Evaluating Performance using Ratio of Execution Times

Tomas Kalibera

My Background

- PL/Systems
  - R language: GNU R, (Purdue) FastR
  - Java: Ovm, OpenJDK
  - Garbage collection, interpretation, analysis
- Performance/Benchmarking
  - Methodology: modeling non-determinism
  - DaCapo benchmarks: observational study
  - Practice: DaCapo, SPEC CPU/JBB/JVM, Shootout, CD, CSIBE, FFT&kernels – Mono, Java, R
  - Teaching; Evaluate, Dagstuhl workshops

Talking about Performance

(fictional conversations in PL/systems)

Lunch at SW company

Joe: Any numbers yet for your compiler patch?
Ann: 9% on average, no big slowdowns.
Joe: That's really good!
Ann: Yes:) Or too good to be true, have to run more tests.

Coffee at CS dept of a uni

Cristine: How much slower is our VM than production VM X?
John: Now within 2x.
Cristine: Perfect, that allows us to claim our speedups are relevant.

Dissertation (MSc) committee meeting, the student got 18% speedup on FFT with kernel patch and claimed he could speed up applications by 18%

Erik: 18% speedup is far too small. We should reject.
Tim: 18% is great even for just FFT, great work. The generalizing claim is naïve.

Evaluating Time Ratio In Papers

	Papers	Reported Time Ratio
	2011	
ASPLOS	32	22
ISMM	13	9
PLDI	55	27
	2015	
ASPLOS	48	37
ISMM	12	10
PLDI	58	22
Total	218	127 (**58%**)

Important Decisions in Evaluations involving Time Ratio

- Which ratio?
  - Opinions, ratio games and confusion
- Averaging
  - Which mean, averaging over benchmarks
- Error estimate
  - Hardly ever any at all

Warning: some options given in the following are questionable and some are outright wrong!

Time Ratio: But Which One?

GNU-R, byte-code interpreter (B): 58s T_{old}
Purdue FastR (F): 16s T_{new}
(spectralnorm-alt4 [sn5] benchmark)

$\frac{T_{new}}{T_{old}}$	**0.28 (28%)**	**Ratio of execution times**
$1-\frac{T_{new}}{T_{old}}$	**0.72 (72%)**	**Percentage improvement in execution time**
$\frac{T_{old}}{T_{new}}$	**3.63 (363%, 3.63x)**	**Speedup**
$\frac{T_{old}}{T_{new}}-1$	**2.63 (263%)**	**"Percentage improvement in speed"**
$\frac{T_{old}}{T_{old}-T_{new}}$	**1.38 (138%)**	**SALE 250%**

Time Ratio: The Right Baseline?

GNU-R, byte-code compiler (B):	**58s**	T_B
Purdue FastR (F):	**16s**	T_F
GNU-R, AST interpreter (A):	**154s**	T_A

$$\frac{T_F}{T_B} = 0.28$$

$$\frac{T_B}{T_F} = 3.63$$

We reduced execution time to 28% of best performing alternative. We are 3.63x faster.

$$\frac{T_F}{T_A} = 0.10 \qquad \frac{T_B}{T_A} = 0.38$$

$$\frac{T_A}{T_F} = 9.63 \qquad \frac{T_A}{T_B} = 2.66$$

We reduced execution time of an existing system to 10%. The best performing alternative reduced it to 38%. We are 9.63x faster but the alternative only 2.66x faster.

Summarizing over Benchmarks

Language Shootout Benchmark Suite for R: n = 37 benchmarks.
Execution times with FastR: T_{Fi}
Execution times with GNU-R AST: T_{Ai}

Summarizing ratio $\frac{T_A}{T_F}$

$$\frac{1}{n}\sum_{i=1}^{n}\frac{T_{Ai}}{T_{Fi}}=12.91$$

Arithmetic mean of ratios

$$\frac{\sum_{i=1}^{n}T_{Ai}}{\sum_{i=1}^{n}T_{Fi}}=7.00$$

Ratio of sums

$$\sqrt[n]{\prod_{i=1}^{n}\frac{T_{Ai}}{T_{Fi}}}=8.53$$

Geometric mean of ratios

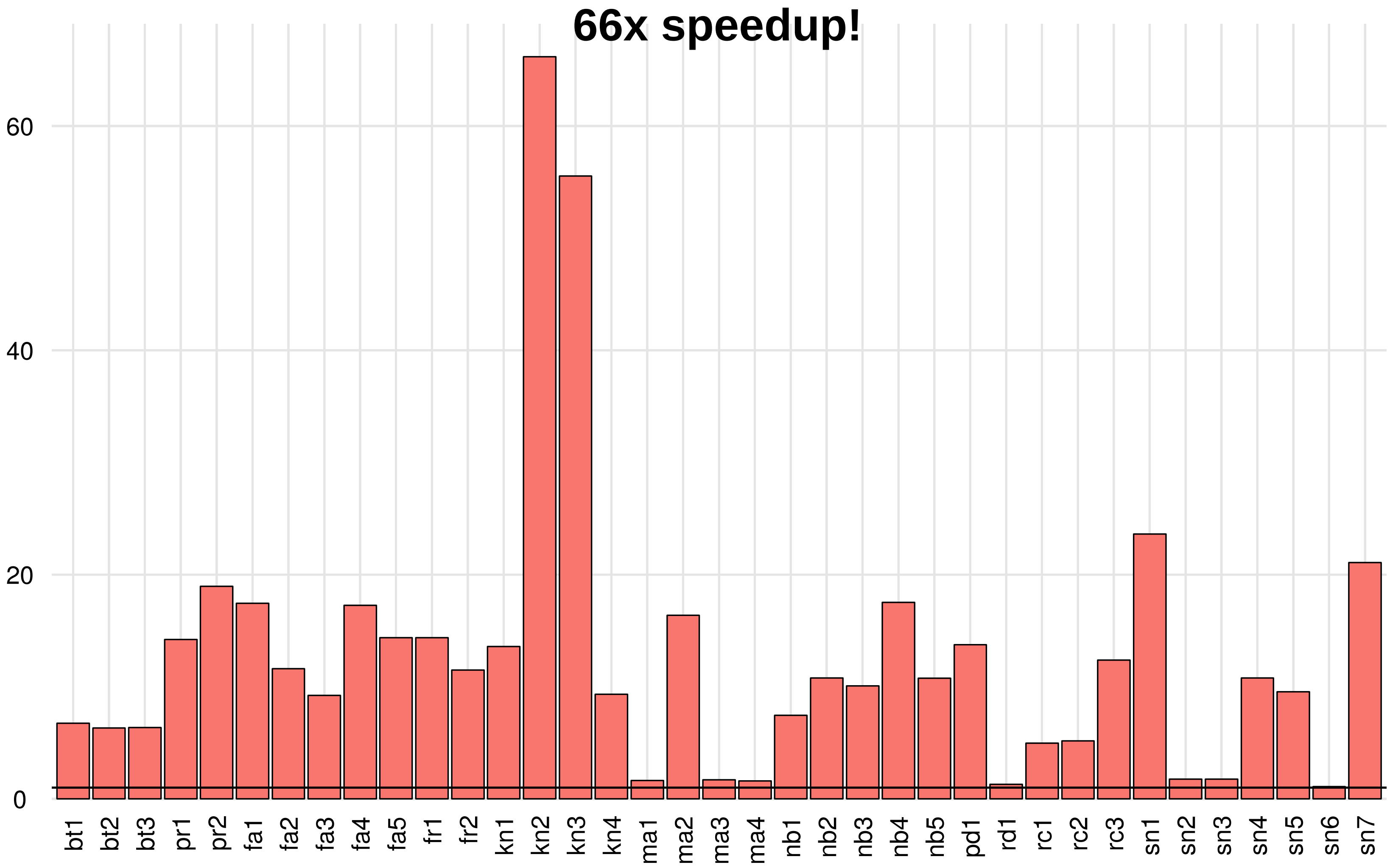
$$\frac{n}{\sum_{i=1}^{n}\frac{T_{Fi}}{T_{Ai}}}=5.02$$

Harmonic mean of ratios

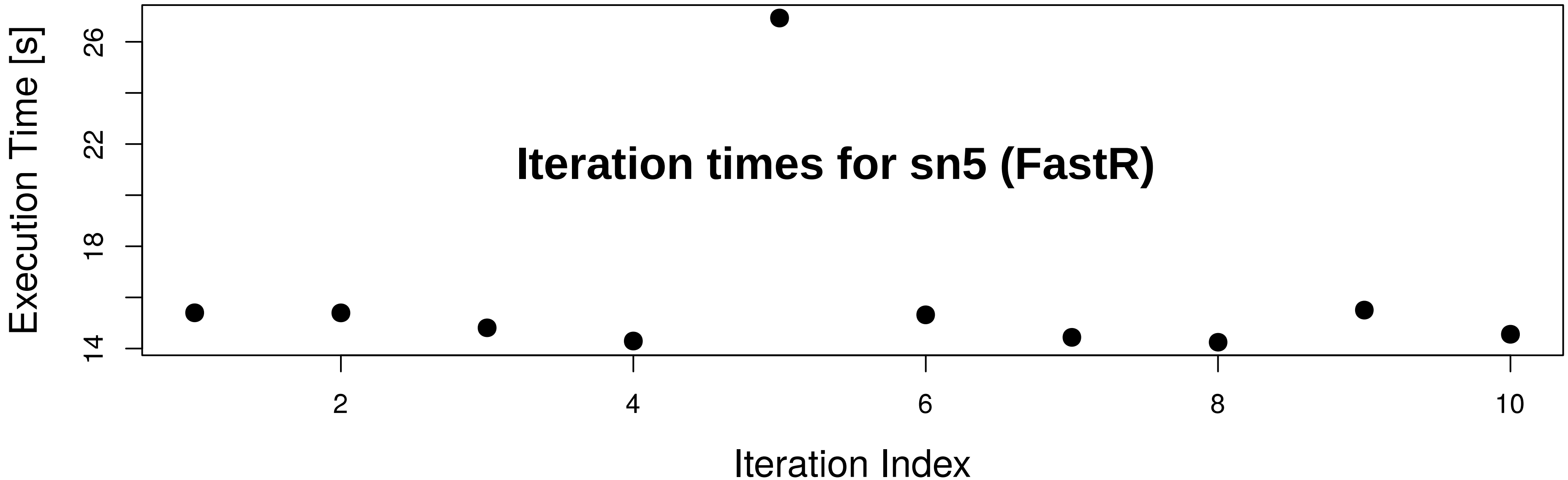
What is Hiding Behind the Mean?

$$\sqrt[n]{\prod_{i=1}^{n} \frac{T_{Ai}}{T_{Fi}}} = 8.53$$

Geometric mean of ratios



Repetition and Error Estimate



Percentile bootstrap 95% confidence interval for the mean

```r
cfsingle <- function(x) {
  means <- sapply(1:10000,
      function(i) mean(sample(x, replace = TRUE)) )
  sort(means)[c(250, 9750)]
}
```

Sn5 with FastR takes 16.6 ± 2.0s with 95% confidence.

Repetition and Error Estimate

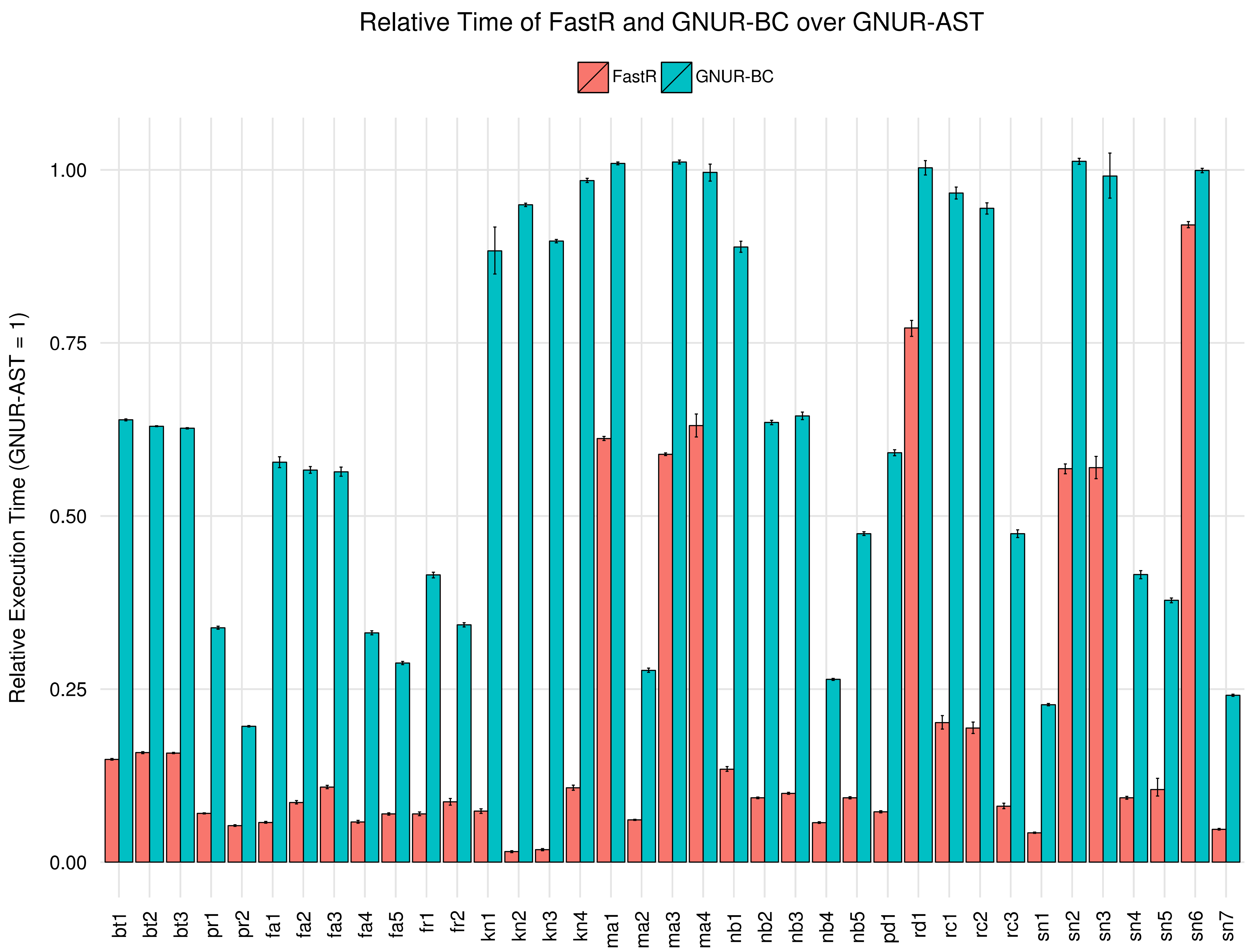
Percentile bootstrap 95% confidence interval for the ratio of means.
Input:

- **x – vector of iteration times for nominator**
- **Y – vector of iteration times for denominator**

```
cfratio <- function(x, y) {
  means <- sapply(1:10000, function(i) {
    xs <- sample(x, replace = TRUE)
    ys <- sample(y, replace = TRUE)
    mean(xs) / mean(ys)
  })
  sort(means)[c(250, 9750)]
}
```

The speedup of FastR over GNU-R AST on sn5 is 9.4 ± 1.1x.

FastR reduces execution time of sn5 over GNU-R AST to 10.8 ± 1.3%.



Relative Time of FastR and GNUR-BC over GNUR-AST
FastR
GNUR-BC
Relative Execution Time (GNUR-AST = 1)
0.00
0.25
0.50
0.75
1.00
bt1
bt2
bt3
pr1
pr2
fa1
fa2
fa3
fa4
fa5
fr1
fr2
kn1
kn2
kn3
kn4
ma1
ma2
ma3
ma4
nb1
nb2
nb3
nb4
nb5
pd1
rd1
rc1
rc2
rc3
sn1
sn2
sn3
sn4
sn5
sn6
sn7

Repetition and Error Estimate

Percentile bootstrap 95% confidence interval for the geometric mean..
Input:
- **xr – vector of ratios (one for each benchmark, calculated as ratio of iteration means))**

```
cfgmean <- function(xr) {

  gmean <- function(x) exp(mean(log(x)))

  gmeans <- sapply(1:10000, function(i)
    gmean(sample(xr, replace = TRUE)) )

  sort(gmeans)[c(250, 9750)]
}
```

The geomean speedup of FastR over GNU-R AST is 8.9 ± 2.7x.

On geomean, FastR reduces execution time over GNU-R AST to 12.4 ± 3.8%.

Summary

- Decisions for R study
  - Ratio for graphs $\frac{T_{new}}{T_{old}}$
  - Ratio in text given as inverse $\frac{T_{old}}{T_{new}}$
  - 95% bootstrap confidence intervals for ratios of individual benchmarks
  - Geometric mean over suite in text with huge disclaimer
- References
  - ISMM'13, Rigorous benchmarking in reasonable time
  - OOPSLA'12, A black-box approach to understanding concurrency in DaCapo
  - VEE'15, A Fast Abstract Syntax Tree Interpreter for R
  - Uni of Kent technical report, https://kar.kent.ac.uk/30809,
    Quantifying Performance Changes with Effect Size Confidence Intervals

Additional Resources

Jain: The Art of Computer Systems Performance Analysis

Lilja: Measuring Computer Performance: A Practitioner's Guide

Kirkup: Experimental Methods: An Introduction to the Analysis and Presentation of Data

NIST/SEMATECH: Engineering Statistics Handbook, http://www.itl.nist.gov/div898/handbook/

Wassermann: All of Statistics: A Concise Course in Statistical Inference

Evaluate Collaboratory: Experimental Evaluation of Software and Systems in Computer Science, http://evaluate.inf.usi.ch/