An Empirical Comparison of Mutant Selection Assessment Metrics

Jie M. Zhang[◇], Lingming Zhang[†], Dan Hao[*], Lu Zhang[*], Mark Harman[◇]
[◇]CREST, University College London, WC1E 6BT, UK.
[†]Department of Computer Science, University of Texas at Dallas, 75080, USA
[*]Key Laboratory of High Confidence Software Technologies (PKU), China

***Abstract*—Mutation testing is expensive due to the large number of mutants, a problem typically tackled using selective techniques, thereby raising the fundamental question of how to evaluate the selection process. Existing mutant selection approaches rely on one of two types of metrics (or assessment criteria), one based on adequate test sets and the other based on inadequate test sets. This raises the question as to whether these two metrics are correlated, complementary or substitutable for one another. The tester's faith in mutant selection as well as the validity of previous research work using only one metric rely on the answer to this question, yet it currently remains unanswered. To answer it, we perform qualitative and quantitative comparisons with 104 different projects, consisting of over 600,000 lines of code. Our results indicate a strong connection between the two types of metrics (R^2 = 0.8622 on average), providing evidence that it may be valid to adopt only one metric.**

I. Introduction

Software testing plays an essential role in assuring software quality. Well-designed tests[1] may detect more faults than poorly-designed ones [3], [4], [5]. Mutation testing has been shown to have powerful fault-revealing ability [6], [7], [8], [9]. Recent advanced mutation tools have also gained adoptions from industry, e.g., mutation testing has been used by developers from The Ladders, Sky, Amazon, State Farm, Norways e-voting system, and the Linux kernel [10], [11], [12]. In (first order [13]) mutation testing, a set of faulty programs (i.e., *mutants*) are generated by injecting one error at a time according to some transformation rules (i.e., *mutation operators*). A test set is said to kill a mutant when at least one of its tests causes behavioral differences between this mutant and the original program. The proportion of killed mutants is called the *mutation score*. Higher mutation scores are preferred, because they indicate that the test set may potentially detect more real faults.

Although powerful in evaluating software tests, mutation testing is currently believed to be expensive, mainly due to the requirements of running a large number of mutants against the test set under evaluation. To tackle this problem, selective mutation testing was proposed [14], and has become a popular way of reducing the costs: it is adopted by many mutation tools (e.g., PIT [10], Major [15], and Javalanche [16]), and is also widely used in various mutation testing applications, such as real fault simulation [17], [18], test generation [19], [20], [21], fault localization [22], regression testing [23], [24], and program repair [25].

In selective mutation testing, it is crucial to select a subset of mutants that can represent the whole set of mutants [26]. Therefore, mutant selection assessment metrics (abbreviated as "metrics" in this paper) to measure the representativeness of the selected mutants are important. To date, two types of mutant selection assessment metrics have been proposed: the adequate-test-based type and inadequate-test-based type [26]. Both types of metrics compare the selected mutants and the whole set of mutants in evaluating the fault-revealing ability of test sets. The former type is based on constructing *adequate* test sets that can kill all the selected mutants, whereas the latter is based on constructing *inadequate* test sets that do not necessarily kill all the selected mutants (see details in Section II).

These two types of metrics, although having different focuses, could be both adoptd for measuring how well the selected mutants represent the whole set of mutants. It is thus important to learn their association for better understanding and application, which is important for both testers and researchers. For testers, if they learn that these two metrics provide different conclusions regardint the same mutant subset, they may get confused with which metric to choose and even wonder whether the metrics are designed properly. For researchers, we found that the current work exploring mutant selection either adopts only one type of metric or uses both without further comparison. For example, Barbosa et al. [27] used the adequate-test-based metric to determine sufficient mutation operators in C language; Zhang et al. [28] used the adequate-test-based metric to compare different mutant selection techniques; Zhang et al. [1] and Gopinath et al. [29] used the adequate-test-based and inadequate-based metrics respectively to investigate the scalability of selective mutation testing; Zhang et.al. [2] investigated different mutant selection techniques using both types of metrics separately but without any comparison between them. A big difference in the metric values may reveal that it is essential to consider both metrics in the experiments.

In this paper, we present the first study to investigate the relationship of the two types of mutant selection assessment metrics. We first make direct comparison between different metrics through collecting the their values calculated based on identical mutant subsets. Additionally, in selective mutation

[1] In this paper, following existing work on selective mutation testing [1], [2], we regard each test method as a "test".

testing, with different experimental factors (e.g., the number of tests, test density, and the number of mutants), the mutants selected in the same strategy may have very different representativeness. We then investigate whether the two types of metrics show similar or different patterns regarding their associations with the factors. For each of the two aspects mentioned above, we conduct qualitative analysis through presenting and observing figures, as well as quantitative analysis through regression analysis. The analysis results reveal that the values of the two types of metrics have strong connection, with an average *Adjusted* R^2 of 0.8622. Additionally, they have similar associations with different experimental factors. These results provide proof that the two types of metrics could be substituted for one another with high probability.

In the paper, by default, we use mutation tool PIT to generate mutants, and adopt the random mutant selection strategy[2] following previous related work [28], [1]. To reduce the threats caused by such default experimental design, we replicate our study with another popular mutant selection strategy: operator based mutant selection[3]. The replication shows similar results which further verify the validity of our conclusions.

To summarize, the paper makes the following contributions:
(1) Empirical evidence on the strong association of the two metrics. We perform our study on 104 GitHub projects (summing up to 606,886 SLOC and 827,971 mutants). The results indicate that the two types of metrics are strongly connected with an average *Adjusted* R^2 of 0.8622. We also observe similar patterns regarding their associations with the seven experimental factors. The findings firstly provide evidence that the metrics have strong association, and thus that it may be reasonable to adopt only one metric when assessing the representativeness of selected mutants in scientific research activities.
(2) Advice and implication. Our study also reveals a number of other interesting findings and practical implications based on regression analysis: the strategy for dealing with equivalent mutants has negligible impact on the mutant selection results (validating the treatment of equivalent mutants in most previous work on selective mutation testing [2], [26], [30], [31], [1], [28] for the first time). We also confirm the previous finding [1], [32] that larger projects may need smaller ratio of mutants to achieve an expected representativeness.

The rest of this paper is organized as follows. Section II introduces the background of the two types of metrics. Section III introduces the experimental design process. Section IV presents the comparison results. Section V discusses the threats through empirical comparison between two mutant selection approaches and two most popular mutation tools. Section VII concludes.

II. Mutant Selection Metrics

In selective mutation testing, the selected mutants are supposed to represent the whole set of mutants in evaluating the fault-revealing ability of tests. Previous work have proposed three categories of mutant selection techniques: operator-based mutant selection [33], [28], random-based mutant selection [1], [28], [1], and element-based mutant selection [2]. In this paper, as in previous work [1], we choose random-based mutant selection, which has been found to represent the state-of-the-art for mutant selection approaches [2], [28].

To measure the representativeness of the selected mutants, two types of mutant selection assessment metrics have been proposed: the first one is based on tests that are adequate for the selected mutants [26]; the other one is based on inadequate tests [34]. To explain the details of the two types of metrics, for a program, we use M_e to represent the set of equivalent mutant[4] that are not killed by any way, and thus should be removed as they are useless in evaluating the test. The remaining mutants are called *non-equivalent mutants*, i.e., M_{ne}. We use M_s to represent the subset of mutants selected from M_{ne} (i.e., $M_s \subset M_{ne}$). M_s is supposed to represent M_{ne} in selective mutation testing.

Adequate-Test-based Metric. This metric is calculated based on adequate test sets for the selected mutants [26]. In particular, for the set of selected mutants M_s, we first construct a group of adequate test sets $\{AT_1, AT_2, ..., AT_n\}$, each of which (i.e., AT_i, where $1 \leq i \leq n$) is required to kill all the mutants in M_s. Then we execute the whole set of mutants M_{ne} against each AT_i, recording the number of mutants within M_{ne} killed by each AT_i (denoted as $\#M_{ne}(AT_i)$). We use $MS(M_{ne}, AT_i)$ to represent the mutation score of AT_i on M_{ne} (i.e., $MS(M_{ne}, AT_i) = \frac{\#M_{ne}(AT_i)}{\#M_{ne}}$). Finally, we calculate to what extent M_s represents M_{ne} using these adequate test sets through the following formula:

$$S(M_s) = \frac{1}{n}\sum_{i=1}^{n} MS(M_{ne}, AT_i) \quad (1)$$

In this formula, $S(M_s)$ is the representativeness of selected mutants M_s when evaluating any test set. The values of $S(M_s)$ range from 0 to 1. The larger $S(M_s)$ is, the better M_s can represent M_{ne}. Ideally, when $S(M_s)$ is equal to 1, M_s is as sufficient as the whole set M_{ne} in evaluating a test set.

Inadequate-Test-based Metric. The inadequate-test-based metric is defined as the correlation criteria between the real mutation score calculated on M_{ne} and the approximate mutation score calculated on M_s [34]. In particular, we randomly construct a group of test sets $\{NT_1, NT_2, ..., NT_n\}$, each of which (i.e., NT_i where $1 \leq i \leq n$) is not necessarily adequate to M_s (i.e., may not kill all the mutants in M_s). Then, we execute each test set NT_i against both M_s and M_{ne}, recording the mutation score of NT_i on the selected mutants M_s (denoted as $MS(M_s, NT_i)$) and the mutation score of NT_i on the whole set of non-equivalent mutants M_{ne} (denoted as $MS(M_{ne}, NT_i)$). Finally, we calculate the correlation coefficient based on

[2] To select mutants from the whole mutant set randomly.
[3] To select mutants generated by a subset of mutation operators.
[4] Strictly speaking, equivalence is not decidable, but we use the term here to decide those mutants that are unkillable with the test suite (see more in Section III-D).

Table I
THE 7 EXPERIMENTAL FACTORS

Category	Factor	Explanation
Program scale	$SLOC$	program size (i.e.,the executable lines of code)
	M	total number of mutants
	M_{ne}	number of non-equivalent mutants
	R_{ne}	proportion of non-equivalent mutants to all mutants
Tests	T	total number of tests (i.e., test methods in JUnit testing)
	T_d	test density (i.e., the number of tests per hundred mutants)
Metric	T_c	average size of constructed test sets

the pairs between $MS(M_s, NT_i)$ and $MS(M_{ne}, NT_i)$ using the following formula:

$$C(M_s) = Corr(\langle MS(M_s, NT_i), MS(M_{ne}, NT_i)\rangle, 1 \leqslant i \leqslant n) \quad (2)$$

In this formula, $C(M_s)$ represents the representativeness of selected mutants M_s at evaluating any test set, while $Corr$ represents a correlation coefficient. Note that the ideal $C(M_s)$ is supposed to be around 1. The closer to 1 $C(M_s)$ is, the better M_s can represent the whole set of mutants M_{ne}. So far, the most widely adopted correlation coefficient function in selective mutation testing are R^2, $Kendall\ \tau_b$, and $Pearson$ [35], [2]. In particular, R^2 [36] is a determination coefficient that is used to evaluate how close the data are to the fitted regression line; $Kendall\ \tau_b$ [36] is a rank correlation coefficient, and can be used to evaluate the concordance of the orderings of $MS(M_s, NT_i)$ and $MS(M_{ne}, NT_i)$[5]; $Pearson$ [37] is a linear correlation coefficient that measures the strength of the linear relationship between $MS(M_s, NT_i)$ and $MS(M_{ne}, NT_i)$. The value range is [0,1] for R^2, and [-1,1] for $Kendall\ \tau_b$ and $Pearson$.

III. EXPERIMENTAL DESIGN

We statistically compare the two types of metrics in terms of their values, and their associations with seven different experimental factors that might have influence on them (see Section III-B).

Our experimental study aims to answer the following three research questions:

RQ1: Are the two types of metrics consistent with each other qualitatively? This research question investigates the consistency of different metrics through observing the associations between different variables (i.e., metrics and experimental factors), presented by figures.

RQ2: Are the two types of metrics consistent with each other quantitatively? This research question investigates the consistency of different metrics through observing the results of regression analysis between different variables.

The research questions check whether the two types of metrics are complementary or substitutable for one another. They are important for researchers: the outcome would indicate whether the previous related work using only one type of metric have serious threats. It would also provide guidelines on future research investigating selective mutation testing on whether both types of metrics should be adopted.

[5] The concordance refers to the fact that if M_s evaluates NT_i to be better than another test set NT_j, M_{ne} will do the same. Thus, high $Kendall\ \tau_b$ may indicate that M_s can differentiate the quality of different test sets as well as M_{ne} does.

Additionally, to reduce the threats caused by mutant selection strategies (random mutant selection by default), we replicate our experiments with another popular mutant selection strategy: operator based mutant selection. These threat studies are presented in Section V.

A. Subjects, Tests, and Mutants

To avoid bias in subject selection, we started from the top 1000 popular Java projects from Github in June of 2015. 712 projects are removed since they are either multi-module or cannot be successfully processed by the tools we used (i.e., Maven[6] and PIT[7]). Furthermore, 184 very small projects are removed, since there is not enough data to perform effective analysis. Specifically, we remove the projects that belong to one of the following three criteria: (1) the program size is smaller than 100 SLOC (i.e., lines of code), (2) the number of tests is smaller than 10, (3) the number of generated mutants is smaller than 200. Finally, the remaining 104 projects are regarded as the target subjects in this study.

The sizes of the remaining projects range from 170 to 77,593 SLOC[8], adding up to 606,886 SLOC. Each subject has a suited test suite, which is collected by the developers during its development. For each subject, we use PIT (i.e., with all its 14 mutation operators[9]) as the base tool to generate mutants, as PIT has been demonstrated to be more robust [39] to enable a large-scale experimental study. More details of the experimental study (including the implementation, the subjects, and experimental data) are available on our homepage [40].

B. Independent Variables

We consider the following independent variables:

IV1: Mutant Selection Metrics. We study two types of mutant selection metrics: (1) adequate-test-based metric, represented as *AMetric*; (2) inadequate-test-based metric, represented as *BMetric*. For *BMetric*, following previous work [35], [2], we use three widely adopted correlation coefficient functions: R^2, $Kendall\ \tau_b$, and $Pearson$. For ease of presentation, we refer to R^2, $Kendall\ \tau_b$, and $Pearson$ based *BMetric* as $BMetric_r$, $BMetric_k$, and $BMetric_p$, respectively.

IV2: Experimental Factors. Based on Formulae (1) and (2), we realize that the two types of mutant selection assessment metrics may be affected by the number of mutants, the total number of tests, and the approach to test suite construction when calculating the metric values. Thus, we check all the possible experimental factors in terms of three aspects: the program scale, the tests, and metric construction. Finally, seven factors are identified, as Table I shows, and are used to investigate their associations with the two types of metrics.

IV3: Ratios of Selected Mutants. Random mutant selection has been shown to be state-of-the-art selective mutation testing

[6] https://maven.apache.org/

[7] PIT is a popular mutation tool for Java programming language, which is accessible at http://pitest.org/.

[8] The number of lines of executable code is reported by *LocMetrics*, http://www.locmetrics.com.

[9] Note that with all the mutation operators, PIT does not have the issue of not subsuming branch coverage [38].

approach [2], [41], [1], [28]. According to previous work [1], [2], for most projects, 1% mutants may have a metric value above 0.8 [1], and 5% mutants can provide precise mutation testing results (i.e., mutation score) [2]. Therefore, in this study, we use five ratios of randomly selected mutants (denoted as r):1%, 2%, 3%, 4%, and 5%.

Additionally, to reduce the threats of validity of our results, we explore the following independent variable in Section V:

IV4: Mutant Selection Strategies. Random mutant selection and operator-based mutant selection are the two dominant mutant selection strategies [2]. In this paper, we mainly investigate the former strategy, and also check if using the latter strategy would yield similar results afterwards.

C. Dependent Variables

We use regression analysis to statistically analyze the associations between the metric values and the experimental factors. The same as previous work on statistics [43], [44], we consider the following two widely adopted dependent variables as a measurement of our regression analysis.

DV1: p-value. In statistics, the p-value [45] is used to test the null hypothesis that the coefficient of the variable of interest is equal to zero (no effect). A low p-value (the 0.05 threshold is typically adopted) indicates that the null hypothesis can be rejected, and thus the responding variable has a non-zero effect on the outcome of the model.

DV2: Adjusted R^2. The *Adjusted* R^2 [46] is widely adopted to measure how well the data fit a statistical model. It is a modified version of R^2, which is adjusted for the number of predictors in the model. The closer *Adjusted* R^2 is to 1, the better the data fit the model.

D. Experimental Procedure

To perform qualitative and quantitative analysis to compare the two types of metrics, we firstly conduct selective mutation testing to collect raw data on the 104 subjects (see Section III-D1) and use figures to present the associations between different sets of data (see Section III-D2). We then perform regression analysis to further confirm the associations. In the end, we replicate our experiments with another popular mutation tool and another mutant selection strategy to investigate the validity of the results. In this section, we present the experimental process of each step. More details can be further referred in Section IV and Section V.

1) Data Collection: First, we conduct mutation testing on all the subjects with the PIT, and collect the set of non-equivalent mutants for each subject. Following the same procedures in the previous work [34], [28], [2], [1], we adopt the mutants that cannot be killed by any test to approximate equivalent mutants, i.e., taking the mutants that can be killed by any test as non-equivalent mutants. More discussion about the impacts of equivalent mutants can be referred to Section IV-B1.

Second, we randomly select mutants from the set of non-equivalent mutants to construct different ratios of selected mutant sets (i.e., $r \in \{1\%, 2\%, 3\%, 4\%, 5\%\}$). To reduce the bias of random selection, for each subject, we randomly construct 50 sets of mutants $M_1, M_2, ..., M_{50}$ for each ratio r following the previous work [2].

Third, for each set of selected mutants $M_i (1 \leqslant i \leqslant 50)$, we construct test sets according to the mutant selection metrics. For *AMetric*, as previous work [2], [1] did, we construct 20 adequate test sets for M_i, each of which is constructed by randomly including one test at a time, until all the mutants in M_i are killed. For *BMetric*, as previous work did [2], we use 100 different test sets each of which is constructed by randomly selecting tests from the original test suite. Thus, the size of each test set (i.e., number of tests in a test set) is a random number between 1 and the total number of tests for the subject. Note that when constructing the adequate test sets, the newly added tests are not guaranteed to kill additional mutants that were not killed by the existing tests, since in practice developers construct such test suites without knowing whether each test kills additional mutants ahead of time.

Fourth, for each set of selected mutants, we calculate the metric values according to Formulae (1) and (2). Following the previous work [28], [2], [1], [29], we calculate the average results for *AMetric* based on the 50 mutant sets selected with specific r each of which has 20 adequate test sets and the average results for *BMetric* based on the 50 mutant sets selected with specific r each of which has 100 non-adequate test sets by using each of the three correlation values (i.e., R^2, $Kendall\ \tau_b$, and $Pearson$).

Through the preceding procedure, we collect raw data for *AMetric* and *BMetric*, which serve as the object of the following statistical analysis.

2) Qualitative Analysis: To analyze the data we collected, we use figures to present the metric values that belong to the same proportion of selected mutants[10], and observe the consistency.

Next, to compare the two metrics' associations with the seven experimental factors, we plot figures between each metric and each experimental factor to observe the general association patterns.

3) Quantitative Analysis: To further confirm the observations and conclusions drawn from the data collected qualitatively, we perform simple and multiple regression analysis between different metrics and between metrics and different experimental factors. If the association is non-linear, we transform the factors into other forms (e.g., its logarithmic value) using curve fitting so that linear models still fit [47]. In particular, we perform simple regression analysis to test the simple hypothesis of association [48], i.e., how an experimental factor is associated with a metric.

4) Experiments Replication: Lastly, to check whether mutant selection strategies are threats to the observed conclusions, we reproduce the results with different mutant selection strategies (random and operator-based mutant selection). We then check if we can get similar conclusions. Positive results

[10] The values of the two metrics for each proportion are calculated based on identical mutant subsets.

would reveal that our results and conclusions are reliable on different mutation tools and different mutant selection strategies.

IV. Results and Analysis

In this section, we present our results and analysis of the qualitative analysis (in Section IV-A) and quantitative analysis (in Section IV-B). For each kind of analysis, we introduce the results of direct comparison between different metric values and the results of factor association comparison respectively. The investigation about whether mutation tools or mutant selection strategy would yield similar results is presented not in this section but in Section V.

It is worth mentioning that both types of metrics are proposed to measure the representativeness of the selected mutants, it is interesting to check the consistency of their measurement results. In our study, all the values of *AMetric* and *BMetric* are collected based on identical selected mutants, and thus are comparable.

A. Qualitative Analysis

We introduce the associations between different metrics in Section IV-A1, and the associations between the metrics and some experimental factors in Section IV-A2.

1) Qualitative Direct Comparison: Based on the experimental procedure described in Section III-D, for each project we would get the values of different metrics under each ratio r (i.e., $r \in \{1\%, 2\%, 3\%, 4\%, 5\%\}$) of selected mutants. When performing direct comparison, we then plot these metric values and observe the associations between the two types of metrics. In particular, as *BMetric* has three types (i.e., $BMetric_r$, $BMetric_k$, $BMetric_p$), three figures are presented, with the *AMetric* as the horizontal axis, and each of the *BMetric* as the vertical axis.

Figure 1 presents the qualitative comparison results. The black line in each figure represents the function $y = x$. The first sub-figure shows the association between *AMetric* and $BMetric_r$. From this sub-figure, there is a lot of space between the colorful lines and the black line, and the space becomes less as the values of *AMetric* increase. Additionally, there is no obvious difference between different colorful lines. Thus, the values of *AMetric* are larger than $BMetric_r$; when the values of *AMetric* increases, *AMetric* and $BMetric_r$ become closer; the ratios of selected mutants have no influence on the relations between the two metrics. The second sub-figure shows the association between *AMetric* and $BMetric_k$. From this sub-figure, the correlation between the values of *AMetric* and $BMetric_k$ is similar to that in the first sub-figure. Besides, the values of *AMetric* also tend to be larger than $BMetric_k$. The third sub-figure shows the association between *AMetric* and $BMetric_p$. From this sub-figure, the values of *AMetric* and $BMetric_p$ are the closest among all the three sub-figures. Additionally, the same as the first two sub-figures, there is no obvious difference between the colorful lines, demonstrating that both types of metrics are well designed and are stable for different selective mutation testing techniques. Thus, the values of *AMetric* tend to be very close to $BMetric_p$ for the five ratios of selected mutants.

From Figure 1, there is obvious qualitative assosiation between *AMetric* and $BMetric_r$/$BMetric_p$/$BMetric_k$. We further present the quantitative analysis results between all the metrics in Section IV-B.

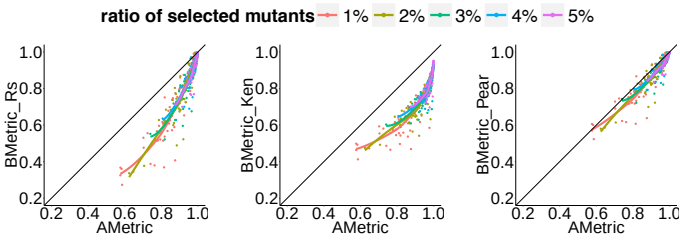

Figure 1. Comparison between metric values. The horizontal and vertical axises are different metrics. The black line in each sub-figure represents the function $y = x$. Each colorful line represents the smoothed conditional mean values for some ratio of selected mutants. The closer these colorful lines are to the black line, the closer the values of the horizontal axis are to those of the vertical axis. From this figure, the metric values are not necessarily equal to each other, but have obvious associations.

2) Qualitative Factor Comparison: We plot figures between each metric and each experimental factor to observe the general association patterns.

In practice, selective mutation testing is often applied with certain experimental settings, and thus it is crucial to compare different metrics in terms of the impacts of different experimental factors on them. For example, previous work [1] found that when using *AMetric* as the mutant selection assessment criterion, larger projects tend to require smaller proportion of selected mutants, while it is still unknown whether such pattern still exists for *BMetric*, or whether such pattern exists for other experimental factors.

Based on the experimental procedure described in Section III-D, for each project we would get the values of different metrics as well as the seven experimental factors under each mutant selection ratio. To further compare the two types of metrics on their associations with different factors and search for implications for practical selective mutation testing, we then plot each metric value and each experimental factor to observe and compare their associations. Because there are four metrics and seven experiemntal factors, we present 4*7 = 28 (sub)figures all together. For each figure, we treat the factor value as the horizontal axis, and the metric value as the vertical axis.

Figure 2 presents the results. Each sub-figure includes one experimental factor. From these figures, for the two types of metrics, the shapes of the lines in different sub-figures are similar, indicating that the two types of metrics have similar associations with the seven factors. For all the metrics, the lines of code, the number of non-equivalent mutants, the test number, and size of adequate tests show non-linear associations with the metric values, and may impact the metric values in a great deal. Specifically, among different projects, when the values of these factors increase, the metric values (of

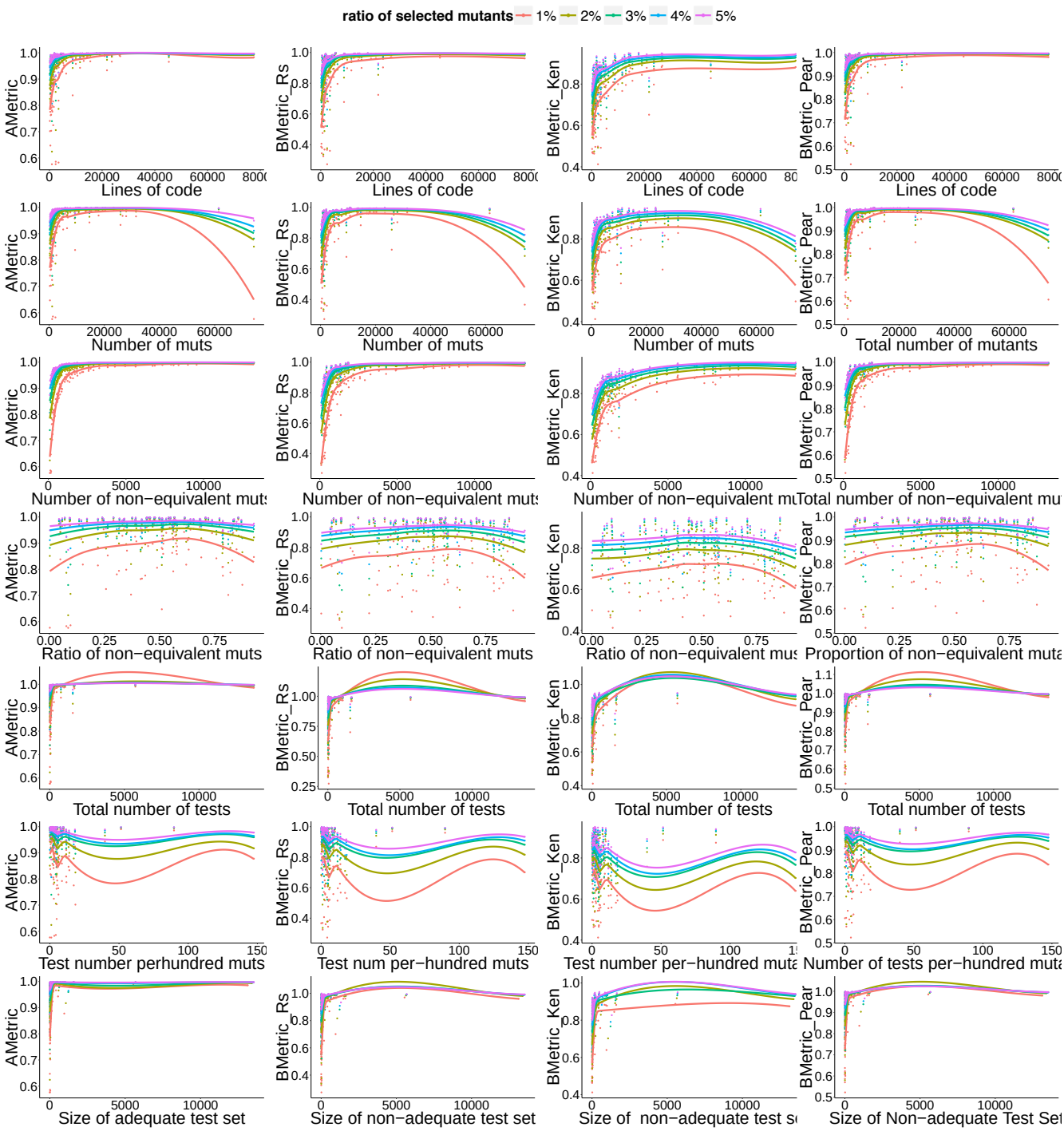

Figure 2. Associations between the seven factors and metrics. The left column presents the results of *AMetric*, and the remaining columns present the results of *BMetric*. $BMetric\_Rs$, $BMetric\_Ken$, $BMetric\_Pear$ represent $BMetric_r$, $BMetric_k$, and $BMetric_p$ seperately. In each sub-figure, the horizontal axis represents the value of each factor; the vertical axis represents the value of *AMetric* or *BMetric*. Different colors represent the five ratios of selected mutants. Each point in the figure represents the corresponding metric value for each experimental factor, and the lines represent the smoothed conditional mean values of points.

the same ratio of selected mutants) would increase rapidly at first, then begin to flatten, and finally reach a stable value that is close to 1.0. For example, consistent with previous work [1], [32], a program with more mutants tend to have higher metric values under the same ratio of mutant selection.

The findings above indicate that developers should adjust the proportion of selected mutants according to the sizes of different projects: smaller projects may need larger proportion of mutants to have a good representativeness.

Also, our results firstly show that no metric has clear changing trend with the growth of test density or proportion of non-equivalent mutants, indicating that these two factors do not have clear impact on the metric values. For *BMetric*, as the average size of the non-adequate test sets is nearly half

of the size of the total tests, the association between the size of non-adequate test sets and the metric is very similar to the association between the size of the total tests and the metric values. More details for the analysis of the impacts of different experimental factors can be referred to Section IV-B1.

Based on our observations from the above qualitative analysis in dicrect value comparison and factor association comparison, we have the following conclusion:

Finding 1: The adequate-test-based metric (i.e., *AMetric*) and the inadequate-test-based metrics (i.e., $BMetric_r$, $BMetric_k$, and $BMetric_p$) are consistent with each other qualitatively.

B. Quantitative Analysis

From previous results presented in Section IV-A and Section IV-A2, there are associations between different metrics as well as between the metrics and some experimental factors. In this section, we further explore the associations through regression analysis.

Table II
TRANSFORMATION RULES FOR METRIC VALUES. Y AND t REPRESENT THE VALUE OF THE TWO TARGET METRICS. b_0, b_1, AND b_2 ARE THE FITTED COEFFICIENTS.

Curve Fitting Models	Equation
Logarithmic	$Y = b_0 + (b_1 * ln(t))$
Inverse	$Y = b_0 + (b_1/t)$
Power	$Y = b_0 * (t^{b_1})$
Quadratic	$Y = b_0 + (b_1 * t) + (b_2 * t^2)$
Exponential	$Y = b_0 * e^{(b_1 * t)}$

1) Quantitative Factor Comparison: From Figure 2, we observe that: 1) The two types of metrics have similar patterns towards their associations with factor values; 2) Some factors, such as the number of non-equivalent mutants and the total lines of code, have obvious non-linear association with the two types of metrics. In this section, to quantitatively measure and compare the association between each factor and the metric values, we again use curve fitting, and check the *Adjusted* R^2 of the fitted models.

In particular, for each factor, we do curve fitting based on the five curve estimation models introduced in Table II. Each factor has five transformations under each ratio r on each metric. Thus, we perform $5*7*5*4 = 700$ regression analysis.

The same as previous related work [49], [50], we divide the correlation degree between the metric values and the factors into three categories according to the values of *Adjusted* R^2: (1) strong correlation: *Adjusted* $R^2 \geqslant 50\%$; (2) moderate correlation: $25\% \leqslant$ *Adjusted* $R^2 < 50\%$; (3) weak correlation: *Adjusted* $R^2 < 25\%$.

The analysis results are shown in Table III. The last seven columns show the *Adjusted* R^2 values of the fitted model between each factor and the corresponding metric. For example, when analyzing the association between the lines of code ($SLOC$) and *AMetric*, the *Adjusted* R^2 is 0.3271 when using curve estimation model $AMetric x = b_0 + b_1 * ln(SLOC)$, while is 0.2399 when using curve estimation model $AMetric = b_0 + 1/SLOC$. Different colors in the table represent different degrees of association. Darker color represents stronger correlation.

Table III
BIVARIATE REGRESSION RESULTS.

r	Metric	IV	Experimental factors						
			$SLOC$	M	M_{ne}	R_{ne}	T	T_d	T_c
1%	*AMetric*	lnx	0.31	0.28	0.57	0.04	0.22	-0.01	0.32
		x	0.13	0.02	0.30	0.00	0.03	-0.01	0.03
		$-1/x$	0.23	0.27	-0.01	0.09	0.17	0.04	0.31
		$x + x^2$	0.22	0.35	0.47	0.04	0.08	-0.02	0.08
	$BMetric_r$	lnx	0.43	0.39	0.58	0.00	0.27	0.00	0.27
		x	0.18	0.06	0.36	-0.01	0.04	-0.01	0.04
		$-1/x$	0.34	0.32	-0.01	0.03	0.22	0.00	0.23
		$x + x^2$	0.29	0.39	0.53	0.01	0.10	-0.02	0.10
	$BMetric_k$	lnx	0.44	0.38	0.52	-0.01	0.30	-0.01	0.30
		x	0.22	0.09	0.45	-0.01	0.06	-0.01	0.06
		$-1/x$	0.30	0.25	0.00	0.01	0.21	0.00	0.22
		$x + x^2$	0.33	0.38	0.57	0.01	0.14	-0.02	0.14
	$BMetric_p$	lnx	0.40	0.36	0.56	0.00	0.25	0.00	0.25
		x	0.16	0.05	0.33	-0.01	0.03	-0.01	0.03
		$-1/x$	0.33	0.31	-0.01	0.04	0.22	0.01	0.23
		$x + x^2$	0.26	0.36	0.49	0.01	0.09	-0.02	0.09
2%	*AMetric*	lnx	0.27	0.28	0.51	0.01	0.18	0.00	0.25
		x	0.11	0.04	0.24	0.00	0.018	-0.01	0.02
		$-1/x$	0.21	0.26	-0.01	0.01	0.18	0.00	0.29
		$x + x^2$	0.18	0.25	0.39	0.03	0.05	-0.02	0.06
	$BMetric_r$	lnx	0.36	0.35	0.51	0.00	0.23	0.00	0.23
		x	0.14	0.07	0.28	-0.01	0.03	-0.01	0.03
		$-1/x$	0.30	0.28	-0.01	0.01	0.22	-0.01	0.22
		$x + x^2$	0.23	0.30	0.43	0.01	0.07	-0.02	0.07
	$BMetric_k$	lnx	0.42	0.39	0.50	-0.01	0.29	-0.01	0.29
		x	0.20	0.12	0.41	-0.01	0.05	-0.01	0.05
		$-1/x$	0.29	0.25	0.00	-0.01	0.23	-0.01	0.23
		$x + x^2$	0.31	0.35	0.53	0.00	0.12	-0.02	0.12
	$BMetric_p$	lnx	0.32	0.31	0.49	0.00	0.21	0.00	0.21
		x	0.12	0.06	0.25	-0.01	0.02	-0.01	0.02
		$-1/x$	0.27	0.25	-0.01	0.00	0.22	-0.01	0.22
		$x + x^2$	0.21	0.27	0.40	0.01	0.06	-0.02	0.06
3%	*AMetric*	lnx	0.26	0.26	0.50	0.02	0.19	-0.01	0.23
		x	0.01	0.03	0.22	0.00	0.02	-0.01	0.02
		$-1/x$	0.20	0.26	-0.01	0.03	0.19	0.01	0.29
		$x + x^2$	0.17	0.25	0.37	0.03	0.05	-0.02	0.05
	$BMetric_r$	lnx	0.34	0.32	0.51	0.00	0.23	-0.01	0.22
		x	0.13	0.05	0.27	-0.01	0.03	-0.01	0.03
		$-1/x$	0.30	0.28	-0.01	0.01	0.23	0.00	0.23
		$x + x^2$	0.22	0.29	0.42	0.01	0.07	-0.02	0.07
	$BMetric_k$	lnx	0.43	0.40	0.51	-0.01	0.30	-0.01	0.30
		x	0.21	0.11	0.42	-0.01	0.06	-0.01	0.06
		$-1/x$	0.30	0.27	0.00	0.00	0.22	-0.01	0.22
		$x + x^2$	0.32	0.36	0.55	0.00	0.13	-0.02	0.13
	$BMetric_p$	lnx	0.32	0.30	0.49	0.00	0.22	-0.01	0.21
		x	0.12	0.05	0.25	-0.01	0.02	-0.01	0.02
		$-1/x$	0.28	0.27	-0.01	0.01	0.23	0.00	0.23
		$x + x^2$	0.20	0.27	0.39	0.01	0.06	-0.02	0.06
4%	*AMetric*	lnx	0.27	0.26	0.49	0.02	0.17	0.00	0.20
		x	0.10	0.02	0.22	-0.01	0.02	-0.01	0.02
		$-1/x$	0.20	0.27	-0.01	0.04	0.17	0.01	0.24
		$x + x^2$	0.17	0.26	0.36	0.03	0.05	-0.02	0.05
	$BMetric_r$	lnx	0.33	0.30	0.50	0.01	0.22	-0.01	0.22
		x	0.13	0.05	0.26	-0.01	0.02	-0.01	0.02
		$-1/x$	0.27	0.26	-0.01	0.02	0.21	0.00	0.21
		$x + x^2$	0.21	0.29	0.41	0.01	0.07	-0.02	0.07
	$BMetric_k$	lnx	0.43	0.39	0.51	-0.01	0.31	-0.01	0.30
		x	0.20	0.10	0.43	-0.01	0.06	-0.01	0.06
		$-1/x$	0.30	0.26	0.00	0.00	0.23	-0.01	0.23
		$x + x^2$	0.32	0.36	0.56	0.01	0.13	-0.02	0.13
	$BMetric_p$	lnx	0.31	0.29	0.49	0.01	0.21	-0.01	0.21
		x	0.12	0.04	0.25	-0.01	0.02	-0.01	0.02
		$-1/x$	0.26	0.25	-0.01	0.02	0.21	0.00	0.21
		$x + x^2$	0.20	0.27	0.39	0.01	0.06	-0.02	0.06
5%	*AMetric*	lnx	0.29	0.30	0.51	0.01	0.18	0.00	0.20
		x	0.11	0.05	0.24	-0.01	0.02	-0.01	0.02
		$-1/x$	0.22	0.30	-0.01	0.01	0.16	0.00	0.20
		$x + x^2$	0.19	0.26	0.39	0.03	0.05	-0.02	0.05
	$BMetric_r$	lnx	0.33	0.30	0.50	0.01	0.22	-0.01	0.22
		x	0.12	0.04	0.26	-0.01	0.02	-0.01	0.02
		$-1/x$	0.29	0.28	-0.01	0.02	0.22	0.00	0.23
		$x + x^2$	0.20	0.27	0.40	0.01	0.07	-0.02	0.07
	$BMetric_k$	lnx	0.45	0.40	0.51	-0.01	0.31	-0.01	0.31
		x	0.21	0.11	0.43	-0.01	0.06	-0.01	0.06
		$-1/x$	0.32	0.28	0.00	0.00	0.23	-0.01	0.23
		$x + x^2$	0.32	0.36	0.56	0.01	0.14	-0.02	0.14
	$BMetric_p$	lnx	0.32	0.29	0.49	0.01	0.21	-0.01	0.21
		x	0.12	0.04	0.25	-0.01	0.02	-0.01	0.02
		$-1/x$	0.28	0.27	-0.01	0.02	0.22	0.00	0.23
		$x + x^2$	0.19	0.26	0.39	0.01	0.06	-0.02	0.06

Next, we statistically compare the two types of metrics from their curve fitting results, and analyze some factors from the aspect of the associations between metrics and their values.

Factor comparison. From table III, when we make vertical comparison between the *Adjusted* R^2 of different metrics, we

can observe a lot of similarity from several aspects. First, different curve estimation models have similar performance rankings on the two types of metrics. For example, from the color of different cells, the first three models are better than the remaining two models no matter for which metric and which selective mutation testing technique. Second, each curve estimation model has similar performance on different metrics. For example, when choosing 1% mutants, model $y = b_0 + b_1 ln(x)$ for factor M_{ne} has *Adjusted* R^2 values of 0.7552, 0.7857, 0.8185, 0.7531 for each metric separately, which are very close.

When we make horizontal comparison between the *Adjusted* R^2 with different factors, we can also observer a lot of similarity. For example, for each metric we can sort the correlated factors in the descendent order of correlation degrees: the number of non-equivalent mutants > the total number of mutants > the lines of code > the number of constructed tests > the total number of tests.

The similarity of fitting results between different metrics further confirms our initial finding through qualitative analysis, i.e., the two types of metrics have similar patterns regarding their associations with the seven experimental factors.

Specific factor analysis. All the five related factors mentioned above (i.e., the number of non-equivalent mutants, the total number of mutants, the lines of code, the number of constructed tests, the total number of tests) are direct measurements of programs, mutants, and tests. The measurements of programs and mutants (i.e., $SLOC$, M, and M_{ne}) have impacts on the mutant selection values because they are directly related to the number of mutants under selection. The fact that the number of non-equivalent mutants has the strongest correlation with metric values confirm our reasoning, since the set of non-equivalent mutants are the base mutants under selection. The direct measurements of tests (i.e., T and T_c) also have impacts on the mutant selection metrics. There are two guesses. First, the number of tests have impacts simply because projects with more tests tend to have larger size and thus have more base mutants under selection. Second, the number of tests have impacts even when projects have similar size, i.e., due to test density. However, also consistent with the qualitative observations, our results demonstrate that test density T_d does not have clear impact on metric values. Therefore, the direct measurements of tests have impacts on metric values may simply due to their correlation with the number of base mutants under selection.

We further focus on the remaining factor, R_{ne} (i.e., the proportion of non-equivalent mutants). As prior work did [34], [28], [2], [1], we take the mutants that cannot be killed by the original test suite as equivalent mutants, which may actually include some non-equivalent mutants that are not killed by the original test suites. That is, this traditional equivalent mutant handling approach may induce some threat to the research in mutation testing, including our own study. We realize that the projects with high-quality test suites would have a very small number of such missing non-equivalent mutants and usually have large R_{ne} due to its high-quality test suites. Therefore, projects with higher R_{ne} may suffer less from the threat to validity, i.e., R_{ne} can actually be used to reflect the bias resulting from such equivalent mutant handling. Surprisingly, according to Table III and Figure 2, all the metric values have negligible correlation with factor R_{ne}, indicating that the fault-revealing ability of original test suites does not have impacts on mutant selection. That is, strategy for dealing with equivalent mutants in selective mutation testing (including our own work) is observed not to be a threat to validity.

Finding 2: We do not observe clearn correlation between the proportion of non-equivalent mutants and the metric values, indicating that strategies for dealing with equivalent mutants barely impact mutant selection results.

Although the metrics are consistent with each other in values and the association patterns with the seven experimental factors, minor differences can be observed: compared with *AMetric*, $BMetric_r$, and $BMetric_p$, $BMetric_k$ shows a slightly different association pattern. To illustrate, in Figure 2, the subfigures related to the $Kendall\ \tau_b$ usually have more divergent data points. Additionally, from these figures, as the lines of code and the number of non-equivalent mutants increase, *AMetric*, $BMetric_r$ and $BMetric_p$ increase dramatically at first, then become steady and approach 100%, while $BMetric_k$ keeps increasing all the time, and is only around 0.9 even for the largest projects. Note that R^2 and $Pearson$ can reflect the linear association between two test-evaluation techniques (i.e., using selected mutants or using all mutants), while $Kendall\ \tau_b$ can reflect whether two test-evaluation techniques perform equally in differentiating the quality of different test sets [35]. Thus, in selective mutation testing, the selected mutants may have high linear association with the whole set of mutants in evaluating the quality of test sets, but are slightly less stable in differentiating the fault-revealing ability of different test sets.

Based on the above observations from the quantitative analysis including direct value comparison and factor association comparison, we have the following conclusion:

Finding 3: The adequate-test-based metric (i.e., *AMetric*) and the inadequate-test-based metrics (i.e., $BMetric_r$, $BMetric_k$, and $BMetric_p$) are consistent with each other quantitatively.

V. Extended Analysis

In this section, we explore whether mutant selection strategies would yield different results and conclusions.

We then try to reproduce our experimental results with operator-based mutant selection strategy. Similar with previous work [2], we choose mutants generated by four mutation operators of PIT: *InlineConstantMutator*, *MathMutator*, *VoidMethodCallMutator*, *NegateConditionalsMutator*. Because some projects do not have mutants generated from one or several of these four operators, we remove them from the total subjects.

Finally 87 projects are left, and thus our results are based on these 87 projects.

1) Metric Value Comparison: Figure 3 lists the metric comparison results of operator-based mutants selection strategy, which shows similar patterns as Figure 1, and thus we get the same conclusions as we did for random mutant selection: *AMetric* is very close to $BMetric_p$, and has obvious correlation with $BMetric_r$ and $BMetric_k$.

At the same time, we observe minor differences between the comparison results for random and operator-based mutant selection. In particular, both *AMetric* and *BMetric* have higher values in operator-based mutant selection (most values are above 0.7), mainly because operator-based mutant selection using the four mutation operators selects more mutants than random mutant selection which selects 1% to 5% mutants in our experiments.

2) Association Analysis: We also compare the comparison results in terms of the associations between metric values and the seven experimental factors. Similarly, we present the association patterns between the number of non-equivalent mutants and the metric values, shown in Figure 4. Comparing with Figure 4, the patterns are similar as in random mutant selection. For $BMetric_r$, the smooth line of operator-based mutant selection is less smoothy than random mutant selection. We suspect the reason to be that in operator-based mutant selection, the number of subjects is smaller than in random mutant selection (i.e., 87 vs 104).

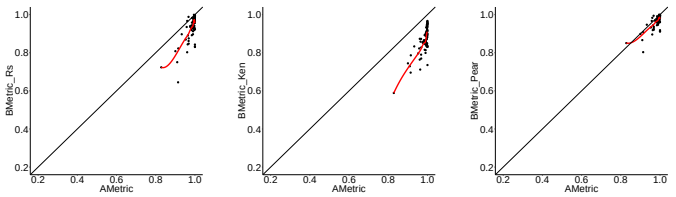

Figure 3. Metric values comparison for operator-based mutant selection, which shows very similar patterns to Figure 1.

In summary, the comparison results above indicate that mutant selection strategy does not affect our correlation and comparison results on *AMetric* and *BMetric*.

> **Finding 7:** Mutant selection strategies do not affect our correlation and comparison results on *AMetric* and *BMetric* based on our observation.

A. Threats to Validity

The threat to internal validity lies in the code implementation of the empirical study. To reduce the possible faults in implementation, the first two authors reviewed the code and experimental scripts of the empirical study carefully.

The threats to external validity mainly lie in the subjects, tests, and mutants. To reduce the threat from subjects, we collected and used a large number of Java projects from open-source community, but they may be not representative for other programs, especially programs in other languages. To reduce the threat from tests, following previous work [2], [1], we use the original test suites collected during software development. These threats can be further reduced by repeating our study on more projects in various programming languages (e.g., C, C++, C#, and Python), with different types of tests, as well as using more mutation testing tools.

The threats to construct validity lie in how we measure the influence of the possible factors and the number of sampling points in regression analysis. To reduce this threat, we used two types of regression analysis (i.e., bivariate regression and multiple regression) and two widely used measurements in statistics (i.e., p-value and *Adjusted* R^2). Also, we chose the number of points similarly as the previous work did [35], [2] to reduce the corresponding threat.

VI. Related Work

First proposed by DeMillo et al. [6] and Hamlet [7], mutation testing is drawing more and more attention in both the academia and industry. Jia and Harman [53] provided a comprehensive survey for mutation testing. Due to the high cost of mutation testing, selective mutation testing has been widely studied and even applied to recent applications of mutation testing. In this section we first introduce the related work on reducing the cost of mutation testing. Then, we further discuss the application of selective mutation testing in the main applications of mutation testing.

A. Cost Reduction Techniques

As it is usually very expensive to execute tests on each mutant, many researchers focus on designing various techniques to reduce the cost of mutation testing. In particular, we divide these techniques into two categories: techniques on reducing the number of mutants in mutation testing (i.e., selective mutation testing) and techniques on reducing the compilation/execution time of mutants (denoted as optimized mutation testing in this work).

1) Selective Mutation Testing: To reduce the number of mutants in mutation testing, selective mutation testing is proposed to select a subset of mutants to represent all the mutants. The techniques in selective mutation testing are mainly classified into operator-based mutant selection [54], [55], [33], [27], [34], [31], [56], [57] and random mutant selection [58], [55], [28], [54].

Operator-based mutant selection aims to reduce the number of mutants by carefully choosing mutation operators before generating mutants. In particular, Offutt et al. [33] proposed to use five *sufficient mutation operators* to generate mutants that achieve almost the same effectiveness as the mutants generated by all mutation operators. Following their work, many researchers focused on detecting *sufficient mutation operators*. In particular, Barbosa et al. [27] presented 6 guidelines to determine sufficient mutation operators, with which they finally determined 10 sufficient mutation operators for C programs. Gligoric et al. [31] studied the sufficient mutation operators for concurrent programs. Furthermore, Namin et al. [34] proposed

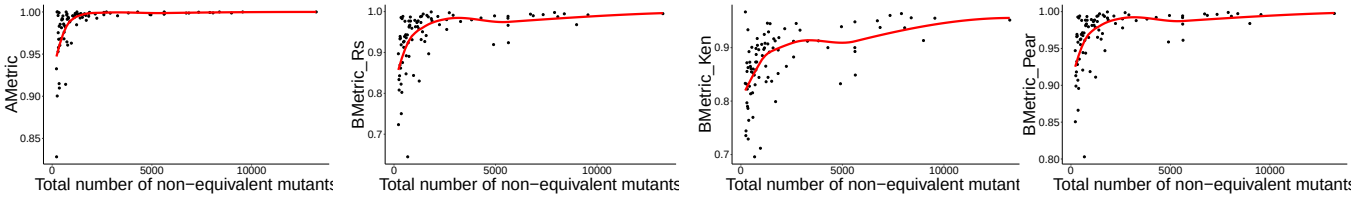

Figure 4. Associations between the number of non-equivalent mutants and the metric values for operator-based mutant selection. The patterns are similar patterns to those shown in Figure 1.

to combine the execution information of a subset of mutants, and identified 28 sufficient mutation operators based on their result analysis.

Random mutant selection aims to randomly select mutants from the whole set of mutants, which are generated by all the mutation operators. After Acree et al. [58] proposed the first random mutant selection technique, Mathur and Wong [54], [55] empirically studied this technique by randomly selecting x% mutants, which are generated by the 22 mutation operators in Mothra [59]. Although random mutant selection seems trivial, it has been demonstrated to be at least as effective as operator-based mutant selection [28].

Most of these work in selective mutation testing focuses on presenting various selection techniques. They simply directly used one or two of the selection metrics without much consideration. In this work, we studied how different factors impact the metrics used in those work, and how different metrics compare and correlate with each other.

2) Optimized Mutation Testing: Another way of reducing the cost of mutation testing is to reduce the compilation or execution time of mutants. To reduce the compilation time of mutants, Demillo et al. [60] changed a compiler to compile all mutants in the same pass to save mutant generation and compilation time. Later on, Untch et al. [61] proposed the schema-based mutation testing to encode all mutants into one meta-mutant. Note that mutant compilation time reduction is not widely studied recently, since (1) the compilation time can be negligible compared with mutant execution time; (2) modern mutation tools mainly operate on the bytecode, assembly code, or intermediate representation levels.

To reduce the execution time of a mutant, Howden [62] proposed the concept of *weak mutation testing*, which treats a mutant as being weakly killed when the mutant causes any internal state change. In this way, the mutants do not need to be executed completely, and thus test execution time can be saved. Later, Woodward and Halewood [63] proposed *firm mutation testing*, which is a compromise of weak mutation testing and traditional mutation testing. Offutt and Lee [64] conducted an empirical study on weak mutation testing and found that weak mutation testing can be better applied in unit testing of non-critical applications. Krauser et al. [65] and Offutt et al. [66] ran mutants in parallel to speed up mutation testing. Zhang et al. [67] proposed to prioritize and reduce tests to speed up mutation testing. To facilitate mutation testing for evolving programs, Zhang et al. [68] also proposed to speed up mutation testing for the current program by reusing the mutant execution results of the previous version. Recently, Just et al. [52] analyzed the infected states of mutant execution to speed up mutant execution by 40% on average.

B. Applications of Mutation Testing

Although mutation testing was originally proposed for evaluating test suites, it is gaining more and more applications recently. Many researchers have used mutation testing to provide guidance in test generation to automatically generate high-quality tests [69], [70], [71], [72], [21], [73], [19], [13]. Since real faults can be hard to find and small in number, mutation testing is also widely used to simulate faults in the evaluation of software testing techniques [74], [75], [76], [77]. Researchers have demonstrated that mutants can be used as valid substitute for real faults in software testing experimentation [74], [18], [17]. More recently, mutation testing has also been used to help with program debugging. A number of mutation-based fault localization techniques [22], [78], [79], [80] have been proposed to use mutation faults to localize real faults based on their similarity. Debroy and Wong also utilized mutation testing to suggest potential fixes for buggy programs [25]. Zhang et.al. [81] used program variants, which can be regarded as mutants, to make the current tests execute more paths and help to detect more software bugs.

Effective selective mutation testing is also crucial for almost all the new applications of mutation testing. For example, using all mutant-killing contraints to guide test generation may choke the underlying test generation solver or engine, and using all mutants to help fault localization can be expensive. In fact, a number of previous work already applied selective mutation testing to the new applications of mutation testing, e.g., test generation [73] and fault localization [22]. Therefore, our study on the mutant selection metrics can also better guide those new applications of mutation testing.

Besides the related work above, Papadakis et.al. [82] found that the presence of subsumed mutants can be a threat to mutation testing. In our work we are investigating the effect of metrics that are defined, by previous work, in terms of all mutants, and as such, include all subsuming mutants and the mutants they subsume. This means we are not at liberty to pick and choose the subsuming mutants we might want to include when assessing the metric. However, future work might investigate whether the results could change when considering only subsuming mutants.

VII. Conclusion

In this paper, we presented the first study to compare two types of mutant selection assessment metrics. The experimental results on 104 GitHub projects provide the first scientific justification of the strong connection between the exisiting two metrics in measuring the representativeness of a mutant subset. We also found that the strategy for dealing with equivalent mutants and test density have negligible impact for mutant selection.

References

[1] J. Zhang, M. Zhu, D. Hao, and L. Zhang, "An empirical study on the scalability of selective mutation testing," in *Proc. ISSRE*, 2014, pp. 277–287.

[2] L. Zhang, M. Gligoric, D. Marinov, and S. Khurshid, "Operator-based and random mutant selection: Better together," in *Proc. ASE*, 2013, pp. 92–102.

[3] M. Dhok and M. K. Ramanathan, "Directed test generation to detect loop inefficiencies," in *Proc. FSE*. ACM, 2016, pp. 895–907.

[4] M. Ermuth and M. Pradel, "Monkey see, monkey do: Effective generation of gui tests with inferred macro events," in *Proc. ISSTA*, 2016.

[5] S. H. Jensen, S. Thummalapenta, S. Sinha, and S. Chandra, "Test generation from business rules," in *Proc. ICST*. IEEE, 2015, pp. 1–10.

[6] R. A. DeMillo, R. J. Lipton, and F. G. Sayward, "Hints on test data selection: Help for the practicing programmer," *Computer*, vol. 11, no. 4, pp. 34–41, 1978.

[7] R. G. Hamlet, "Testing programs with the aid of a compiler," *IEEE Transactions on Software Engineering*, no. 4, pp. 279–290, 1977.

[8] S. Mirshokraie, A. Mesbah, and K. Pattabiraman, "Guided mutation testing for javascript web applications," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 429–444, 2015.

[9] M. Kusano and C. Wang, "Ccmutator: A mutation generator for concurrency constructs in multithreaded c/c++ applications," in *Proc. ASE*. IEEE, 2013, pp. 722–725.

[10] H. Coles, T. Laurent, C. Henard, M. Papadakis, and A. Ventresque, "Pit: a practical mutation testing tool for Java," in *Proc. ISSTA*. ACM, 2016, pp. 449–452.

[11] I. Ahmed, C. Jensen, A. Groce, and P. E. McKenney, "Applying mutation analysis on kernel test suites: An experience report," in *Proc. ICSTW*. IEEE, 2017, pp. 110–115.

[12] A. Groce, I. Ahmed, C. Jensen, and P. E. McKenney, "How verified is my code? falsification-driven verification (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov 2015, pp. 737–748.

[13] M. Harman, Y. Jia, and W. B. Langdon, "Strong higher order mutation-based test data generation," in *Proc. FSE*, 2011, pp. 212–222.

[14] A. P. Mathur, "Performance, effectiveness, and reliability issues in software testing," in *Proc. COMPSAC*, 1991, pp. 604–605.

[15] R. Just, F. Schweiggert, and G. M. Kapfhammer, "Major: An efficient and extensible tool for mutation analysis in a java compiler," in *Proc. ASE*, 2011, pp. 612–615.

[16] D. Schuler and A. Zeller, "Javalanche:efficient mutation testing for java," in *Proc. FSE*, 2009, pp. 297–298.

[17] R. Just, D. Jalali, L. Inozemtseva, M. D. Ernst, R. Holmes, and G. Fraser, "Are mutants a valid substitute for real faults in software testing," in *Proc. FSE*, 2014, pp. 654–665.

[18] J. H. Andrews, L. C. Briand, and Y. Labiche, "Is mutation an appropriate tool for testing experiments?" in *Proc. ICSE*, 2005, pp. 402–411.

[19] G. Fraser and A. Zeller, "Mutation-driven generation of unit tests and oracles," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 278–292, 2012.

[20] L. Zhang, T. Xie, L. Zhang, N. Tillmann, J. de Halleux, and H. Mei, "Test generation via dynamic symbolic execution for mutation testing," in *Proc. ICSM*, 2010, pp. 1–10.

[21] M. Papadakis, N. Malevris, and M. Kallia, "Towards automating the generation of mutation tests," in *Proc. AST*, 2010, pp. 111–118.

[22] M. Papadakis and Y. L. Traon, "Effective fault localization via mutation analysis: A selective mutation approach," in *Proc. SAC*, 2014, pp. 1293–1300.

[23] Y. Lou, D. Hao, and L. Zhang, "Mutation-based test-case prioritization in software evolution," in *Proc. ISSRE*, 2015, pp. 46–57.

[24] Y. Lu, Y. Lou, S. Cheng, L. Zhang, D. Hao, Y. Zhou, and L. Zhang, "How does regression test prioritization perform in real-world software evolution?" in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 535–546.

[25] D. Vidroha and W. W. Eric, "Using mutation to automatically suggest fixes for faulty programs," in *Proc. ICST*, 2010, pp. 65–74.

[26] A. J. Offutt, G. Rothermel, and C. Zapf, "An experimental evaluation of selective mutation," in *Proc. ICSE*, 1993, pp. 100–107.

[27] E. F. Barbosa, J. C. Maldonado, and A. M. R. Vincenzi, "Toward the determination of sufficient mutant operators for C," *Software: Testing, Verification and Reliability*, vol. 11, no. 2, pp. 113–136, 2001.

[28] L. Zhang, S.-S. Hou, J.-J. Hu, T. Xie, and H. Mei, "Is operator-based mutant selection superior to random mutant selection?" in *Proc. ICSE*, 2010, pp. 435–444.

[29] R. Gopinath, A. Alipour, I. Ahmed, C. Jensen, and A. Groce, "How hard does mutation analysis have to be, anyway?" in *Proc. ISSRE*, 2015, pp. 216–227.

[30] E. S. Mresa and L. Bottaci, "Efficiency of mutation operators and selective mutation strategies: An empirical study," *Software: Testing, Verification and Reliability*, vol. 9, no. 4, pp. 205–232, 1999.

[31] M. Gligoric, L. Zhang, C. Pereira, and G. Pokam, "Selective mutation testing for concurrent code," in *Proc. ISSTA*, 2013, pp. 224–234.

[32] R. Gopinath, M. A. Alipour, I. Ahmed, C. Jensen, and A. Groce, "On the limits of mutation reduction strategies," in *Proc. ICSE*. ACM, 2016, pp. 511–522.

[33] A. J. Offutt, A. Lee, G. Rothermel, R. H. Untch, and C. Zapf, "An experimental determination of sufficient mutant operators," *ACM Transactions on Software Engineering and Methodology*, vol. 5, no. 2, pp. 99–118, 1996.

[34] A. Siami Namin, J. H. Andrews, and D. J. Murdoch, "Sufficient mutation operators for measuring test effectiveness," in *Proc. ICSE*, 2008, pp. 351–360.

[35] M. Gligoric, A. Groce, C. Zhang, R. Sharma, M. A. Alipour, and D. Marinov, "Comparing non-adequate test suites using coverage criteria," in *Proc. ISSTA*, 2013, pp. 302–313.

[36] F. Lindskog, A. Mcneil, and U. Schmock, *Kendall tau for elliptical distributions*. Springer, 2003.

[37] I. Lawrence and K. Lin, "A concordance correlation coefficient to evaluate reproducibility," *Biometrics*, pp. 255–268, 1989.

[38] P. Ammann, "Transforming mutation testing from the technology of the future into the technology of the present [online]," in *Mutation Workshop*, 2015.

[39] M. Delahaye and L. du Bousquet, "A comparison of mutation analysis tools for Java," in *Proc. QSIC*, 2013, pp. 187–195.

[40] "homepage," https://github.com/SEITest/SMTMetrics.

[41] R. Gopinath, A. Alipour, I. Ahmed, C. Jensen, and A. Groce, "How hard does mutation analysis have to be, anyway?" in *ISSRE*, 2015, pp. 216–227.

[42] R. Gopinath, I. Ahmed, M. A. Alipour, C. Jensen, and A. Groce, "Does choice of mutation tool matter?" *Software Quality Journal*, pp. 1–50, 2016.

[43] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2012, vol. 821.

[44] R. P. Barke, H. Jenkins-Smith, and P. Slovic, "Risk perceptions of men and women scientists," *Social Science Quarterly*, pp. 167–176, 1997.

[45] P. H. Westfall and S. S. Young, *Resampling-based multiple testing: Examples and methods for p-value adjustment*. John Wiley & Sons, 1993, vol. 279.

[46] O. J. Dunn and V. A. Clark, *Applied statistics: analysis of variance and regression*. John Wiley & Sons, Inc., 1986.

[47] J. L. Horowitz, "Semiparametric estimation of a regression model with an unknown transformation of the dependent variable," *Econometrica: Journal of the Econometric Society*, pp. 103–137, 1996.

[48] J. C. Fry *et al.*, *Biological data analysis: a practical approach.* IRL Press Ltd, 1993.

[49] A. Nicodemo, M. Araujo, A. Ruiz, and A. Gales, "In vitro susceptibility of stenotrophomonas maltophilia isolates: Comparison of disc diffusion, etest and agar dilution methods," *Journal of Antimicrobial Chemotherapy*, vol. 53, no. 4, pp. 604–608, 2004.

[50] L. Eriksson, T. Byrne, E. Johansson, J. Trygg, and C. Vikström, *Multi-and megavariate data analysis basic principles and applications*. Umetrics Academy, 2013.

[51] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (mae)?–arguments against avoiding RMSE in the literature," *Geoscientific Model Development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[52] R. Just, M. D. Ernst, and G. Fraser, "Efficient mutation analysis by propagating and partitioning infected execution states," in *Proceedings of the 2014 International Symposium on Software Testing and Analysis*, 2014, pp. 315–326.

[53] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE Transactions on Software Engineering*, vol. 37, no. 5, pp. 649–678, 2011.

[54] W. E. Wong, A. P. Mathur, and J. C. Maldonado, "Mutation versus all-uses: An empirical evaluation of cost, strength and effectiveness," in *Software Quality and Productivity*, 1995, pp. 258–265.

[55] W. E. Wong and A. P. Mathur, "Reducing the cost of mutation testing: An empirical study," *Journal of Systems and Software*, vol. 31, no. 3, pp. 185–196, 1995.

[56] Y. Jiang, S.-S. Hou, J. Shan, L. Zhang, and B. Xie, "Contract-based mutation for testing components," in *Proc. ICSM*, 2005, pp. 483–492.

[57] ——, "An approach to testing black-box components using contract-based mutation," *International Journal of Software Engineering and Knowledge Engineering*, vol. 18, no. 1, pp. 93–117, 2008.

[58] A. T. Acree, T. A. Budd, R. A. DeMillo, R. J. Lipton, and F. G. Sayward, "Mutation analysis." Georgia Institute of Technology, Tech. Rep., 1979.

[59] B. Choi, R. A. DeMillo, E. W. Krauser, R. Martin, A. Mathur, A. J. Offutt, H. Pan, and E. H. Spafford, "The mothra tool set (software testing)," in *Proc. ICSS*, 1989, pp. 275–284.

[60] R. A. DeMillo, E. W. Krauser, and A. P. Mathur, "Compiler-integrated program mutation," in *Proc. COMPSAC*, 1991, pp. 351–356.

[61] R. H. Untch, A. J. Offutt, and M. J. Harrold, "Mutation analysis using mutant schemata," in *Proc. ISSTA*, no. 3, 1993, pp. 139–148.

[62] W. E. Howden, "Weak mutation testing and completeness of test sets," *IEEE Transactions on Software Engineering*, no. 4, pp. 371–379, 1982.

[63] M. Woodward and K. Halewood, "From weak to strong, dead or alive? an analysis of some mutation testing issues," in *Proc. STVA*, 1988, pp. 152–158.

[64] A. J. Offutt and S. D. Lee, "An empirical evaluation of weak mutation," *IEEE Transactions on Software Engineering*, vol. 20, no. 5, pp. 337–344, 1994.

[65] E. W. Krauser, A. P. Mathur, and V. J. Rego, "High performance software testing on simd machines," *IEEE Transactions on Software Engineering*, vol. 17, no. 5, pp. 403–423, 1991.

[66] A. J. Offutt, R. P. Pargas, S. V. Fichter, and P. K. Khambekar, "Mutation testing of software using a mimd computer," in *Proc. ICPP*, 1992.

[67] L. Zhang, D. Marinov, and S. Khurshid, "Faster mutation testing inspired by test prioritization and reduction," in *Proc. ISSTA*, 2013, pp. 235–245.

[68] L. Zhang, D. Marinov, L. Zhang, and S. Khurshid, "Regression mutation testing," in *Proc. ISSTA*, 2012, pp. 331–341.

[69] R. DeMilli and A. J. Offutt, "Constraint-based automatic test data generation," *IEEE Transactions on Software Engineering*, vol. 17, no. 9, pp. 900–910, 1991.

[70] S. Xu and T. J. Frank, "Forecasting the efficiency of test generation algorithms for combinational circuits," *Journal of Computer Science and Technology*, vol. 15, no. 4, pp. 326–337, 2000.

[71] M. Liu, Y.-F. Gao, J. Shan, J.-H. Liu, L. Zhang, and J. Sun, "An approach to test data generation for killing multiple mutants," in *Proc. ICSM*, 2006, pp. 113–122.

[72] D. Hao, L. Zhang, M. Liu, H. Li, and J. Sun, "Test-data generation guided by static defect detection," *Journal of Computer Science and Technology*, vol. 24, no. 2, pp. 284–293, 2009.

[73] L. Zhang, T. Xie, L. Zhang, N. Tillmann, J. d. Halleux, and H. Mei, "Test generation via dynamic symbolic execution for mutation testing," in *Proc. ICSM*, 2010, pp. 1–10.

[74] H. Do and G. Rothermel, "On the use of mutation faults in empirical assessments of test case prioritization techniques," *IEEE Transactions on Software Engineering*, vol. 32, no. 9, pp. 733–752, 2006.

[75] D. Hao, T. Lan, H. Zhang, C. Guo, and L. Zhang, "Is this a bug or an obsolete test?" in *Proc. ECOOP*, 2013, pp. 602–628.

[76] L. Zhang, D. Hao, L. Zhang, G. Rothermel, and H. Mei, "Bridging the gap between the total and additional test-case prioritization strategies," in *Proc. ICSE*, 2013, pp. 192–201.

[77] M. Staats, G. Gay, M. Whalen, and M. Heimdahl, "On the danger of coverage directed test case generation," in *Proc. FASE*, 2012, pp. 409–424.

[78] L. Zhang, L. Zhang, and S. Khurshid, "Injecting mechanical faults to localize developer faults for evolving software," in *Proc. OOPSLA*, 2013, pp. 765–784.

[79] S. Moon, Y. Kim, M. Kim, and S. Yoo, "Ask the mutants: Mutating faulty programs for fault localization," in *Software Testing, Verification and Validation (ICST), 2014 IEEE Seventh International Conference on*, 2014, pp. 153–162.

[80] X. Li and L. Zhang, "Transforming programs and tests in tandem for fault localization," in *SPLASH/OOPSLA*, 2017.

[81] J. Zhang, Y. Lou, L. Zhang, D. Hao, L. Zhang, and H. Mei, "Isomorphic regression testing: executing uncovered branches without test augmentation," in *Proc. FSE*. ACM, 2016, pp. 883–894.

[82] M. Papadakis, C. Henard, M. Harman, Y. Jia, and Y. Le Traon, "Threats to the validity of mutation-based test assessment," in *Proc. ISSTA*. ACM, 2016, pp. 354–365.