mrstudyr: Retrospectively Studying the Effectiveness of Mutant Reduction Techniques

Colton J. McCurdy
Allegheny College

Phil McMinn
University of Sheffield

Gregory M. Kapfhammer
Allegheny College

***Abstract*—Mutation testing is a well-known method for measuring a test suite's quality. However, due to its computational expense and intrinsic difficulties (e.g., detecting equivalent mutants and potentially checking a mutant's status for each test), mutation testing is often challenging to practically use. To control the computational cost of mutation testing, many reduction strategies have been proposed (e.g., uniform random sampling over mutants). Yet, a stand-alone tool to compare the efficiency and effectiveness of these methods is heretofore unavailable. Since existing mutation testing tools are often complex and language-dependent, this paper presents a tool, called *mrstudyr*, that enables the "retrospective" study of mutant reduction methods using the data collected from a prior analysis of all mutants. Focusing on the mutation operators and the mutants that they produce, the presented tool allows developers to prototype and evaluate mutant reducers without being burdened by the implementation details of mutation testing tools. Along with describing *mrstudyr*'s design and overviewing the experimental results from using it, this paper inaugurates the public release of this open-source tool.**

I. Introduction

Software developers may introduce errors into a program's source code that could result in a human fatality [1]. Running a set of tests, frequently called a test suite, often aids in detecting the faults that cause a program to function incorrectly [2]. Yet, simply testing a program is not sufficient. In order for testing to establish a confidence in the correctness of the program under test, the test suite needs to be of a high level of quality [3].

Mutation testing is a widely-recognized technique for assessing the quality of a test suite [4]. While there are many potential faults for a program, mutation testing focuses on those that are "close" to the correct version, with the expectation that they will be representative of all faults [5]. Of the methods for evaluating test quality, mutation testing is widely considered the strongest test criterion in terms of its capability to necessitate the creation of tests that find many faults [6].

Although mutation testing effectively requires tests to detect faults, it has significant drawbacks in its computational cost and the amount of necessary human interaction, often making it impractical to use [4], [7], [8]. A major computational cost of mutation testing comes from executing each test case in a test suite for the many generated mutants [9], [10]. Executing a small, representative set of mutants against the test suite has previously been proposed as a technique to reduce the cost of mutation testing [5], [11], [10]; this reduction strategy is categorized by Offutt and Untch as a "do fewer" approach [12].

There are several mutant reduction techniques in the "do fewer" category, with mutant sampling being a simple method that randomly selects a subset of all mutants [11]. In addition to being conceptually simple [4], mutant sampling has been experimentally shown to outperform other more sophisticated methods [13]. Two sub-techniques within mutant sampling are called uniform random sampling and sampling over operators [4], [8], [14]. For both of these sub-techniques, a threshold for the maximum percentage of selected mutants is set to x, which is then either applied to the entire set of mutants or to each set of mutants produced by an operator [4], [8], [14].

Prior work has found the smallest value of x that still produces a representative set of mutants [5], [15]. Yet, these efforts normally required the experimenters to integrate a reduction technique into an existing mutation testing system before performing a mutation testing experiment [16], [17]. Since mutation testing tools are often complex — according to the Count Lines of Code (`cloc`) tool the PIT mutation testing system contains over 43,000 lines of non-commented Java code and thousands of lines of build and configuration files — this approach to studying mutant reduction methods has a high upfront cost. That is, researchers in this field must grasp the complexities of a mutation testing tool before they can experimentally evaluate new techniques for mutant reduction.

As a means for obviating the need for researchers to grasp a complex mutation testing system, this paper advances the idea of retrospectively studying mutant reducers. After using a tool like PIT to collect data about which operators ran and what mutants they produced, a retrospective analysis applies strategies like uniform random sampling to the mutant data, thereby quickly facilitating an understanding of a reduction method's trade-offs. Only after researchers comprehend how the mutant reducers work in the intended domain must they then grapple with the complexities of the chosen tool.

Since retrospective analysis still requires tool support, this paper presents *mrstudyr*, a tool for evaluating mutant reduction techniques in retrospect. Accepting data in a generalized format from a single run of a mutation testing system, *mrstudyr* applies mutant reduction strategies and calculates their efficiency and effectiveness. In addition to being capable of retrospectively analysing mutant reduction techniques from various domains, *mrstudyr* is well-documented and has been released on GitHub under an open-source license [18].

As studying the mutant data retrospectively removes the need to comprehend the complexities of a target environment, mutant reduction methods can be extended to new domains such as that of relational database schemas [19], [20], [21]. Ensuring that a database's schema has correctly specified integrity constraints is important because these entities ensure that only valid data enters the database. Even though there are 971,373 questions about databases on StackExchange, the

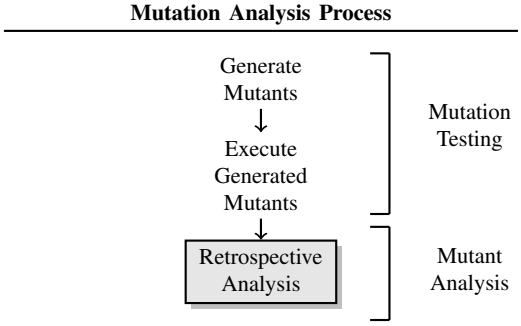

Fig. 1: The phases of the mutation analysis process.

technical question and answer website [22], little prior work has focused on testing these integrity constraints [19].

Since it is important to assess the quality of tests for relational database schemas, recent work has proposed and evaluated database-aware mutation analysis techniques [19], [20], [21]. Although the presented method and tool are general, this paper illustrates the retrospective study of mutant reducers and the use of *mrstudyr* in the area of mutation analysis for database schemas. In addition to describing the implementation of *mrstudyr* and overviewing results from applying it to the retrospective study of database schema mutation, this paper inaugurates the public release of this analysis tool. In summary, the key contributions of this paper are as follows:

- A well-documented and easy-to-use tool, *mrstudyr*, that:
  - supports using mutant reduction methods retrospectively as a way to study trade-offs in efficiency and effectiveness without having to understand the implementation of a complex mutation testing tool.
  - accepts a generalized input format, is extendible to various domains, and is released as a free and open-source package in the R programming language.
- Using database schemas taken from real-world database-centric applications, preliminary results from using *mrstudyr*, highlighting the benefits of mutant reduction and the ease with which these results may be obtained.

II. Implementation and Use of the *mrstudyr* Tool

A. Objectives

When performed with tools such as Major [23], the process of mutation testing, as displayed in Figure 1, involves the use of operators to generate mutants for a specific program and then the execution of tests to determine how they kill the mutants. The outcome of this phase is the higher-is-better mutation score, or the ratio of the number of killed mutants to the number of mutants generated [7]. In many cases, as shown in Figure 1, it is necessary to perform various analyses of the mutants. For instance, testers may want to see which mutants were not killed so as to determine if they are equivalent (i.e., semantically the same as the original program) or, alternatively, indications of ways to improve the test suite.

This paper presents another type of mutant analysis: the retrospective study of mutant reduction techniques. Leveraging data collected during mutation testing (e.g., the name of an operator that produced a mutant, the kill-status of a mutant, and the costs of producing and analysing a mutant), this method supports the study of mutant reduction techniques. This type of retrospective analysis allows testers to ask and answer questions like "what would the mutation score be if only a random 20% of the mutants were executed?" While questions of this nature could be executed through, for instance, either a manual analysis or a bespoke program, this paper presents *mrstudyr*, a tool that makes it easy to effectively pose and answer questions about methods for mutant reduction.

TABLE I: The generalized data format accepted by *mrstudyr*.

Identifier	Configuration	Entity	Operator	Mutant Type	Kill Status	Time (ms)
fpbpyn2	SQLite	CoffeeOrders	FKCColumnPairR	NORMAL	true	59
fpbpyn2	SQLite	CoffeeOrders	FKCColumnPairR	NORMAL	true	59
fpbpyn2	SQLite	CoffeeOrders	FKCColumnPairR	NORMAL	true	88
fpbpyn2	SQLite	CoffeeOrders	FKCColumnPairR	NORMAL	true	54
fpbpyn2	SQLite	CoffeeOrders	FKCColumnPairE	NORMAL	true	56
fpbpyn2	SQLite	CoffeeOrders	FKCColumnPairE	NORMAL	true	49

Currently, *mrstudyr* can perform a widely-studied mutant reduction technique: mutant sampling [4], [5], [8]. Due to its extensible design, the *mrstudyr* tool could be extended to perform specific reduction techniques such as E-selective [24], where mutants are only created by the following operators: ABS, UOI, LCR, AOR, and ROR [8]. Additionally, reduction methods that group mutants with clustering algorithms [5] could also be incorporated into *mrstudyr*. To support the integration of new algorithms for mutant reduction, *mrstudyr* is accompanied with extensive documentation that explains the inputs, outputs, and behavior of the main functions [18].

B. Input Format

By accepting a generalized input format, the *mrstudyr* tool can be used in a variety of testing domains to assess the efficiency and effectiveness of mutant reduction techniques. Table I gives a snippet of a data set analyzed in the experimental study of this paper. Organized so that each row furnishes data about an individual mutant, *mrstudyr*'s data is in a "tidy" format [25] that primarily focuses on the entity under test, the configuration in which mutation testing was performed, and the operator that produced each mutant. Since mutation testing systems can produce different types of mutants (e.g., normal, still-born, or equivalent) [26], *mrstudyr*'s input files also record the type of the mutant. Additionally, the tool tracks the status of the mutant and one of many different execution timings (e.g., mutant generation or analysis time); as necessary, the tool can also be extended to process more input types.

C. Conducting Experiment Campaigns

Figure 2 gives the structure for the campaign of experiments that *mrstudyr* conducts to collect the reduced mutant data from a single reduction technique. The "analyze" algorithm expects the mutant data from performing mutation testing on all mutants. Since the presented tool currently focuses on mutant sampling, the *reduction technique* algorithm requires as input an arbitrary x, to be chosen as the maximum percentage for the number of mutants to be analyzed from a set. Following

function analyze ($data$)
where
- *data* is data collected from performing mutation testing on all mutants
- x is the maximum percentage for the number of mutants
- *percentages* is the list of chosen x values for mutant sampling
- t is the trial index of the experiment
- *trials* is the total number of trials set for each configuration of technique
- *reduced* is the reduced mutant data returned from a single reduction method
- d is the data from the configuration of a reduction method and x value

$d \leftarrow \varnothing$, $t \leftarrow 0$
for each $x \in$ *percentages*
  for $t \leq$ *trials*
    let *reduced* $\leftarrow$ *reduction technique* ($data$, x)
    let $d \leftarrow$ **append**(d, *reduced*)
    $t \leftarrow t + 1$
return d
end function

Fig. 2: An experiment function for studying mutant sampling.

the recommendations of Traeger et al. [27] and Arcuri and Briand [28], for the preliminary results presented in this paper, the maximum threshold for the number of trials that *mrstudyr* runs for each configuration of a reduction approach is set to 30, thereby controlling for the randomness inherent in both a reduction method's behavior and execution time.

D. Implementation as an R Package

The R programming language is commonly used because it facilitates reproducible research [29], [30]. It does so by providing anyone interested with the necessary data and code to recreate the analyses of the researcher [31]. In the R language, the established way to share code is via a package, which is easy to distribute and often includes data, code, documentation, and test cases [30], [32]. Since *mrstudyr* has been released as an R package, installing it requires four commands. First, `install.packages("devtools")`, then `library(devtools)` to install and load the `devtools` [33] package, respectively. The `devtools` package is necessary because it facilitates the installation and maintenance of *mrstudyr* as well as its dependencies. Finally, to install *mrstudyr* [18] from the popular Git repository hosting service, GitHub [34], use the following command: `devtools::install_github("mccurdyc/mrstudyr")`. Then, load *mrstudyr* using `library(mrstudyr)`.

E. Tool Usage

We designed the *mrstudyr* tool to make it simple to perform a thorough and automated empirical analysis of mutant reduction techniques. The common structure of an R package expects that externally-collected data is stored in the `inst/extdata` folder. This is the location where *mrstudyr* looks to find the mutation data, stored as a comma-separated value file; in this paper, we ran the *SchemaAnalyst* tool [35] to generate the data used as input to *mrstudyr*.

The "Reduction Techniques", as referenced in Figure 3, are performed following the provision of mutant data, the "Original Data", to *mrstudyr*. Using *mrstudyr* to perform analyses and create visualisations requires the call of a single function per reduction technique; the functions are `create_random_sampling_graphs()` and `create_operator_sampling_graphs()`. Both functions accumulate the reduced data over 30 trials into a single data set, as shown by the "Accumulated Reduced Data" ellipse in Figure 3. After performing a reduction technique, it is evaluated in the "Efficiency and Effectiveness Analysis" phase that results in the output of these values: mutation score (MS), cost reduction (Red.), correlation (Corr.), and error (Err.).

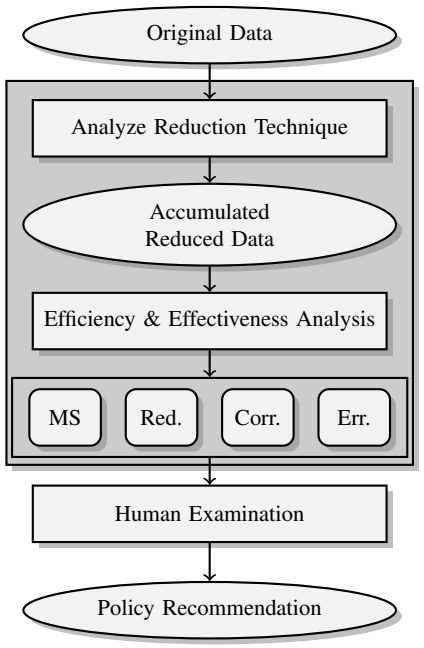

Fig. 3: The inputs and outputs of the *mrstudyr* tool.

In this figure, the dark square represents the *mrstudyr* tool and its constituent parts, a rectangle stands for a process, a rectangle with rounded edges is a calculation performed by *mrstudyr*, and an ellipse symbolises an input or output.

While mutation score and cost reduction are calculated in the function performing the analysis, a correlation coefficient and two error metrics are calculated by `analyze_calculations()`, where the function's input is the accumulated data from a reduction technique and the output is a new data set with these three effectiveness values.

The *mrstudyr* tool employs correlation to determine how the mutation score arising from the reduced set of mutants corresponds to the score produced by the full mutant set. Kendall's τ_b is a measure of correlation between -1 and 1, representing a strong negative and strong positive association, respectively, with 0 indicating that there is no correlation [36]. Following Inozemtseva and Holmes, we adopt the Guildford scale to describe a correlation, with the absolute value of a coefficient being described as "low" when it is less than 0.4, "moderate" when it is between 0.4 and 0.7, "high" between 0.7 to 0.9, and "very high" when it is greater than 0.9 [37]. The errors calculated by *mrstudyr* are the widely-used root mean square error (RMSE) and mean absolute error (MAE) [38], both "lower-is-better" metrics showing the difference between the mutation scores for the reduced and full set of mutants.

In the human examination phase of Figure 3, *mrstudyr* presents the results of the analysis phase by using Wickham's graphing package, `ggplot2` [39], to create easy-to-grasp visualisations that help a user to construct

TABLE II: Schemas analyzed in the empirical study.

Schema	Tables	Columns	Checks	Foreign Keys	Not Nulls	Primary Keys	Uniques	∑Constraints
CoffeeOrders	5	20	0	4	10	5	0	19
Employee	1	7	3	0	0	1	0	4
Inventory	1	4	0	0	0	1	1	2
Iso3166	1	3	0	0	2	1	0	3
JWhoisServer	6	49	0	0	44	6	0	50
MozillaPermissions	1	8	0	0	0	1	0	1
NistWeather	2	9	5	1	5	2	0	13
Person	1	5	1	0	5	1	0	7
Products	3	9	4	2	5	3	0	14
Total	21	114	13	7	71	21	1	113

a recommendation regarding which reducer should be incorporated into a mutation testing tool. Examples of these visualisations, along with a screencast [40] and more documentation are found on *mrstudyr*'s GitHub page [18].

III. Preliminary Study

To demonstrate the effectiveness and domain extensibility of *mrstudyr*, we studied the mutants of the nine schemas in Table II with the presented tool. Similar to the studies of Wong and Mathur [15], *mrstudyr* performed mutant sampling with x value increments larger than 1%. Specifically, for this experiment, *mrstudyr* analyzed x at 1% and 10%, then increased by 10% intervals to a maximum value of 90%. By setting the granularity of the experiment to 10% intervals, *mrstudyr* reduces the cost of performing retrospective analysis, while confirming trends from prior work, as shown in Figure 4. For each of the sampling techniques currently supported by *mrstudyr*, the mutant reducer is invoked at every x percentage for each of the schemas under test and for a total of 30 trials.

Figure 4 is a box-and-whisker plot with the schemas on the horizontal-axis and the mutation scores of the reduced sets after random sampling at the x values of 1%, 10%, 20%, and 40% on the vertical-axis. These values were chosen as the x values because they serve to confirm a previously observed trend of decreasing errors between the original and reduced sets' mutation score as x increases. Moving from top-left to bottom-right, the boxes in Figure 4 show this decrease in error.

This trend occurs because the mutation scores of reduced sets from small percentages are often very volatile and can thus vary largely based on one or few mutants; in contrast, the mutation scores of sets with greater percentages are substantially more stable. In the top-left of Figure 4, x is evaluated at 1%. In this quadrant of Figure 4, the calculated RMSE, 12.090, is very high with respect to the same metric at greater percentages, while the correlation coefficient, 0.385, is classified as "low" according to Guildford scale. In the top-right quadrant, where x is 10%, stability is already evident in the reduced sets' mutation scores. At this percentage, an RMSE of 4.082 is much lower than at 1% and the correlation is "moderate" with a coefficient value of 0.654. This same trend of decreasing RMSE values and increasing correlation coefficients remains true for x values of 20% and 40%. When randomly sampling 20% and 40% of the mutants, RMSE is 2.485 and 1.568 and the correlation coefficients are "high" for both, with values of 0.763 and 0.852, respectively.

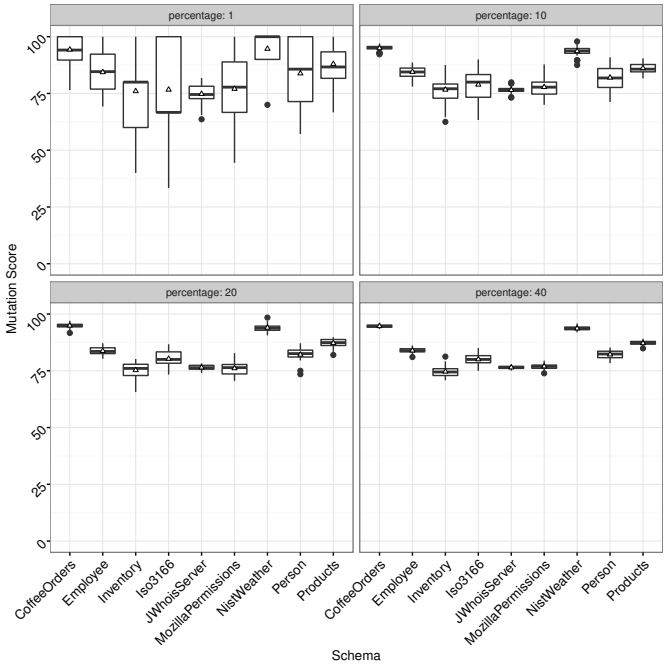

Fig. 4: Mutation scores per database schema over 30 trials.

Each box represents the inter-quartile range (IQR), or the measure of statistical dispersion that is the difference between the 1st and 3rd quartiles. In this plot the whiskers extend up to 1.5 times the IQR and the line across the middle of the box marks the median. Additionally, the triangle in the boxes denotes the mean and the filled circles extending beyond the whiskers correspond to outliers.

These results demonstrate that it is possible to easily use *mrstudyr* to confirm the prior results of Wong and Mathur in the new application domain of database schemas. More studies can now be run as *mrstudyr* is available from GitHub [18].

IV. Conclusions and Future Work

Although mutation testing is well-recognized as a way to assess test suite quality, it may be too costly to practically use. As such, various methods have been developed to decrease the cost of mutation testing. Performing these reduction techniques in the past has required researchers and experimenters to incorporate a reduction method into an, often complex, mutation testing tool. The *mrstudyr* tool alleviates the burden of implementing each approach by analyzing reduction techniques in retrospect, a potentially more cost-effective method.

By retrospectively analyzing the data collected from a prior analysis of all mutants, the *mrstudyr* tool is able to decrease the upfront human-implementation costs by obviating the need for researchers and industrialists to fully understand the domain complexities associated with a mutation testing system. Furthermore, *mrstudyr* provides an easy-to-use and rapid way to assess the efficiency and effectiveness of mutant reduction methods. In addition to being detailed in this paper, *mrstudyr* has been released under an open-source license on a GitHub site that features extensive documentation and a screencast [18]. In future work, we plan to extend the functionality of *mrstudyr* by integrating additional mutant reduction techniques, thereby allowing for a more comprehensive comparison of the techniques' efficiency and effectiveness.

REFERENCES

[1] K. J. Vicente, K. Kada-Bekhaled, G. Hillel, A. Cassano, and B. A. Orser, "Programming errors contribute to death from patient-controlled analgesia: Case report and estimate of probability," *Canadian Journal of Anesthesia*, vol. 50, no. 4, 2003.

[2] G. M. Kapfhammer, "Regression testing," in *The Encyclopedia of Software Engineering*, 2010.

[3] G. M. Kapfhammer, "Software testing," in *The Computer Science Handbook*, 2004.

[4] R. Gopinath, A. Alipour, I. Ahmed, C. Jensen, and A. Groce, "Do mutation reduction strategies matter?" *Oregon State University, Technical Report*, 2015.

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

[6] P. Ammann and A. J. Offutt, *Introduction to Software Testing*. Cambridge University Press, 2008.

[7] R. Just, G. M. Kapfhammer, and F. Schweiggert, "Using conditional mutation to increase the efficiency of mutation analysis," in *Proceedings of the 6th International Workshop on Automation of Software Test*, 2011.

[8] R. Gopinath, A. Alipour, I. Ahmed, C. Jensen, and A. Groce, "An empirical comparison of mutant selection approaches," *Oregon State University, Technical Report*, 2015.

[9] R. Just, G. M. Kapfhammer, and F. Schweiggert, "Using non-redundant mutation operators and test suite prioritization to achieve efficient and scalable mutation analysis," in *Proceedings of the 23rd International Symposium on Software Reliability Engineering*, 2012.

[10] A. J. Offutt, G. Rothermel, and C. Zapf, "An experimental evaluation of selective mutation," in *Proceedings of the 15th International Conference on Software Engineering*, 1993.

[11] 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, 1995.

[12] A. J. Offutt and R. H. Untch, "Mutation 2000: Uniting the orthogonal," in *Mutation Testing for the New Century*, 2001.

[13] L. Zhang, S.-S. Hou, J.-J. Hu, T. Xie, and H. Mei, "Is operator-based mutant selection superior to random mutant selection?" in *Proceedings of the 32nd International Conference on Software Engineering*, 2010.

[14] L. Zhang, M. Gligoric, D. Marinov, and S. Khurshid, "Operator-based and random mutant selection: Better together," in *Proceedings of the 28th International Conference on Automated Software Engineering*, 2013.

[15] A. P. Mathur and E. W. Wong, "An empirical comparison of data flow and mutation-based test adequacy criteria," *Software Testing, Verification and Reliability*, vol. 4, no. 1, 1994.

[16] R. A. DeMillo, D. S. Guindi, W. M. McCracken, A. J. Offutt, and K. N. King, "An extended overview of the Mothra software testing environment," in *Proceedings of the 2nd International Workshop on Software Testing, Verification, and Analysis*, 1988.

[17] K. N. King and A. J. Offutt, "A Fortran language system for mutation-based software testing," *Software: Practice and Experience*, vol. 21, no. 7, 1991.

[18] C. J. McCurdy, "*mrstudyr* software tool," 2016. [Online]. Available: https://github.com/mccurdyc/mrstudyr

[19] P. McMinn, G. M. Kapfhammer, and C. J. Wright, "Virtual mutation analysis of relational database schemas," in *Proceedings of the 11th International Workshop on Automation of Software Test*, 2016.

[20] P. Mcminn, C. J. Wright, and G. M. Kapfhammer, "The effectiveness of test coverage criteria for relational database schema integrity constraints," *Transactions on Software Engineering and Methodology*, vol. 25, no. 1, 2015.

[21] C. J. Wright, G. M. Kapfhammer, and P. McMinn, "Efficient mutation analysis of relational database structure using mutant schemata and parallelisation," in *Proceedings of the 8th International Workshop on Mutation Analysis*, 2013.

[22] "StackExchange query: Prevalence of SQL, schema, and integrity constraints," 2016. [Online]. Available: http://goo.gl/lXBm8f

[23] R. Just, G. M. Kapfhammer, and F. Schweiggert, "MAJOR: An efficient and extensible tool for mutation analysis in a Java compiler," in *Proceedings of the 26th International Conference on Automated Software Engineering*, 2011.

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

[25] H. Wickham, "Tidy data," *Journal of Statistical Software*, vol. 59, no. 1, 2014.

[26] C. J. Wright, G. M. Kapfhammer, and P. McMinn, "The impact of equivalent, redundant, and quasi mutants on database schema mutation analysis," in *Proceedings of the 14th International Conference on Quality Software*, 2014.

[27] A. Traeger, E. Zadok, N. Joukov, and C. P. Wright, "A nine year study of file system and storage benchmarking," *ACM Transactions on Storage*, vol. 4, no. 2, 2008.

[28] A. Arcuri and L. Briand, "A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," *Software Testing, Verification and Reliability*, vol. 24, no. 3, 2014.

[29] G. M. Kapfhammer, "Empirically evaluating regression testing techniques: Challenges, solutions, and a potential way forward," in *Proceedings of the 1st International Workshop on Regression Testing*, 2011.

[30] G. M. Kapfhammer, P. McMinn, and C. J. Wright, "Hitchhikers need free vehicles! Shared repositories for statistical analysis in SBST," in *Proceedings of the 9th International Workshop on Search-Based Software Testing*, 2016.

[31] R. Gentleman and D. T. Lang, "Statistical analyses and reproducible research," *Journal of Computational and Graphical Statistics*, 2012.

[32] H. Wickham, *R Packages*. O'Reilly Media, Inc., 2015.

[33] H. Wickham and W. Chang, "CRAN — package devtools," 2016. [Online]. Available: https://goo.gl/Je5EO2

[34] "Github," 2016. [Online]. Available: https://github.com/

[35] P. McMinn, C. J. Wright, C. Kinneer, C. J. McCurdy, M. Camara, and G. M. Kapfhammer, "*SchemaAnalyst*: Search-based test data generation for relational database schemas," in *Proceedings of the 32nd International Conference on Software Maintenance and Evolution*, 2016.

[36] A. I. McLeod, "Kendall rank correlation and mann-kendall trend test," 2016. [Online]. Available: https://goo.gl/fZ7jc8

[37] L. Inozemtseva and R. Holmes, "Coverage is not strongly correlated with test suite effectiveness," in *Proceedings of the 36th International Conference on Software Engineering*, 2014.

[38] 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, 2014.

[39] H. Wickham and W. Chang, "CRAN — package ggplot2," 2016. [Online]. Available: https://goo.gl/bapiJI

[40] C. J. McCurdy, "Screencast of mrstudyr," 2016. [Online]. Available: https://goo.gl/rZCkTU