Risk of the ecological fallacy: the elephant in the empirical software engineering room

Bennett Narron
North Carolina State University
Raleigh, North Carolina
bynarron@ncsu.edu

Akond Rahman
North Carolina State University
Raleigh, North Carolina
aarahman@ncsu.edu

Manish Singh
North Carolina State University
Raleigh, North Carolina
mrsingh@ncsu.edu

ABSTRACT

We will compose the abstract when the paper is finished.

Keywords

Ecological Inference, Ecological Fallacy, Aggregation, Empirical Software Engineering

1. INTRODUCTION

Posnett *et al.* (2011) challenged previous works regarding the empirical study of software systems by positing that methodologies for collecting and examining data in software projects did not take into account the risk of the *ecological fallacy*–a logical fallacy that arises from inferring conclusions across different levels of aggregation [16]. The hierarchical structure of a software system presents many levels of aggregation for examination. A project may be decomposed into modules, packages, or files, each of which may be empirically evaluated for quality, distribution, collaboration, and productivity. As the study of a population may provide some insight about its individuals, studying the various levels of aggregation of a software system is may be useful for understanding, for instance, where software defects exists, when and why they surface, and how best to minimize them. Using one level of aggregation (e.g., a population) to infer information about some dis-aggregated entity (e.g., an individual) is called *ecological inference*. Ecological inference implies the risk of the ecological fallacy–an idea that had been overlooked prior to the publication of *Understanding ecological inference in empirical software engineering*.

Choosing the correct level of aggregation (e.g., file-level) is critical when testing hypotheses on one or more of these variables [16]. But where should one start? Top-to-bottom? At file-level? The authors set forth the understand the intricacies of ecological inference and how well hypotheses hold up across differing levels of aggregation.

This paper is intended to provide an overview of the effect of the risk of the ecological fallacy on empirical software engineering, before and after the publication by Posnett *et al.* . We will begin by providing a brief background on ecological inference and the risk of ecological fallacy, and followed by a high-level overview of the study of empirical software engineering. We will then begin discussing a selection of papers published prior to the acknowledgement of the risk of the ecological fallacy in empirical software engineering. We will follow the brief survey by further discussion of the paper by Posnett *et al.* , and then we will continue with publications that followed and how they attempt address the elephant in the room. We will conclude with some discussion about the effectiveness of this newfound awareness and how effectively it is being addressed.

2. BACKGROUND AND CONCEPTS

Before we begin with the survey, it will be beneficial to have a proper introduction to the origins of ecological inference and the risk of ecological fallacy and a greater understanding of the aims of empirical software engineering. An overview of these topics will provide a modest foundation for understanding the quandary that the risk of the ecological fallacy imposes on empirical software engineering.

2.1 Ecological Inference and the Risk of Ecological Fallacy

In criticism of medical statistical methods, *The Economist*, a publication famous for its editorials on everything zeitgeist, printed an article titled, *Signs of the Times*, that derived statistically significant correlations between emergency room patients, their ailments, and their astrological signs [19]. They reported that those born under the zodiac sign of Leo are 15% more likely to be admitted to the hospital for "gastric bleeding" than those born under any other sign, while Sagittarians are 38% more likely to be laid up with a broken arm. Both of the aforementioned statistics meet the typical 95% certainty threshold. So what in Hades is going on here?

Piantadosi *et al.* , in a publication in the *American Journal of Epidemiology* in 1988, would argue that ecological analyses suffer from the same confounding biases as individual-level analyses, though perhaps more severely [15]. Ecological inference introduces a vast amount of moving parts, as the factors that affect the targeted study group are compounded with those of the aggregation of that group, leading to an abundance of confounding factors and threats to validity to consider. Piantadosi *et al.* offer no solution beyond an urge to researchers to analyze data sensibly and perform *ad hoc* measures to minimize confounding bias. Thus, as statistical analyses of the most granular subject comes with its own set of potential biases, ecological inference comes with the risk of the ecological fallacy. So, Sagittarians need not worry about their brittle arms or Leos about their bloody guts, as "statistically-significant" correlations are easy to spot when due diligence has not been performed to eliminate as much bias as possible.

2.2 Empirical Software Engineering

Empirical Software Engineering (ESE) aims for observable outcomes in the study of software systems, including quality and productivity [16]. Ideally, these studies are conducted using large sample sizes (e.g. numerous software projects) in order to leverage statistical methods for hypothesis testing and to gather large quantities of data for mining methods and machine learning for building software tools to support programming tasks [16]. Inevitably, the need to decompose data, such as software projects, into analyzable data creates the opportunity for ecological inference to occur and, thus, an increased risk in the ecological fallacy.

3. SURVEY OVERVIEW

Note that the organization of the following sections occurs chronologically, beginning with papers published prior to 2011, follow by a summary of the paper by Posnett *et al.* on ecological inference, and concluding with papers published in 2012 or later. Each subsection may be viewed as a singleton summary of a publication. The reader may read them all serially or at whim (like watching episodes of *Seinfeld*), as more in-depth discussion will occur after all papers have been presented.

4. SURVEY - APPROACHING 2011

The following papers were composed before Posnett *et al.* published on the prevalence of the ecological inference in Empirical Software Engineering. For each paper, we will provide a bullet point format, defining keywords described by each author, summarizing select ideas presented in each paper, and finally, describing how the paper used ecological inference to test hypotheses or perform analysis.

4.1 Mining metrics to predict component failures [12]

4.1.1 *Keywords:*

- *Software metrics*: Measurement tools derived from software repositories to perform quantitative analysis.
- *Fault prediction*: A methodology to predict failures by minign and analyzing software repositories.

4.1.2 *Summary:*

TODO:// A brief summary of the paper.

4.1.3 *Use of Ecological Inference*

Nagappan *et al.* took a holistic approach where analysis was performed on one level (did not differentiate between aggregated or dis-aggregated level).

4.2 Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings [9]

4.2.1 *Keywords:*

- *Classification*: This term refers to the categorization of data values into discrete classes. For example, a binary classification will have two categories or classes - ”YES” and ”NO”.
- *Software Defect Prediction*: This term refers to the process of identifying error prone software modules by means of data mining techniques. This helps in efficient allocation of resources to high-risk software segments.
- *Data Mining*: It is a process of analyzing data from various sources and analyzing and deriving useful insights and patterns from the data , that can be used to make better decision in future.
- *Statistical Methods*: Statistics deals with the collection, interpretation, presentation and analysis of data. It offers various methods to estimate and correct for any bias within a sample and data collection procedures.

4.2.2 *Summary:*

TODO:// A brief summary of the paper.

4.2.3 *Ecological Inference in ESE*

Posnett *et al.* provided a conceptual framework on how prediction models can vary if metrics that are collected at aggregated with that of metrics collected at non-aggregated level. The ”Ecological Inference in Empirical Software Engineering” paper makes use of the conclusion from ”Benchmarking Classification Models..” paper - that simple statistic measures like TPR, FPR do not work well in a software defect prediction context as it is possible for two groups to use the same model on same data set and yet come with different results just because they had different threshold values. So the authors use ROC and statistical testing methods in their experiment to model defects.

4.3 Predicting failures with developer networks and social network analysis [10]

4.3.1 *Keywords:*

- *Developer network*: A network that infers the developer dependencies for the developed software artifacts.
- *Social network analysis (SNA)*: Analyzing the collaborative structure that can be inferred directly, and indirectly from software module dependencies and developer dependencies.
- *Failure prediction*: A methodology to predict failures by mining and analyzing software repositories.

4.3.2 *Summary:*

TODO:// A brief summary of the paper.

4.3.3 *Use of Ecological Inference*

Meneely *et al.* took a holistic approach where social network analysis was performed on one level.

4.4 The Influence of Organizational Structure on Software Quality: An Empirical Case Study [13]

4.4.1 *Keywords:*

- *Organizational Structure*: The hypothetical structure that is used to categorize the responsibilities and assigner-assignee relationship inside that organization.

- *Software Quality Metric*: Empirical concepts to evaluate the quality of a software
- *Traditional Software Quality Metrics*: Empirical concepts that are derived from software artifacts to quantify/or predict software quality
- *Organizational Software Quality Metrics*: Empirical concepts that are derived from organizational aspects of a software organization

4.4.2 Summary:

TODO:// A brief summary of the paper.

4.4.3 Use of Ecological Inference

Nagappan *et al.* take a holistic organizational approach, disregarding the difference of aggregated and dis-aggregated levels.

4.5 Does Distributed Development Affect Software Quality?: An Empirical Case Study of Windows Vista [5]

4.5.1 Keywords:

- *Distributed Development*: This term refers development of software that is spread across various levels - building, campus, locality, continent, world. Different teams working at different locations work together to build a complex software system.
- *Collocated Development*: This refers to development of software systems in which the teams are working in the same location - say same building or floor. In collocated development, teams can reach out to each other and communication and collaboration becomes relatively easy.
- *Software Quality*: This refers to the quality of software that is being produced. Quality is typically measured in terms of failure/bugs found in the software. A poor quality software will have more bugs and encounter failures.
- *Outsourcing*: It is a special case of distributed development model in which different companies work on building a complex software system.

4.5.2 Summary:

TODO:// A brief summary of the paper.

4.5.3 Use of Ecological Inference

Bird *et al.* take a holistic approach, and discuss the global distributed model of software development, at different levels of separation and its effect on software quality.

4.6 Cross-project defect prediction: a large scale experiment on data vs. domain vs. process [20]

4.6.1 Keywords:

- *Post-release Defects*: Defects of a project that are discovered after the release of the software.
- *Cross-project Defect Prediction*: A methodology to use one defect prediction model, to predict defects of another project.
- *Similarity between Projects*: Metrics that measure the similarity between two projects.

4.6.2 Summary:

TODO:// A brief summary of the paper

4.6.3 Use of Ecological Inference

Posnett *et al.* provided a conceptual framework on how prediction models can vary if metrics that are collected at aggregated with that of metrics collected at non-aggregated level. This paper does not address the findings achieved at local and global level of software projects.

4.7 A systematic and comprehensive investigation of methods to build and evaluate fault prediction models [1]

4.7.1 Keywords:

- *Fault Prediction Models*: "binary classifiers typically developed using one of the supervised learning techniques from either a subset of the fault data from the current project or from a similar past project." [8]
- *Fault Proneness*: the number of defects detected in a software component (e.g., class)
- *Cost-Effectiveness*: refers to the return on investment of time and resources expended to acheive a desired outcome. In the context of this paper, cost-effectiveness is measured per model by the ratio of the percentage of the number of lines of code examined (% NOS) to the percentage of totals faults discovered (% faults). Cost effectiveness is used as a variable for success criteria.
- *Verification*: in testing, the act of ensuring that a particular software system meets specifications and meets its intended purpose.

4.7.2 Summary:

TODO:// A brief summary of the paper.

4.7.3 Use of Ecological Inference

While it seems that predecessory related works focused on performance metrics and evaluation of various singular or combinatorial data-mining and fault prediction models, this paper had the intension of systematically exploring the space, and the authors were eager for results. The outcome was modest, suggesting that what is best for any given system is highly dependent upon the evaluation criteria applied. The authors conclude that it is important that predictive models are justified in context by any evaluation criteria. These results paved the way for the systematic approach to predictive modeling employed by the authors of the original paper, where evaluation criteria of such models was questioned at different levels of aggregation/disaggregation– a shortcoming of the procedures documented in this paper.

4.8 Studying the impact of social structures on software quality [2]

4.8.1 Keywords:

- *Measure of Discussion Contents*: Metrics related to software project development discussion that can be extracted from bug reports, stack traces, source code, and software repositories to measure software quality and software defects.
- *Measure of Social Structures*: Metrics related to developer dependencies during software development, and maintenance that can be extracted from source code, and software repositories to measure software quality and software defects.
- *Measure of Communication Dynamics*: Metrics related to developer communications during software development, and maintenance namely number of messages, length of messages, reply time, and interestingness, and workflow measures. These measures can be extracted from source code, and software repositories to measure software quality and software defects.
- *Post-Release Defects*: Software defects and failures discovered after deployment or release of software.

4.8.2 Summary:

TODO:// A brief summary of the paper.

4.8.3 Use of Ecological Inference

Posnett *et al.* provided a conceptual framework on how prediction models can vary if metrics that are collected at aggregated with that of metrics collected at non-aggregated level. This paper performed a holistic approach where analysis was performed on one level (did not differentiate between aggregated or dis-aggregated level).

4.9 Studying the impact of dependency network measures on software quality [14]

4.9.1 Keywords:

- *Dependency network*: An illustration where software modules are presented as nodes, and edges are represented for dependency between two modules.
- *Ego network*: A network that consists of the modules itself and the other modules it is dependent on.
- *Global network*: A network that consists of all the modules and all the other modules each other is dependent on.

4.9.2 Summary

TODO:// A brief summary of the paper

4.9.3 Use of Ecological Inference

Nguyen *et al.* analyze social network analysis on two levels namely local, and global which are labeled as ego, and global, respectively.

5. ECOLOGICAL INFERENCE IN ESE

Discussion about Posnett, et al paper

6. SURVEY - FORWARD FROM 2011

The following papers were composed after Posnett *et al.* published on the prevalence of the ecological inference in Empirical Software Engineering. For each paper, we will provide a bullet point format, defining keywords described by each author, summarizing select ideas presented in each paper, and finally, describing how the paper addressed ecological inference to perform more informed hypothesis testing and analysis.

6.1 Think Locally, Act Globally: Improving Defect and Effort Prediction Models [3]

6.1.1 Keywords:

- *Global Model*: A prediction model that is created by using all the features available in the dataset.
- *Local Model*: A prediction model that is created by using a subset of the features available in the dataset. The other subsets are used in testing the predictive performance of the prediction model.
- *Goodness of Fit*: A statistical testing mechanism that is used to test the predictive performance of any prediction model. Example of some popular goodness of fit tests include Pearson's test, chi-squared test, R-square measure (used in this paper) etc.
- *Connecting Global and Local Models*: The concept of using results from a local model and use it to build a better predictor at the Global level. Posnett et al. in their work [16] stated that the same prediction model can behave differently at different levels of the same software. Menzies et al. [11] stated that smaller subsets of a typical software engineering dataset can provide better insight with respect to prediction. In this paper, Bettenburg et al. uses these findings to build better global models.

6.1.2 Summary:

TODO:// A brief summary of the paper.

6.1.3 Acknowledgement of Ecological Inference

Bettenburg *et al.* use the findings Posnett *et al.* as a motivational aspect and try to discover if the prediction results obtained at a dis-aggregated level can be applied to get better prediction results at the aggregated level.

6.2 Recalling the imprecision of cross-project defect prediction [18]

6.2.1 Keywords:

- *Empirical Software Engineering*: Focuses on experiments involving software systems (software products, processes, and resources). The purpose of these experiments is to collect data that can be used to validate theories about the processes involved in software engineering.
- *Fault Prediction*: To make "use of a plethora of structural measures in order to predict faults, even in the absence of a fault history... Example structural measures include lines of code, operator counts, nesting

depth, message passing coupling, information flow-based cohesion, depth of inheritance tree, number of parents, number of previous releases the module occurred in, and number of faults detected in the module during the previous release [4]." Accurate fault prediction at an early stage of development, despite the lifecyle phase, allow for code to be fixed at a lower cost [4].

- *Code Inspection*: The practice of reviewing code for any defects or process improvements. May be performed by person, team of people, and/or a model built for fault detection.
- *Cross-Project Prediction*: "using data from one project to predict defects in another [18]." As the title of the paper suggests, this is an imprecise practice.

6.2.2 Summary:

TODO:// A brief summary of the paper.

6.2.3 Use of Ecological Inference

Rahman *et al.* demonstrate the consideration of the risk of ecological inference before proceeding, explicitly stating that their choice of file-level analysis was directly motivated by the publication by Posnett *et al.* .

6.3 Bug prediction based on fine-grained module histories [7]

6.3.1 Keywords:

- *Bug Prediction*: This term refers to the ability to predict a future bug by building a model using the historical metrics, which are mined from version histories of software modules.
- *Fine-grained Prediction*: This term refers to the use of method-level details while using historical metrics, mined from version histories of software modules, to build a model for predicting bugs.
- *Fine-grained Histories*: It refers to the use of fine-grained metrics, by creating metadata or a structure which will store the history or change information at very fine grained level , for example, capturing details of a method level change instead of package level or file level change alone.
- *Historical Metrics*: They are metrics coming from various categories such as code-related metrics, process-related metrics, organizational metrics and geographical metrics for capturing version history in different categories.

6.3.2 Summary:

TODO:// A brief summary of the paper.

6.3.3 Use of Ecological Inference

Ecological Inference in Empirical Software Engineering makes use of the conclusion from *Benchmarking Classification Models for software defect prediction* - that simple statistic measures like TPR, FPR do not work well in a software defect prediction context as it is possible for two groups to use the same model on same data set and yet come with different results just because they had different threshold values. So the authors use ROC and statistical testing methods in their experiment to model defects. The authors of this paper use this information and argue that method-level fine-grained historical metrics give interesting bug-prediction models which yield better results as compared to file-level or package-level historical metics.

6.4 Method-level bug prediction [6]

6.4.1 Keywords:

- *Bug Prediction*: This term refers to the ability to predict a future bug by building a model using the historical metrics, which are mined from version histories of software modules.
- *Fine-grained Prediction*: This term refers to the use of method-level details. Here, the author computes source code metrics at the method level along with change metrics to do fine-grained prediction
- *Fine-grained source code changes*: It refers to the code changes happening in the source code at the fine-grained level of methods as compared to changes observed at the file-level or the package-level.
- *Code Metrics*: They are metrics which are directly computed from the source code itself. There are two traditional suites of code metrics : CK metrics and a SCM, which is a set of metrics directly computed at the method level. They are used by author to generate bug prediction models.

6.4.2 Summary:

TODO:// A brief summary of the paper.

6.4.3 Use of Ecological Inference

Posnett *et al.* provided a conceptual framework on how prediction models can vary if metrics that are collected at aggregated with that of metrics collected at non-aggregated level. The "Ecological Inference in Empirical Software Engineering" paper makes use of the conclusion from "Benchmarking Classification Models.." paper - that simple statistic measures like TPR, FPR do not work well in a software defect prediction context as it is possible for two groups to use the same model on same data set and yet come with different results just because they had different threshold values. So the authors use ROC and statistical testing methods in their experiment to model defects. The authors of this paper use this information and argue that method-level fine-grained changed metrics give interesting bug-prediction models which yield better results as compared to file-level or package-level metrics or method-level source code metrics.

6.5 How, and why, process metrics are better [17]

6.5.1 Keywords:

- *performance*: The authors "compare the performance of different models in terms of both traditional measures such as AUC and F-score, and the newer cost-effectiveness measures"

- *stability*: The authors "compare the stability of prediction performance of the models across time and over multiple releases"
- *portability*: The authors "compare the portability of prediction models: how do they perform when trained and evaluated on completely different projects?"
- *stasis*: The authors "study stasis, viz.., the degree of change (or lack thereof) in the different metrics, and the corresponding models over time. [They] then releate these changes with their ability to prevent defects."

6.5.2 Summary:

TODO:// A brief summary of the paper.

6.5.3 Use of Ecological Inference

It appears that every paper that follows (chronologically) Posnett *et al.* 's recognition of the risk of ecological fallacy has to, in some way at least once, describe some measure to taken to avoid the risk of it occurring. This paper is not an exception. In threats to validity, as mentioned previously, the authors describe an extra measure taken to cross-compare their results with analyses run per project. It appears that, even today, researchers in defect prediction are still determining what the ecological fallacy means in terms of the organization of their research and their analysis and results.

7. DISCUSSION

Discuss what changed, how did it change, is it sufficient?

Posnett *et al.* raised an issue that the community has not yet fully realized how to sort out.

8. CONCLUSIONS

text here

9. REFERENCES

[1] E. Arisholm, L. C. Briand, and E. B. Johannessen. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. *Journal of Systems and Software*, 83(1):2–17, 2010.

[2] N. Bettenburg and A. E. Hassan. Studying the impact of social structures on software quality. In *Program Comprehension (ICPC), 2010 IEEE 18th International Conference on*, pages 124–133. IEEE, 2010.

[3] N. Bettenburg, M. Nagappan, and A. E. Hassan. Think locally, act globally: Improving defect and effort prediction models. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, pages 60–69. IEEE Press, 2012.

[4] D. Binkley, H. Feild, D. Lawrie, and M. Pighin. Software fault prediction using language processing. In *Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION, 2007. TAICPART-MUTATION 2007*, pages 99–110. IEEE, 2007.

[5] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy. Does distributed development affect software quality?: an empirical case study of windows vista. *Communications of the ACM*, 52(8):85–93, 2009.

[6] E. Giger, M. D'Ambros, M. Pinzger, and H. C. Gall. Method-level bug prediction. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 171–180. ACM, 2012.

[7] H. Hata, O. Mizuno, and T. Kikuno. Bug prediction based on fine-grained module histories. In *Proceedings of the 34th International Conference on Software Engineering*, pages 200–210. IEEE Press, 2012.

[8] Y. Jiang, J. Lin, B. Cukic, and T. Menzies. Variance analysis in software fault prediction models. In *Software Reliability Engineering, 2009. ISSRE'09. 20th International Symposium on*, pages 99–108. IEEE, 2009.

[9] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *Software Engineering, IEEE Transactions on*, 34(4):485–496, 2008.

[10] A. Meneely, L. Williams, W. Snipes, and J. Osborne. Predicting failures with developer networks and social network analysis. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 13–23. ACM, 2008.

[11] T. Menzies, A. Butcher, A. Marcus, T. Zimmermann, and D. Cok. Local vs. global models for effort estimation and defect prediction. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, pages 343–351. IEEE Computer Society, 2011.

[12] N. Nagappan, T. Ball, and A. Zeller. Mining metrics to predict component failures. In *Proceedings of the 28th international conference on Software engineering*, pages 452–461. ACM, 2006.

[13] N. Nagappan, B. Murphy, and V. Basili. The influence of organizational structure on software quality: an empirical case study. In *Proceedings of the 30th international conference on Software engineering*, pages 521–530. ACM, 2008.

[14] T. H. Nguyen, B. Adams, and A. E. Hassan. Studying the impact of dependency network measures on software quality. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–10. IEEE, 2010.

[15] S. Piantadosi, D. P. Byar, and S. B. Green. The ecological fallacy. *American Journal of Epidemiology*, 127(5):893–904, 1988.

[16] D. Posnett, V. Filkov, and P. Devanbu. Ecological inference in empirical software engineering. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, pages 362–371. IEEE Computer Society, 2011.

[17] F. Rahman and P. Devanbu. How, and why, process metrics are better. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 432–441. IEEE Press, 2013.

[18] F. Rahman, D. Posnett, and P. Devanbu. Recalling the imprecision of cross-project defect prediction. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, page 61. ACM, 2012.

[19] *The Economist*. Signs of the times, February 22, 2007.

[20] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 91–100. ACM, 2009.