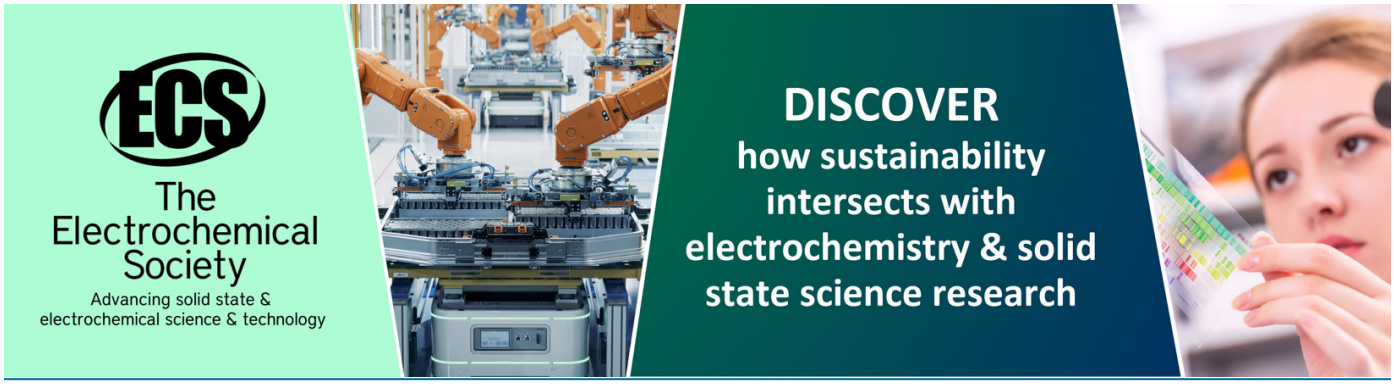
Application and improvement of capture-recapture model for crowdsourced testing



Application and improvement of capture-recapture model for crowdsourced testing

Yi Yao, Yuchan Liu*

Command and Control Engineering College, Army Engineering University of PLA, Nanjing, Jiangsu, 210007, China

*Corresponding author's e-mail:1700710318 @mails.guet.edu.cn

Abstract. Crowdsourced testing has become an effective alternative to traditional testing. However, crowdsourced testing is inherently difficult to manage. In this paper, objective information such as defect count is considered to evaluate the task process. We researched how the capture-recapture model estimates the total number of defects in a crowdsourced testing scenario. By comparing the difference between crowdsourced testing and traditional testing, incremental sampling methodology and the best estimator for crowdsourced testing were proposed to estimate the number of defects, so as to improve the ability to evaluate the completion degree of crowdsourced testing task.

1. Introduction

Crowdsourced testing is an emerging mode. Crowdsourced testing platforms usually plan the end of crowdsourced testing tasks according to the personal experience of project managers. They usually close crowdsourced testing tasks based on time or testers [1-3]. These experience-based decisions can lead to unsatisfactory software quality and can result in wasted testing resources. Therefore, the trade-off of "when to stop testing" in crowdsourced testing is critical [4-6]. Therefore, it is worthwhile to evaluate the test task process by using the objective information of the test, such as the number of defects.

CRC model has been widely applied in biology [7-10], and animal population size is estimated based on overlaps generated by multiple captures [11-16]. Recently, a similar solution [17-20] has been proposed in software engineering. The catch-recapture model in biology can be transferred to software inspection, and the overall scale of software defects can be estimated. The existing CRC models are mainly divided into four types according to the difference of defect detection probability and testers' detection capability, as shown in Table 1. Based on the four types of CRC models, researchers have developed a variety of estimators to calculate the total size number.

In this paper, CRC model is introduced to estimate the number of defects in crowdsourced testing. First, the differences between crowdsourced testing and traditional testing were compared. We used the incremental sampling methodology to deal with all the test report in time order, grouped them as dynamic input data into a fixed size of samples. Then the best CRC estimator was selected through experimental analysis of the number of testers and the number of reports to estimate the defect count. The experiments use data sets of 16 project in crowdsourced testing platform, and the results show that the prediction performance of M_h-SC and M_{th}-SC estimators for crowdsourced testing is significantly better than other estimators, and far better than the prediction results on the traditional data sets, which can basically accurately reflect the completion degree of the test task.

Table 1. Classification of CRC models and estimators.

Models	Tester's detection capability	Bug detection probability	Estimators	Acronym
M_0	Identical	Identical	Maximum Likelihood	M_0-MLE[11]
M_t	Different	Identical	Maximum Likelihood Chaos	M_t-ML[11] M_t-Ch[14]
M_h	Identical	Different	Jackknife Chaos Sample Coverage	M_h-JK[12] M_h-Ch[13] M_h-SC[16]
M_{th}	Different	Different	Sample Coverage	M_{th}-SC[15-16]

2. Improvements in crowdsourced testing scenario

All CRC estimators studied are suitable for traditional testing scenarios. Traditional testing typically delivers software to a testing team in software testing centre, with smaller staff and fewer reports to submit. The frequency of each defect in the report is also greatly low when the test report base is small. Estimation of defects using the CRC model is usually done after the test has been substantially completed to determine whether the test still needs to continue. In this scenario, the training set mainly takes the report of each tester as the unit sample and whole training set is relatively fixed.

However, crowdsourced testing requires a large number of testers, each of whom can submit reports multiple times. The platform needs to process a large number of test reports in a timely manner. The test report is in a constantly changing process and cannot keep the training set stable. Moreover, when the test report base is very large, the frequency of defects with less difficulty in the sample is also greatly increased.

Therefore, this paper compares the two test scenarios and proposes improvement measures for the application of CRC model in crowdsourced testing scenarios.

2.1. Processing test reports

Crowdsourced testing is in a distributed test environment. All testers check for defects in their own test environment and submit test reports to the crowdsourced testing platform over the network, and they can submit report multiple times. In this case, the number of test reports is increasing, and the training set cannot be kept relatively fixed. So, taking all test reports of a tester as a sample does not meet the requirements of real-time prediction. To manage the test process in real time, we use each test report received as a sample. In addition, the platform needs to process the test reports in real time to continuously transform the newly arrived report data into the form that can be directly used as the input data of the prediction model.

CRC models typically use a 0-1 matrix [7], as shown in Table 2. Each row represents a sample received by the platform and each column represents a defect detected. The values of i-th row and j-th column in the table is denoted by X_{ij}.

$$X_{ij}=\begin{cases}1 & \textit{The i-th defect is detected in the j-th report}\\ 0 & \textit{other}\end{cases}$$

Table 2.Test reports defect matrix.

Defect# / Report#	1	2	…	i	…
1	1	0	…	0	…
…	…	…	…	…	…
j	0	1	…	1	…
…	…	…	…	…	…

We adopt incremental sampling methodology [21] to process the test report received by the platform. It improves the reliability and defensibility of sampled data by reducing variability. Considering that crowdsourced testing reports are submitted in chronological order, we add a new blank line at the end of the table when a new test report reach crowdsourced testing platform. And then we check for defects detected in the report. For existing defects in the table, we mark the corresponding value as 1. For defects that are not in the table, we add a new column to the table, which means a new defect, and mark the value as 1 in the new column in the last row. The rest of the spaces in the table are marked with 0. Finally, the training matrix required by the CRC model is generated according to the table.

In the process of crowdsourced testing, we need to maintain the test report defect matrix table dynamically to record all the information of the test reports. For this table, some parameters that are useful for CRC model can be clearly obtained: 1) The number of rows of valid values in the table represents the number of samples; 2) The number of columns of valid values in the table represents the number of different defects captured in all samples; 3) The number of defects found in the j-th report can be obtained by summation of the values in the j-th line; 4) The number of times that the i-th defect is detected be obtained by summation of the values in the j-th column. These important parameters can be obtained by simple processing of the matrix table, which can greatly improve the computational efficiency of the CRC model.

2.2. Selecting the best estimator

Biological researchers generally use large data sets [8,16,20] to conduct comprehensive assessments of CRC models, while software engineering studies have been limited to relatively small data sets, usually consisting of a small number of testers and defects [13,14,23]. Recommendations and comparisons of defect estimation results in software engineering and animal size estimation results are based on these limited data sets.

2.2.1. The number of testers. A survey [22] shows that in the traditional testing data set, there are rarely test teams have more than 8 members, and test teams contain less than 5 members account for 25%. In fact, crowdsourced testing belongs to the category with a large number of testers, and the number of testers is usually over 80. Since the CRC model works based on the amount of overlap of defects detected by different testers, it is unclear to a large number of testers what impact the CRC model's performance will have. The direct application of the CRC model to the dataset of crowdsourced testing may result in a reduction in accuracy of prediction. Therefore, it is necessary to analyze the various estimators of CRC model based on the different data sets where the number of testers is different. Briand [22] has analyzed the impact of the number of testers on these estimators mostly in table 1. They compared a dataset of 73 testers and found two conclusions: 1) As the number of testers increased, the accuracy and accuracy of the estimator improved directly; 2) With the increase in the number of testers, the accuracy of M_h-CH, M_h-SC, and M_{th}-SC estimators improves faster than other estimators. The number of testers in the dataset in this study may be much more than 73. Therefore, we apply the conclusions of this large dataset to the crowdsourced testing scenario, and M_h-CH, M_h-SC and M_{th}-SC estimators will be the main focus in this experiment.

2.2.2. Defect detection frequency. The traditional testing datasets have small base and the frequency of defect detected is low. Typically, each tester will be able to detect a small number of defects, and the number of defects being recaptured will be small, perhaps only once or twice per defect. In this situation of sparse data [23], M_h-JK and other estimators are no longer applicable, and new low-order estimators M_t-CH and M_h-CH are proposed to correct errors. In contrast, crowdsourced testing datasets have a large base and defects are detected with a relatively high frequency. Most defects will be found by multiple testers, so crowdsourced testing is more suitable for high-order estimators of CRC model, such as M_0-ML, M_t-ML, M_h-JK, M_h-SC, and M_{th}-SC, etc.

3. Experimental Verification

3.1. Research Problems

Q1: Whether all CRC estimators can accurately predict the number of defects according to crowdsourced testing datasets? Is there a CRC estimator with the best estimated performance?

Q2: What is the difference between the estimated results of the CRC model for the datasets of traditional testing and crowdsourced testing? Can the estimated performance of the best estimator in crowdsourced testing exceed that of traditional testing?

Q3: Does the number of defects estimated by the CRC model accurately reflect the degree of completion of the test task?

3.2. Experimental setup

The datasets analysed by this experiment is from the crowdsourced testing platform of MOOC [24]. These test projects examined in this study were developed for use in software testing competitions and were seeded with different numbers of defects in advance. The test involved nearly hundreds of testers. In order to make a comparison with the traditional testing, the experiment also delivered the crowdsourced testing task to 5 students in the laboratory for the traditional team testing. Therefore, the defect datasets of the traditional testing of the same software can be obtained. These datasets will be used to compare the predicted performance with the crowdsourced testing data set on the CRC model.

In order to examine the estimated performance of the CRC model on the crowdsourced testing dataset, we used the following two evaluation indicators to evaluate the improved model: relative error (RE) and defect coverage (DC).

3.3. Experimental results

3.3.1. Answers to Q1

The experiment conducted defect number estimation for 16 crowdsourced testing datasets. Table 4 shows the median and variance of RE of the final prediction results of all seven CRC estimators.

Table 3. Prediction RE of different estimators.

	M_0-MLE	M_t-ML	M_t-Ch	M_h-JK	M_h-Ch	M_h-SC	M_{th}-SC
RE (median)	0.03	0.03	0.02	0.07	0.03	0.02	0.01
RE (variance)	0.08	0.07	0.06	0.13	0.10	0.04	0.03

It can be seen that the median values of most projects are close to 0 in the end, indicating that the predicted number of defects is almost the same as the actual number of defects. Mth-SC has the minimum median value, followed by Mh-SC and Mt-CH, while Mh-JK has a relatively poor prediction performance. From variance, prediction of Mth-SC estimator is quite accurate. Although the median values of Mh-SC and Mt-CH are the same, the variance values of Mt-CH is much higher than that of Mh-SC. By comparing the median and variance values of RE, the experiment shows that Mh-JK estimator is not effective in crowdsourced testing. Mt-CH, Mh-SC, and Mth-SC estimators are always accurate in predicting the total number of defects in crowdsourced testing, and are significantly better than other estimators. The number of predicted defects is almost equal to the actual situation, with a deviation of less than 5%.

3.3.2. Answers to Q2

Two types of defect datasets of crowdsourced testing and traditional testing were estimated by using best estimators (Mth-SC), and the estimated results were compared. Table 5 shows the median and variance of RE in the whole estimation process.

Table 4. Prediction RE of Mth-SC estimators.

	Test scenarios	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
RE (median)	Traditional	0.29	0.20	0.15	0.13	0.10	0.08	0.06	0.06	0.05	0.05
	crowdsourced	0.32	0.22	0.16	0.12	0.08	0.05	0.03	0.02	0.01	0.01
RE (variance)	Traditional	0.27	0.24	0.22	0.20	0.19	0.17	0.14	0.12	0.11	0.10
	crowdsourced	0.30	0.30	0.29	0.23	0.15	0.20	0.15	0.17	0.10	0.04

At the beginning of the task, the decline rate of median and variance of traditional testing is much higher than that of crowdsourced testing. The decline rate gradually decreases near the end of the task. This may be due to the large base of test reports for crowdsourced testing. The number of valid reports was too small to accurately predict the number of defects when the test reports were initially selected proportionally. With the increase of the test reports, the median value of RE on the datasets is decreasing, and the prediction results become more and more accurate. The experiment shows that the median and variance values of Mh-SC and Mth-SC are both lower than the traditional testing, indicating that the estimation performance of CRC model in crowdsourced testing exceeded the traditional testing.

The detailed performance of the Mth-SC (best estimator) during the estimation process is shown in Figure 1.

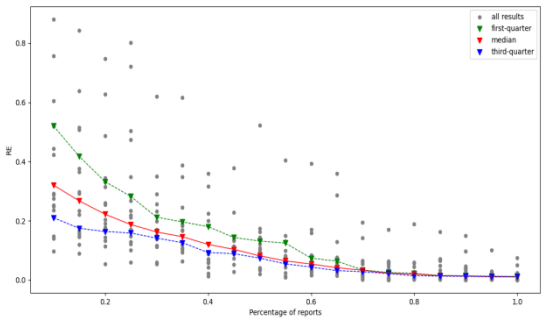


Figure 1. RE of Mth-SC estimators.

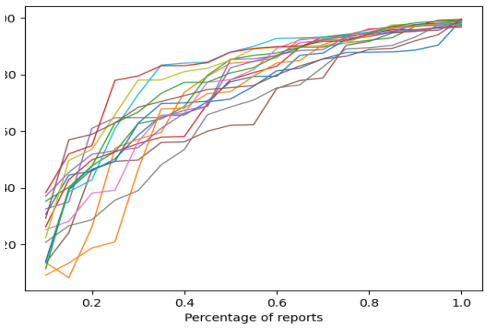


Figure 2. DC trend diagram of projects for the Mth-SC estimator.

3.3.3. Answers to Q3

In order to be able to assess test task progress according to number of defects, we analysed the DC in the estimation process. Figure 2 shows the DC of all crowdsourced testing projects by the Mth-SC estimator during the entire estimation process.

The DCs of all projects increase as the number of test reports increases and eventually stabilize. At the end of the task, the number of projects whose DC is more than 95% accounted for about 93.75% of the total. It can be seen that the number of defects estimated by the CRC model can basically accurately reflect the degree of completion of the test task.

4. Conclusion

In this paper, we apply CRC model to crowdsourced testing to predict the total number of potential defects in the software. We have improved the original CRC model. Not only is the incremental sampling methodology presented to handle a large number of test reports, but also the best estimator suitable for crowdsourced testing is selected based on the number of testers and the frequency of defect detection. The model can improve the accuracy of defect number prediction and reflect the completion of test tasks well.

Acknowledgments
This work is supported by National Key Research and Development program of China (No: 2018YFB1403400), National Natural Science Foundation of China (No: 61702544), National Natural Science Foundation of Jiangsu Province, China (No: BK20160769, Bk20141072), and China Postdoctoral Science Foundation (No: 2016M603031).

References
[1] Harman, Mark, Mao, et al. A survey of the use of crowdsourcing in software engineering[J]. Journal of Systems & Software, 2017.
[2] Zhang X, Feng Y, Liu D, Chen Z, and Xu B. Research progress of crowdsourced software testing[J]. Journal of Software, vol. 29(1), pp. 69–88, 2018.
[3] https://www.softwaretestinghelp.com/
[4] Lewis, William E. Software testing and continuous quality improvement[M]. CRC press, 2016.
[5] Garg M, Lai R, Huang S J. When to stop testing: a study from the perspective of software reliability models[J]. IET Software, 2011, 5(3):263-273.
[6] Iqbal J, Ahmad N, Quadri S M K. A Software Reliability Growth Model with Two Types of Learning[C]// IEEE Second International Conference on Image Information Processing. IEEE, 2014.
[7] Wang J, Yang Y, Krishna R, et al. iSENSE: Completion-Aware Crowdtesting Management[C]// 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). ACM, 2019.
[8] Shan Q, Rong G, Zhang H, et al. An Empirical Evaluation of Capture-Recapture Estimators in Software Inspection[C]// Software Engineering Conference. IEEE, 2015.
[9] Rong G, Zhang H, Shao D. Does Detecting more Defects Contribute to Better Estimation? An Empirical Investigation to the Capture-Recapture Method[J]. 2015.
[10] Liu G, Rong G, Zhang H, et al. The adoption of capture-recapture in software engineering[C]// International Conference on Evaluation & Assessment in Software Engineering. ACM, 2015.
[11] Otis D L K, Burnham K P, White G C, et al. Statistical Inference From Capture Data on Closed Animal Populations[J]. Journal of the American Statistical Association, 1978, 76(374).
[12] Chao A. Estimating Population Size for Sparse Data in Capture-Recapture Experiments[J]. Biometrics, 1989, 45(2):427-438.
[13] Burnham K P, Overton W S. Estimation of the size of a closed population when capture probabilities vary among animals[J]. Biometrika, 1978,65(3):625-633.
[14] Chao A. Estimating the population size for capture-recapture data with unequal catchability.[J]. Biometrics, 1987, 43(4):783-791.
[15] Chao A, Lee S M, Jeng S L. Estimating Population Size for Capture-Recapture Data When Capture Probabilities Vary by Time and Individual Animal[J]. Biometrics, 1992, 48(1):201-216.
[16] Chao L A. Estimating population size via sample coverage for closed capture-recapture models.[J]. Biometrics, 1994, 50(1):88-97.
[17] Eick S G, Loader C R, Long M D, et al. Estimating software fault content before coding[C]// International Conference on Software Engineering. IEEE, 1992.
[18] M. Roper, Estimating Fault Numbers Remaining After Testing[C]// 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, Luxembourg, 2013, pp. 272-281.
[19] Walia G S, Carver J C. Evaluating the Effect of the Number of Naturally Occurring Faults on the Estimates Produced by Capture-Recapture Models[C]// International Conference on Software Testing Verification & Validation. IEEE, 2009.
[20] Briand L C, El Emam K, Freimut B G, et al. A Comprehensive Evaluation of Capture-Recapture Models for Estimating Software Defect Content[J]. Software Engineering IEEE Transactions on, 2000, 26(6):518-540.

[21] L. Mora-Applegate and M. Malinowski, Incremental sampling methodology[C]// Interstate Technology and Regulatory Council (ITRC), Tech. Rep., 2012.
[22] G. Walia, J. Carver and N. Nagappan, The effect of the number of inspectors on the defect estimates produced by capture-recapture models[C]// 2008 ACM/IEEE 30th International Conference on Software Engineering, Leipzig, 2008, pp. 331-340.
[23] G. S. Walia and J. C. Carver, The effect of the number of defects on estimates produced by capture-recapture models[C]// 2008 19th International Symposium on Software Reliability Engineering (ISSRE), Seattle, WA, 2008, pp. 305-306.
[24] http://www.mooctest.net/login2