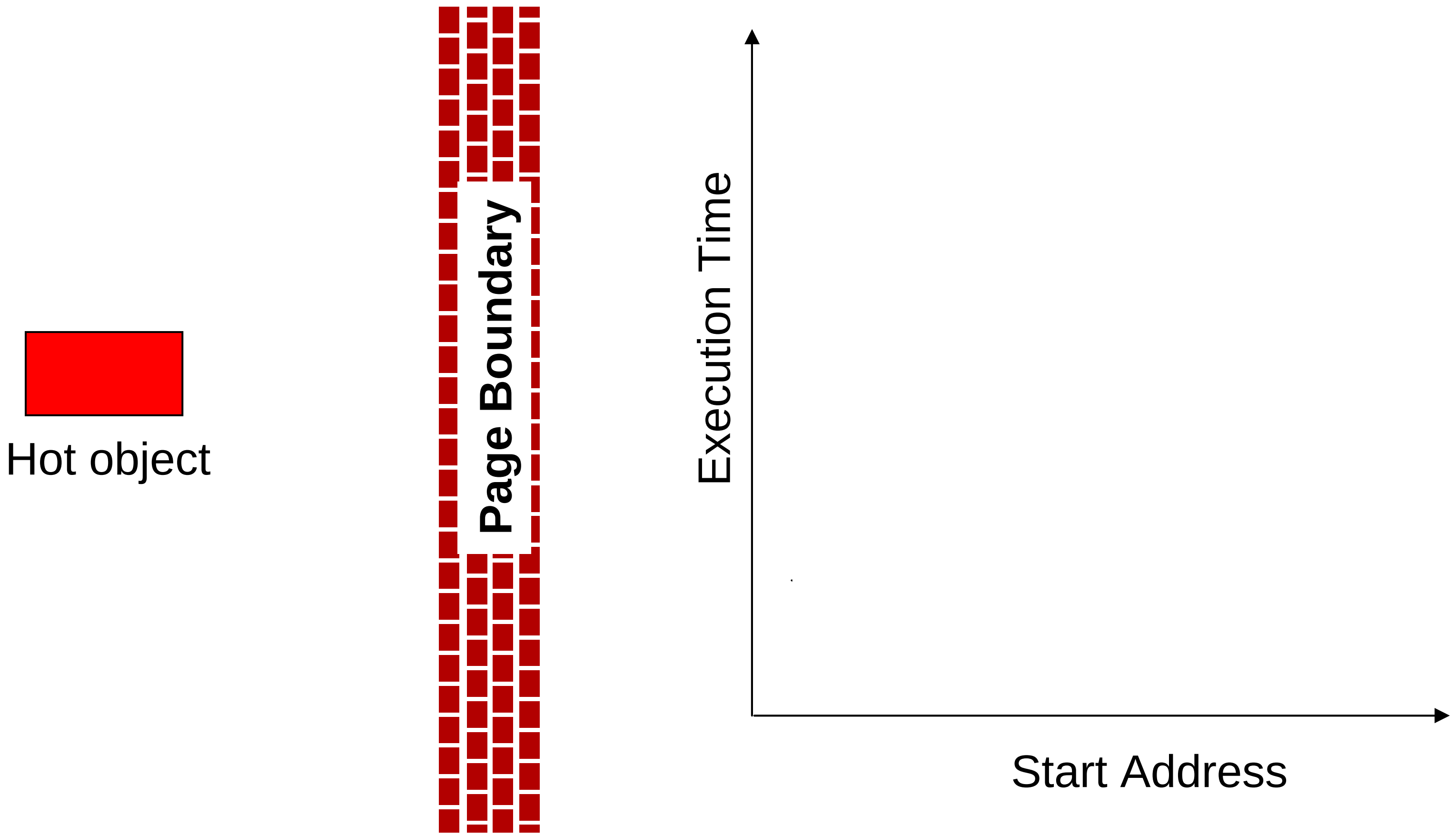


- Observer Effect
  - Measurement delay caused the commuter to wait at the crossing gates
- This study was easy to conduct but it is not right!

A small perturbation drastically changes the data



A small perturbation drastically changes the data



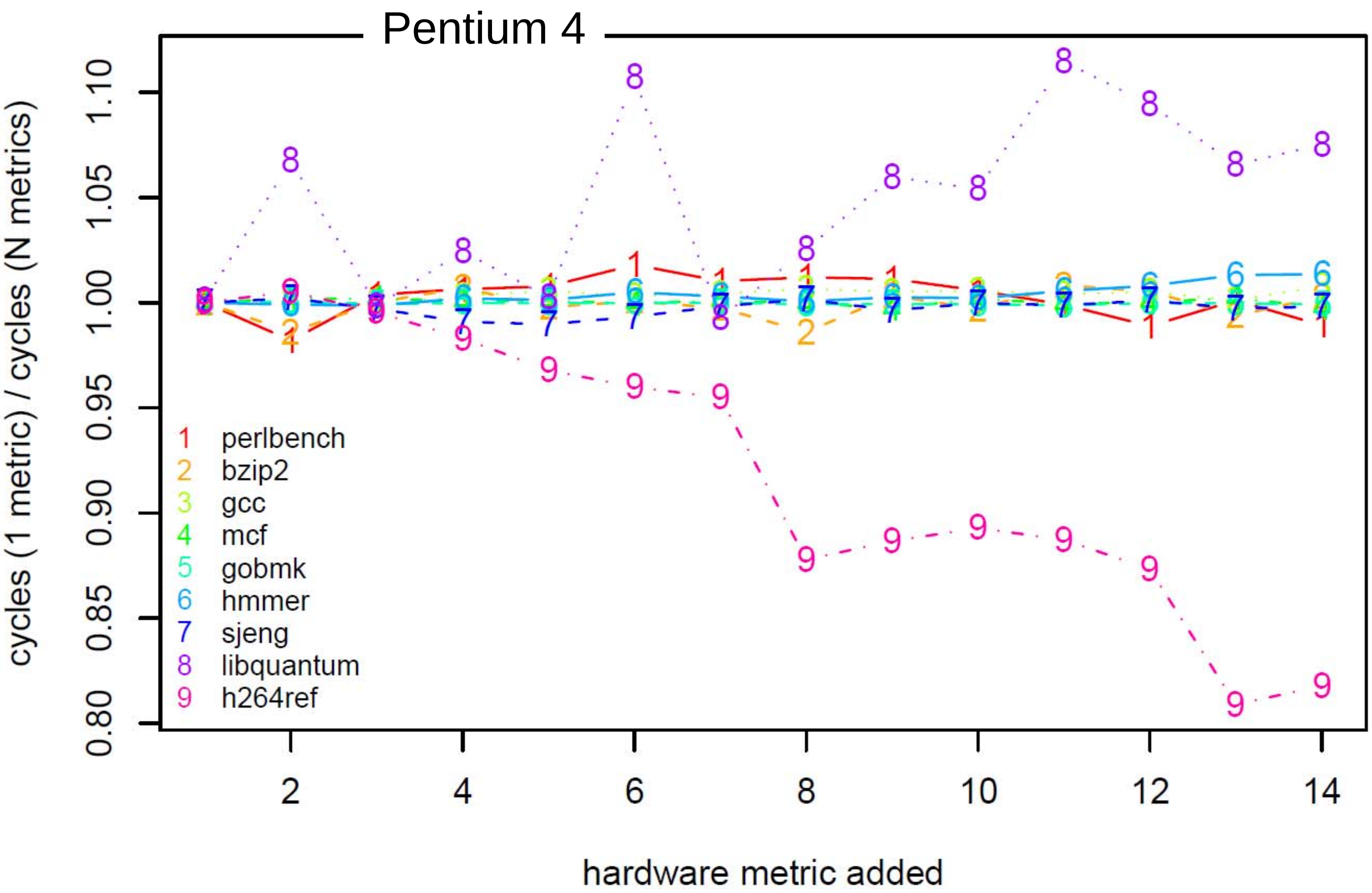
Why do objects shift around?

```
           main() {
0xF000       while (...) {
             recordAtStart()
0xF004         ...
             }
           }
0xF800     void f () {
0xF004       ...
           }
```

Why do objects shift around?

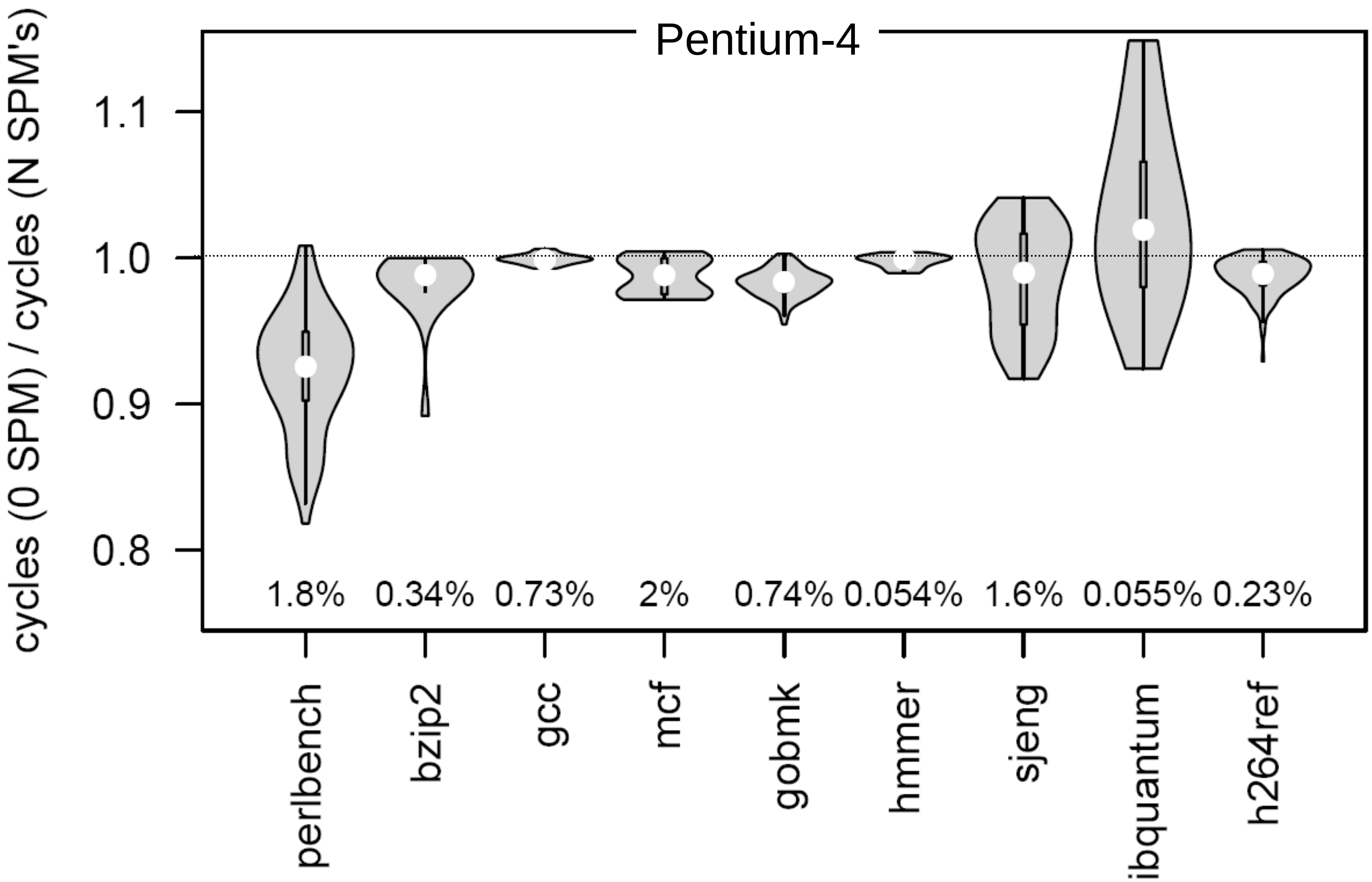
```
          main() {
0xF000      recordAtStart()
0xF004      while (...) {
              ...
            }
0xF800    } recordAtEnd()
0xF004    void f () {
0xF008      ...
          }
```

Observer effect due to shifts when collecting hardware metrics



The "insignificant" shift when using hardware counters causes observer effect!

Observer effect due to shifts when collecting software metrics



Small instruction overhead does not translate to small observer effect

Where did the observer effect come from?

- Instrumentation pushed code and data through some boundary
- These boundaries are everywhere...
  - Cache block sizes
  - Cache sizes
  - Page sizes
  - TLB sizes
  - ...
- Unfortunately, the problem gets worse...

Research model in experimental sciences:
(iii) Evaluate the new idea

Ask your co-workers

And draw broad conclusions:

Bridge cuts commute in half!

If you believe that study...
I have another bridge to sell you!

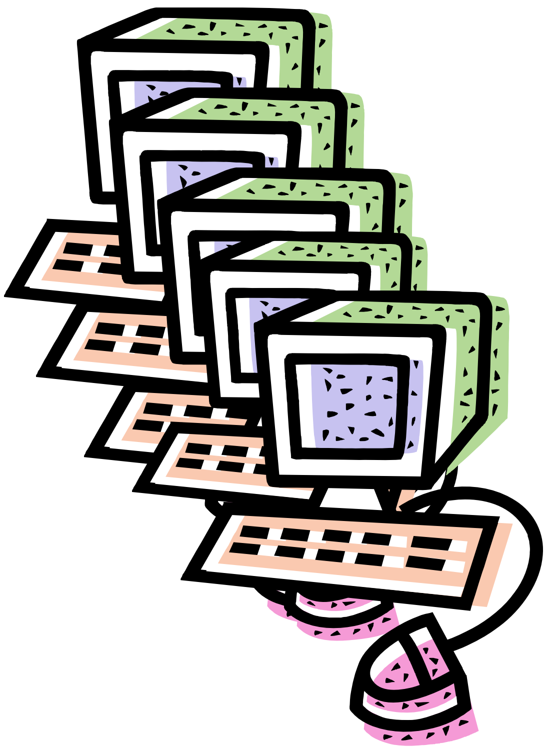
The study is easy to do but it is not right:

It is potentially biased

Your co-workers

may not be representative of the whole population

For experimental computer science

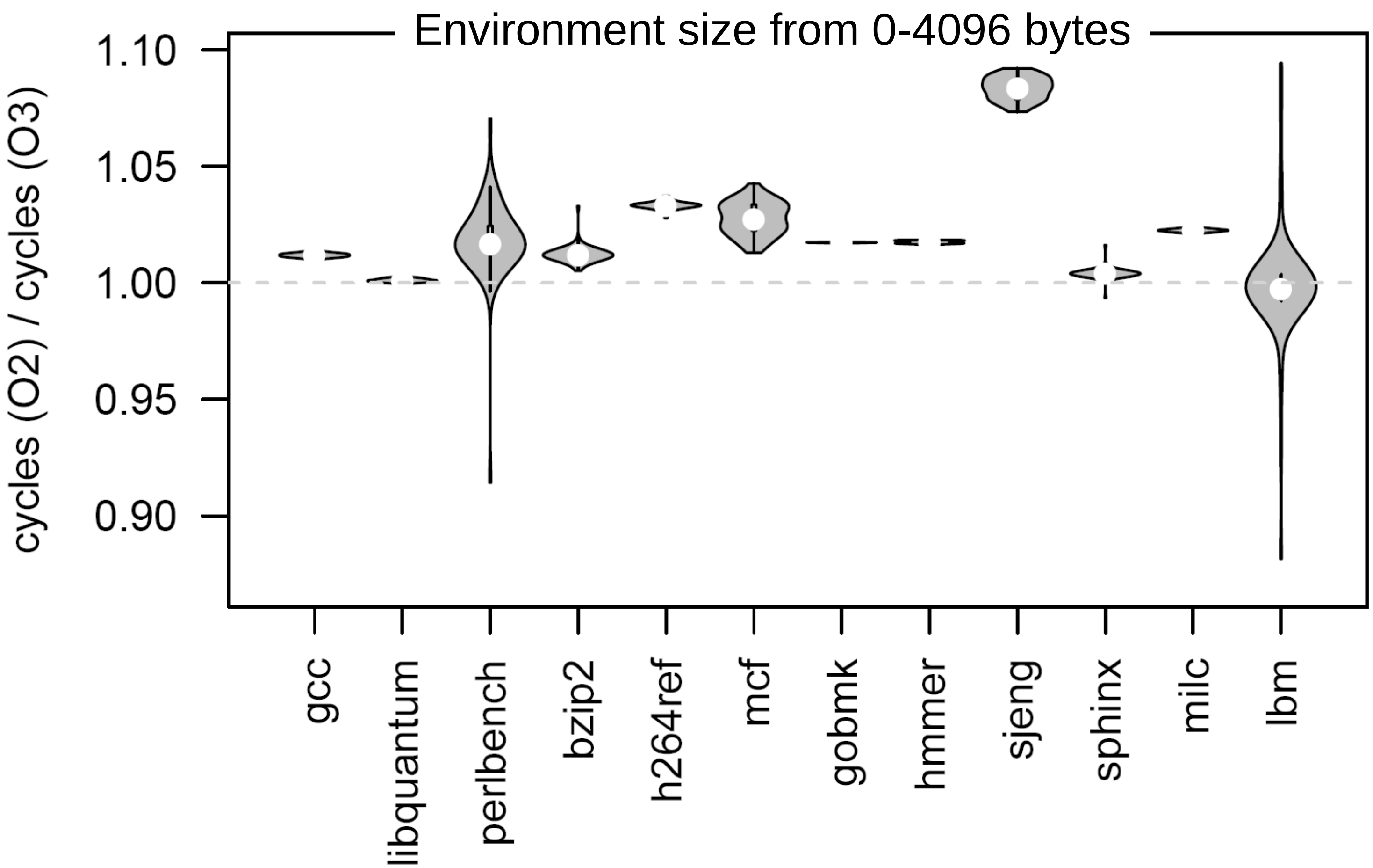


Your experimental setup

may not be representative of the whole population

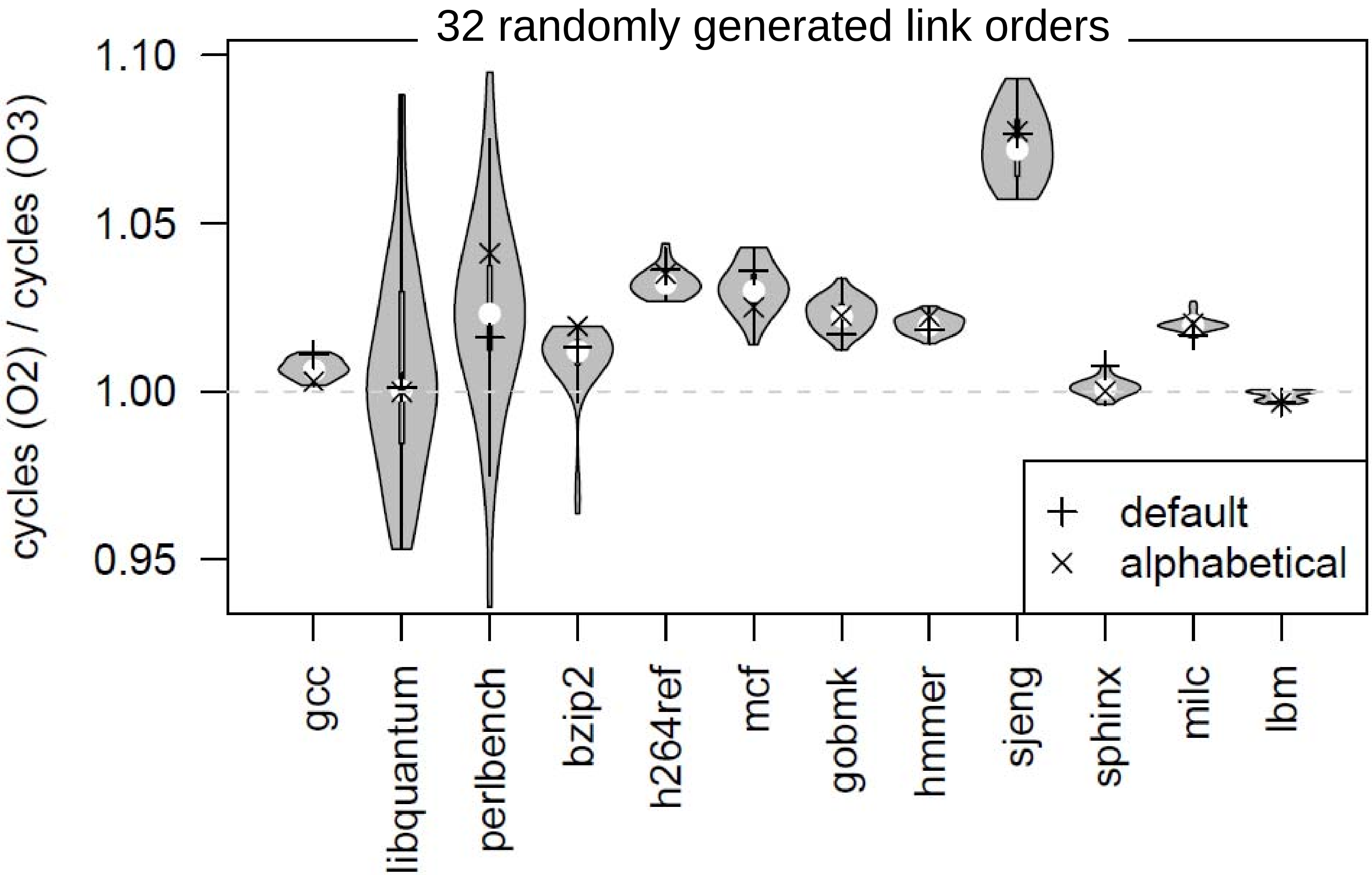
This phenomenon is called measurement bias

Trying different setups: environment variables (Core2/gcc)



The setting of irrelevant environment variables can lead to contradictory conclusions

Trying different setups: link order (Core2/gcc)



Order of .o files can lead to contradictory conclusions

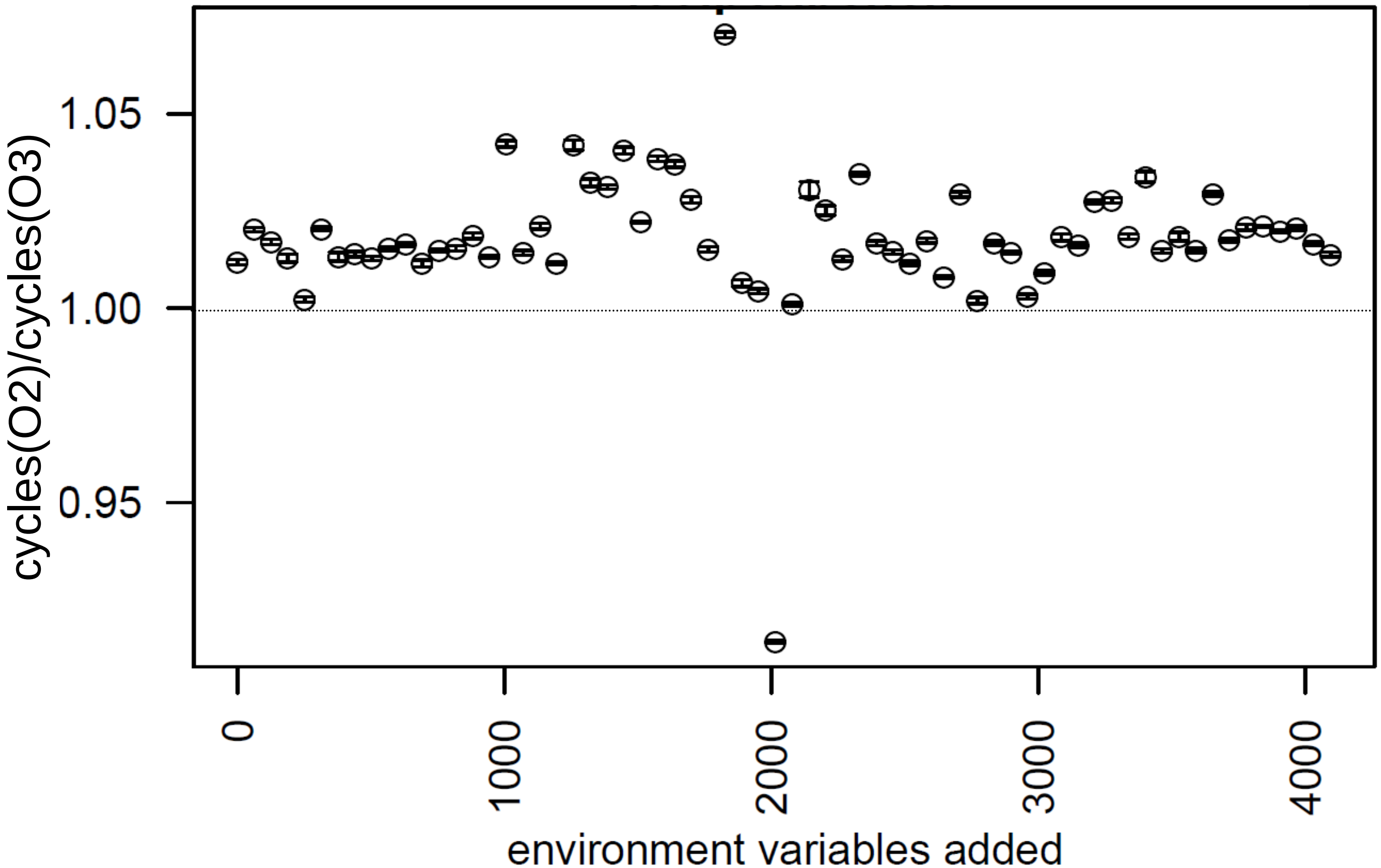
Default or alphabetical order may not be best

Where did the measurement bias come from?

- Environment variables and link order affect code and data layout
- Many other sources of measurement bias
  - Domain independent
    - E.g., temperature of the room
  - Domain dependent
    - E.g., heap size for a garbage collected system
  - We cannot easily suppress bias!

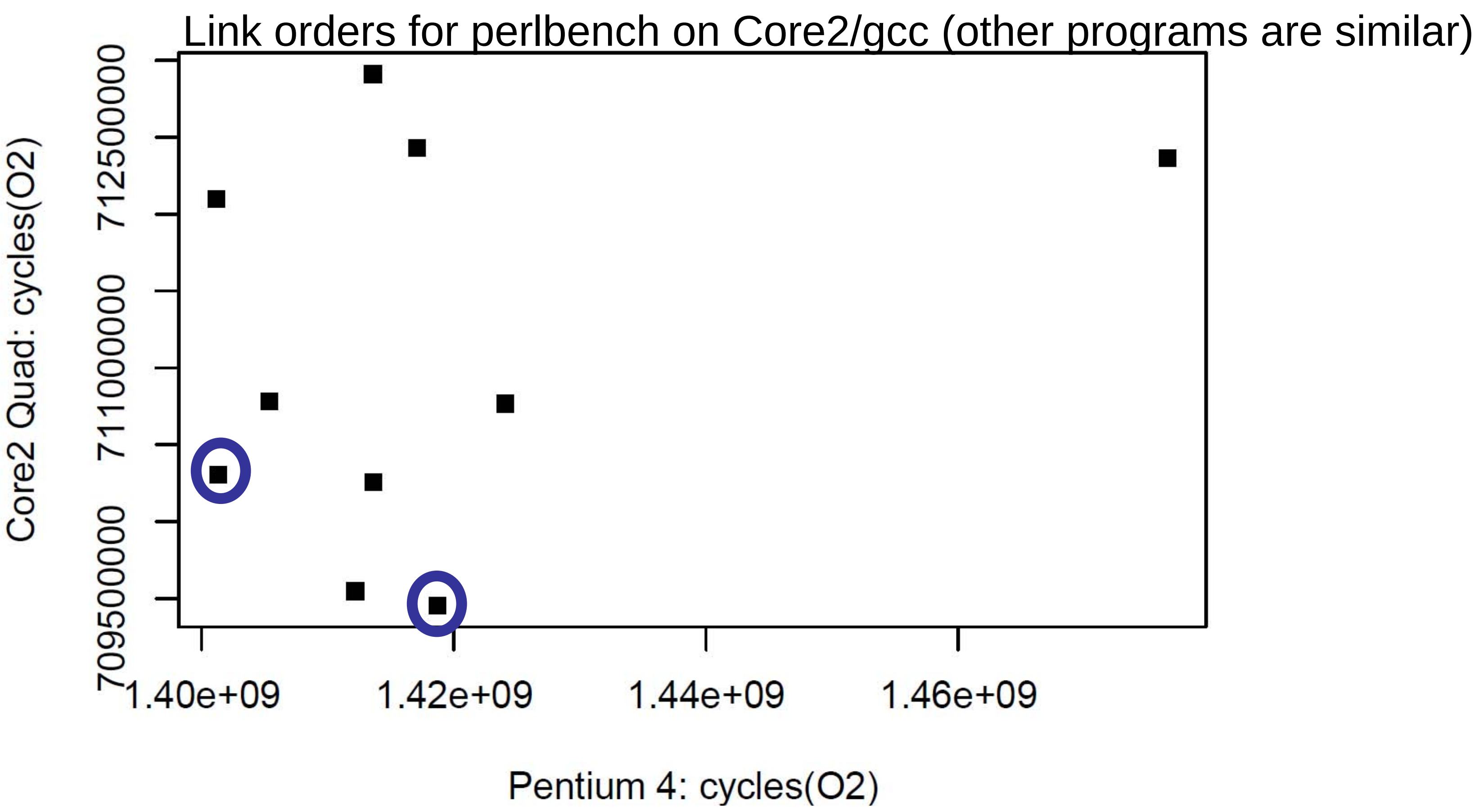
Is measurement bias predictable?

perlbench on Core2/gcc (other programs are similar)



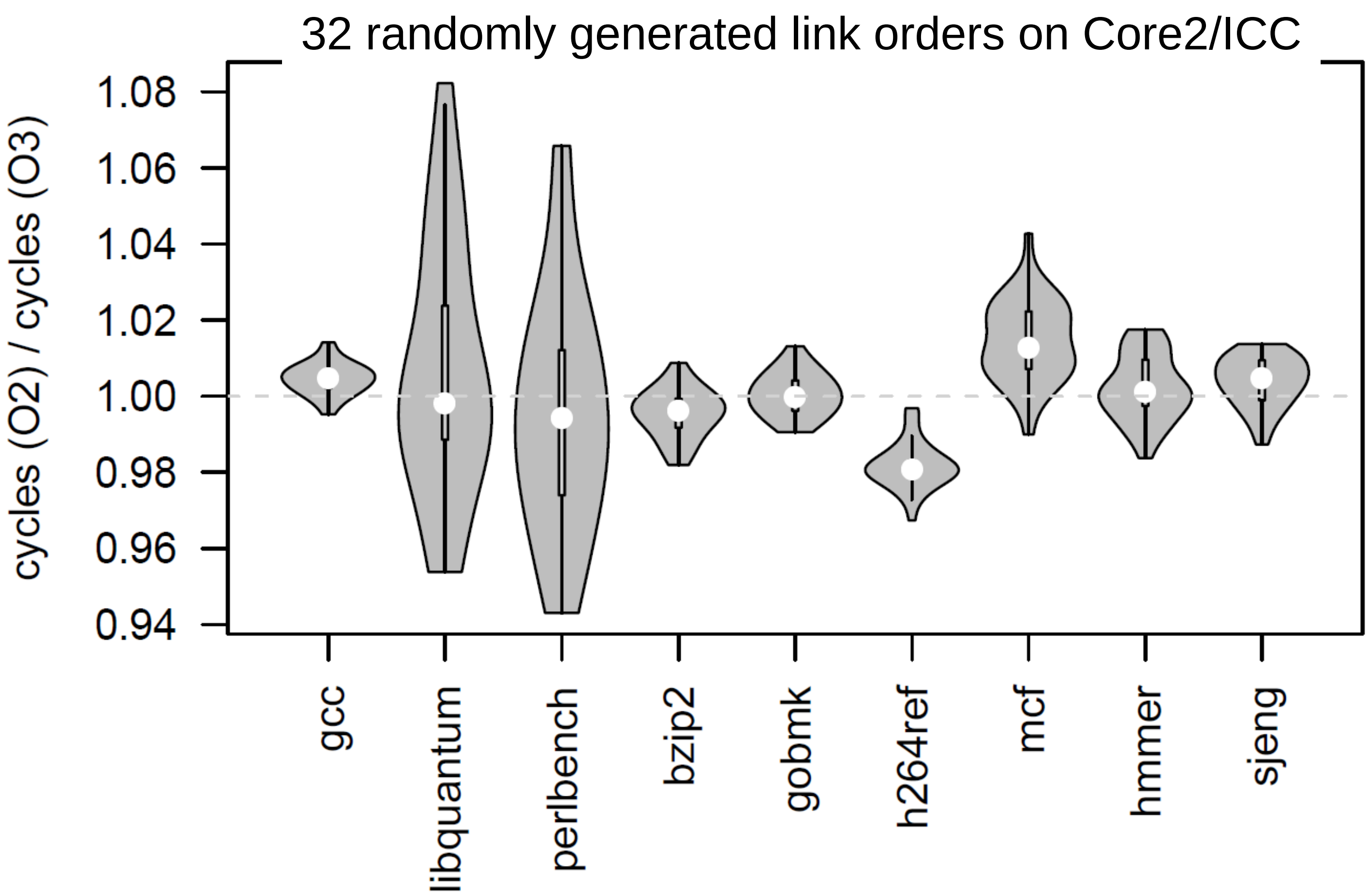
We saw no obvious trends

Are these phenomena consistent across microprocessors?



Different microprocessors have different “best” link orders

Are these phenomena caused by a poor compiler?



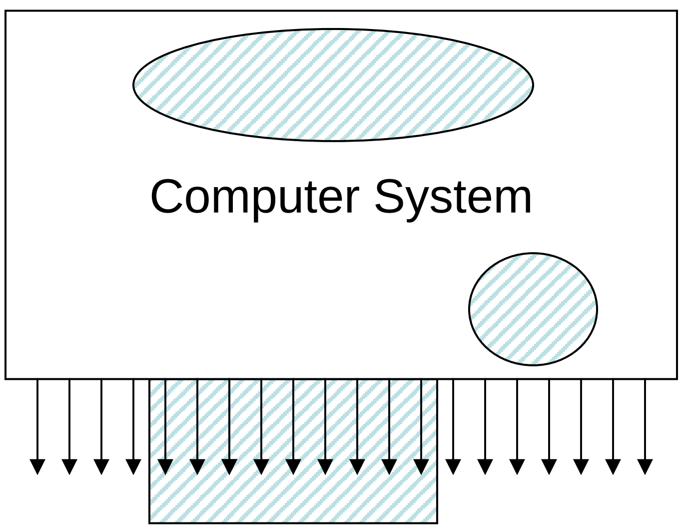
Intel’s C compiler exhibits the same phenomena

Are these phenomenon caused by a poor methodology?

- SPEC CPU 2006 C benchmarks with train inputs
- Used best practices
  - Minimally invasive instrumentation
  - Lightly loaded machines, local disks, …
  - 15 runs for each experiment
  - Reproduced experiments on 4 architectures and two compilers

It is incredibly difficult to explain these phenomena

Need to reason with information about system and data from system



Unavailable information

Unavailable data

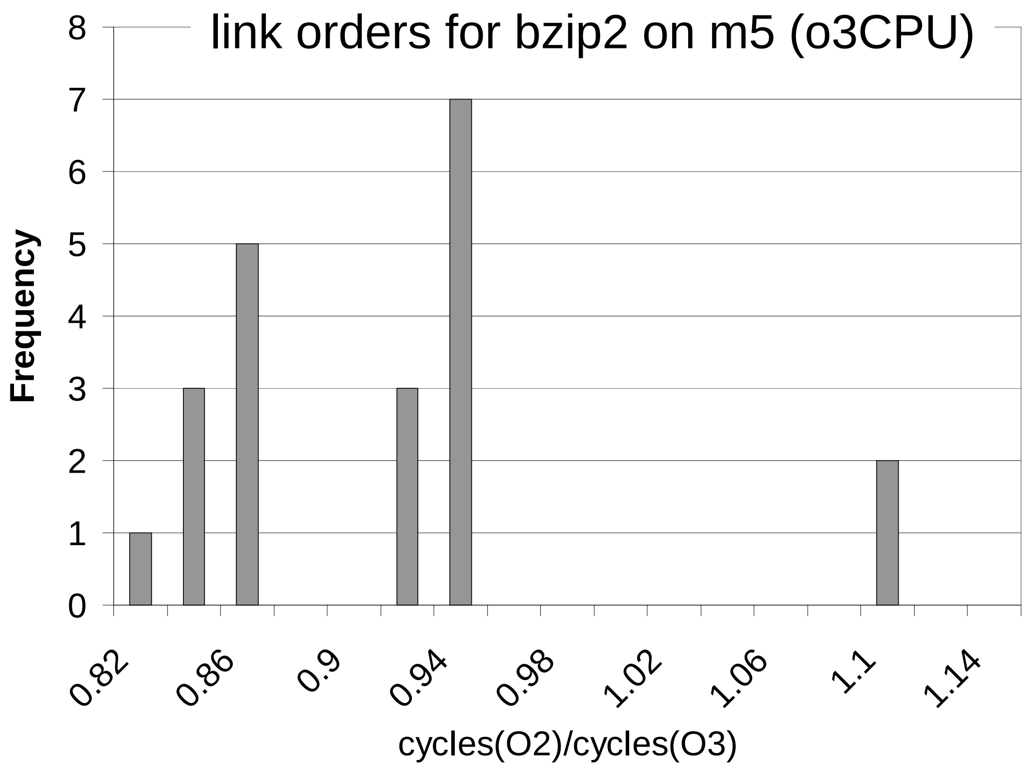
We can guess but it will be very hard to pin point the exact underlying causes

How do we deal with these issues

- We surveyed all papers from ASPLOS 2008, PACT, PLDI, and CGO 2007
- 88 papers had an evaluation section
- What do they do with observer effect and measurement bias?

36 papers used simulations

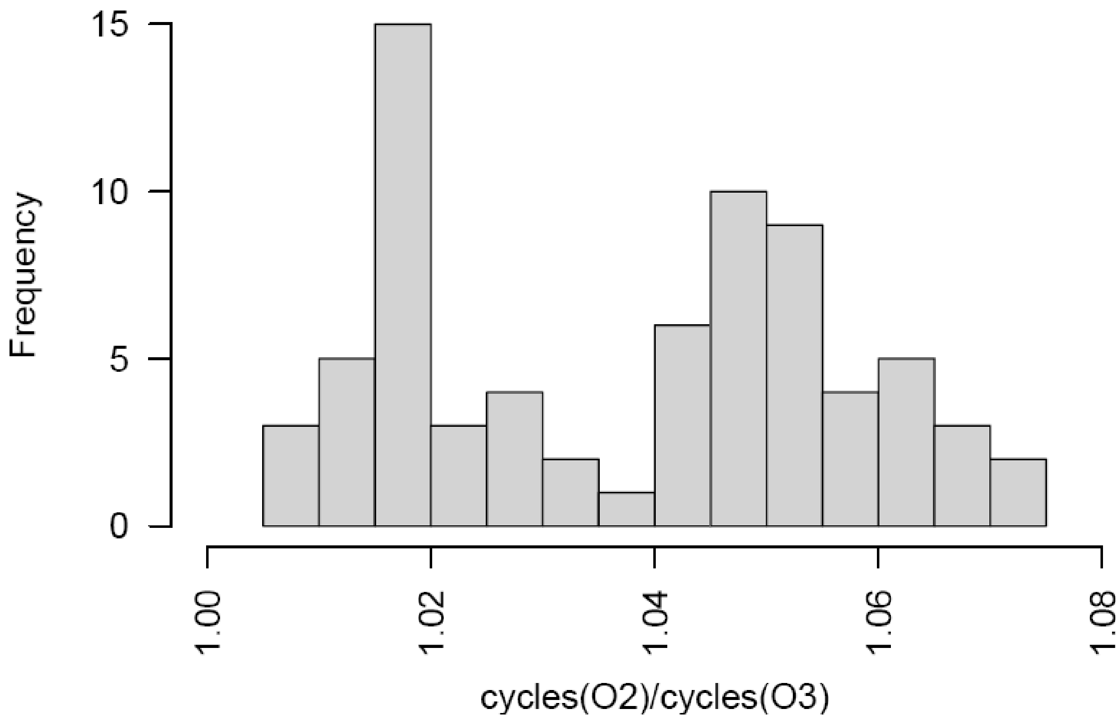
- Simulators avoid observer effect but can it avoid measurement bias?



Simulators also suffer from measurement bias

83 papers used more than one benchmark

- Can a diverse benchmark suite statistically factor out bias?

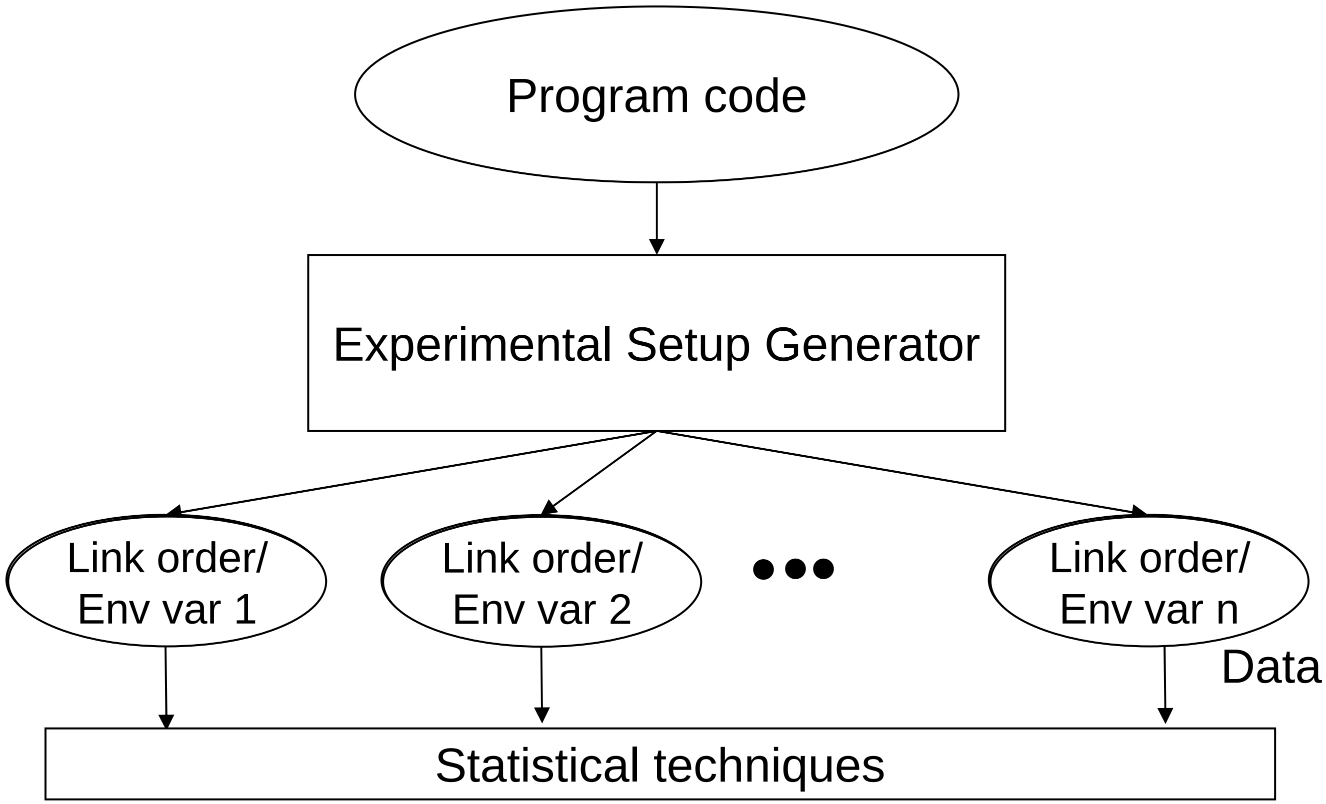


Maybe, but the 12 SPEC 2006 C benchmarks are not enough

How other sciences deal with these issues?

- Use many measurement setups
  - Statistically factor out bias
- Causality analysis
  - Confirm that the conclusions from the data are actually correct
- I'll show how we can use these techniques

Using many measurement setups: Setup randomization

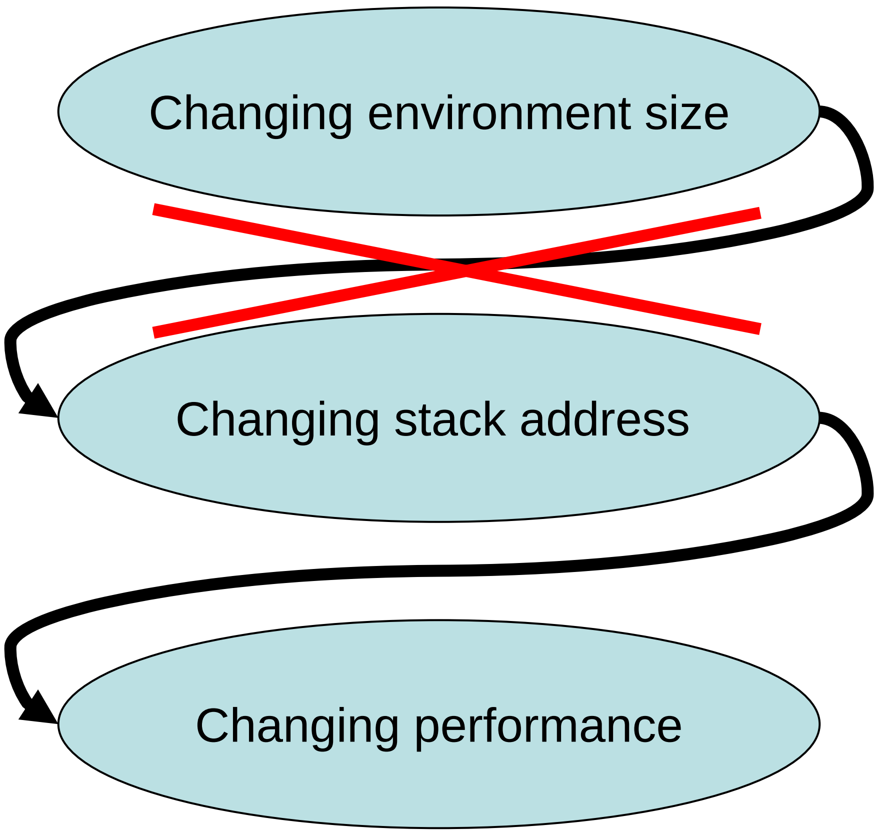


Popular in the social sciences

but its effectiveness depends on the representativeness of setups

Causality Analysis

1. Analyze data to arrive at hypothesis
2. Perform interventions to test hypothesis
   e.g., Fix stack address
3. Validate hypothesis
   e.g., changing environment variables does not change performance



Popular in sciences but difficult

A call to action

- We must reward careful experimentation
  - We need a cultural change!
- We need better workloads
  - Representative of your problem domain

A call to action (continued)

- We need more information on processors
  - Follow Sun's footsteps!
- We need better hardware support
  - Need metrics for all key components
  - Need mapping from HW to software events

Key lessons

- Small instrumentation ≠ Small observer effect
- Observer effect and bias are
  - Unpredictable,
  - Commonplace, and
  - Large enough to obfuscate data
- Other experimental sciences expend great effort to work around these phenomena
  - We have had it easy but we don't have it right…

Acknowledgements

- My collaborators
  - Liz Bradley, Matthias Hauswirth, Dan Knights, Mike Mozer, Todd Mytkowicz, Peter Sweeney
- Feedback on this talk
  - Mike Hind and many other colleagues at IBM
  - My research group and other friends at CU
- IIWSC for inviting me