Transformed Vargha-Delaney Effect Size

Conference Paper · September 2015

DOI: 10.1007/978-3-319-22183-0_29

3 authors, including:

Geoffrey Kenneth Neumann
University of Stirling
12 PUBLICATIONS **20** CITATIONS

SEE PROFILE

Mark Harman
University College London
439 PUBLICATIONS **13,704** CITATIONS

SEE PROFILE

Transformed Vargha-Delaney Effect Size

Geoffrey Neumann[1], Mark Harman[2], and Simon Poulding[3]

[1] University of Stirling, UK
[2] University College London, CREST centre, UK
[3] Blekinge Institute of Technology, Sweden

Abstract Researchers without a technical background in statistics may be tempted to apply analytical techniques in a ritualistic manner. SBSE research is not immune to this problem. We argue that emerging rituals surrounding the use of the Vargha-Delaney effect size statistic may pose serious threats to the scientific validity of the findings. We believe investigations of effect size are important, but more care is required in the application of this statistic. In this paper, we illustrate the problems that can arise, and give guidelines for avoiding them, by applying a 'transformed' Vargha-Delaney effect size measurement. We argue that researchers should always consider which transformation is best suited to their problem domain before calculating the Vargha-Delaney statistic.

Keywords: Vargha and Delaney, Effect Size, Threats to Validity

1 Introduction

Software engineering researchers, and SBSE researchers in particular, have been increasingly adept at applying statistical analysis to their empirical data. In addition to measuring and reporting the statistical significance of the data, many researchers also rightly report an effect size. Statistical significance indicates how likely an observed difference between, for example, two randomized search algorithms is genuine rather than a result of chance. By contrast, the effect size provides an indication of the practical importance of any observed difference between the algorithms, while taking into account their inherent variability.

Vargha and Delaney's effect size measure [17] is regarded as a robust test when assessing randomized algorithms such as those used in SBSE [3]. The test returns a statistic, $\hat{A}$ (often denoted $\hat{A}_{12}$ in SBSE research), that takes values between 0 and 1; a value of 0.5 indicates that the two algorithms are stochastically equivalent, while values closer to 0 or 1 indicate an increasingly large stochastic difference between the algorithms.

One of the most attractive properties of the Vargha-Delaney test is the simple interpretation of the $\hat{A}$ statistic: for results from two algorithms, A and B, $\hat{A}_{AB}$ is simply the expected probability that algorithm A produces a superior value to algorithm B. If $\hat{A}_{AB} = 0.7$ then Algorithm A is expected to 'beat' Algorithm B 70% of the time, which may lead us to conclude the Algorithm A is 'better' than Algorithm B (and with a fairly large effect size). We can think of $\hat{A}_{AB}$ as a contest between algorithms A and B, repeated over a number of trial applications,

with a draw counting equally for both algorithms. Other effect size statistics do not have this intuitive interpretation.

However, if we are to ensure this contest has a *correct* interpretation, we must be careful to compare the 'right' data values from the algorithms. In this paper we argue that the data observed for two techniques or algorithms may need to undergo transformation before we calculate the $\hat{A}$ statistic. While data transformations of the types we describe here are a common practice in statistics, we contend they are not applied in SBSE research as often as they should be. In the following section we demonstrate that the Vargha-Delaney test can be misinterpreted when it is applied to untransformed data, leading to serious threats to validity that can reverse the scientific findings of the study that contains them.

2 Misapplication of Vargha-Delaney Effect Size

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10
Algorithm A	6.02	0.01	0.00	0.02	0.04	0.05	0.01	0.03	15.00	9.04
Algorithm B	0.05	0.05	2.00	0.09	0.08	0.06	0.05	0.09	0.09	0.08

(a) Untransformed Data

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10
Algorithm A	6.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	15.00	9.00
Algorithm B	0.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

(b) Transformed Data

Figure 1. Illustrative Example of Three SBSE Algorithm Results

Consider the illustrative data that might be obtained from two different algorithms, set out in Figure 1(a), in which the problem demands that lower values are better than higher values. Suppose the values recorded in each trial are execution times in seconds. In this case, $\hat{A}_{AB} = 0.315$. As lower scores are better here, this would indicate that A is the clear winner out of A and B.

The analysis is deceptively simple, and in fact it is *oversimplified*. Unfortunately, many SBSE researchers (the present authors included) might have been tempted to simply apply $\hat{A}$ to raw (untransformed) data. This oversimplification can lead to serious threats to the construct validity of the scientific findings. To illustrate this threat to validity, suppose we are interested in execution time because our use-case involves a user who will become irritated should they have to wait too long for an answer. It is widely believed that response times lower than 100 ms are, for all practical purposes, imperceptible to users [4].

If this is our use-case, then we should, at least, first transform the data to reflect the fact that values lower than 100 ms are considered equally good (essentially 'instantaneous'). The transformed data for this response-time scenario is depicted in Figure 1(b). In this more *use-case-compliant* analysis, $\hat{A}_{AB} = 0.615$, completely *reversing* the findings of the study; Algorithm A is now seen to be the loser, not the winner. As can be seen, applying Vargha-Delaney effect size

to untransformed data can result in a potential threat to validity, able to undermine the scientific findings of the entire study. Transforming the data prior to the application of the Vargha-Delaney test addresses this threat.

We refer to this approach as *Pre-Transforming Data (PTD)*. It is one of two techniques discussed in this paper for avoiding these important threats to validity. As an alternative to transforming data before $\hat{A}$ is calculated, the comparisons that are performed during the $\hat{A}$ test could be modified. We refer to this as the *Modified Comparison Function* (MCF) technique. PTD is simpler to apply and normally sufficient, although the MCF technique is strictly more expressive, and so may be necessary in some scenarios. For example, when the quality of a result is a function of multiple values, pareto dominance could be incorporated into an MCF while PTD could not achieve this.

The key to the correct application of the Vargha-Delaney effect size measure lies in considering what it *means* for one algorithm to be superior to another. This is a question that goes to the very heart of any empirical investigation of algorithms. Non-parametric statistical tests such as Vargha-Delaney are championed because of a lack of assumptions. However, we have to be careful not to relinquish one set of questionable assumptions, only to tacitly incorporate another set of highly flawed assumptions. In particular, we ought not to be seduced by the apparent precision of our measurements, but should be ready to transform our data to represent the *appropriate precision* of the solution space. In the remainder of this paper we give guidelines to illustrate how researchers might choose different transformations for different scenarios.

3 Transforming Data and the $\hat{A}$ Comparison Function

Calculating $\hat{A}$ requires the comparison of each data point in one sample set with each data point in the other. The calculation is implemented in many statistical packages and is widely used. One advantage of using PTD is that the experimenter need not change the computation of $\hat{A}$ at all. Rather, they simply apply their existing $\hat{A}$ computation to the pre-transformed data, in order to avoid the threat to validity.

When PTD is used, $\hat{A}$ is calculated only after the data has been transformed using a many-to-one function, f_{ptd}. The function f_{ptd} captures the precision inherent in the judgement of whether one algorithm is superior to another for our proposed use-case. f_{ptd} can be used to 'bucket together' any observed values that should be considered equivalent for the purposes of algorithm comparison (such as all those timings lower than 100ms in response time).

By contrast, MCF modifies the way in which the comparisons themselves are performed. Given two numeric results x_1 and x_2, $\hat{A}$ simply determines which is true out of $x_1 > x_2$, $x_1 < x_2$ and $x_1 = x_2$. With the MCF technique, this comparison is replaced by a function, f_{mcf}, which takes, as input, two results x_1 and x_2. These may be simple numeric values or may be more complex data structures. f_{mcf} returns one of three possible outcomes (x_1 *is better than* x_2, x_1 *is worse than* x_2 and x_1 *equals* x_2).

In the remainder of this paper we consider possible ways in which f_{ptd} and f_{mcf} might be defined for different use-case scenarios, thereby illustrating our proposed approach to the transformed Vargha-Delaney effect size measure.

Using Moore's law: An observation originally made in 1965 by Gordon Moore stated that the number of transistors on a dense integrated circuit approximately doubles every two years [11]. There are various interpretations of the speed up implications achievable simply by advances in underlying hardware. However, if Algorithm A is faster than Algorithm B by 10% we can safely assume that this 'advantage' is equivalent to less than 6 months of Moore's law. The precise determination of bucketing remains for the researcher to define and justify, but we might consider exponentially larger bucket sizes to take account of the erosion of performance advantages due to Moore's law.

Implementation differences: For problems concerning computational efficiency bucketing techniques could be set so that only improvements that are greater than those matched by mere implementation improvements are counted. Areas of implementation that we might consider include parallelization or hardware and software changes. In the case of parallelization it is possible to estimate how much time could be saved if a serial process were run in parallel. Amdahl's law [2] states that if a process is currently being run entirely sequentially and s is the proportion of time spent executing parts of the process that cannot be parallelized, then parallelizing it would cause it to run $1/(s + \frac{(1-s)}{N})$ times faster where N is the number of available processors. If the cost of additional parallel computation can be assessed, then this coulld be used to determine a threshold for bucketing.

Categorical Thresholds: In some situations there is a natural boundary that could be used as a threshold for equivalence bucketing. Often this will be a boundary between an unacceptable result and an acceptable result (as in the case of branch coverage distance [5]).

Singh and Kahlon [15] and Shatnawi [14] give thresholds for object oriented programming metrics such as information hiding, encapsulation, class complexity, inheritance, class size, cohesion and coupling. These metrics can be used to predict certain characteristics of 'poorly designed' programs (those above threshold have a 'code smell'). The fitness function that guides search-based refactoring to remove such smells might use the raw OOP metric value to guide it [7,12]. However, when it comes to assessing effect size, we should only consider a 'win' occurring when the metric value indicates the smell is removed (i.e. the metric value moves within threshold).

Thresholds can be defended and justified in many ways. There are often precedents of considering certain threshold values. For example Schneider [13] favours precision above 0.6 and recall above 0.7 (based on work by Ireson [9]), while McMinn states that the proportion of runs in which a branch is covered in 50 runs during testing needs to be more than 60% [10]. PTD could be used here, accompanied by a justification of the transformation applied to the data

based on these threshold values. Doing so will strengthen the conclusion validity by ruling out 'trivial wins'.

Time Scales: Ali et al. [1] review test case generation and state that a difference of a few minutes in how long a test case takes to run is unlikely to be of practical importance. One might characterise improvements in terms, of whether the difference is noticeable (100ms delay threshold), whether it can be used interactively (a few second delay), whether it can be achieved overnight (perhaps more suitable to regression test optimisation [18]) and so on. These could be used as thresholds for PTD.

Decomposition: If fitness is calculated from more than one metric then this provides many more options for Transformed VD. Many ideas from multi objective optimization, such as pareto dominance, could be explored even though the problem itself was not optimized as a multi objective problem. This could simply mean using a MCF, which only returns a difference between two solutions when one solution *pareto dominates* the other (it achieves at least equal performance in every objective and better performance in at least one objective [16]).

However, a much more sophisticated and problem specific MCF could be used. Objectives could be prioritized, each objective could have its own thresholds and conditional statements could be used so that the importance of one objective depends on the results of other objectives. In addition to objectives, fitness functions obtained by testing on a set of problem instances are also potentially decomposable. Comparisons could, for example, take into account growth functions across a set of problem instances of increasing complexity so that scalability can be compared. Hsu notes the importance of both objective prioritization and growth functions in software testing [8].

This discussion has highlighted many ways in which two solutions can be compared which cannot be replicated simply by pre-transforming the data, clearly demonstrating that this is one area in which the MCF technique provides much more power than both standard $\hat{A}$ and also PTD. However, PTD is simple: once transformed, the data can be compared using any existing VD test.

4 Conclusion

We have demonstrated how ritualistic application of the Vargha-Delaney $\hat{A}$ effect size may reverse technical findings, leading to a fundamentally flawed scientific analysis. The problem applies, not only to work on Search Based Software Engineering (SBSE) [6], but any work involving the comparison of randomised algorithms, although we chose to illustrate the problem using examples drawn from SBSE. This serious threat to validity cannot be overcome by simply avoiding the use of $\hat{A}$, because effect size reporting is essential. To address this problem, we proposed two approaches that enable the Vargha-Delaney measure to be applied to transformed data. Finally, we observe that although our paper focuses on the Vargha-Delaney effect size, there is no reason why these ideas could not be applied to other statistical tests used by SBSE researchers.

References

1. Ali, S., Briand, L.C., Hemmati, H., Panesar-Walawege, R.K.: A systematic review of the application and empirical investigation of search-based test case generation. Software Engineering, IEEE Transactions on 36(6), 742–762 (2010)
2. Amdahl, G.M.: Validity of the single processor approach to achieving large scale computing capabilities. In: Proceedings of the April 18-20, 1967, spring joint computer conference. pp. 483–485. ACM (1967)
3. Arcuri, A., Briand, L.: A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. Software Testing, Verification and Reliability (2012)
4. Department of Defense: US DOD MIL-STD 1472-F: Human engineering standard (1999)
5. Harman, M., Jia, Y., Zhang, Y.: Achievements, open problems and challenges for search based software testing (keynote). In: 8^{th} IEEE International Conference on Software Testing, Verification and Validation (ICST 2014). Graz, Austria (April 2015)
6. Harman, M., Jones, B.F.: Search based software engineering. Information and Software Technology 43(14), 833–839 (Dec 2001)
7. Harman, M., Tratt, L.: Pareto optimal search-based refactoring at the design level. In: 9^{th} annual conference on Genetic and evolutionary computation (GECCO 2007). pp. 1106 – 1113. ACM Press, London, UK (Jul 2007)
8. Hsu, H.Y., Orso, A.: Mints: A general framework and tool for supporting test-suite minimization. In: Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on. pp. 419–429. IEEE (2009)
9. Ireson, N., Ciravegna, F., Califf, M.E., Freitag, D., Kushmerick, N., Lavelli, A.: Evaluating machine learning for information extraction. In: Proceedings of the 22nd international conference on Machine learning. pp. 345–352. ACM (2005)
10. McMinn, P.: How does program structure impact the effectiveness of the crossover operator in evolutionary testing? In: Search Based Software Engineering (SSBSE), 2010 Second International Symposium on. pp. 9–18. IEEE (2010)
11. Moore, G.E., et al.: Cramming more components onto integrated circuits. Proceedings of the IEEE 86(1), 82–85 (1998)
12. O'Keeffe, M., Ó Cinnédie, M.: Search-based refactoring: an empirical study. Journal of Software Maintenance 20(5), 345–364 (2008)
13. Schneider, K., Knauss, E., Houmb, S., Islam, S., Jürjens, J.: Enhancing security requirements engineering by organizational learning. Requirements Engineering 17(1), 35–56 (2012)
14. Shatnawi, R.: A quantitative investigation of the acceptable risk levels of object-oriented metrics in open-source systems. Software Engineering, IEEE Transactions on 36(2), 216–225 (March 2010)
15. Singh, S., Kahlon, K.: Object oriented software metrics threshold values at quantitative acceptable risk level. CSI Transactions on ICT 2(3), 191–205 (2014)
16. Srinivas, N., Deb, K.: Multi-objective function optimization using non-dominated sorting genetic algorithms. Evolutionary Computation 2 (3), 221–148 (1995)
17. Vargha, A., Delaney, H.D.: A critique and improvement of the CL common language effect size statistics of McGraw and Wong. Journal of Educational and Behavioral Statistics 25(2), 101–132 (2000)
18. Yoo, S., Harman, M.: Regression testing minimisation, selection and prioritisation: A survey. Journal of Software Testing, Verification and Reliability 22(2), 67–120 (2012)