TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science Institute of Systems Architecture, Operating Systems Group

HYPOTHESES AND EXPERIMENTS

Maksym Planeta (mplaneta@os.inf.tu-dresden.de)
Julian Stecklina
Björn Döbel

Dresden, 2017/4/12

00 Earn a Beverage!

For every lecture-friendly real-world example of the mistakes I present in this lecture, I offer the finder a beverage of his/her choice at the ASCII.

Just send me a mail.

01 Outline

01 What's This About?

What do we want to achieve with a research paper:

- Communicate research findings.
- **Convince others** of the correctness of these findings.

01 One Example: NewNix

Consider a hypothetical operating system *NewNix* that we might have implemented.
What do you want to show?

- The network stack performs better than X.
- The disk subsystem is more efficient than X.
- It uses less energy for task Y than X.

These are called *hypotheses*.

01 A Scientific Hypothesis

- aligns with preliminary data,
- can be used to make a **prediction**,
- predictions can be **tested** with experiments and
- **falsified** (at least in principle),

01 Hypothesis Example 1

"NewNix's TCP/IP stack is better than Linux' stack."

01 Hypothesis Example 1

"NewNix's TCP/IP stack is better than Linux' stack."

What is the prediction?
How can it be tested and falsified?

01 Hypothesis Example 2

Observed: Lot of time spent in socket buffer code for small packet handling in Linux' TCP/IP stack.

Hypothesis: *"Complex socket buffer data structures are a cause for inefficient handling of small packets."*

Predicted: A network stack with simple data structures, will be able to handle more small packets in the same amount of time.

01 Experiments

How do we know that a hypothesis is true? In general, we can't. Hypothesis can be

- supported (doesn't make them universally true!) or,
- **refuted**.

Systems research is mostly empirical, so we use **experiments** to do that.

See *The Logic of Scientific Discovery* (original *Logik der Forschung*) by Karl Popper.

02 Outline

02 Designing Experiments

Paper has certain hypotheses (claims). They have to be verified by experiments. Design experiments to

- test explicit and implicit assumptions (don't need to be in the final paper)
- substantiate claims made in the paper by testing predictions.

02 Designing Experiments

Experiments must be **repeatable**:

- **Automate! Automate! Automate!**
  - System configuration
  - Booting
  - Your Experiment
  - Analysis and Plotting
- Document system and system configuration.
- *Publish and keep* raw data, source code, scripts, random seeds . . .

If an experiment cannot be repeated, it's result is **meaningless**.
See Ten Simple Rules for Reproducible Computational Research.

02 Reasons to Repeat

Reasons for **you** to repeat:

- Code was buggy,
- Code was inefficient,
- Configuration incomplete,
- Documentation incomplete,
- Need new data.

Reasons for **others** to repeat:

- **Validate your findings**,
- **Refute your findings**,
- Improve on your work.

Experiments must be repeatable **by others**.

02 Watch out for . . .

System configuration can be particularly hard. Watch out for:

- Latency benchmark
  - Power management
  - IRQ rate throttling
  - IRQ affinity
  - . . .
- Memory-intense benchmark
  - NUMA
  - Huge-page support
  - memory speed
  - . . .
- Throughput/compute benchmark
  - Hyperthreading
  - Thread affinity
  - Compiler changes
  - Room temperature `:-D`
  - . . .

02 Where do my Cycles go?

Taking timestamps at predefined points in the program (**tracing**).

- Easy to interpret
- Fine-grained measurements have high overhead

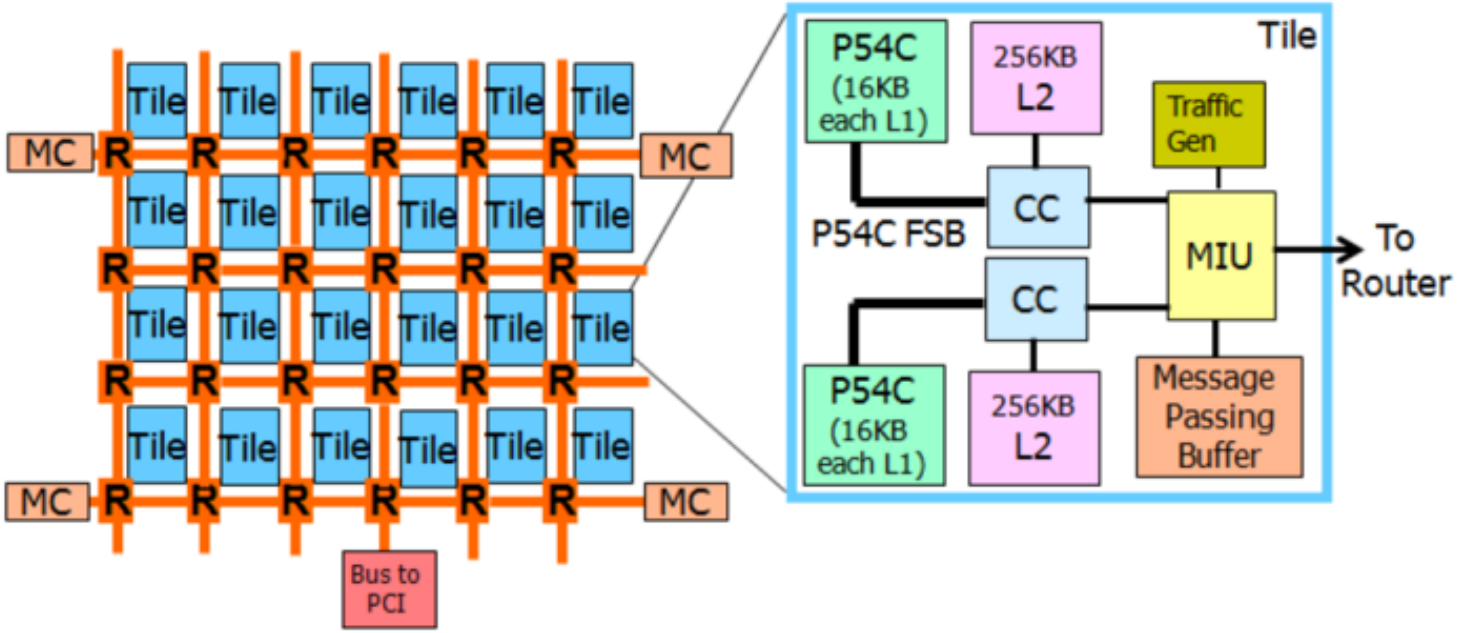
Interrupting the system regularly and see where it is (**statistical profiling**).

- As low overhead as you want
- Fine-grained measurements slow
- Latency measurements hard

Use the right tool for the job, but be aware of the limitations. **Verify** with unmodified data!

02 A Tale of Caution [4]

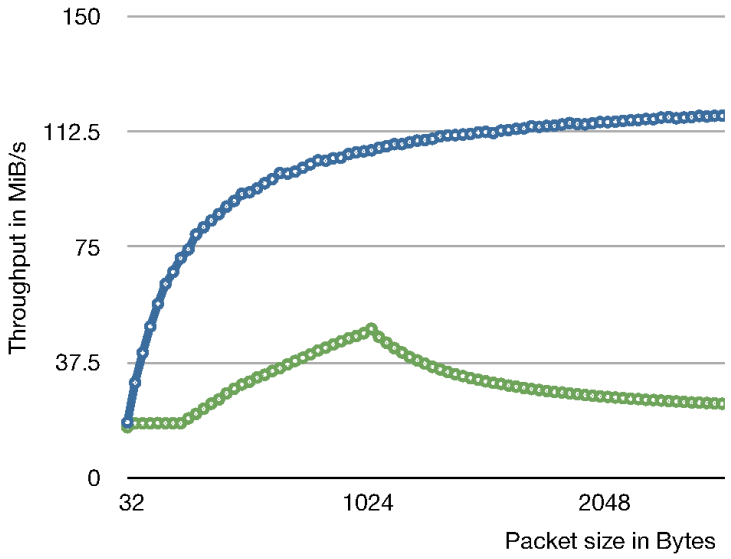
Investigated memory throughput as a function of packet size on the *Singlechip Cloud Computer* (SCC).



Source: The 48-core SCC Processor: the Programmer's View

02 Confusion

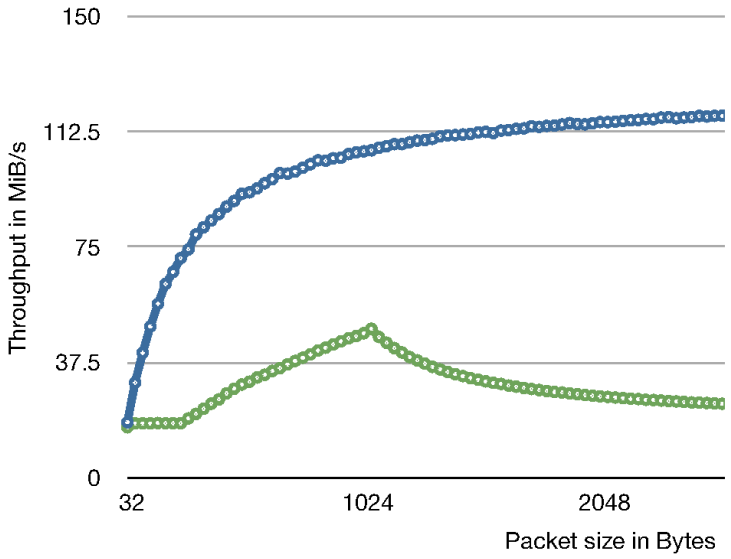
Expected: monotonously incr. curve
Got: strange peak at 1024 bytes



02 Confusion

Expected: monotonously incr. curve
Got: strange peak at 1024 bytes

Got blue curve while iterating over packet size in **increasing** order, green curve by iterating in **descending** order.

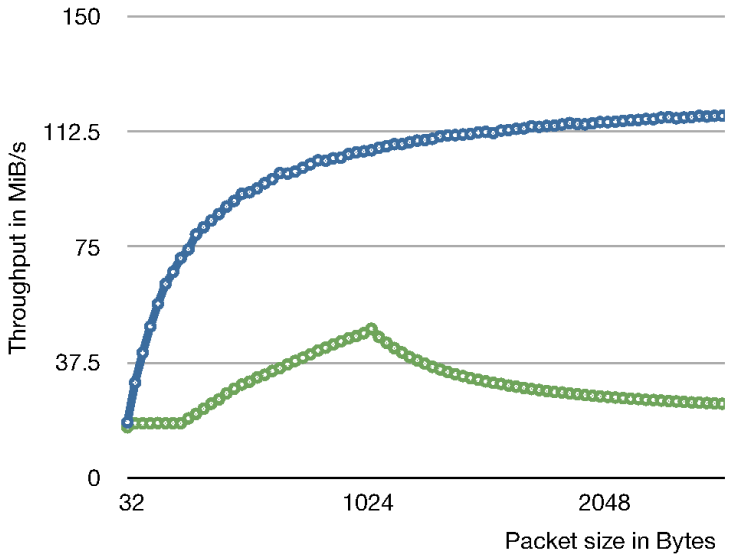


02 Confusion

Expected: monotonously incr. curve
Got: strange peak at 1024 bytes

Got blue curve while iterating over packet size in **increasing** order, green curve by iterating in **descending** order.

Measurements influence each other!



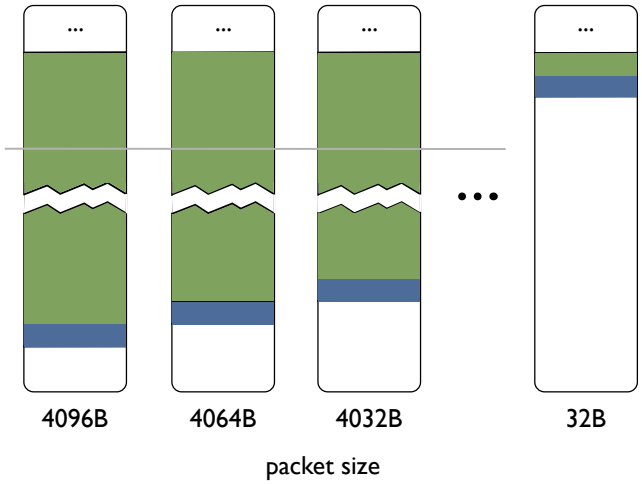
02 SCC Background

SCC has a write-back cache, but only allocates a cache line, when data is **read**, not when data is written. Instruction sequences such as

```
*ptr =  42;
 foo = *ptr;
```

can miss the cache **twice**. Repeated writes without reads will **always** miss.

02 Descending Measurement



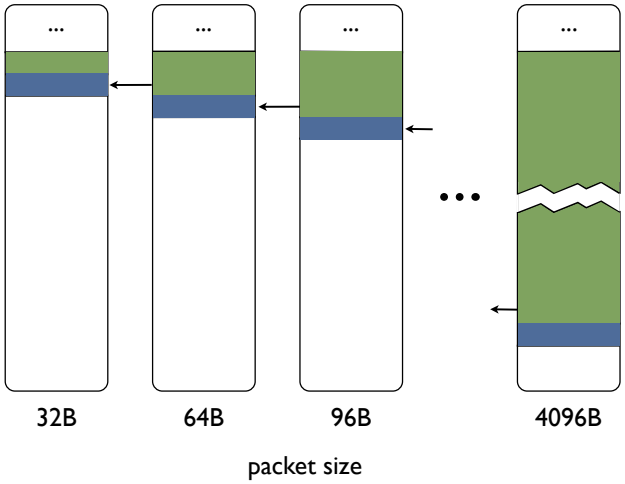
Green measurement buffer was allocated on the stack and only written.

Grey line indicates maximum stack usage before measurement, i.e. memory partly in cache.

Blue is the function prologue (read and written).

Memory writes miss below the grey line and partly hit above the grey line. Best performance for largest buffer that fits!

02 Ascending Measurement



Function prologue allocates cache line for next benchmark iteration.

All writes hit the cache (if it is large enough). Best performance for largest buffer.

Better solution is to manually prefetch the buffer before each experiment. Order doesn't matter anymore.

02 Lessons Learned

- If the order of experiments change their result, they are interfering with each other.
- Results of interfering experiments **useless** (usually)!
- Sanitize the environment before starting experiments:
  - Reboot
  - Warmup runs
  - Cache
  - Memory Consumption
  - ...
- Discuss your experiment setups with other people!

03 Outline

03 Measuring

We want to measure a value x (*random variable*), which could be:

- network throughput,
- CPU utilization,
- power consumption,
- ...

Our experimentation setup is slightly inaccurate and will give us slightly different values x_i everytime we repeat the experiment (*random sample*).

What value is "correct"? How to report it?

03 Descriptive Statistics 101

(Arithmetic) Average	$\mu_X = \bar{x} \quad = \frac{1}{n}\sum_{i=1}^{n} x_i$
Median	Middle value in an ordered data set
Mode	Most frequent value
Sample Variance	$\sigma_x^2 \quad = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$
Sample Standard Deviation	$\sigma_X \quad = \sqrt{\sigma_x^2}$

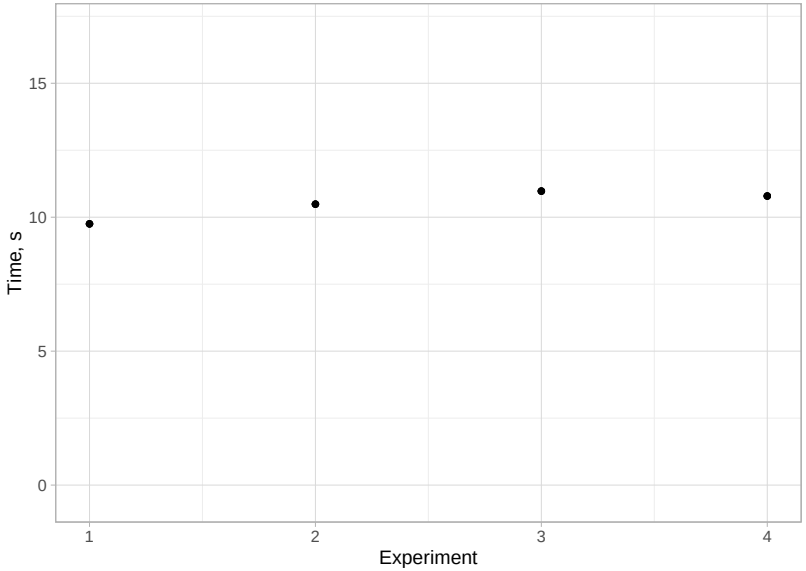
The $\frac{1}{n-1}$ vs. $\frac{1}{n}$ issue is due to Bessel's correction.

03 Is 101 Statistics enough?

- Statistics may be hard.
- But sometimes simple things are just fine.
- Try to end up with the least work

03 First look

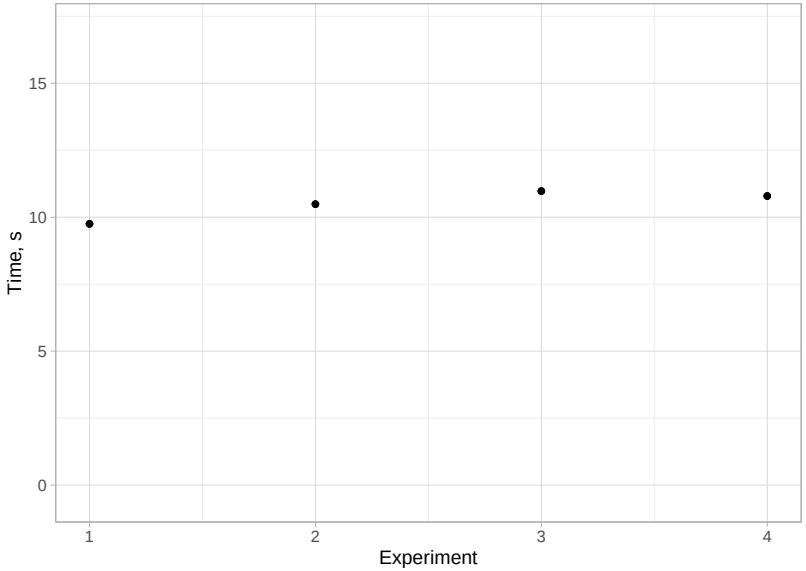
- We have averages, how to compare them?



03 First look

- We have averages, how to compare them?
- Chebyshev's inequality:

$$Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

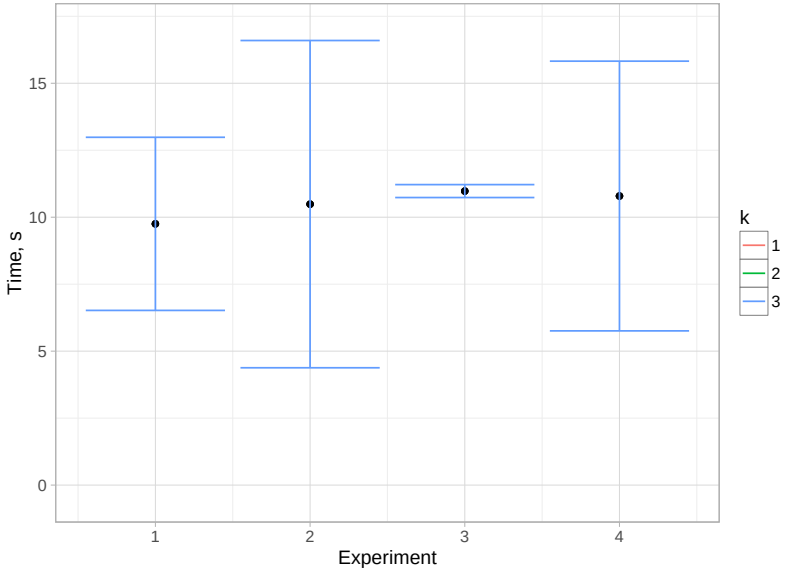


03 First look

- We have averages, how to compare them?
- Chebyshev's inequality:

$$Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

- $k = 3 : \frac{N_{in}}{N} = 89\%$

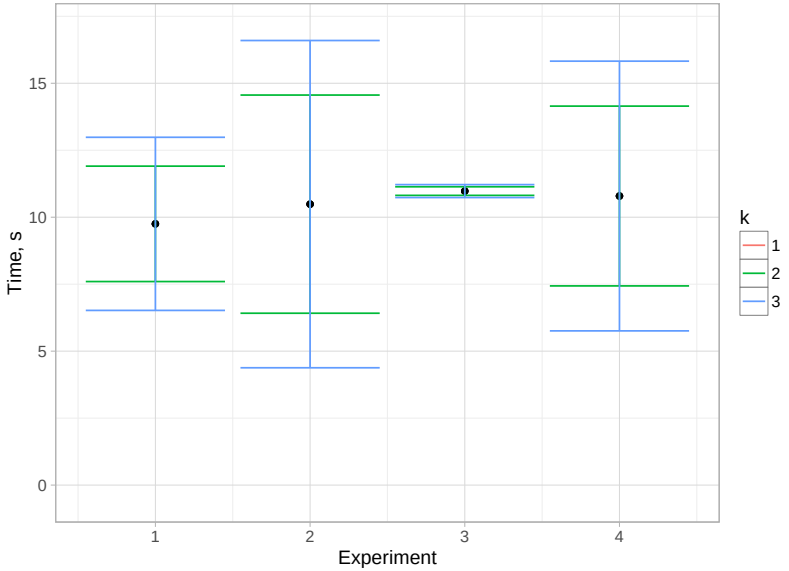


03 First look

- We have averages, how to compare them?
- Chebyshev's inequality:

$$Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

- $k = 3 : \frac{N_{in}}{N} = 89\%$
- $k = 2 : \frac{N_{in}}{N} = 75\%$

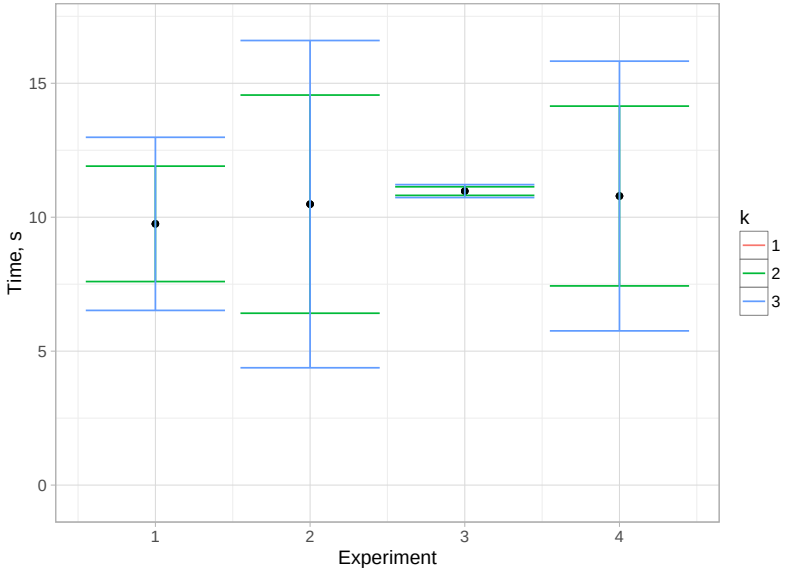


03 First look

- We have averages, how to compare them?
- Chebyshev's inequality:

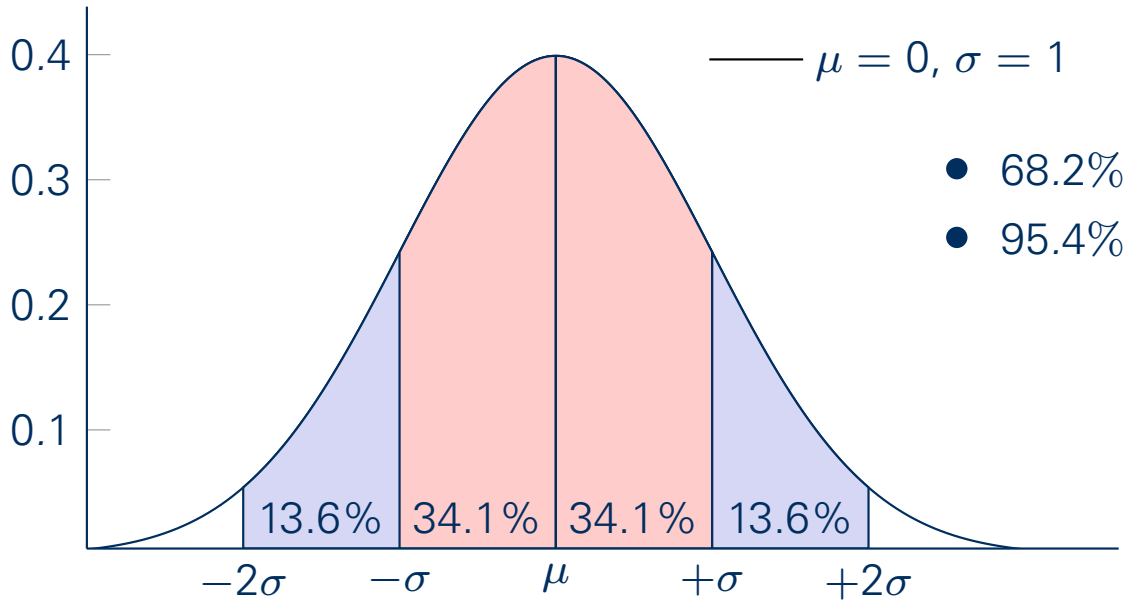
$$Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

- $k = 3 : \frac{N_{in}}{N} = 89\%$
- $k = 2 : \frac{N_{in}}{N} = 75\%$



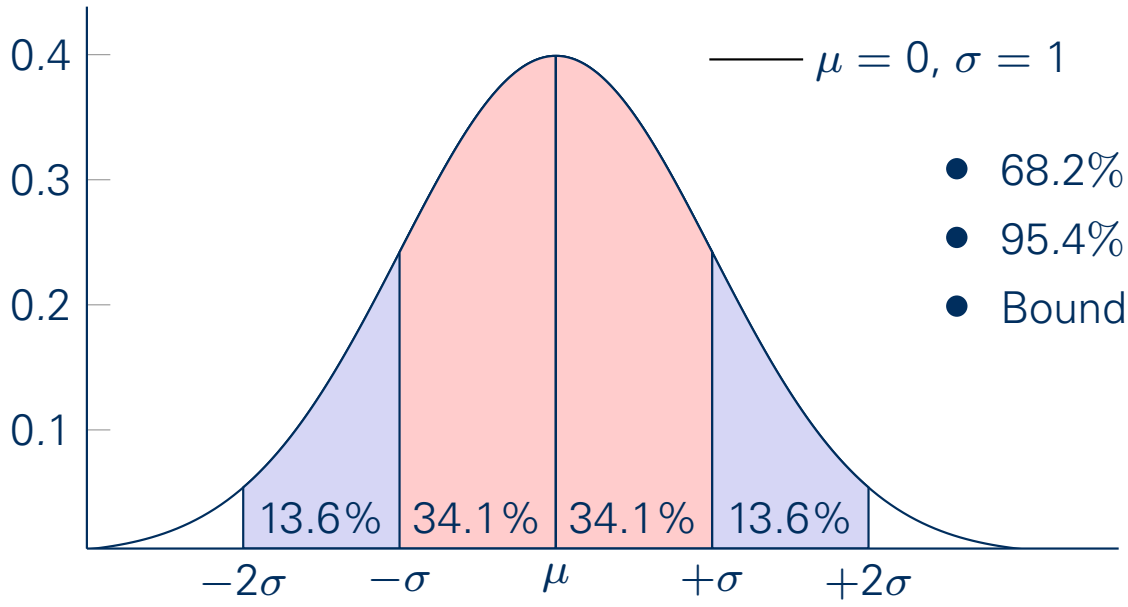
Can we narrow down the regions?

03 Three sigma rule of thumb



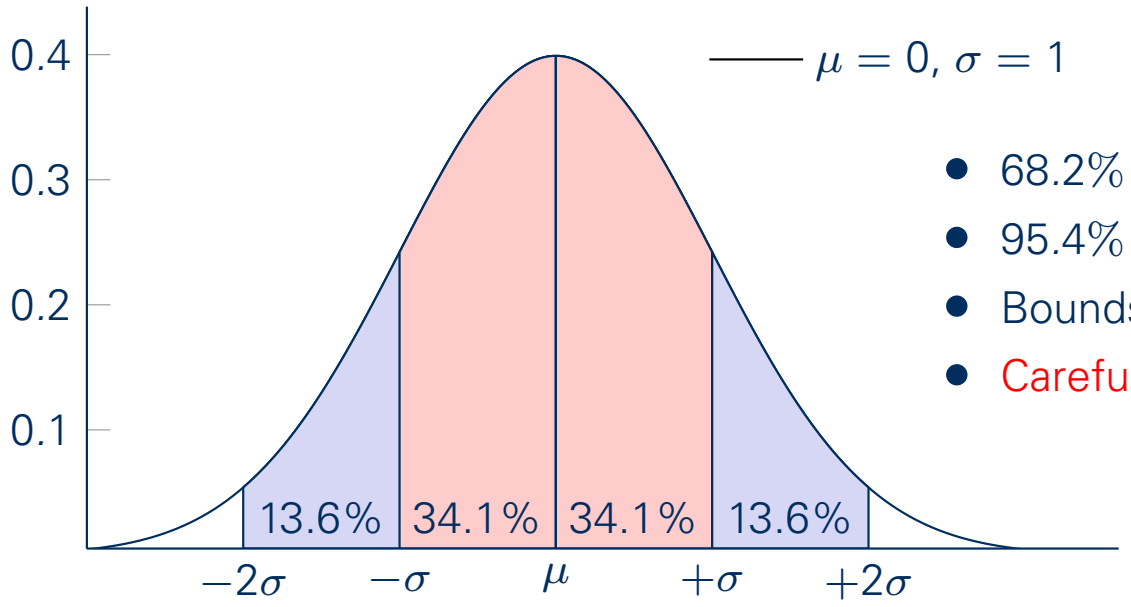
- 68.2% within $\mu \pm \sigma$
- 95.4% within $\mu \pm 2\sigma$

03 Three sigma rule of thumb



- 68.2% within $\mu \pm \sigma$
- 95.4% within $\mu \pm 2\sigma$
- Bounds are tighter

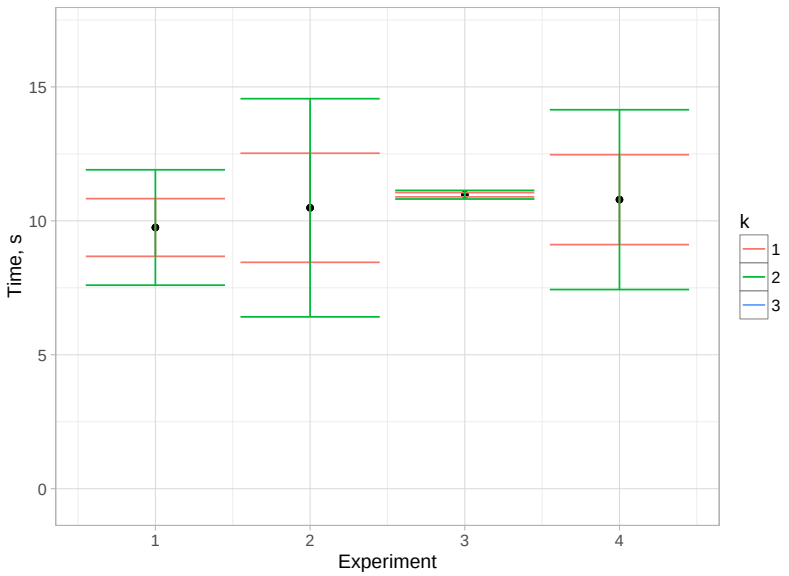
03 Three sigma rule of thumb



- 68.2% within $\mu \pm \sigma$
- 95.4% within $\mu \pm 2\sigma$
- Bounds are tighter
- Careful!

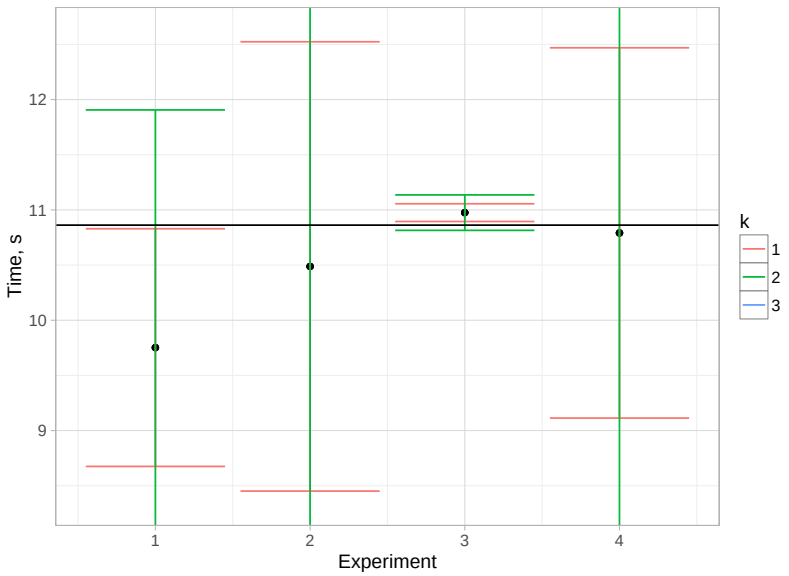
03 How tighter bounds look like?

- According to Chebyshev's inequality:
  75% of data points lies within $\mu \pm 2\sigma$
- According to three sigma rule: 68% of data points lies withing $\mu \pm \sigma$



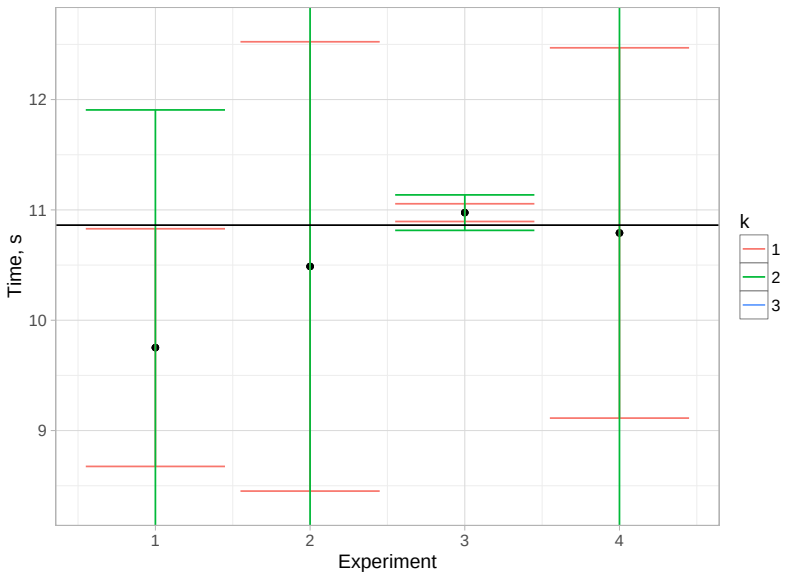
03 How tighter bounds look like?

- According to Chebyshev's inequality:
  75% of data points lies within $\mu \pm 2\sigma$
- According to three sigma rule: 68% of data points lies withing $\mu \pm \sigma$



03 How tighter bounds look like?

- According to Chebyshev's inequality:
  75% of data points lies within $\mu \pm 2\sigma$
- According to three sigma rule: 68% of data points lies withing $\mu \pm \sigma$
- There is now free lunch: distribution *has* to be normal.



03 Normal Distribution

Lots of data can be approximated as **normally distributed**. Especially, when it is the result of multiple random influences.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$P(a \leq X \leq b) = \int_a^b f(t)\,\mathrm{d}t$$

Central Limit Theorem: Arithmetic mean of a large sample is approx. normally distributed.

03 How do we know if the distribution is normal?

1. Work out a mental model
2. Decide if it *logical* to be normally distributed
   Latency typically is not.
3. Use statistical tools to prove that the distribution may be normal
4. Visualization for initial diagnosis:
   - Scatter plot
   - Histogram
   - Box plot
   - Violin plot
   - QQ-plot
5. Make it normally distributed by applying CLT

03 Enforcing normality

Statement: Our system is faster on average

- Given correct experiment setup, draw enough samples
- Mean of a sample is normally distributed (CLT)
- Compare means
- ???
- PROFIT!!!

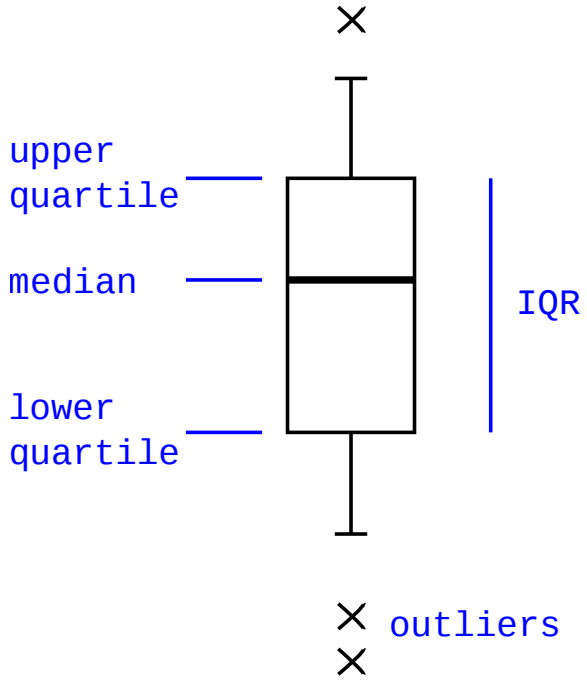
- Not always enough
- Interested in worst case?
- See EVT, GEV distribution (Weibull, Frechet, Gumbel)

03 Box Plots

Present a sample by its median and quartiles. Use of the whiskers is not uniform. Popular choices are:

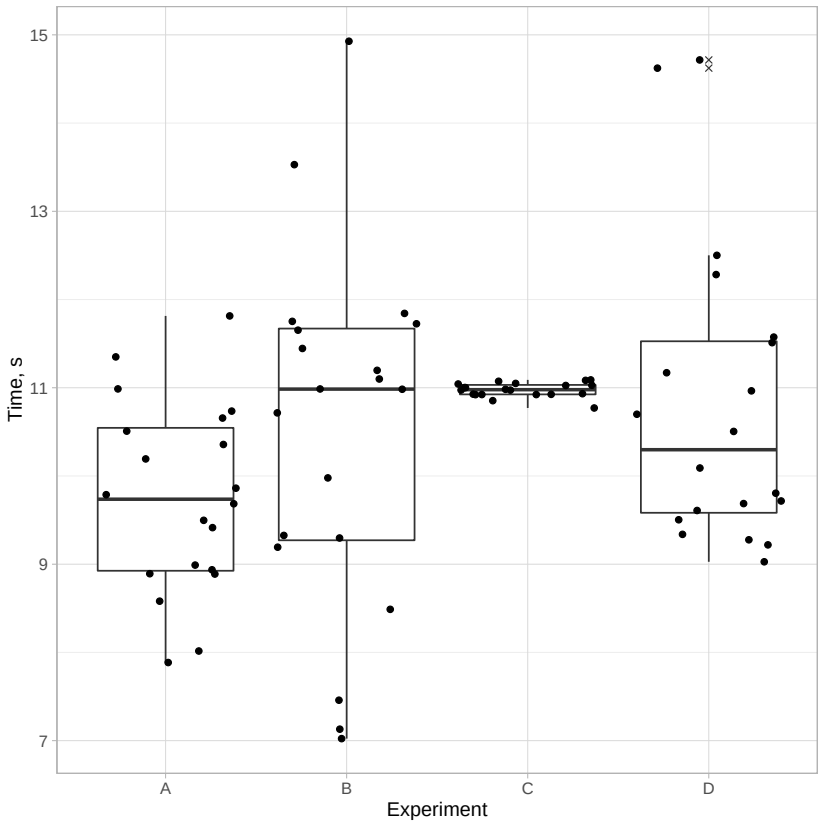
- smallest point within 1.5 IQR of lower quartile to largest in upper quartile,
- min / max,
- $\mu \pm \sigma$

Points outside whiskers are drawn individually.



03 Look at more points

- Combination of scatter plot and box plot
- Outliers are crosses
- Data points are dots
- What doesn't look normal?

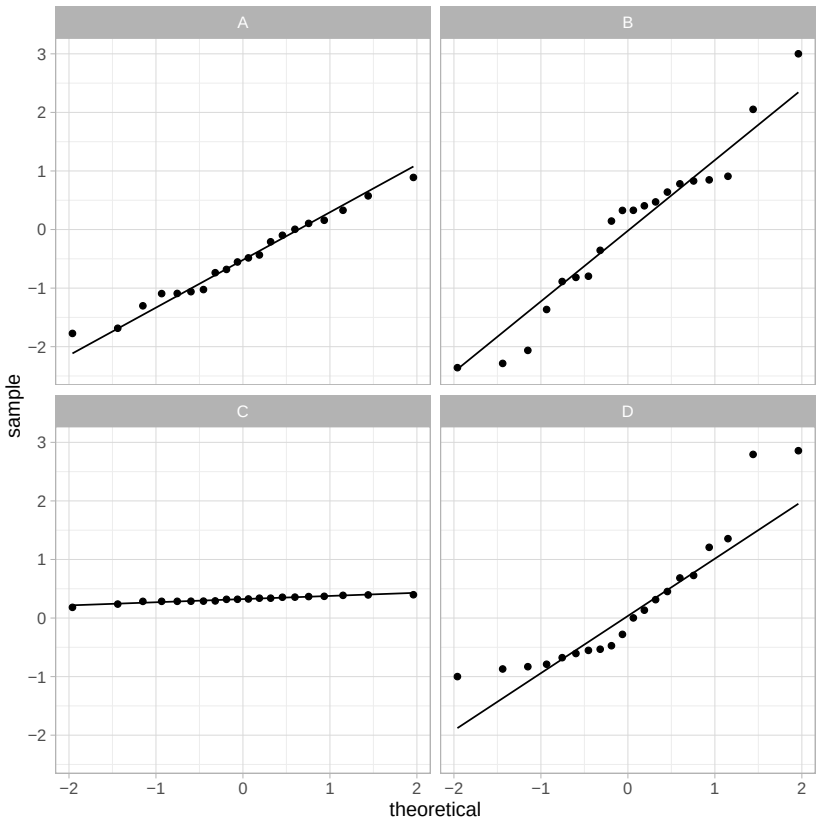


03 QQ-plots for the rescue

- Compare sample against theoretical distribution
- Better to use standard score

$$z = \frac{x - \mu}{\sigma}$$

- A and C are clearly normal

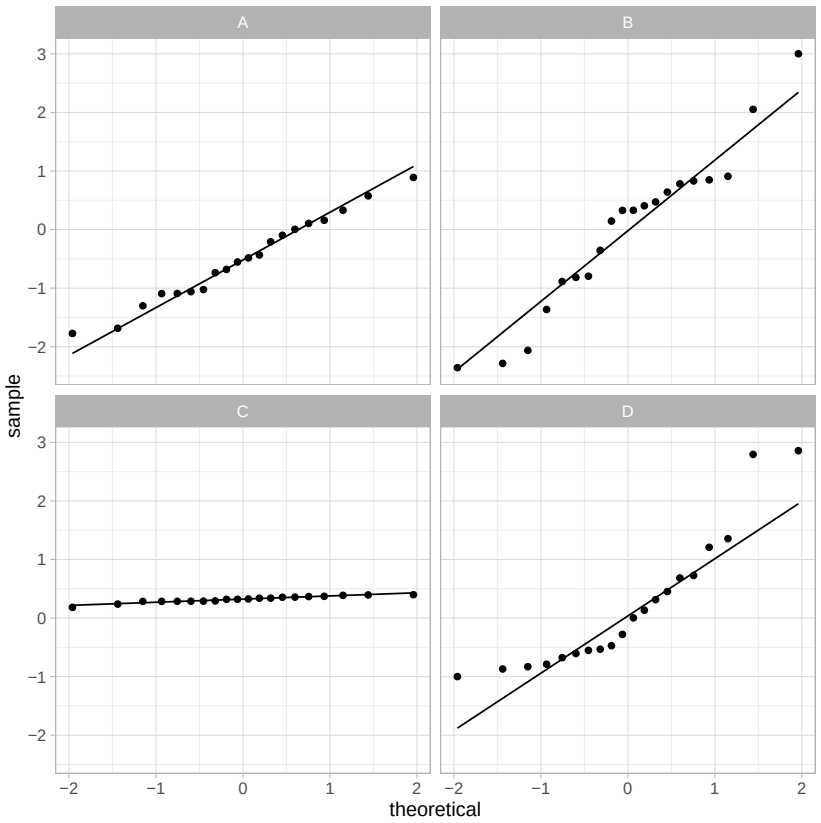


03 QQ-plots for the rescue

- Compare sample against theoretical distribution
- Better to use standard score

$$z = \frac{x - \mu}{\sigma}$$

- A and C are clearly normal
- B and D are problematic



03 Normality tests

- Shapiro-Wilk test
- Can be used to decide that distribution is *not* normally distributed
- Usually works better for small samples
- A, B, C may be normal
- D is not

Experiment	A	B	C	D
p-value	0.95	0.4	0.26	0.005

03 Parametric statistics

Why to bother about distribution at all?

- Parametric and non-parametric tools
- Parametric tools are generally more powerful
  Examples: Three sigma rule, CI for mean [3], ANOVA, Pearson correlation
- Non-parametric tools can be applied in more situations
  Examples: Chebyshev's inequality, CI for median, non-parametric ANOVA, Spearman correlation

03 Mean vs. median

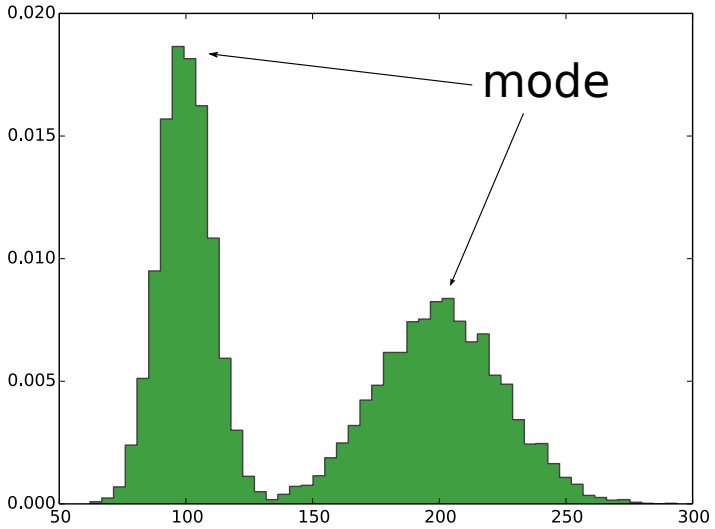
- Both non-parametric (do not assume distribution)
- Median is robust, mean is not
- CI for mean is parametric, CI for median is not
- Compromise: *truncated* mean

the first two results were excluded as warm-up runs [4]

03 Multimodal Distributions

Some random samples have multiple **modes** ("peaks"). Each one is an interesting data point.

- Median and mean are misleading!
- Do histgram or violin plot.



03 Example: Reporting results of HPL [2]

- Piz Daint supercomputer
- High-Performance Linpack
- Max: 77.38 Tflops/s
- Theoretical max:
  94.5 Tflops/s

03 Example: Reporting results of HPL [2]

- Piz Daint supercomputer
- High-Performance Linpack
- Max: 77.38 Tflops/s
- Theoretical max: 94.5 Tflops/s

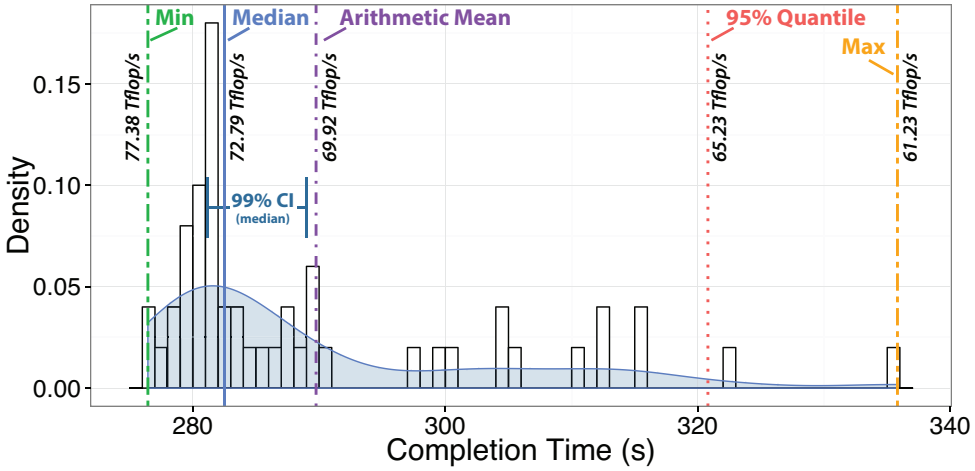


Figure 1: Distribution of completion times for 50 HPL runs.

03 Histograms may be misleading [6]

- Histograms and density plots show the same
- Histograms may be very different depending on the shift
- Hard to get exact data from density plots

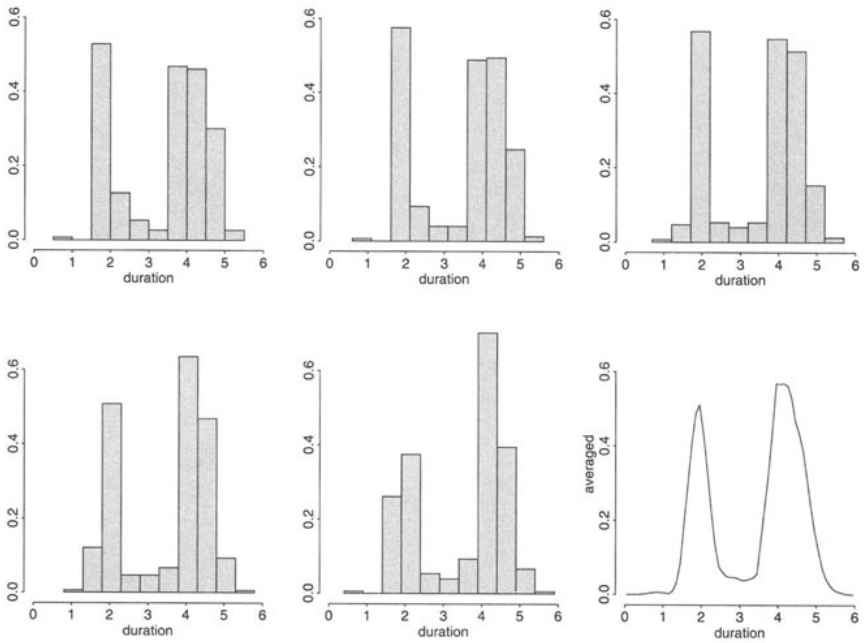


Figure 5.8: Five shifted histograms with bin width 0.5 and the frequency polygon of their average, for the Old Faithful geyser `duration` data. The code used is in the scripts for this chapter.

03 Descriptive Statistics Recap

- **Aggregated values** might be misleading.
- Indicate **dispersion** in your plots!
- Multimodal results require extra care.
- May require additional experiments

03 Significance of Results

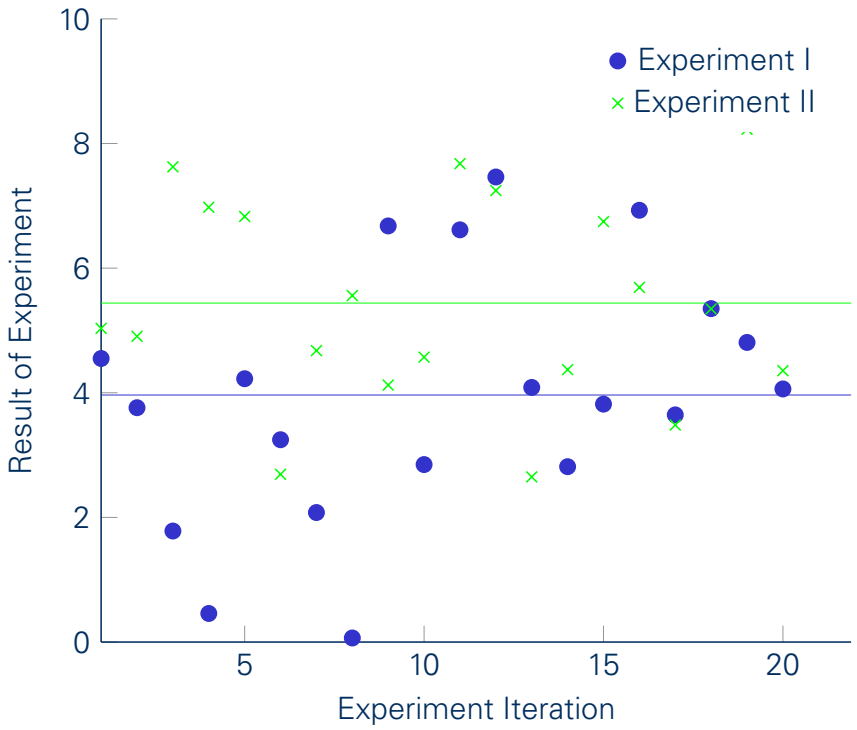
A result is significant, if it is unlikely to have happened by chance.
In practice, a **rule of thumb** for a large number of data points is

Any change that is several times larger than σ is **probably** significant.

Effects in the order of σ are suspicious, below σ are very likely **meaningless**.

03 Example: Small Data Sets

Suppose you have a system property that is expensive to test, you got the following data:



$$\mu_1 = 3.9640$$
$$\sigma_1 = 1.9690$$
$$\mu_2 = 5.4395$$
$$\sigma_2 = 1.6110$$

Clearly, our rule of thumb is not applicable.

03 Hypothesis Testing

"Simplifying socket buffer data structures leads to **different** CPU usage for small network packets."

$$H_1 : \mu_{simple} - \mu_{original} \neq 0$$

The right way to "prove" this hypothesis is to **reject** the **null hypothesis** that results are chance results:

$$H_0 : \mu_{simple} - \mu_{original} = 0$$

We usually can do this only in a probabilistic way by giving a probability p for H_0. If p is below the **significance level** α, H_0 can be rejected. Popular significance levels are 0.05 or 0.01.

If we can reject H_0, our result is statistically **significant**, otherwise **inconclusive**.

03 Understanding Significance

Given 100 experiments of a worthless change (H_0 is true) to our system with a significance level of 0.05:

- 95 will be inconclusive,
- 5 will be significant! (**Type I error**).

Given 100 experiments of a real change (H_0 is false) to our system with a significance level of 0.05:

- 95 will be significant,
- 5 will be inconclusive! (**Type II error**).

03 t-Tests

For normally distributed results with the same σ, Student's t-test can be used to reject H_0.

For normally distributed results with different σ, Welch's t-test can be used to reject H_0.

03 t-Test Practice

The math is complex, but RTFM of SciPy gives us:

```
>>> v1
array([ 4.55004876,  3.76050393,  1.78153774,  0.45846504,  4.22565852,
        3.2464959 ,  2.07829922,  0.06514428,  6.67741018,  2.84793141,
...
>>> v2
array([  9.4764887 ,   6.66139287,   8.12849474,   7.60792901,
        10.73051496,   6.5853035 ,   6.93264452,   6.09349537,
,,,
>>> scipy.stats.ttest_ind(v1, v2, equal_var=False)
(array(-2.5280026596739273), 0.015926207185724916)
```

$p = 0.0159$
With significance level of 0.05, we can reject H_0.
With significance level of 0.01, the test is **inconclusive**.

03 Confidence Interval

A way to assess the accuracy of e.g. the mean of a random sample is to use confidence intervals.

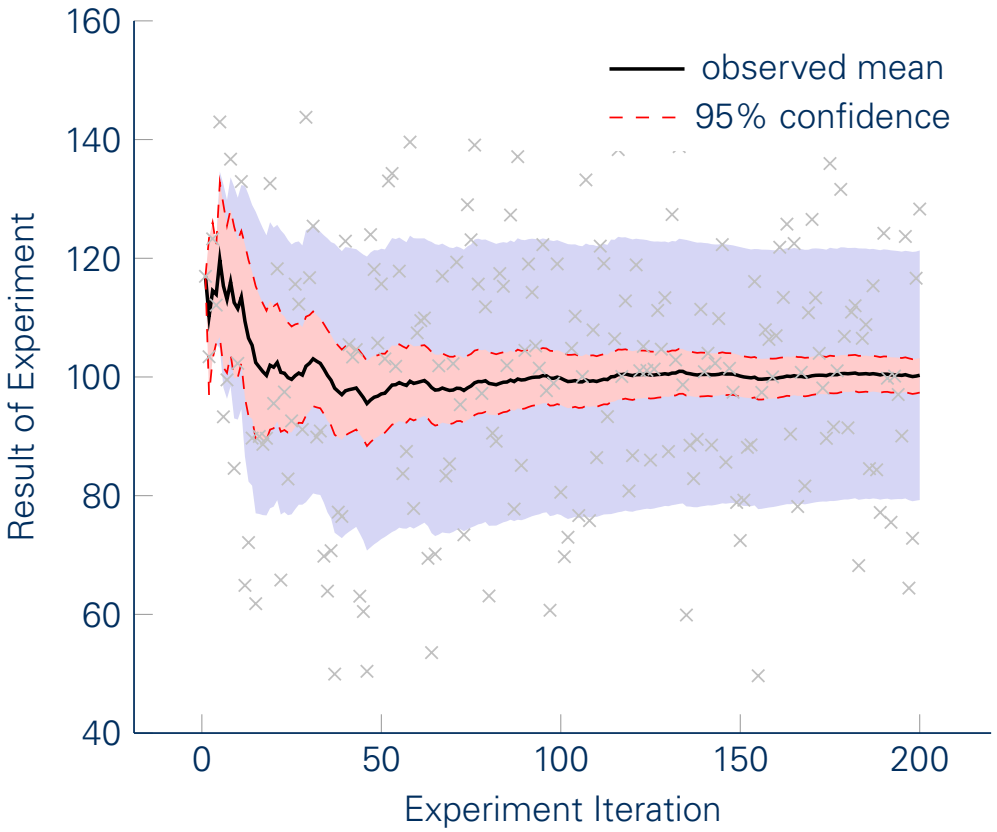
The confidence is given as $(1 - \alpha)$.

What does it mean: If you

- draw a random sample,
- compute the 95% confidence interval for the mean,

the "real" mean will be in this interval with 95% confidence.

03 Confidence Interval



03 Confidence Intervals vs. t-Tests

Use confidence intervals, if there is **nothing to compare to** or the **sample is large**.

Use t-tests to establish the likelyhood whether something **changed** with respect to another measurement, especially for **very small samples**.

t-tests can also test for the likelyhood of a specific mean.

03 Statistics and Computer Science

Literature on statistics focuses on social studies, experimental research in medicine, . . .

Their experiments are:

- expensive,
- time-consuming,
- **manual**.

Make most out of **little data** with advanced statistical methods.

Systems experiments:

- cheap,
- fast,
- can be **automated**.

Gather **lots of data** and can get away with simpler statistical methods.

03 Statistics Recap

- Repeat experiments until you can be certain its result is not by chance.
  - Check multimodality and dispersion!
  - Use statistical methods to check your results:
    - confidence intervals
    - t-tests
- Understand significance

03 Statistics Recap

- Repeat experiments until you can be certain its result is not by chance.
  - Check multimodality and dispersion!
  - Use statistical methods to check your results:
    - confidence intervals
    - t-tests
- Understand significance

If you torture the data enough, nature will always confess.

Ronald Coase

04 Outline

04 Normalization [1]

Suppose we have a benchmark suite comprised of **benchmarks E, F, H, I, K** and results for three **processors R, M, Z**.

It's tempting to normalize results to one processor and compute the arithmetic mean for each processor. It's also **wrong**.

TABLE I. Incorrect Use of the Arithmetic Mean on Normalized Numbers

Benchmark	Processor		
	R	M	Z
E	417 (1.00)	244 (0.59)	134 (0.32)
F	83 (1.00)	70 (0.84)	70 (0.85)
H	66 (1.00)	153 (2.32)	135 (2.05)
I	39,449 (1.00)	33,527 (0.85)	66,000 (1.67)
K	772 (1.00)	368 (0.48)	369 (0.45)
Arithmetic mean	(1.00)	(1.01)	(1.07)

The numbers in parentheses are normalized to Machine R.

Source: How Not to Lie with Statistics: The Correct Way to Summarize Benchmark Results

04 Normalization [1]

Suppose we have a benchmark suite comprised of **benchmarks E, F, H, I, K** and results for three **processors R, M, Z**.

It's tempting to normalize results to one processor and compute the arithmetic mean for each processor. It's also **wrong**.

TABLE II. Same Raw Data, but Different Results

Benchmark	Processor		
	R	M	Z
E	417 (1.71)	244 (1.00)	134 (0.55)
F	83 (1.19)	70 (1.00)	70 (1.00)
H	66 (0.43)	153 (1.00)	135 (0.88)
I	39,449 (1.18)	33,527 (1.00)	66,000 (1.97)
K	772 (2.10)	368 (1.00)	369 (1.00)
Arithmetic mean	(1.32)	(1.00)	(1.08)

The numbers in parentheses are normalized to Machine M.

Source: How Not to Lie with Statistics: The Correct Way to Summarize Benchmark Results

04 Arithmetic vs. Geometric Mean

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} v_i$$

$$\bar{x}_{\mathrm{geom}} = \left(\prod_{i=1}^{n} v_i\right)^{\frac{1}{n}} = \frac{1}{n}\sum_{i=1}^{n} \ln v_i$$

For non-negative v_i: $\bar{x}_{\mathrm{geom}} \leq \bar{x}_{\mathrm{arith}}$.

Geometric mean $\bar{x}_{\mathrm{geom}}$ usually used for interest rates and growth factors.

04 Normalization (cont'd)

The geometric mean gives the same relative "ranking" of benchmarks regardless of normalization.

The geometric mean is the **only** mean with this property.

TABLE V. Correct Use of the Geometric Mean

Benchmark	Processor		
	X	Y	Z
1	20 (1.00)	10 (0.50)	40 (2.00)
2	40 (1.00)	80 (2.00)	20 (0.50)
Geometric mean	(1.00)	(1.00)	(1.00)

The numbers in parentheses are normalized to Machine X.

Source: How Not to Lie with Statistics: The Correct Way to Summarize Benchmark Results

04 Normalization (cont'd)

The geometric mean gives the same relative "ranking" of benchmarks regardless of normalization.

The geometric mean is the **only** mean with this property.

TABLE VI. The Geometric Mean Is Independent of Normalization

Benchmark	Processor		
	X	Y	Z
1	20 (2.00)	10 (1.00)	40 (4.00)
2	40 (0.50)	80 (1.00)	20 (0.25)
Geometric mean	(1.00)	(1.00)	(1.00)

The numbers in parentheses are normalized to Machine Y.

Source: How Not to Lie with Statistics: The Correct Way to Summarize Benchmark Results

04 Normalization Rules

- Don’t average normalized numbers with the arithmetic mean!
- Use geometric mean instead.

04 Benchmark Suites [5]

Benchmark suites are sets of several individual benchmarks and designed to test certain aspects of a system. Popular examples are SPEC benchmarks:

- CPU (SPEC CPU)
- High Performance Computing (SPEC MPI)
- Java (SPECjbb)
- Network filesystem (SPECsfs)
- Power (SPECpower)

Using only subsets of benchmark suites is highly frowned upon.

04 SPEC CPU2006

SPEC CPU2006 is comprised of (more or less) real-life applications:

12 Integer Benchmarks:

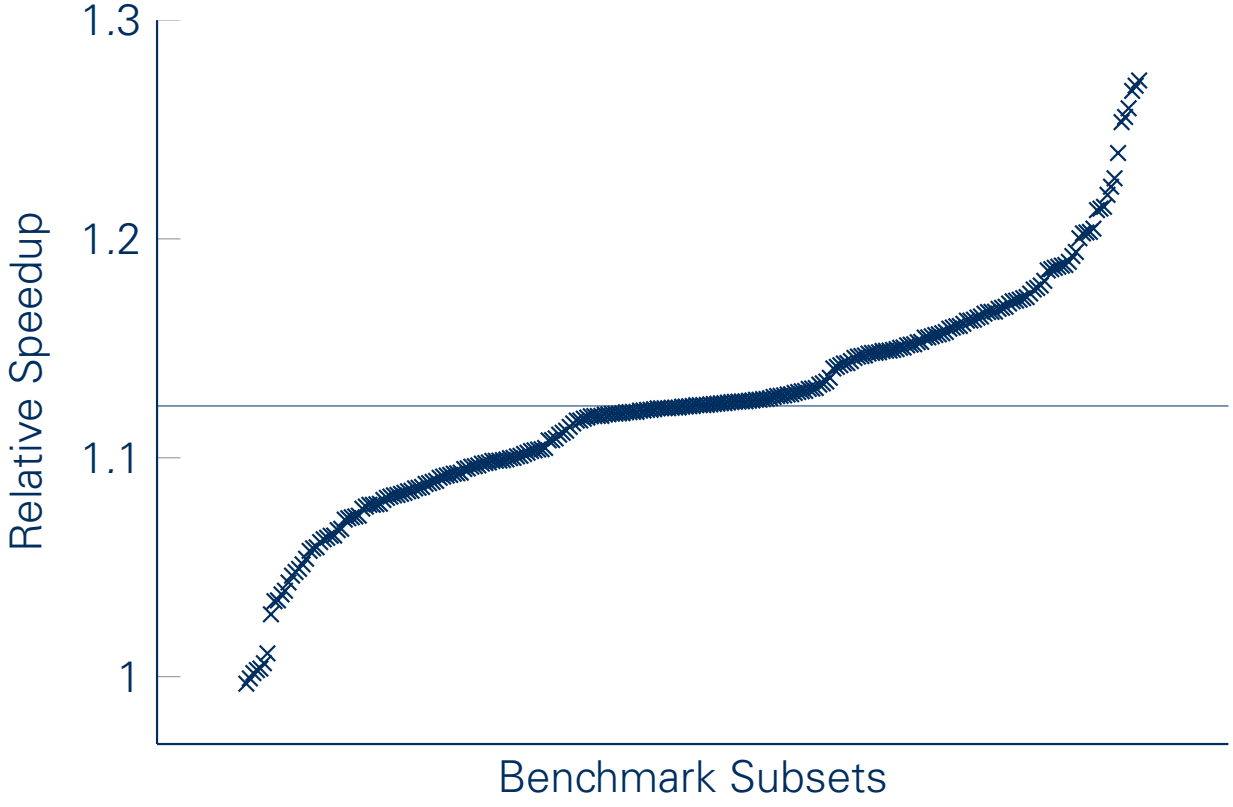
- perlbench
- bzip2
- gcc
- h264ref
- mass transport vehicle scheduling
- ...

18 Floating Point Benchmarks:

- Partial Differential Equations
- Fluid Dynamics
- Simplex Linear Program Solver
- ...

Choosing subsets will unfairly highlight particular system properties, but not give a good estimate of overall performance.

04 Suite Subsets vs. Speedup



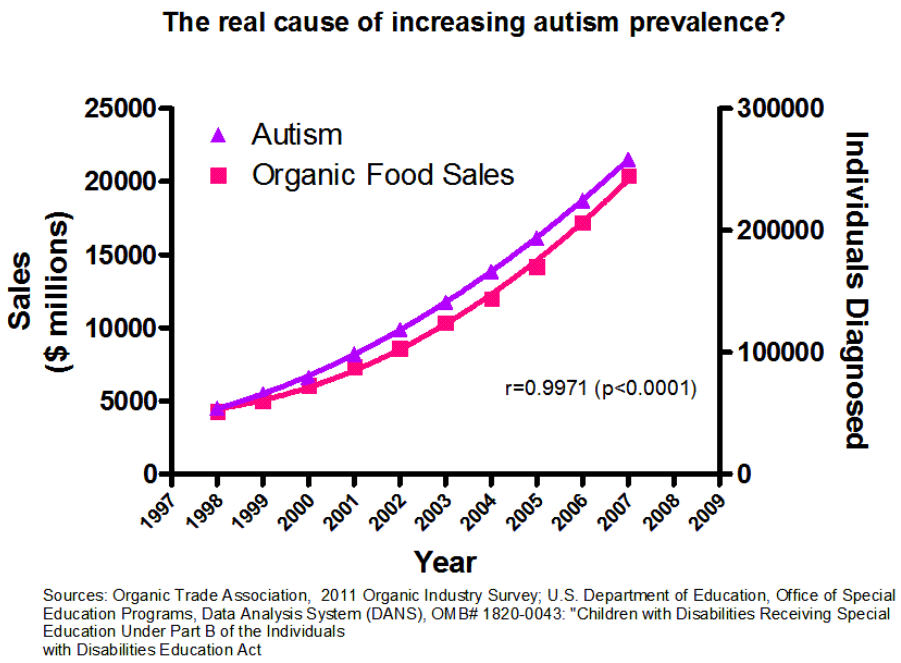
04 Benchmark Suites

- If a subset is used, then a reason must be given.
- “Those were the only ones I got working.” is a bad reason.

04 Correlation is not Causation

If A correlates to B, this can mean:

- A causes B.

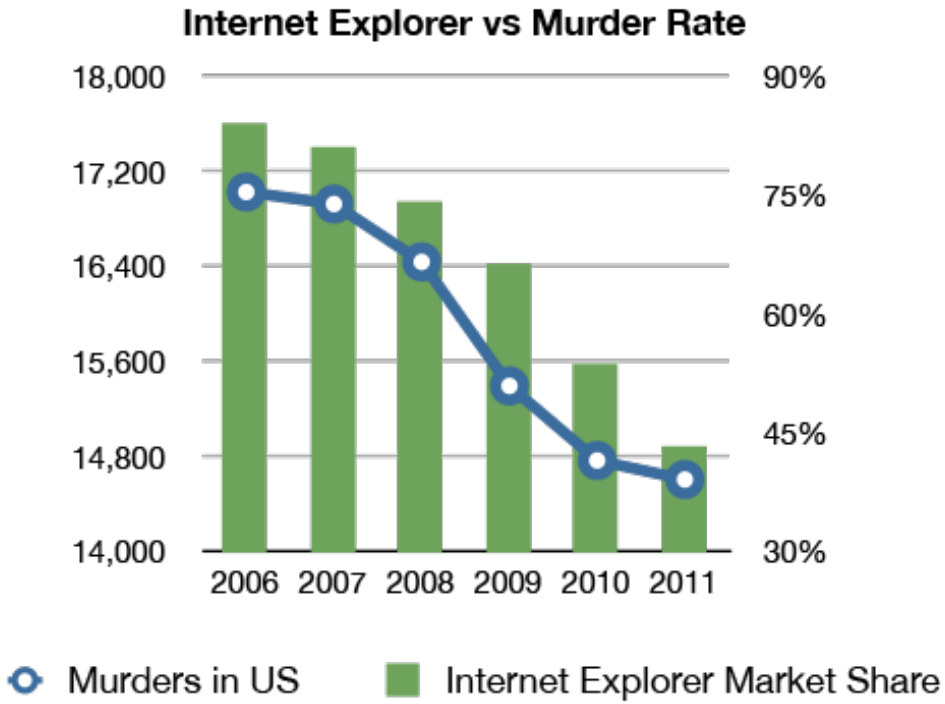


Source: BoingBoing

04 Correlation is not Causation

If A correlates to B, this can mean:

- A causes B.



Source: @altonncf Twitter

04 Correlation is not Causation

If A correlates to B, this can mean:

- A causes B.
- B causes A.
- A causes B **and** B causes A.
- C causes A and B.
- Nothing. (Chance result)

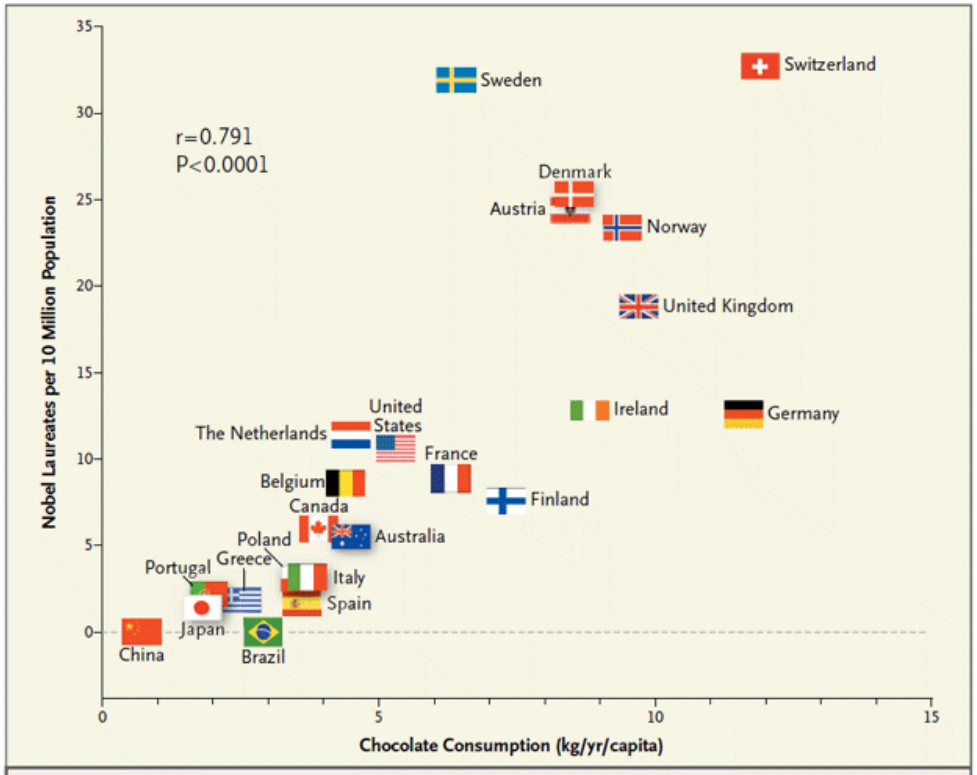
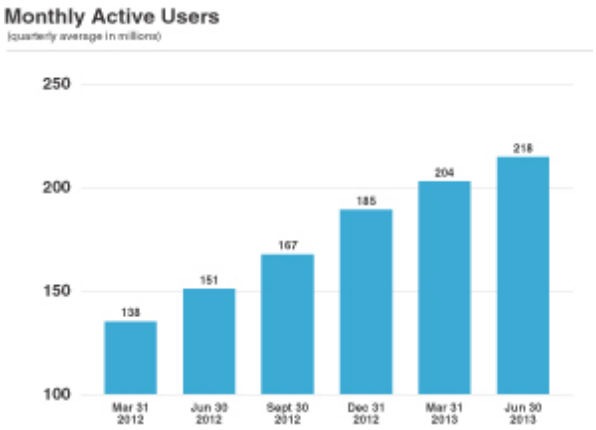


Figure 1. Correlation between Countries' Annual Per Capita Chocolate Consumption and the Number of Nobel Laureates per 10 Million Population.

NEW ENGLAND JOURNAL OF MEDICINE

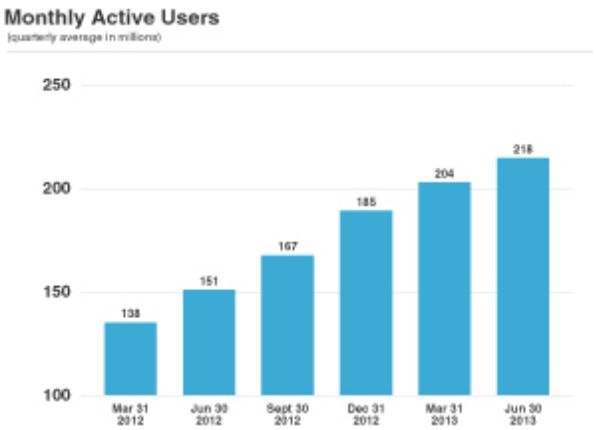
05 Outline

05 Biases

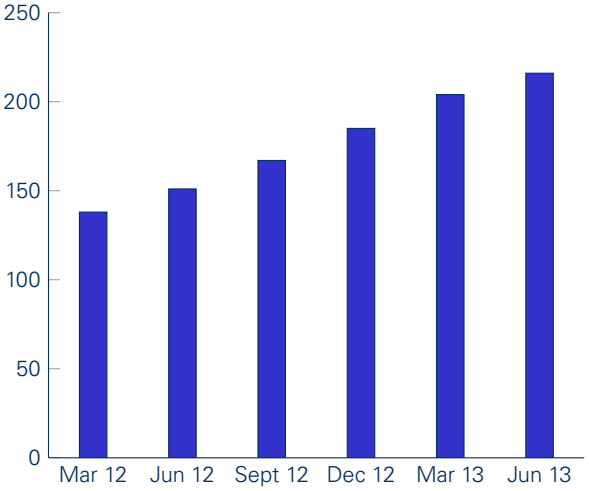


Source: Twitter IPO

05 Biases

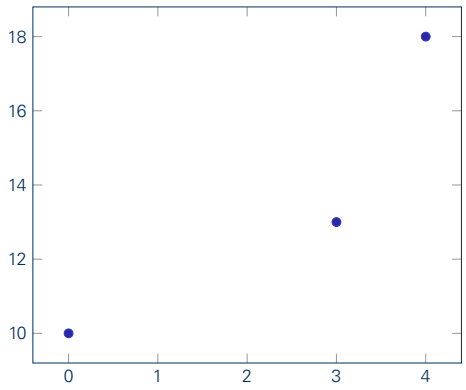


Source: Twitter IPO

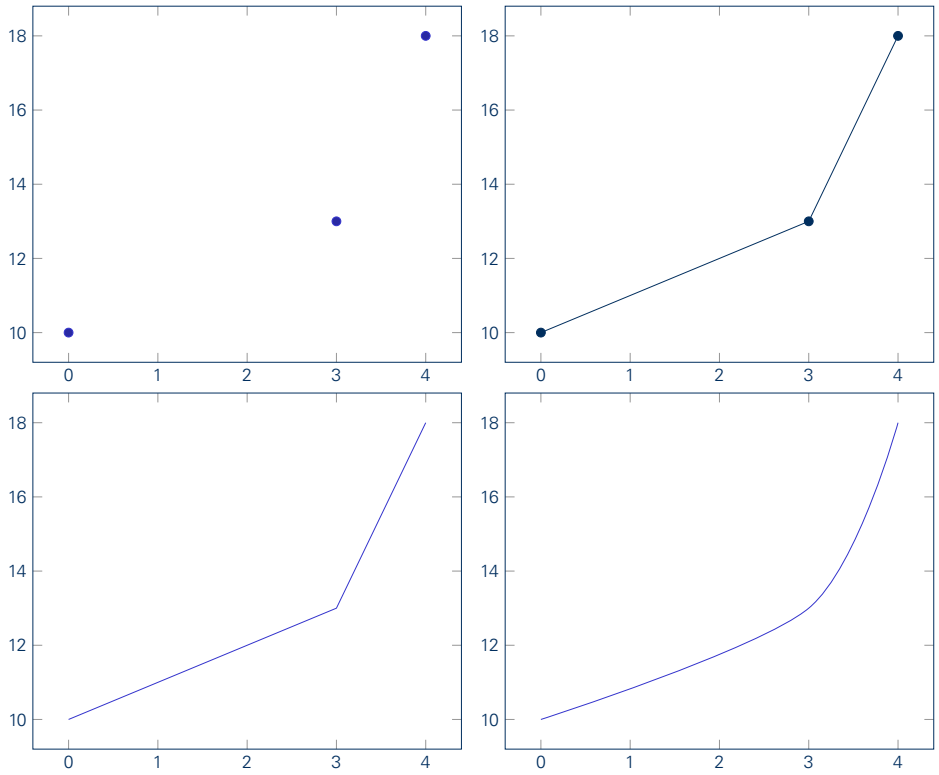


Axes starting at a non-zero value exaggerate changes!

05 Connecting the Dots

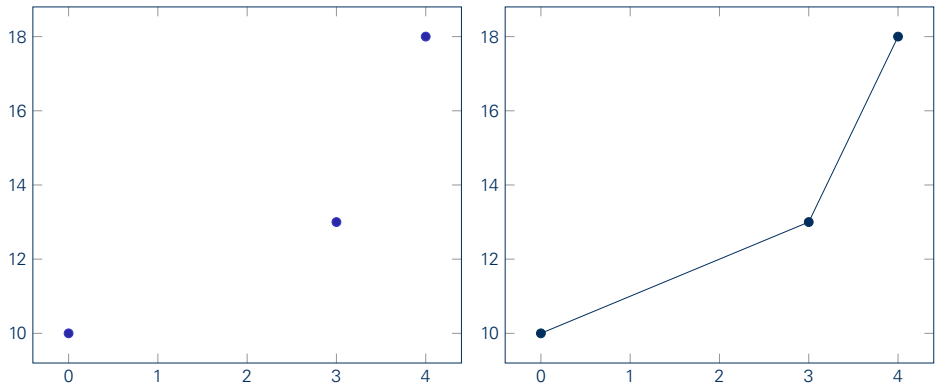


05 Connecting the Dots



Which is the best visualization of the data?

05 Connecting the Dots

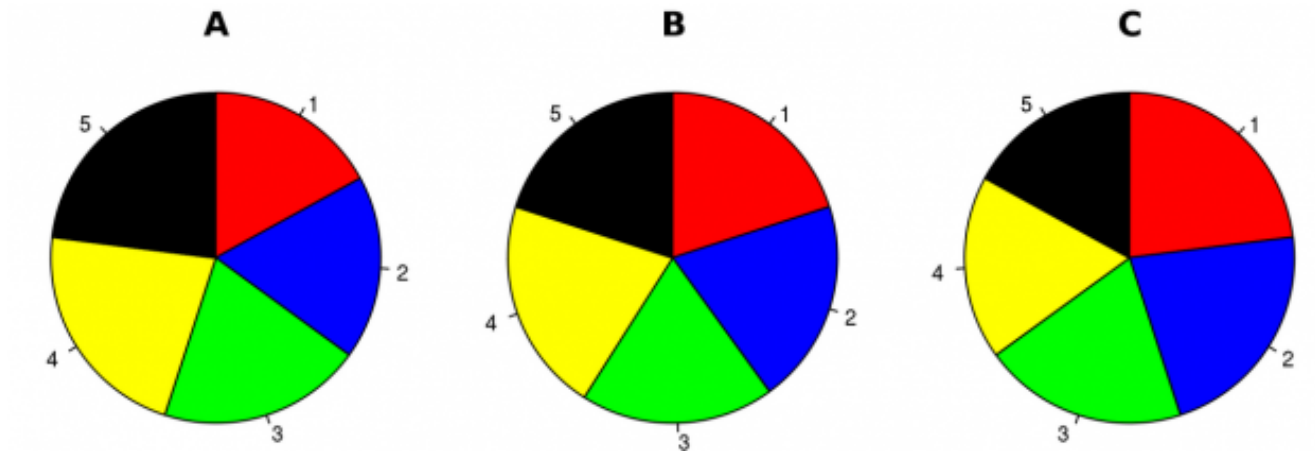


Which is the best visualization of the data?

Lines indicate **trends**.

Not useful for independent benchmarks.

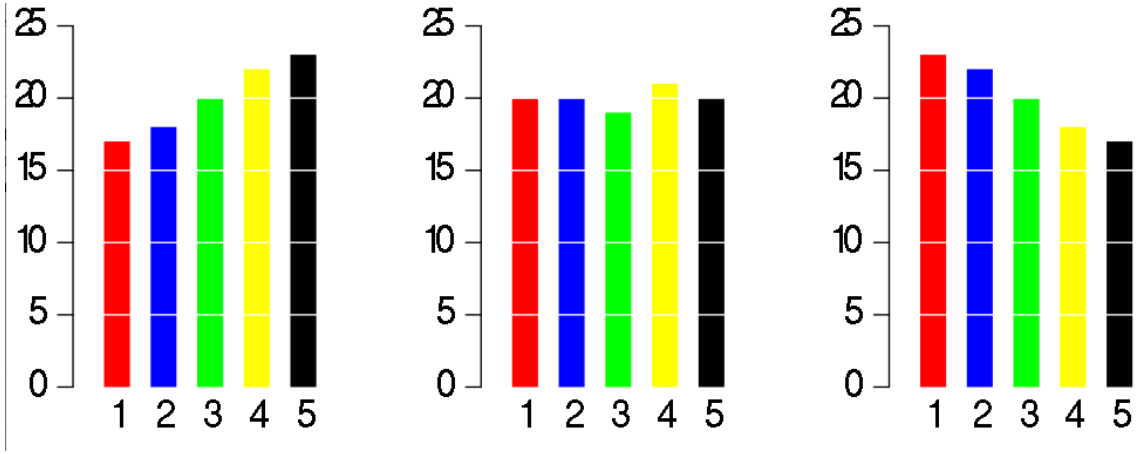
05 Pie Charts



Source: BusinessInsider / Wikipedia

What is the second best result in each chart?

05 Pie Charts



Source: BusinessInsider / Wikipedia

What is the second best result in each chart?

Hard to compare sizes without values as labels.
3D pie charts are even worse!

06 Outline

06 Take Away Points

Automate experiments, because you will redo them

Hypotheses must be **substantiated** with data.

Experiments must be **repeatable by others** and ideally **automated**.

Numbers don't lie, but analyses do. Use proper statistical methods!

Present results without cheating!

06 Literature I

Philip J Fleming and John J Wallace.
How not to lie with statistics: the correct way to summarize benchmark results.
Communications of the ACM, 29(3):218–221, 1986.

Torsten Hoefler and Roberto Belli.
Scientific benchmarking of parallel computing systems: Twelve ways to tell the masses when reporting performance results.
In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 73. ACM, 2015.

Jean-Yves Le Boudec.
Performance evaluation of computer and communication systems, volume 12.
EPFL Press Lausanne, 2010.

06 Literature II

Markus Partheymüller.
Exploring inter-core message-passing for fiasco on the scc.
Master's thesis, TU Dresden, 2011.

David A Patterson.
Computer architecture: a quantitative approach.
Elsevier, 2011.

William N Venables and Brian D Ripley.
Modern applied statistics with S-PLUS.
Springer Science & Business Media, 2013.