War of the Benchmark Means: Time for a Truce

John R. Mashey
Techviser
mash at heymash dot com

Abstract— For decades, computer benchmarkers have fought a War of Means. Although many have raised concerns with the geometric mean (GM), it continues to be used by SPEC and others. This war is an unnecessary misunderstanding due to inadequately articulated implicit assumptions, plus confusion among populations, their parameters, sampling methods, and sample statistics. In fact, all the Means have their uses, sometimes in combination. Metrics may be *algebraically* correct, but *statistically* irrelevant or misleading if applied to population distributions for which they are inappropriate. Normal (Gaussian) distributions are so useful that they are often assumed without question, but many important distributions are not normal. They require different analyses, most commonly by finding a mathematical transformations that yields a normal distribution, computing the metrics, and then back-transforming to the original scale. Consider the distribution of relative performance ratios of programs on two computers. The normal distribution is a good fit *only* when variance and skew are small, but otherwise generates logical impossibilities and misleading statistical measures. A much better choice is the *lognormal* (or log-normal) distribution, not just on theoretical grounds, but through the (necessary) validation with real data. Normal and lognormal distributions are similar for low variance and skew, but the lognormal handles skewed distributions reasonably, unlike the normal. Lognormal distributions occur frequently elsewhere are well-understood, and have standard methods of analysis.

Everyone agrees that "Performance is not a single number," … and then argues about which number is better. It is more important to understanding populations, appropriate methods, and proper ways to convey uncertainty. When population parameters are estimated via samples, *statistically* correct methods must be used to produce the appropriate means, measures of dispersion, Skew, confidence levels, and perhaps goodness-of-fit estimators. If the wrong Mean is chosen, it is difficult to achieve much. The GM predicts the mean relative performance of *programs*, not of workloads. The usual GM formula is rather unintuitive, and is often claimed to have no physical meaning. However, *it is the back-transformed average of a lognormal distribution*, as can be seen by the mathematical identity below. Its use is not onlystatistically appropriate in some cases, but enables straightforward computation of other useful statistics.

$$GM = \overline{x}_G = \left(\prod_{i=1}^{n} x_i \right)^{\left(\frac{1}{n}\right)} = \exp\left(\frac{1}{n} \sum_{i=1}^{n} \ln\left(x_i\right) \right)$$

"If a man will begin in certainties, he shall end in doubts, but if he will be content to begin with doubts, he shall end with certainties."
– Francis Bacon, in Savage [1].

Index Terms—Benchmarking, Geometric Mean, Lognormal Distribution

1 GM considered harmful…

People have long argued the merits of various means – arithmetic (AM), harmonic (HM), geometric (GM). In 1986, Fleming and Wallace [2] argued strongly for use of the GM. In 1988, Smith [3] wrote the opposite, as has John [4] just recently, going so far as to conclude:

> "Geometric mean does not represent anything meaningful while aggregating performance metrics over a benchmark suite."

Among the fine textbooks that discuss the issue are Jain's 1991 book [5] and Lilja's 2000 text [6]. Both discuss sampling and populations, whose importance in benchmarking is often forgotten. Successive editions of Hennessy and Patterson [7] have offered increasingly nuanced analyses of performance issues. All these credible sources express concerns (or worse!) about the GM.

2 But people keep using it anyway

2.1 Livermore FORTRAN Kernels (LFK)

McMahon's Livermore Fortran Kernels [8], successfully used for 20+ years, are well-constructed and carefully-explained benchmarks that produce 72 data values plus many summaries. McMahon [9] is still worth rereading for good benchmarking practice.

> "No single rate quotation is sufficient or honest. These measures show a realistic variation in Fortran cpu performance that has stood the test of time…
>
> The performance of the standard "as is" LFK test (no modification) correlates well with the performance of the majority of cpu-bound, Fortran applications, and hence, of diverse workloads…

> The best central measure is the Geometric Mean (GM) of 72 rates because the GM is less biased by outliers than the Harmonic (HM) or Arithemetic (AM) (sic)…
>
> More accurate projection of cpu workload rate may be computed by assigning appropriate weights for each kernel."

The reason above is the one commonly quoted for the use of the GM, and it is true, but has likely diverted understanding of the GM's mathematical foundation.

2.2 Digital Review CPU 2 (DR CPU2) benchmarks

During the 1980s and early 1990s, Digital Review magazine published "CPU 2 benchmark suite" comparisons, a set of performance ratios for 34 Fortran benchmarks. using the MicroVAX II as a base, and yielding "MVUPS." They computed the GM of the ratios as a central measure, but also gave standard deviations, confidence intervals, and did jackknife statistical analyses. The 34 programs were identified by name, and the full set of details published, plus comprehensible graphs [10].

2.3 Computer Vendors in 1980s

In the 1980s, computer and CPU vendors made wider use of the GM in analyzing internal benchmark suits and describing the performance of their computers in their external performance documents. Published results included real application codes or publicly available variants thereof, such as the Berkeley SPICE circuit simulator, useful well-known benchmarks (like Linpack or Livermore Fortran Kernels), Digital Review's CPU 2. They also included benchmarks that were often counterproductive, but demanded by customers anyway. Benchmark results were often expressed in incommensurate metrics, like Dhrystones, Whetstones, MFLOPS, run-times, VUPS, or MVUPS... Benchmark sets varied widely among vendors.

Vendors often normalized results to one of their own products or to the VAX 11/780, and customers liked this consistency as they could more easily do future performance estimation. These documents first tried to characterize the performance of the company's own products, usually yielding a set of relative performance ratios, expressed in {MIPS, VAX-MIPS, IBM-mips, VUPS, or our-own-mips}, to set expectations for existing customers. Then vendors would do their best to compare with other vendors' products, which was expensive to do well. Often vendors ran slightly different versions of code or used different inputs, yielding inconsistent results difficult to compare by customers.

Most vendors used some internal suite of programs they believed (rightly or wrongly) to be "typical" of their customers' programs, and which they used for architectural and software tuning. But these benchmarks were often not distributable, or even recognizable by customers, so vendors needed to run many others as well. Many benchmark numbers were printed, with the hope that each customer might find *something* they understood. Customers liked seeing many real benchmarks, especially if they were recognizable and related to their own work. Engineers would carefully say "No one number predicts performance, your mileage may vary", and offer various means, ranges, histograms. Still, many customers demanded simple estimators for relative CPU speed, either one number, or perhaps two (integer and float).

Following are just a few samples of the many performance reports written then. These were 20-50 pages long, and they consumed much time and effort.

1985 Sun [11]
1987 HP [12], MIPS [13]
1988 AMD [14], Apollo [15]
1989 Digital Equipment [16], HP [17], SGI [18]

3 Kinds of benchmark analyses

This section introduces terms for processes that people have used for years, but for which the author knows no short standard names.

3.1 WCA

A *Workload Characterization Analysis* (WCA) is the process of gathering information, at various levels of detail, about existing workloads. At the simplest level, the result might be a list of programs known to be important, and a sample run-time for each.[1] A really thorough study yields well-characterized distributions of program run-times, frequencies of execution, fraction of total time consumed in the environment studied, for each relevant system, plus historical trends used to estimate likely changes. These are expensive, long-term efforts as continual data collection is normally required. They are normally done by owners of expensive computers or large numbers of them.

3.2 SERPOP

A somewhat orthogonal type of benchmark analysis could be called a *Sample Estimation of Relative Performance Of Programs* (SERPOP). A good SERPOP analysis constructs a multi-element benchmark suite that is a sample of some population of *programs*. It requires certain assumptions to be met regarding the sampling

[1] Some computers spend most of their existence running one program repeatedly, and some people buy hundreds for this purpose at high energy physics labs, at some bioinformatics sites, and for some graphics "render-farms." WCA is relatively easy in this case.

process, and it requires an appropriate model of the population's distribution. Then it can produce a meaningful and mean, indices of dispersion, confidence levels, and goodness of fit measures. It is very desirable to identify the specific codes, describe their nature, to enable users to select only those relevant, and recomputed metrics as needed. Anonymous benchmarks are not particularly convincing, no matter how good they are.

A few of the many SERPOP analyses include LFK, the NAS kernels, DR CPU2, parts of many vendor performance documents, and SPEC CPU benchmarks. Many architectural studies have done SERPOP analyses as input for new computer designs. A SERPOP analysis always admits to uncertainty, and preferably quantifies it.

If an individual benchmark produces one metric, it is not a SERPOP itself, but might be included as an element of a SERPOP. The 1980s performance documents tried to do this by gathering numbers for Dhrystone, Whetstone, Linpack, and others, normalizing them, and then displaying the distribution.

Conversely, a set of benchmarks used in a SERPOP analysis can be treated as a single data point, discarding all information except the total run-time.[2]

3.3 WAW

Workload Analysis with Weights (WAW) is done by people who have done extensive WCA, who thus know the parameters of the workload *population*.[3] In purest form, WAW predicts the performance of *workloads* under different assumptions. If the WCA is good enough, it can give concrete results of the form "System A is 2.2X faster than System B on our workload."

In practice, people do as much WCA as they can afford, and use WAW when they can. They employ SERPOP information when they must, which is often, as WAW copes poorly with missing data and cannot make some kinds of predictions. Many people use a combination, in which they use WCA+WAW for data they can measure cost-effectively, with SERPOP to provide relative performance estimates as needed to estimate missing data. Most WAW calculations are algebraic, with little or no statistical inference.

[2] SPEC89 did this with the NASA7 benchmark. In effect, this converts a set of programs into a workload with specific weightings, and then makes that workload into a benchmark, so the relative weighted run-times really matter, and the total run-time may be dominated by one element.

[3] Note that a good SERPOP requires that the programs be representative, while saying little about the frequency of usage. A good WAW requires serious *workload characterization* to be done, at least to the extent of identifying the programs that consume large fractions of any resource.

4 Instructive 1987 Example – VAX 8700

This example examines an instructive vendor CPU performance characterization of the 1980s, as an example of a reasonable SERPOP.

4.1 VAX 8700/8800 Paper

McInnis, Kusik, and Bhandarkar [19] included an educational performance discussion, showing 99 benchmarks comparing the new VAX 8700 to the VAX 11/780.

```
3.0–3.4      4.0%   FFFD

3.4–3.8      6.1%   FFFFFF
3.8–4.2      7.1%   IFFFFDL
4.2–4.6     19.2%   IIFFFFFFFFFDDDDDDLL
4.6–5.0     11.1%   CIIIFFLLLLL
5.0–5.4     15.2%   FFFFDDDDDDLLLLL
5.4–5.8     23.2%   CIFFFDDDDDDDDDDDLLLLLL
5.8–6.2      6.1%   CCFDLL
6.2–6.6      3.0%   DDL
6.6–7.0      2.0%   DD
7.0–7.4      3.0%   DLL
```

Table 1 - Example from Digital Equipment [19]

The authors provided the overall statistics shown in Table 2, the subsample Medians, plus (not shown) Minimum, Maximum, Standard Deviation, Coefficient of Variation, Median, and Inter-quartile Range. The author derived subsample GMs by assuming each point lies at the midpoint of its range.

Description	**#**	**Metric**	**Value**
Sample HM	**99**	**HM**	**4.84**
Sample GM	**99**	**GM**	**4.92**
Sample AM	**99**	**AM**	**5.01**
Sample Std Dev	**99**	**STDV**	**.88**
Sample Median	**99**	**Mdian**	**5.0**
Subsample			**Estimated**
C: Cobol	4	GM	5.6
I: Fortran Integer	7	GM	4.3
F: Fortran Single	32	GM	4.7
D: Fortran Double	32	GM	5.3
L: LISP	24	GM	5.3

Table 2 – Statistics for Table 1

4.2 What this data really is … and is not

This is a tiny sample from a huge population of programs, chosen by Digital presumably to be somewhat "representative." Programs are categorized into 5 major types by language, but not specifically identified in the paper. Presumably, each benchmark's run-time is long

enough to avoid significant measurement error.
The paper presents no run-times, only run-time ratios of VAX 11/780 time divided by VAX 8700 time.

Neither the actual run-time of any benchmark, nor the ratio of run-times within a single system enter into these calculations in any way, because this *sample is not a workload*. It is unlikely that *any* computer in the world, other than a few benchmarking systems at Digital, *ever* ran this particular set of programs as a *workload*, and no one but the benchmark administrator ever cared about the total run-time.

This set of benchmarks is a *sample* used to estimate parameters of the population. Its subsamples might be used to estimate performance of programs of the same type.[4] Had benchmarks been explicitly identified, more precise estimates might be made by specific customers who had programs whose relative performance was known to be well-correlated with some benchmark.

The paper assumes no run-times, frequencies, or weighting factors, i.e., the usual output of WCA. How could there it? This is a characterization of a *system and its software*, not of the myriads of differing customer workloads. Even using the same ISA, operating system, and compilers, the relative performance of the VAX 8700 on these programs varies from <3.4X to >7X.

Given those constraints, and the lack of any run-times whatsoever, is there any use to such data? Actually, this sort of treatment is *quite* useful to real customers, to computer architects and compiler writers.

4.3 What can be learned in VAX 8700 SERPOP?

The most important performance information above lies in the *distribution* of the performance ratios and this sort of display makes that clear.[5]

The closeness of HM, GM, AM and Median indicates low skew in the overall distribution. On the other hand, the multi-modal nature of the distribution, with two peaks nearly a Standard Deviation apart, is a strong hint that there exist several distinct subpopulations, and that it would be wise to understand their nature.

Suppose a VAX 11/780 customer is considering purchase of a VAX 8700, perhaps in comparison with one or more inexpensive pre-owned VAX 11/780s.

The customer's knowledge of the proposed workload could range from near-zero to near-perfect.

4.3.1 Near-zero knowledge

Suppose the customer knows little or nothing about the workload expected on the VAX 8700.[6] The wide range of ratios warns the customer that they must characterize their own workload well (WCA) if they want to make good predictions about workloads. Although 68.7% of the data points fall in the range 4.2-5.8, that still leaves 31.3% elsewhere so the customer can expect substantial variation - VAX 8700 is *not* just a VAX 11/780 with a faster clock, but has substantial architectural differences. Had the range been 4.5-5.5, with 70% of points in the range 4.8-5.2, most people would likely assume the VAX 8700 was simply 5X faster on most user programs, making both WCA and WAW easy.

4.3.2 Some knowledge of workload

Suppose the benchmark subsample categories given are meaningful to the customer. As shown in Table 2, the subsample GMs differ somewhat. The data looks like samples from two different populations: {F, I} and {C, D, L}, the first with means ~4.4, and the second with means ~ 5.4. In this case, customers know there are relatively many samples for {F, D, L}, and not very many for {I, C}, and their confidence in the statistics of latter groups would be lower.

A user might well use these numbers to do quick performance estimation for workloads, using subsample GMs. Suppose the customer spends 8 hours per day running D-type jobs, 4 hours running I-type, and the other 12 running an unknown mix. A simple estimator for the total run-time for the same 24-hour mix would be: T(8700) = 8/5.3 + 4/4.3 + 12/4.92 = 4.9 hours. This combines SERPOP (to estimate relative performance of programs) with WAW (to apply relevant weights).

4.4 Good knowledge of workload

Suppose the customer's workload is dominated by a few programs, and that the benchmarks had been identified more specifically. If the customer knows from past history that their codes correlate well with specific benchmarks, they ignore the statistics, and just use the ratios of the related benchmarks. They then use their knowledge of their workload weightings to compute the relative performance, i.e. a simple WAW that incorporates a few of the SORPOP data points.

[4] This of course is analogous to Spec's split into integer and floating point groups immediately after SPEC89.

[5] The VAX 8700 was typically described as 6 VUPS, which might have been computed from a different set of measurements, or perhaps marketing.

[6] Suppose the machine will be used to run programs not yet written, as certainly happens for a machine with a five-year life. Suppose the buyer of the machine is a computer center, and provides service to others who generate changing workloads.

4.5 Near-perfect knowledge of workload

Suppose the customer's workload is dominated by one known program. Given the substantial dispersion, the customer should ignore the data, insist on benchmarking their code on the new system, and spend serious time tuning and analyzing code, if they have such access.[7]

This sort of data is a sample, whose statistics help estimate certain population parameters, and whose subsamples and components improve estimates, if coupled with appropriate workload knowledge, the more, the better. This set of benchmarks is *not* anybody's workload. It is a sample of a program population, not a sample of workload populations.

This process and the resulting advice are similar to those of SPEC, and that is not accidental. In fact, many vendors were following similar approaches, and the SPEC process was designed to build on them.

5 SPEC History – "Put Up or Shut Up"

Although working for (sometimes bitter) competitors, the 1980s vendor performance documents' authors usually knew each other. They often referenced each others' work, shared ideas, traded benchmark numbers, and sometimes benchmark code. Competition was fierce, but good ideas tended to propagate among the competitors.

SPEC (www.specbench.org) was founded in late 1988, by Apollo, HP, MIPS[8], Sun, and EE Times magazine, which quoted benchmarks about which we vendors sometimes complained. EE Times editors challenged us to stop whining and build some better benchmarks. We (vendors) were frustrated by the scarcity of industry-wide CPU benchmarks whose behaviors matched those of the real programs that we used internally and programs seen in customer benchmarking. All were weary of wasting time on Dhrystone,[9] Whetstone, and others that induced bizarre compiler tunings, marketing demands for new instructions, and sometimes outright cheating, but did little to improve performance on real programs. Good performance documents were extremely expensive to produce, and yet, customers still complained that inter-vendor comparisons were poor.

The first SPEC meeting[10] rapidly developed a consensus that we should cooperate to develop metrics over which competition would be meaningful to our customers, and that would actually help improve system performance. We agreed on the basic approach, and several likely benchmarks, based on long internal experience. We recruited other vendors to buy into this approach. A year later we released the SPEC89 CPU benchmarks [20]. Results started to be widely reported in early 1990 [21].

SPEC89 used 10 benchmarks with times normalized to a base system, the VAX 11/780.[11] SPEC always emphasized that there was no single number, used a graphical presentation format that reinforced that idea, and strongly urged publications to print all 10 numbers, so that users could ignore inapplicable benchmarks. Although many of us preferred separate integer and floating means, we started with a single mean. Many people told us that if we wanted to eliminate the confusion of VAX-MIPS, VUPS, Dhrystone-mips, etc, that we needed one number. It was emphatically pointed out to us that if we didn't specify one number, that other people would do it for us, differently. However, people quickly started computing the additional GMs for the integer and floating subsamples and by SPEC92, they had been officially separated, and the effort was extended with additional benchmark sets, in line with trying to find programs to which customers could relate directly.

From the beginning, SPEC always hoped that that the creation of consistent results data would lead to interesting analyses. The author [22] used the accumulated data to show the poor correlation between vendor MIPS-ratings and SPEC results. As hoped, the benchmarks were also heavily studied in universities. For example, in 1995, Giladi and Ahituv [23] analyzed SPEC92 and found benchmark redundancies and other interesting results.

SPEC has always been concerned about realistic "social engineering" issues of benchmarking. If SPEC were successful, fierce pressure would develop around its metrics. Mighafori, Jacoby, and Patterson [24] studied the resulting issues in SPEC92, and offered suggestions incorporated in SPEC95. As will be seen later, not only must programs be chosen to be "representative", but unlike normal samples, SPEC benchmarks could not be truly random samples. Worse, the choice of a program for SPEC induces intense study and tuning that changes

[7] They might not, the program might be third-party software not yet ported to the machine for which purchase is considered.

[8] The author was the MIPS founding representative, and was heavily involved in the early work, through SPEC92. He edited most of the MIPS Performance Briefs [7], which used GMs, and whose presentation style had some influence on SPEC89. All comments in this paper are his alone.

[9] Including the original Dhystone author, Reinhold Weicker, who later joined SPEC and contributed strongly over many years.

[10] Held in the "neutral ground" of a bar owned by Stan Baker of EE Times. The Transaction Processing Council, started about a month earlier, yielding two major industry groups pursuing (differing) benchmark issues.

[11] Technically, any system could have been chosen without changing the relative performance ratios. But, in practice, in the workstation and server market of the late 1980s, it was politically impossible to choose anything but the VAX 11/780. Fortunately, DIGITAL still owned many 11/780s, joined SPEC early, and kindly ran many long benchmarks.

the resulting numbers, and those effects must be managed. SPEC knew that benchmarks would age, and become unrepresentative of user programs, so: SPEC89→SPEC92→SPEC95→SPEC2000→SPEC200?

6 Benchmark statistics

6.1 Statistics background

Most of the statistical theory needed can be found in Lilja [6], Chapters 3 and 4, or especially Jain [5], Chapters 12 and 13, plus section 29.10 (Lognormal distribution), 29.12 (Normal, or Gaussian Distribution. DeCoursey [25] is a good general text, including use of Excel. Good and Hardin [26] offer useful cautions. If texts are not handy, the terms are easily found on the Web.

6.2 Definitions

Many analyses start with *n* x *m* runtimes for n programs on m systems:

$P_{i = 1, n}$ Programs, sometimes with Weights $W_{i = 1,n}$

$S_{j = 1, m}$ Systems

T_{ij} Run-time of P_i on S_j

Performance ratios are given as the ratio of S_j's performance to that of S_k's running program Pi with some specific input. Mathematically, any system could be chosen as the base, and k can be assumed to be 1 when omitted. Larger ratios imply higher speed.

$$R_{ijk} = T_{ik}/T_{ij}$$

Of course, some benchmarks report results in units other than time, such as Dhrystones, Whetstones, i.e., X_{ik} and X_{ij}. If these are inversely proportional to run-times for identical achieved work, they can be normalized without ever knowing the run-times.

$$R_{ijk} = X_{ij}/X_{ik}$$

In general, once the R_{ijk} are computed, SERPOP analyses ignore the T_{ij}, just the opposite of WAW analyses.

6.3 Assumptions for SERPOP

The usefulness of a SERPOP analysis depends on the goodness of several assumptions. **A1** and **A2** deal with initial selection of sample and its size. **A3** and **A4** needed to produce trustable base data. **A5** is unnecessary for correct statistics, but is helpful to the consumers of the statistics, so should not be ignored.

A1: Programs and inputs should be "Representative"

This is necessarily vague. Computer architects and systems software people often study detailed micro-architectural statistics of internal codes and customer codes that they see. Although such programs vary greatly, synthetic benchmarks often show characteristics unlike any real programs, and are usually avoided.[12]

Experienced people do accumulate knowledge of the kinds of programs seen in real life in their domains. This is a sample of *programs*, not of workloads containing the programs. Hence, people normally use unweighted techniques, although if an important class of programs is seriously underweighted in a sample, people might use weightings. On the other hand, if that class of program actually behaves differently than the rest of the sample, the sample might be split into separate subsamples, and results computed for each. For example, integer programs were likely under-represented in SPEC89, and the right response was to split integer and floating point programs, because they often behave differently. Likewise, in the VAX 8700 example, the Integer {I} programs are under-represented.

Likewise, the chosen input should be "representative,"[13] as the results reflect program and input.

A2: Small sample sizes should be avoided

One can learn something from even a handful of programs, if they are well-chosen. It is wonderful, but often expensive, to obtain a complete set of run-times for 30 different programs on multiple systems.[14] Some vendors have collected hundreds of programs for their internal workloads used for architectural and compiler tuning, but many of these programs are difficult to convert into distributable benchmarks.[15]

A3: Measurement Repeatability

Assume that multiple executions of the same program with same input on same system yield a distribution of run-times with Standard Deviation << Mean, so that a Mean (or Median) is a good estimate of population mean.

[12] For example, 1980s CPU and compiler designers disliked Dhrystone because it just didn't act like real programs. Among other things, the number of instructions/function call was very low, and it dealt only with 30-character strings. Synthetic benchmarks can be useful, but great care is needed to avoid over-application of their results.

[13] Berkeley Spice was quickly chosen at the first SPEC meeting, as we all knew it for a floating-point-intense code of which commercial versions were widely used, and we all used it as a benchmark. Input decks that vendors were willing to share proved to run too quickly, and another was found with adequate run-time. To our chagrin, it later was discovered that the specific code mostly exercised memory allocation and did little floating point computing! We *knew* it was floating point, so we didn't check.

[14] For samples of 30 or more, the normal distribution is usually a reasonable approximation to the *t*-distribution, which simplifies analysis.

[15] Some are proprietary. Others are difficult to use as industry-standard benchmarks for a plethora of causes that are often painful surprises.

People normally do increase the run-time of short benchmarks to avoid clock measurement problems on the fastest machine for which it is expected to be used.[16] They also tend to reduce the time of long-running codes for convenience. Such behavior implies that run-times are only considered relevant in producing ratios among systems. The relative sizes of the T_{ij} within each S_j are considered irrelevant, exactly opposite to WCA and WAW, which assume that sample run-times really matter, as they do for those analyses.

A4: Performance ratios must vary little across inputs

Suppose T_{ij} are recalculated across a wide variety of inputs. For some programs, T_{ij} is approximately constant across all inputs, while for others, it varies wildly. Variability of T_{ij} is perfectly acceptable, but the reported R_{ijk} should be near the mean of a distribution with small standard deviation.

This is true for many kinds of programs, but not all. For example, programs that depend on convergence properties of floating point computations may behave wildly different on different machines, although this less of an issue in an era of IEEE Floating Point. Jain[5], Chapter 11 discusses "Ratio Games," and those warnings are well-taken here. Finally, data-size-dependent codes sometimes disobey this assumption in the presence of different cache sizes and memory designs.[17]

A5: Programs should be "recognizable"

In practice, credibility is improved if individual benchmarks are identified and recognizable. Even better, it may help some users select specific benchmarks for their own WAW analyses.

6.4 Population distributions and their metrics

Statistics can be computed from most distributions, but they may not mean much if the methods mismatch the-distribution and there are many ways to go wrong. Normal distributions are common, but cannot be assumed. It useful to compute the first four moments Arithmetic Mean, Standard Deviation, Skew (Skewness), and Kurtosis, plus Confidence interval.[18]
A normal distribution has a Skew of zero. Skew is negative if the distribution has a long tail to the left (smaller) or positive if there is a long tail to the right. Large Skew indicates the presence of one or more outliers that need to be examined carefully, or a mixture of samples whose means differ substantially, or in general, that the distribution is not normal, and hence one must be very careful in the interpretation of the Mean.

A normal distribution has a Kurtosis of zero. Positive Kurtosis indicates a peakier distribution more concentrated around its mean, and a negative Kurtosis one spread further than a normal distribution. A strongly positive Kurtosis probably means that the sample contains many elements whose values are correlated.

One would also want to compute the X% (commonly 95%) confidence interval, i.e., the interval that has X% chance of including the real population mean, as the sample mean, of course, is only an estimator.

6.5 Normal and lognormal distributions

If a distribution is not normal, it is standard practice to hunt a transformation of the data that might yields a normal, compute the statistics on the transformed data, and then invert the transformation to return to the domain of the original data. In any case, people use Goodness-of-Fit metrics, such as the Coefficient of Determination, to find transformations that work acceptably.

Of particular interest here is the lognormal distribution, i.e., one for which $\ln X_i$ (or any log, such as $\log_{10} X_i$) is normal. Normal distributions arise by aggregations of many *additive* effects, while lognormals arise from combinations of *multiplicative* effects, like clock rate differences, compiler optimizations, or memory system design differences. Good discussions of log-normal (or log-normal) distributions can be found in Limpert and Stahel [27], from which the following figure are taken, and in more detail in Limpert, Stahel, and Abbt [28].

Figure 1(a) shows the appearance of a distribution with mean 100 and standard deviation 2. It has the characteristic appearance of a lognormal distribution, and of course, the mean (100) does not appear a good measure of central tendency, given the large Skew.

[16] Benchmarks that run for a fixed time and produce a metric of (units of work done) are very convenient in this regard, as they work well for many years. Unfortunately, many real programs do not work this way.
[17] This fact has often been used in benchmarketing, creating meaneuvers to use benchmarks whose particular size favors one side or the other.
[18] In Excel, the relevant functions are AVERAGE, STDEV, SKEW, and KURTOSIS. CONFIDENCE is used for confidence interval calculations.

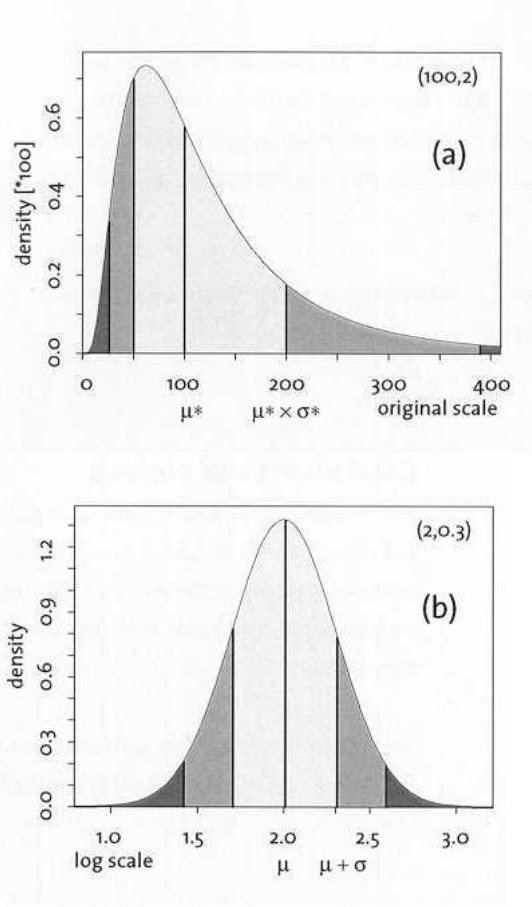


Figure 1 –Limpert and Stahel[27]

Using a $\log_{10}$ transformation, Figure 1(b) now looks like a typical normal distribution, its mean is computed as usual, and then back-transformed:

$$\overline{x}_A = \frac{1}{n}\sum_{i=1}^{n} \log_{10} x_i$$

$$Mean = \exp(\overline{x}_A) = GM$$

In a normal distribution, 68% of the data would lie in the interval Mean +/- Standard Deviation in Fig b.

A lognormal distribution with small standard deviation is shaped like a normal distribution, so some data sets get good fits with either. As the standard deviation increases, the tail will stretch increasingly to the right, i.e., positive Skew, and the fit with normal worsens.

6.6 Distributions of run-times T_{ij}

It is difficult to find any distribution that generally fits distributions of run-times. Suppose P_i is run on S_j with every input ever given to P_i on any system.[19] For each P_i, the resulting distribution of T_{ij} might be normal, with tiny variance. It might be normal with substantial variance. It might be any of several highly-skewed distributions, including lognormal. It might have multiple discrete peaks, possibly of arbitrary number.

Consider all the programs that might be run on Sj, each with one input. Is there any reason to believe that there is a preferred distribution of run-times? If not, there is no reason to believe that any sample will have any particular distribution either. Benchmarkers often adjust times up or down via input changes. Weights are essentially arbitrary, and so are weighted benchmark times unless the workload is very well-characterized.

With good workload characterization, one may compute reasonable statistics for programs (as used in that workload), but it is very difficult to say much about programs' run-times independent of workload. But normalized ratios are a very different story.

6.7 Distribution of performance ratios R_{ij}

R_{ij} (for j!=base) **cannot**, in general, be normally distributed, because it produces impossible results for both simple examples and real cases. Suppose S1 and S2 are systems whose performance is clearly equal, but making either the base causes the other to look 1.25X faster. This simply cannot be true, as meaningful statistics should not be changed just by labeling. The sums of the logarithms add to zero, as one would expect of equal systems.

	S1	S2	
	Ti1	Ti2	
P1	2.00	4.00	
P2	4.00	2.00	
	Ri11	Ri21	ln(Ri21)
P1	1.00	0.50	-0.69
P2	1.00	2.00	0.69
AM	1.00	1.25	0.00
GM	1.00	1.00	
	Ri12	Ri12	ln(Ri11)
P1	0.50	1.00	-0.69
P2	2.00	1.00	0.69
AM	1.25	1.00	0.00
GM	1.00	1.00	

Table 3 - Another example of AM of ratios

[19] A large, but finite number.

That example might seem contrived, but it is just a simplified example from SPEC2000 results.

The EINUX A4800 (1800Mhz AMD Opteron) and IBM eServer pSeries 690 Turbo (1700Mhz POWER4+) have SPECint2000 scores of 1080 and 1077, respectively, i.e., identical. The Einux system is more than 2X faster than the IBM on 197.parser, while IBM is more than 2X faster on 181.mcf.[20] Most of the ratios lie in the range .8X to 1.2X. Overall, computing AMs of the SPECratios, AM(IBM/Einux) = 1.08 and AM(Einux/IBM) = 108, so each machine is "faster" than the other.

The lognormal distribution has other useful properties., mostly derived from knowing ln(X/Y) = - ln(Y/X). The first 4 statistical moments have useful identity or symmetry properties.

1. AM(ln(Rijk)) = -AM(ln(Rikj))
→ GM(Rijk) = 1/GM(Rikj)

2. STDEV(ln(Rijk)) = STDEV(ln(Rikj))

3. SKEW(ln(Rijk)) = - SKEW(ln(Rikj))

4. KURTOSIS(ln(Rjk)) = Kurtosis(ln(Rkj))

The first moment, has the desired property that the relative performance of two machines is simply inverted upon swap of base. The second moment and fourth moment are invariant, and the third moment changes sign. Using a logarithmic scale, swapping base moves the mean to its new point, then maintains the same shape of the distribution, inverted left for right.

Finally:

5. GM(Rijl) = GM(Rijk) / GM(Rilk)

Having computed the GM of two machines relative to Sk, one can compute the GM of those machines by dividing the corresponding GMs.

However, the higher moments do not work that way, because the GM only describes the Mean, not the shape . For example, Sj might use the same CPU as Sk, but with higher clock rates, and one would expect it to have GM>1, and a small STDEV, and SKEW and KURTOSIS near zero. Suppose GM(Sl) = GM(Sj), but is a different system with different strengths. One would expect it to have larger STDEVs, SKEWs and KURTOSIS, because it is "more different" from Sk than Sj is.

[20] As of this writing. Presumably compiler people are working hard. The SPEC reference #s for the two files are #02136 and #02097. [29]

Real-world performance ratio distributions need more study to assure they are well-modeled by lognormals,[21] or at least know the cases in which they do not. For now, it is assumed, and examples will be analyzed .

7 SERPOP Analysis Examples

7.1 Textbook Example 1

A well-known example is found in Smith [3], and used in Hennessy and Patterson [7], p.35-37, combining Figures 1.15 and 1.16, giving runtimes in seconds:

	Systems			Weightings		
	S1	**S2**	**S3**	**W(1)**	**W(2)**	**W(3)**
P1	1.00	10.00	20.00	0.500	0.909	0.999
P2	1000.00	100.00	20.00	0.500	0.091	0.001
Total	1001.00	110.00	40.00			
AM:W(1)	500.50	55.00	20.00			
AM:W(2)	2.00	18.19	20.00			
AM:W(3)	91.91	18.19	20.00			

Table 4 - Hennessy and Patterson [7]

The GM's known properties are described, with note:

> In general, there is no workload for three or more machines that will match the performance predicted by the geometric means of normalized execution times.

This is a WAW analysis, which has its own set of assumptions about knowledge of the run-times and weights. The example clearly shows that one must know the weightings to know anything at all about the relative performance of the systems on such workloads.

Consider a SERPOP analysis instead, to understand what may be known about the relative performance distributions of these systems. The next table converts the data to ratios, then shows the logs of the ratios, and computes AM, STDEV, and 95% confidence levels. The Mean and confidence levels are transformed back to the original ratio domain. It is not surprising that S1 and S2 have equal GMs, and S3 is considered faster. Of course, the confidence interval is immense, which says that nothing of substance is really known about the relative performance of these systems independent of workload weightings. Any pair of systems with a range of relative performance ratios can be made by workload choices to look faster or slower.

	S1	S2	S3
	Ri11	Ri21	Ri31
P1	1.00	0.10	0.05
P2	1.00	10.00	50.00
Logs	ln(Ri11)	ln(Ri21)	ln(Ri31)
P1	*0.00*	*-2.30*	*-3.00*
P2	*0.00*	*2.30*	*3.91*
AM	*0.00*	*0.00*	*0.92*
STDEV	*0.00*	*3.26*	*4.88*
95%Low	*0.00*	*-9.91*	*-13.94*
95% High	*0.00*	*9.91*	*15.78*
Exp			
Mean(GM)	1.00	1.00	2.50
95%Low	1.00	0.00	0.00
95% High	1.00	20075	7110810

Table 5 - Analysis of Table 4

7.2 Textbook Example 2

Lilja [6] uses a similar example, and writes:

> Unfortunately, although the geometric mean produces a consistent ordering of the systems being compared, it is the wrong ordering.

	Systems		
Program	S1	S2	S3
P1	417	244	134
P2	83	70	70
P3	66	153	135
P4	39449	33527	66000
P5	772	368	369
GM	587	503	499
Rank	3	2	1

Table 6 - Lilja[6], p. 33.

This was a WAW analysis as well. Using S1 as a base for a SERPOP analysis, these results are obtained, including SKEW and KURTOSIS, which require more data points.

	S1	S2	S3
	Ri11	Ri21	Ri31
P1	1.00	1.71	3.11
P2	1.00	1.19	1.19
P3	1.00	0.43	0.49
P4	1.00	1.18	0.60
P5	1.00	2.10	2.09
Logs	ln(Ri11)	ln(Ri21)	ln(Ri31)
P1	*0.00*	*0.54*	*1.14*
P2	*0.00*	*0.17*	*0.17*
P3	*0.00*	*-0.84*	*-0.72*
P4	*0.00*	*0.16*	*-0.51*
P2	*0.00*	*0.74*	*0.74*
AM	*0.00*	*0.77*	*0.81*
STDEV	*0.00*	*0.61*	*0.79*
Skew	*-*	*-0.30*	*0.82*
Kurtosis	*-*	*0.07*	*-0.75*
95%Low	*0.00*	*-1.08*	*-1.60*
95% High	*0.00*	*2.62*	*3.22*
Exp			
Mean(GM)	1.00	2.16	2.26
95%Low	1.00	0.34	0.20
95% High	1.00	13.73	25.09

Table 7 - Analysis of Table 6

First, the confidence intervals overlap substantially, meaning there is little confidence that these systems are substantially different, and that serious workload characterization must be done to make any predictions at all. Second, S1 and S2 are more alike than they are with S3, given the latter's larger STDEV, SKEW, and KURTOSIS.

8 SPEC2000 SERPOP Example

8.1 The data

Following is data for the IBM eServer pSeries 690 Turbo (1700 Mhz), SPEC serial #02136 for CINT2000[22] and #02137 for CFP2000, labeled S2. S1 is a Sun SPARC Ultra 5-10, and its times Ti1 are measured, then rounded, with inputs chosen to yield convenient run-times, and used as the base values. The Ri21 values (SPEC base ratios) are actually computed as Ri21 = 100 * Ti1/Ti2.[23] The last two columns show ln(Ri21) and a z-score computed from ln(Ri21) for Goodness-of-Fit analysis.

[22] The serial # is the last component of the filename in SPEC's archives, and it is unique within a benchmark suite.[29]

[23] I.e., the SPARC Ultra-5 is essentially rated as 100.

	S1 Ti1	S2 Ti2	S2 Ri21*	S2 ln(R..)	S2 z(ln..)
197.parser	1800	398.0	452	*6.11*	*-2.34*
164.gzip	1400	175.0	800	*6.68*	*-0.80*
253.perlbmk	1800	217.0	829	*6.72*	*-0.70*
186.crafty	1000	106.0	943	*6.85*	*-0.36*
175.vpr	1400	133.0	1053	*6.96*	*-0.06*
254.gap	1100	98.6	1116	*7.02*	*0.10*
256.bzip2	1500	133.0	1128	*7.03*	*0.12*
176.gcc	1100	96.4	1141	*7.04*	*0.16*
252.eon	1300	96.7	1344	*7.20*	*0.60*
300.twolf	3000	213.0	1408	*7.25*	*0.72*
255.vortex	1900	114.0	1667	*7.42*	*1.18*
181.mcf	1800	99.8	1804	*7.50*	*1.39*
MEDIAN	1450	124	1122	*7.02*	
HM	1462	131	1005	*6.96*	
GM	1521	141	1077	*6.97*	
AM	1592	157	1140	*6.98*	
STDEV	540	88	377	*0.37*	
SKEW	1.64	2.16	0.12	*-0.96*	
KURTOSIS	3.74	5.19	0.07	*1.76*	**EXP**
95% lo	1252	102	903	*6.75*	853
95% hi	1591	155	1140	*7.22*	1360
Bin<m-2s	511	-19	386	*6.24*	513
Bin<m-s	1052	69	763	*6.61*	743
Bin<m	1592	157	1140	*6.98*	1077
Bin<m+s	2132	244	1517	*7.35*	1560
Bin<m+2s	2672	332	1894	*7.72*	2261
COEDET			0.940	0.88	

Histogram Bin	**R.. Freq**	**Hist Bin**	**ln(R..) Freq**
386	0	6.24	1
763	1	6.61	0
1140	6	6.98	4
1517	3	7.35	5
1894	2	7.72	2
More	0	More	0

Table 8 - SPEC CINT2000 SERPOP Analysis

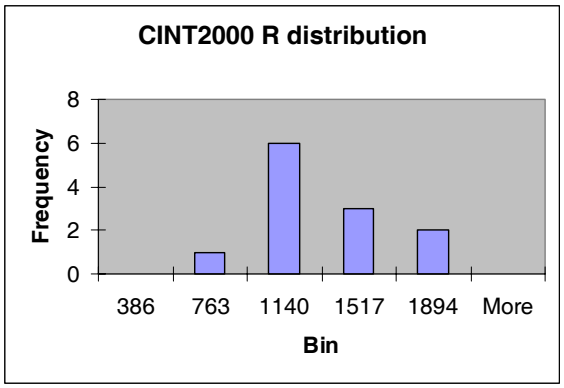
On the next page are plotted distributions using the bins above, and z-plots that should approximate straight lines for normal distributions. The COEDET is the Coefficient of Determination

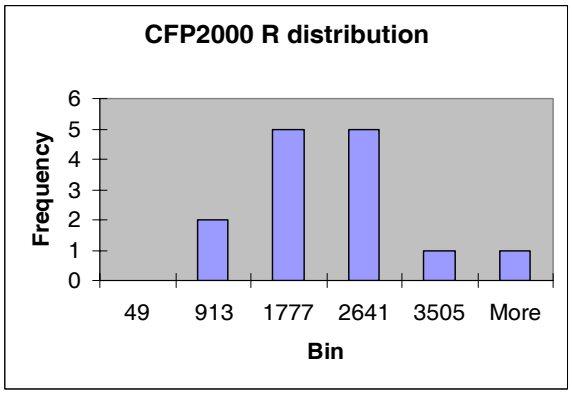
	S1 Ti1	S2 Ti2	S2 Ri21*	S2 ln(R..)	S2 z(ln..)
200.sixtrack	1100	152.0	724	6.58	-1.65
177.mesa	1400	164.0	854	6.75	-1.31
172.mgrid	1800	173.0	1040	6.95	-0.90
188.ammp	2200	210.0	1048	6.95	-0.88
191.fma3d	2100	163.0	1288	*7.16*	-0.45
301.apsi	2600	193.0	1347	*7.21*	-0.36
173.applu	2100	151.0	1391	*7.24*	-0.29
189.lucas	2000	109.0	1835	*7.51*	0.29
187.facerec	1900	98.5	1929	*7.56*	0.39
171.swim	3100	145.0	2138	*7.67*	0.61
168.wupwise	1600	72.1	2219	*7.70*	0.68
179.art	2600	112.0	2321	*7.75*	0.78
183.quake	1300	44.1	2948	*7.99*	1.28
178.galgel	2900	76.4	3796	*8.24*	1.80
MEDIAN	2050	148.0	1613	*7.38*	
HM	1881	111	1440	*7.35*	
GM	1970	123	1598	*7.36*	
AM	2050	133	1777	*7.38*	
STDEV	597	49	864	*0.48*	
SKEW	0.19	-0.29	0.99	*0.07*	
KURTOSIS	-0.69	-0.77	0.83	*-0.69*	**EXP**
95% lo	1708	105	1282	*7.10*	1214
95% hi	2392	161	2272	*7.65*	2104
Bin<m-2s	856	36	49	*6.42*	613
Bin<m-s	1453	84	913	*6.90*	989
Bin<m	2050	133	1777	*7.38*	1598
Bin<m+s	2647	182	2641	*7.86*	2581
Bin<m+2s	3244	230	3505	*8.34*	4169
COEDET			0.900	0.977	

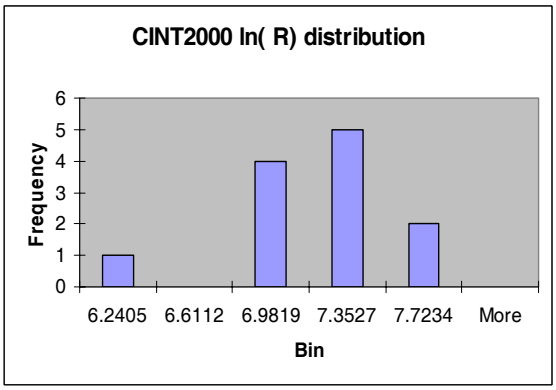
Histogram Bin	**R.. Freq**	**Hist** *Bin*	**ln(R..) Freq**
49	0	*6.418*	0
913	2	*6.897*	2
1777	5	*7.377*	5
2641	5	*7.856*	5
3505	1	*8.335*	2
More	1	*More*	0

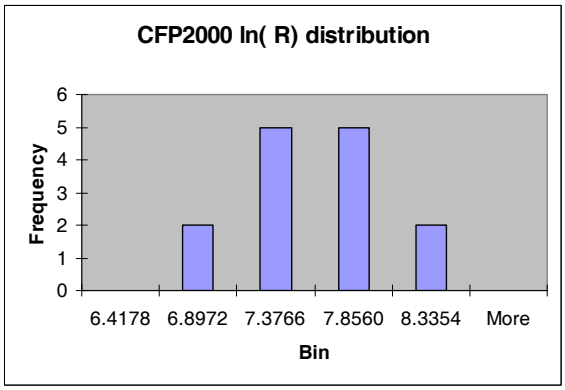
Table 9 - SPEC CFP2000 SERPOP Analysis

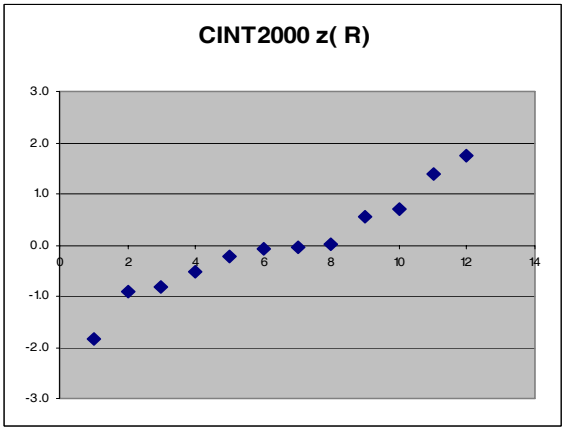
the plot is a good fit to a normal distribution. The EXP column back-transforms from ln(R..) to R.. domain.

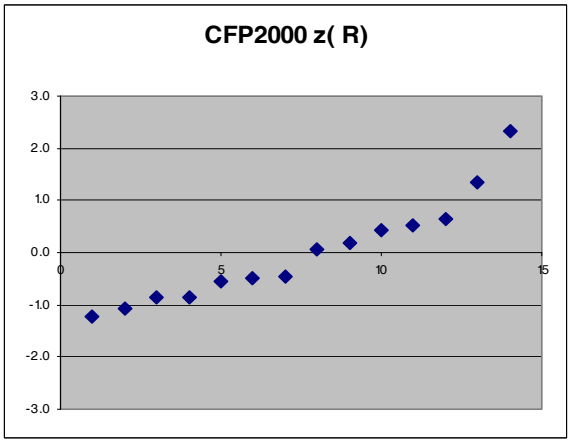


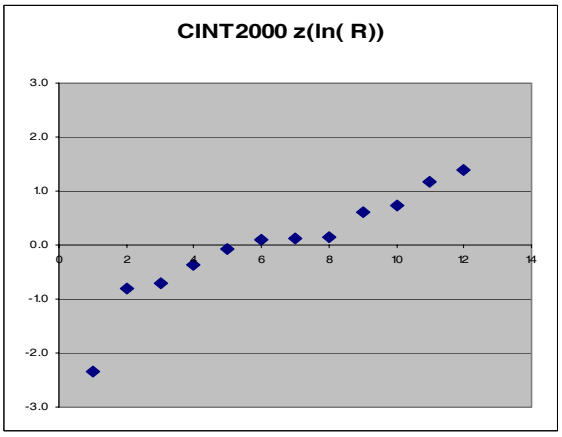












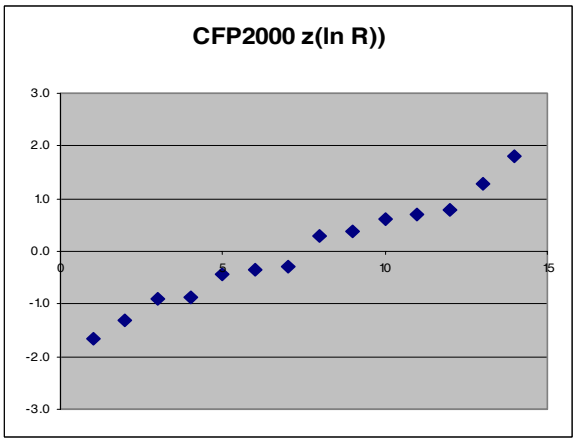


Table 10 - Distributions and z-scores

8.2 Specific observations

Relative performance is always a distribution, not just a number. Assuming **A1-A5**, SERPOP analysis, using lognormal distributions, works well, and yields standard statistical results that quantify uncertainty.

While it is not generally valid to use the AM on a distribution of ratios, in this case, with modest Standard Deviation, normal and lognormal are close, so those numbers were included.

It is especially worth examining the 95% Confidence limits [853-1360 and 1214-2104] as that offers a reasonable estimate of the uncertainty, which of course, is mainly improved by increasing the sample size. The COEDET value for ln(R..) indicates the goodness of it, and that is worth checking. In this case, the normal assumption fits slightly better for CINT2000, and the lognormal slightly better for CFP2000. More studies will appear in future papers, but so far, examination of several sets of data from LFK, DR CPU2, and SPEC look promising, i.e., with lognormal having a good fit, and usually better than the normal fit, as skew increases. That is no surprise, mathematically.

SKEW and KURTOSIS have fairly small values here; large SKEW numbers indicate care must taken, as do large negative KURTOSIS, and large STDEV. These indicate unusual outliers, or perhaps indicate that the benchmark set contains codes with radically different characteristics. For example, in looking at LFK this way, simple scalar machines tend to have well-behaved distributions. Vector or parallel systems tend to show several distinct performance clusters. A large positive KURTOSIS probably indicates that systems being compared are tightly related, or else there is a subset of benchmarks that are unusually well-correlated.

9 Conclusion

WCA, SERPOP, and WAW are all needed, and they sometimes need differing mathematical treatments.

Mathematically, it is somewhat strange to compute AM, GM, and HM on the same sample of *anything*, i.e., the appropriate choice depends on knowledge of the population's distribution, which may not be available. Perhaps people use them intuitively as a measure of dispersion, if that is otherwise not available. Alternatively, it may be that (as in LFK) they are used as surrogates for different populations that have been mixed together, i.e., including vector and parallel systems with serious code tuning (that achieve AM) and small-cache micros (that only yield HM), with GM being a middle estimate. This is somewhat akin to SPEC's use of Base and Peak ratios.

It is very non-obvious to specify a single distribution that usefully describes run-times in general. Computing means on samples of unknown distributions is indeed statistically chancy, even if the algebra works.

The GM is seen as vital to SERPOP analyses, as there are many reasons to believe that distributions of performance ratios are better modeled by lognormal distributions than by normal ones. The former is more general, and the latter has serious statistical problems in the presence of skewed distributions, and even claims that two systems are each faster than the other in real examples. The use of ratios in this case converts unknown run-time distributions into a well-understood lognormal distribution whose analysis is easy.

There is much work to do, but perhaps it is time for truce in the War of Means.

Acknowledgments

The author thanks the large number of people who have worked very hard on SPEC and other benchmarking efforts over the years. Angela Hey provided good comments on early drafts. Dave Patterson and John Hennessy pushed me to think more about the mathematics, as my early informal attempts were insufficiently convincing.

References

[1] Savage, S, "Some Gratuitous Inflammatory Remarks on the Accounting Industry," http://www.stanford.edu/dept/MSandE/faculty/savage/AccountingRemarks.pdf

[2] Fleming, P., Wallace, J. "How Not to Lie With Statistics: The Correct Way to Summarize Benchmarks," *Comm ACM,* Vol 29, No. 3, pp. 218-221, March 1986.

[3] Smith, J., "Characterizing Computer Performance with a Single Number," *Comm ACM,* Vol 31, No. 10, pp. 1202-1206, October 1988.

[4] John, L., "More on finding a Single Number to indicate Overall Performance of a Benchmark Suite," *Computer Architecture News,* Vol. 32, No 1, pp. 3-8, March 2004.

[5] Jain, R., The Art of Computer Systems Performance Analysis, John Wiley and Sons, New York, 1991.

[6] Lilja, D., Measuring Computer Performance – A Practioner's Guide, Cambridge University Press, 2000.

[7] Hennessy, J, Patterson, D., Computer Architecture – A Quantitative Approach, Third Edition, Morgan Kaufmann Publishers, 2003. Earlier editions 1990 and 1996.

[8] McMahon, F., "The Livermore Fortran kernels: A Computer test of numerical performance range," Tech. Rep. UCRL-55745, Lawrence Livermore national Laboratory, Univ. of California, Livermore, 1986. See also:

[9] McMahon, F., "L.L.N.L Fortran Kernels Test" source. www.netlib.org/benchmark/livermore or www.llnl.gov/asci_benchmarks/asci/limited/lfk/README.html

[10] Digital Review, "At the speed of light through a PRISM," Digital Review, pp. 39-42, December 19, 1988.

[11] Spanier, S., "Sun-3 Benchmarks," Sun Microsystems, Aug 1985.

[12] Hewlett Packard, "HP 9000 Series 800 Performance Brief," May 1987.

[13] MIPS Computer Systems, "Performance Brief Part 1: CPU Benchmarks, Issue 3.0," October 1987.

[14] AMD, "Am29000 Performance Analysis,", May 1988.

[15] Apollo Computer, "Apollo Performance Report Version 1.2," Digital Equipment 1988.

[16] Digital Equipment, "RISC Workstation Performance Summary," July 11, 1989.

[17] Hewlett Packard, "Series 300 HP-UX 6.5 Performance Brief," April 1989.

[18] Ralph Humphries, "Performance Report, Revision 1.4, July 1, 1989," Silicon Graphics Computer Systems, Mountain View, CA.

[19] McInnis, D., Kusik, B., Bhandarkar, D., "VAX 8800 System Overview," Proc. COMPCON 1987, pp. 316-321, San Francisco, CA, Feb 1987. Note: VAX 8800 uses two 8700 CPUs.

[20] SPEC, *SPEC Newsletter*, Vol. 1, No. 1, Fall 1989.

[21] SPEC, "Benchmark Results," *SPEC Newsletter,* Vol. 2, Issue 1, Winter 1990.

[22] Mashey, J., "SPEC Results Help Normalize Vendor Mips-Ratings for Sensible Comparison," *SPEC Newsletter,* Vol. 2, Issue 3, Summer 1990.

[23] Giladi, R, Ahituv, N., "SPEC as a Performance Evaluation Measure," *Computer,* Vol. 28, No.8, pp 33-42, Aug 1995.

[24] Mighafori, N., Jacoby, M., and Patterson, D., "Truth in SPEC Benchmarks," *Computer,* Vol 28, No. 8, pp. 33-43, Aug 1995.

[25] DeCoursey, W., Statistics and Probability for Engineering Applications with Microsoft Excel, Newnes, Amsterdam, 2003.

[26] Good, P., Hardin, J., Common Errors in Statistics (and How to Avoid Them), Wiley-Interscience, Hoboken,NJ, 2003.

[27] Limpert, E., Stahel, W., "Life is log-normal! Science and art, life and statistics," ETH Zurich, 1998, http://www.inf.ethz.ch/personal/gutc/lognormal/brochure.html

[28] Limpert, E., Stahel, W., and Abbt, W., "Log-Normal Distributions across the Sciences: Keys and Clues," BioScience Vol 51, No. 5, pp. 341-352, May 2001. Also in: http://www.inf.ethz.ch/personal/gutc/lognormal/bioscience.pdf

[29] SPEC, www.specbench.org