**Temporal Systems and Forecasting**
Volume 1 · Number 1 · 2026

# Dating the Repair Record: Limited Temporal Value of a Simple Issue-Based Edit-Size Proxy

Mark Scott[1] · Oliver Young[2] · Ryan King[3]

[1] Slovakia · [2] Malta · [3] Cyprus

Original publication: November 4, 2026 | Article 703

## Abstract

Can older issue records tell us how large later repairs will be? A four-parameter proxy fitted to 105 SWE-bench Lite pull requests through 2019 barely improved on a constant when applied to 195 later tasks. Mean absolute error in log edit size fell by 0.0062, with a repository-clustered interval crossing zero. A later-window bias contrast was also uncertain and rested on six repositories observed in both windows. The result limits the practical role of these inexpensive descriptors without establishing that newer coding tasks are intrinsically harder.

*Exploratory internal chronological split of SWE-bench Lite. The outcome is reference-patch line count; this is not a coding-agent evaluation or an official benchmark score.*

# Dating the Repair Record: Limited Temporal Value of a Simple Issue-Based Edit-Size Proxy

> Exploratory internal chronological split of SWE-bench Lite. The outcome is reference-patch line count; this is not a coding-agent evaluation or an official benchmark score.


## Abstract

Can older issue records tell us how large later repairs will be? A four-parameter proxy fitted to 105 SWE-bench Lite pull requests through 2019 barely improved on a constant when applied to 195 later tasks. Mean absolute error in log edit size fell by 0.0062, with a repository-clustered interval crossing zero. A later-window bias contrast was also uncertain and rested on six repositories observed in both windows. The result limits the practical role of these inexpensive descriptors without establishing that newer coding tasks are intrinsically harder.


## 1. Edit planning needs an outcome it can actually observe

A coding assistant is expected to change an appropriate amount of code, not merely produce a plausible continuation. Before evaluating that behavior, researchers often need a simpler description of the repair task: how much changed, where it changed, and whether the information available beforehand could have anticipated its size. An issue's length or formatting is inexpensive to obtain, but inexpensive metadata may provide little information about the resulting patch.

The question examined here is whether an issue-record proxy fitted to older pull requests remains informative about the line-count size of later reference repairs. We analyze SWE-bench Lite Jimenez et al., 2023, using its public task records Princeton NLP, 2026 as an observational source. The outcome is the logarithm of one plus added and removed non-test patch lines. It describes the reference artifact, not the assistant's behavior, semantic completeness, reviewer satisfaction, or the minimum change necessary to solve a problem.

Tao and Kamata, 2026 study fine-tuning coding assistants from pairwise preferences about what to change. That research makes the definition and availability of change-related information especially relevant. Our analysis does not implement its training procedure or construct human preference labels. It examines a more modest ingredient in a possible evaluation workflow: whether three readily encoded record descriptors provide a useful chronological baseline for reference edit size.

Two propositions guide the empirical design. A metadata proxy should show an advantage over a declared constant reference on later records before it is treated as useful for planning. In addition, apparent temporal changes should be separated from changes in the repository composition of those records. Neither proposition presumes that performance must deteriorate over time. Weak improvement, inconsistent yearly errors, and uncertain period contrasts remain admissible results.

## 2. Repository records are selected measurement objects

Mining a software repository requires attention to how records are created and selected Bird et al., 2009. A benchmark task pairs an issue with a reference solution under benchmark-specific eligibility rules; it is not a random observation of all developer work. Misclassification can affect software prediction studies Herzig et al., 2013, and the expectations surrounding code review extend beyond the final text of a patch Bacchelli and Bird, 2013. These considerations constrain what line-count forecasting can establish.

Lines changed are also a restricted definition of edit scope. Structural complexity McCabe, 1976 concerns properties that a line count does not directly encode. A short edit can have a large behavioral effect, while a long edit can contain repetitive changes. Test-based reference support has its own limits, as emphasized by the oracle problem Barr et al., 2015. We therefore retain a literal, reproducible count without relabeling it semantic difficulty, repair correctness, or a developer's preferred amount of change.

Other public coding resources provide different empirical units. Model evaluation on coding problems Chen et al., 2021, crowd-sourced programming tasks Austin et al., 2021, paired buggy and corrected functions Lin et al., 2017, and more extensive tests for generated code Liu et al., 2023 cannot simply be substituted for dated repository repairs. Those resources motivate broader future comparisons, but none supplies additional observations to the current regression. The chronological evidence comes only from the captured Lite records.

## 3. The timestamp is the solution pull request's date

The local snapshot contains 300 complete task records from twelve repositories. We read the three acquired JSON pages and require unique instance identifiers and no truncated cells. Source documentation defines created_at as the creation date of the solution pull request. It is not the original issue-opening time, the date a developer first recognized the problem, or the execution date of an evaluated coding model. The distinction is essential to the meaning of temporal transport.

We fit the proxy on the 105 tasks dated through 2019 and evaluate it on the 195 tasks dated from 2020 onward. The fitting period begins on 20 September 2012 and ends on 31 December 2019. Later records extend from 1 January 2020 to 17 July 2023. All records originate in the benchmark's public test split; our internal division does not create an official train-test benchmark protocol or establish that a pretrained model never encountered these repositories.

The older subset contains seven repositories and the later subset eleven. Five later repositories are absent from fitting: Matplotlib, Seaborn, Flask, Pylint, and Sphinx. The proxy therefore contains no repository categorical predictor requiring arbitrary assignment of an unseen level. We keep all later observations in the primary loss comparison, rather than discarding repositories that make transport harder. The later-period composition is part of the practical scope of that comparison.

Predictors are log one plus the issue's regex word count, an indicator for a triple-backtick fence, and an indicator for nonempty hints. The source defines hints as issue comments before the solution pull request's first commit. Such comments need not be available when the issue first opens. Consequently, this is a retrospective descriptor of the pull-request task record, not a validated prediction service operating at issue creation. Empty hints are treated as an observed field state, not proof that no useful discussion existed.

## 4. A frozen proxy and two later-period questions

An ordinary least-squares fit relates log edit size to the three predictors and an intercept. There is no hyperparameter search, repository-specific coefficient, model-generated patch, or test-patch predictor. Its fitted coefficients are frozen before computing later losses. Chronological evaluation principles Tashman, 2000 motivate keeping fitting and assessment periods distinct, while dependence-aware validation Roberts et al., 2017 explains why the later task count alone should not determine inferential precision.

The constant reference is the older sample's mean log edit size. It corresponds to an intercept-only squared-loss fit. It is not claimed to be the absolute-loss-optimal training median. The point-forecast literature Gneiting, 2011 makes this alignment between a fitting objective and a scoring objective

consequential. We deliberately retain the declared constant and evaluate both predictions under the same later absolute error on the log scale, without replacing the reference after seeing the outcome.

For each later task, the primary contrast subtracts the constant's absolute log error from the metadata proxy's error. A negative value favors the proxy. Model 1 estimates the equal-task average contrast for all 195 later tasks. Repository-clustered covariance uses eleven groups, small-sample correction, and a t reference with ten degrees of freedom Cameron and Miller, 2015. Robust covariance methods White, 1980 address variation in residual scale, but they do not make eleven repositories a representative population of software projects.

The secondary question concerns signed error, defined as predicted minus actual log edit size. We contrast 2022-2023 with 2020-2021 after repository effects. Only six repositories appear in both windows. Model 2 therefore uses their 173 tasks, retaining those with direct within-repository temporal comparison and excluding twenty-two tasks only from this model. It uses six repository clusters and t5 intervals. The loss contrast and period contrast intentionally have different denominators and answer different questions.

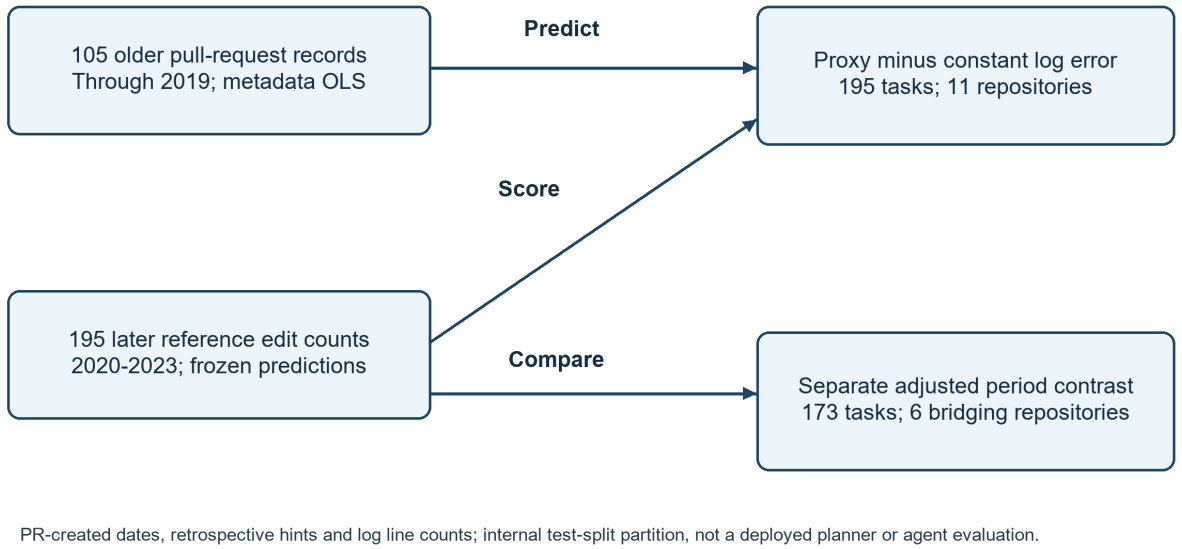



*Figure 1. Older pull-request records fit a frozen metadata proxy; later gold edit counts define loss contrasts, while six bridging repositories support a separate period comparison.*

Time-series cross-validation results Bergmeir et al., 2018 do not automatically validate this repository sample or permit random mixing of its eras. Similarly, selection-bias concerns in model evaluation Varma and Simon, 2006 favor keeping the simple formula fixed rather than searching many metadata combinations on later tasks. Figure 1 makes the separation explicit: later gold patches score the already fitted proxy and never become predictor inputs.

## 5. A weak older fit and a small later advantage

The older fit has R-squared 0.014706. Its intercept is 2.243283; coefficients are -0.077501 for log issue words, 0.039911 for a code fence, and 0.190870 for nonempty hints. These values offer little in-sample explanation of reference edit-size variation. The negative length coefficient is recorded without a narrative that longer issues cause smaller repairs. It is a conditional coefficient in one selected older subset.

Across the 195 later tasks, mean absolute log error is 0.648835 for the proxy and 0.655064 for the constant. The mean difference is -0.006229, with clustered standard error 0.003654 and interval

-0.014370 to 0.001912. The proxy's point estimate is slightly better, but the interval crosses zero. That is insufficient evidence for a dependable planning benefit and is not an equivalence finding or proof that all issue metadata are uninformative.

***Table 1. Later-period error contrasts from frozen predictions.*** *M1 is an intercept-only mean of paired absolute-log-error differences, N195 with eleven repository clusters and t10. M2 regresses signed log error on a 2022-2023 indicator and repository effects, N173 in six repositories present in both windows, with six clusters and t5. Only the period coefficient is displayed for M2; no era-causal interpretation is intended.*

| Model | Term | Estimate (SE) | 95% CI | N |
|---|---|---|---|---|
| M1 | Mean outcome/contrast | -0.0062 (0.0037) | [-0.0144, 0.0019] | 195 |
| M1: fit | R-squared = 0.0000 | - | - | 195 |
| M2 | 2022-2023 minus 2020-2021 window | 0.2310 (0.1102) | [-0.0522, 0.5141] | 173 |
| M2: fit | R-squared = 0.0945 | - | - | 173 |

Model labels: M1 = Held-period issue-proxy minus constant absolute log-scope error; M2 = Later-window signed error with repository composition.

The repository-adjusted later-window signed-error coefficient is 0.230967, with standard error 0.110157 and interval -0.052200 to 0.514133. Model 2 explains 0.094487 of signed-error variation in its bridging subset. The interval includes zero and allows appreciable changes in either inferential direction around the estimate. In particular, an eleven-repository count from Model 1 cannot be reused to make this six-repository comparison appear more precise.

Overlap is uneven even among those six projects. Django contributes fifty-seven earlier-window and thirty-one later-window tasks, while SymPy contributes twenty-four and nine. Matplotlib contributes just one earlier-window task against twenty-two later-window tasks. Thus the adjusted contrast is identified by a highly unbalanced set of within-project comparisons. A repository indicator removes a level difference; it does not manufacture comparable examples of every task type at both dates. The saved project-by-window counts make that limitation inspectable rather than hiding it behind the pooled task total.

The date division also does not describe continuous observation of a software system. There are no repeated measurements of the same unresolved issue as it ages, and no record of a coding assistant's predictions made in each historical year. Each later prediction is produced now by the same frozen descriptive fit. This construction can assess chronological separation of captured task records, but it cannot establish a historical degradation trajectory for a deployed model or infer changes in the source repositories' overall engineering productivity.

## 6. Yearly differences resist a simple deterioration story

Later-year sample sizes are sixty-six in 2020, forty-two in 2021, fifty-seven in 2022, and thirty in 2023. Proxy absolute log errors are 0.694113, 0.521369, 0.641374, and 0.741855 respectively. Constant-reference errors are 0.685942, 0.528861, 0.666866, and 0.741394. The proxy is worse in the first and last years and better in the two middle years. This sequence does not form a monotone deterioration pattern.

Signed errors similarly change direction: -0.250094 in 2020, 0.140182 in 2021, 0.104214 in 2022, and -0.231421 in 2023. These values are on the transformed edit-count scale, not numbers of missing code

lines. Average gold log scope also varies across years. We avoid exponentiating an average signed log error and calling it the average raw-line error, because nonlinear transformations do not preserve those means.

The distinction between absolute and squared error is substantive Willmott and Matsuura, 2005, and the circumstances favoring one summary over another deserve explicit consideration Hodson, 2022. Here absolute log error emphasizes proportional-scale discrepancies while reducing domination by very large patches. That choice does not establish the loss function a developer would use. A deployment study would need a separately justified cost of underestimating versus overestimating an edit budget.

## 7. What information would be needed for a useful planner?

The result gives a narrow answer: these three descriptors, under this older fitting sample and declared scoring rule, provide only a small and uncertain gain over a constant. It does not rule out richer representations of the issue, repository state, dependency graph, or intended behavioral change. Nor does it show that training a coding assistant on preferences fails. A line-count proxy and a preference-trained repair policy optimize different objects.

Snapshot completeness also differs from complete information. Missing-data reasoning Little and Rubin, 2002 helps distinguish an empty captured hints field from an unobserved development process. Measurement-error analysis Carroll et al., 2006 reinforces that a noisy descriptor can have weak predictive association even when the underlying task property matters. Neither consideration licenses filling unavailable discussions or treating an unmeasured semantic task complexity as if it were an observed covariate.

A calendar association cannot identify the cause of a shift. Repository practices, benchmark selection, issue mix, and repair conventions can all change alongside dates. Causal-graph reasoning Greenland et al., 1999 clarifies why a later-window indicator is not an intervention. Work on predictive uncertainty under distribution shift Ovadia et al., 2019 motivates further stress testing, but it does not convert this metadata-only exercise into evidence about uncertainty calibration of a large coding model.

Large individual losses could shape a small-sample comparison, making influence a reasonable follow-up concern Cook, 1977. We have not removed a tail or selected a different fitting formula after inspecting later outcomes. The displayed mean, year summaries, and full retained records remain available for subsequent investigation. The uncertainty should be read with the context-first interpretation encouraged by Wasserstein and Lazar, 2016, rather than converted into a claim based on proximity to a significance threshold.

An unhelpful proxy can add maintenance without solving the planning problem, connecting this result to technical debt in learning systems Sculley et al., 2015. Production-failure diagnosis by distributed tracing, the topic of Yan, 2026, similarly makes the observational context relevant, but it does not validate our issue-date or edit-size proxy. A prospective extension would need to record information available at the actual planning decision and separate repositories during validation. Dataset documentation Gebru et al., 2021 should therefore state timestamp meanings and benchmark filters rather than treating every dated record as a commensurate operational event.

## 8. Conclusion

An older issue-record fit transferred with a very small average advantage over a constant in later reference edit sizes. The uncertainty, nonmonotone yearly errors, and limited repository overlap prevent a strong temporal-performance claim. This evidence favors treating cheap issue descriptors as a transparent baseline to be challenged, rather than as an established measure of what a coding assistant should change.

## References


1. Bird, Christian; Rigby, Peter C.; Barr, Earl T.; Hamilton, David J.; German, Daniel M.; Devanbu, Premkumar. (2009). The promises and perils of mining git. https://doi.org/10.1109/MSR.2009.5069475.
2. Ovadia, Yaniv, et al. “Can You Trust Your Model’s Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift.” *Advances in Neural Information Processing Systems*, vol. 32, 2019. https://arxiv.org/abs/1906.02530.
3. White, Halbert. “A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity.” *Econometrica*, vol. 48, no. 4, 1980, pp. 817-838. https://doi.org/10.2307/1912934.
4. Hodson, Timothy O.. (2022). Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not. Geoscientific Model Development. https://doi.org/10.5194/gmd-15-5481-2022.
5. Lin, Derrick; Koppel, James; Chen, Angela; Solar-Lezama, Armando. (2017). QuixBugs: a multi-lingual program repair benchmark set based on the quixey challenge. Proceedings Companion of the 2017 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity. https://doi.org/10.1145/3135932.3135941.
6. Little, Roderick J. A., and Donald B. Rubin. *Statistical Analysis with Missing Data*. Wiley, 2002. https://doi.org/10.1002/9781119013563.
7. Sculley, D., et al. "Hidden Technical Debt in Machine Learning Systems." *Advances in Neural Information Processing Systems*, vol. 28, 2015. https://papers.nips.cc/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html
8. Wasserstein, Ronald L.; Lazar, Nicole A.. (2016). The ASA Statement on p -Values: Context, Process, and Purpose. The American Statistician. https://doi.org/10.1080/00031305.2016.1154108.
9. Greenland, Sander, Judea Pearl, and James M. Robins. “Causal Diagrams for Epidemiologic Research.” *Epidemiology*, vol. 10, no. 1, 1999, pp. 37-48. https://doi.org/10.1097/00001648-199901000-00008.
10. Liu, Jiawei; Xia, Chunqiu Steven; Wang, Yuyao; Zhang, Lingming. (2023). Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. arXiv:2305.01210. https://arxiv.org/abs/2305.01210.
11. Yan, Biyun. (2026). From Observability to Causality: Detecting Production Failures in Microservices with Distributed Tracing. 2026 9th International Conference on Computer Information Science and Application Technology (CISAT), 1522-1525. https://doi.org/10.1109/CISAT71030.2026.11682113.
12. Jimenez, Carlos E.; Yang, John; Wettig, Alexander; Yao, Shunyu; Pei, Kexin; Press, Ofir; Narasimhan, Karthik. (2023). SWE-bench: Can Language Models Resolve Real-World GitHub Issues?. arXiv:2310.06770. https://arxiv.org/abs/2310.06770.
13. Chen, Mark; Tworek, Jerry; Jun, Heewoo; Yuan, Qiming; Pinto, Henrique Ponde de Oliveira; Kaplan, Jared; Edwards, Harri; Burda, Yuri; Joseph, Nicholas; Brockman, Greg; Ray, Alex; Puri, Raul; Krueger, Gretchen; Petrov, Michael; Khlaaf, Heidy; Sastry, Girish; Mishkin, Pamela; Chan, Brooke; Gray, Scott; Ryder, Nick; Pavlov, Mikhail; Power, Alethea; Kaiser, Lukasz; Bavarian, Mohammad; Winter, Clemens; Tillet, Philippe; Such, Felipe Petroski; Cummings, Dave; Plappert, Matthias; Chantzis, Fotios; Barnes, Elizabeth; Herbert-Voss, Ariel; Guss, William Hebgen; Nichol, Alex; Paino, Alex; Tezak, Nikolas; Tang, Jie; Babuschkin, Igor; Balaji, Suchir; Jain, Shantanu; Saunders, William; Hesse, Christopher; Carr, Andrew N.; Leike, Jan; Achiam, Josh; Misra, Vedant; Morikawa, Evan; Radford, Alec; Knight, Matthew; Brundage, Miles; Murati, Mira; Mayer, Katie; Welinder, Peter; McGrew, Bob; Amodei, Dario; McCandlish, Sam; Sutskever, Ilya; Zaremba, Wojciech. (2021). Evaluating Large Language Models Trained on Code. arXiv:2107.03374. https://arxiv.org/abs/2107.03374.
14. Tao, Hongmo; Kamata, Hiroyuki. (2026). Teaching Coding Assistants What to Change: Fine-Tuning from Pairwise Preferences. Proceedings of the Second Workshop for REsearch on Agent Language Models (REALM 2026), 1363-1374. https://doi.org/10.18653/2026.realm-1.95.
15. Varma, Sudhir and Richard Simon. “Bias in error estimation when using cross-validation for model selection.” *BMC Bioinformatics*, vol. 7, no. 1, 2006. https://doi.org/10.1186/1471-2105-7-91.

16. Carroll, Raymond J., David Ruppert, Leonard A. Stefanski, and Ciprian M. Crainiceanu. “Measurement Error in Nonlinear Models.” 2006. https://doi.org/10.1201/9781420010138.

17. Willmott, CJ; Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Climate Research. https://doi.org/10.3354/cr030079.

18. McCabe, T.J.. (1976). A Complexity Measure. IEEE Transactions on Software Engineering. https://doi.org/10.1109/tse.1976.233837.

19. Austin, Jacob; Odena, Augustus; Nye, Maxwell; Bosma, Maarten; Michalewski, Henryk; Dohan, David; Jiang, Ellen; Cai, Carrie; Terry, Michael; Le, Quoc; Sutton, Charles. (2021). Program Synthesis with Large Language Models. arXiv:2108.07732. https://arxiv.org/abs/2108.07732.

20. Cook, R. Dennis. (1977). Detection of Influential Observation in Linear Regression. Technometrics. https://doi.org/10.1080/00401706.1977.10489493.

21. Bacchelli, Alberto; Bird, Christian. (2013). Expectations, outcomes, and challenges of modern code review. 2013 35th International Conference on Software Engineering (ICSE). https://doi.org/10.1109/icse.2013.6606617.

22. Gebru, Timnit, et al. "Datasheets for datasets" *Communications of the ACM*, 2021. https://doi.org/10.1145/3458723.

23. Cameron, A. Colin, and Douglas L. Miller. “A Practitioner’s Guide to Cluster-Robust Inference.” *Journal of Human Resources*, vol. 50, no. 2, 2015, pp. 317-372. https://doi.org/10.3368/jhr.50.2.317.

24. Tashman, Leonard J.. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. International Journal of Forecasting. https://doi.org/10.1016/s0169-2070(00)00065-0.

25. Bergmeir, Christoph; Hyndman, Rob J.; Koo, Bonsoo. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. Computational Statistics & Data Analysis. https://doi.org/10.1016/j.csda.2017.11.003.

26. Princeton NLP. *SWE-bench Lite*. Hugging Face Datasets, public test split, JSON snapshot downloaded 25 Oct. 2026. https://huggingface.co/datasets/princeton-nlp/SWE-bench_Lite.

27. Roberts, David R., et al. “Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure.” *Ecography*, vol. 40, no. 8, 2017, pp. 913-929. https://doi.org/10.1111/ecog.02881.

28. Barr, Earl T.; Harman, Mark; McMinn, Phil; Shahbaz, Muzammil; Yoo, Shin. (2015). The Oracle Problem in Software Testing: A Survey. IEEE Transactions on Software Engineering. https://doi.org/10.1109/tse.2014.2372785.

29. Gneiting, Tilmann. (2011). Making and Evaluating Point Forecasts. Journal of the American Statistical Association. https://doi.org/10.1198/jasa.2011.r10138.

30. Herzig, Kim; Just, Sascha; Zeller, Andreas. (2013). It’s not a bug, it’s a feature: How misclassification impacts bug prediction. https://doi.org/10.1109/ICSE.2013.6606585.