There is no Random Sampling in Software Engineering Research

Bilal Amir
Sur University College
Sur, Oman
bilal@suc.edu.om

Paul Ralph
University of Auckland
Auckland, New Zealand
paul@paulralph.name

ABSTRACT

Representative sampling is considered crucial for predominately quantitative, positivist research. Researchers typically argue that a sample is representative when items are selected randomly from a population. However, random sampling is rare in empirical software engineering research because there are no credible sampling frames (population lists) for the units of analysis software engineering researchers study (e.g. software projects, code libraries, developers, projects). This means that most software engineering research does not support statistical generalization, but rejecting any particular study for lack of random sampling is capricious.

CCS CONCEPTS

• **General and reference** → **Empirical studies**; *Reference works*; *Surveys and overviews*;

KEYWORDS

Research methodology, sampling, random sampling, purposive sampling

ACM Reference Format:
Bilal Amir and Paul Ralph. 2018. There is no Random Sampling in Software Engineering Research. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3195001

1 INTRODUCTION

Most research involves studying a subset of a larger group of things. Choosing which of many things to study is *sampling*. *Representative* sampling is crucial for positivist research (e.g. experiments, questionnaires, positivist case studies) because properties of unrepresentative samples cannot be statistically generalized to a population.

Sampling strategies can be divided into two broad categories: *probability* (which employs randomness) and *non-probability* (which does not). *Multistage* sampling refers to any sequential combination of probability or non-probability sampling (e.g. purposively sampling GitHub and then randomly sampling projects from GitHub). Sufficiently large probability samples are ostensibly representative due to the law of large numbers.

ICSE'18, May 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05…$15.00
https://doi.org/10.1145/3183440.3195001

Sampling for software engineering (SE) research is particularly interesting and troublesome because SE studies often lack comprehensive population lists, preventing simple random sampling. This raises the following research question:

> ***Research Question:*** *How common is probability sampling in empirical software engineering research?*

To investigate this question, we examined the *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (ESEM) for the past five years.

2 METHOD

The study design is based on common guidelines for systematic literature reviews [2] and systematic mapping studies [3]. Broadly, the idea was to analyze the sampling strategy used in a set of empirical software engineering research papers.

We adopted a two-stage sampling strategy. First, we purposively selected the proceedings of ESEM because 1) most ESEM papers involve sampling; 2) ESEM is a widely considered a good outlet; and 3) ESEM papers are typically shorter than journal papers. Next, we retrieved all 248 full papers from ESEM 2012 to 2016, inclusive, using the ACM Digital Library and IEEE Xplore. We excluded 12 articles that were not empirical, leaving a sample of 236 articles to analyze.

We reviewed each paper and recorded the list of authors, title, conference year, research methodology, sample size, unit of analysis, population and sampling technique in an Excel spreadsheet. Some papers clearly stated the sampling technique. For most papers, however, we had to infer the sampling technique from the text. For example, we inferred expert sampling from the statement "The subjects were five electronic engineers that worked at the ... induction division. These engineers are experts in developing induction hub software" [1]. All ambiguous cases were reviewed by both authors and classified by consensus.

3 RESULTS

Ten different sampling techniques were identified (Table 1). The three most commonly used sampling techniques are purposive, convenience and two-stage sampling. Only eight papers (3%) employed probability sampling. (Five papers with two-stage sampling used probability sampling for one of the stages.)

The distribution of sampling strategies appears stable over time (Figure 1). Indeed, no trends were obvious. Purposive sampling is always the most common approach, followed by convenience sampling. We may not have enough data to detect trends.

While the articles used more than 30 distinct research methodologies, we abstracted these into 10 groups to simplify the analysis. For example, we grouped systematic literature reviews, systematic mapping studies, and thematic synthesis into "literature review." We

Table 1: Sampling Techniques Found in SE

Type	Sampling Method	Code	Definition	Total
Non-probability	Purposive	PS	Select items strategically	145
Non-probability	Convenience	CS	Select items arbitrarily	53
Either	Two stage	TSS	Sequential combination of two sampling strategies	15
Non-probability	Snowball	SS	Select items that are linked	9
Non-probability	Search-based	SBS	Select items in search results	5
Non-probability	Heterogeneity	HS	Purposively select diverse items	4
Probability	Stratified random	STS	Randomly select items to match a known distribution	2
Non-probability	Expert	ES	Purposively select experts	1
Non-probability	Quota	QS	Purposively select items to match a known distribution	1
Probability	Systematic	SYS	Select every *n*th item	1

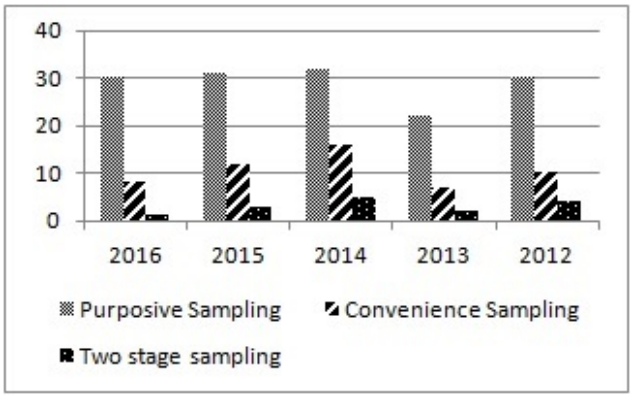

Figure 1: Sampling Method Usage Over Time

grouped social network analysis, mathematical modeling, simulation studies and some software repository mining into "quantitative data analysis."

The most common methodologies were case study, quasiexperiment and (controlled) experiment. All three of these predominately use purposive and convenience sampling. Literature reviews use the widest variety of sampling techniques.

The articles study more than 60 distinct units of analysis. We grouped these into 11 broad categories, of which software project, software professional and code artifact were the most common. All three of these, again, use predominantly purposive and convenience sampling.

Two factors impeded our analysis. Some studies do not name or adequately describe their research method or sampling technique. Many papers misuse terms including *random*, *case study* and *experiment*. Studying a top conference in *empirical* software engineering, we did not expect so much difficulty identifying the articles' research methodologies and sampling techniques.

4 DISCUSSION AND CONCLUSION

The purpose of this study was to investigate the prevalence of random sampling in empirical software engineering research. Based on recent ESEM conferences at least, we conclude that random sampling is extremely rare. Only 3% of *empirical* papers employed probability sampling techniques.

While this kind of study cannot determine why probability sampling is so rare, at least three factors may be implicated: 1) the popularity of case studies and experiments (which historically favor non-probability sampling; 2) some SE research adopts philosophical positions (e.g. interpretivism) incommensurate with random sampling; 3) most phenomena SE researchers care about have no comprehensive population list.

The significance of this third factor cannot be overstated. There are simply no lists (sampling frames) of all the software developers, projects, products, companies, test suites, embedded systems, design diagrams, user stories, personas, requirements specifications or code comments in the world or even in a specific country, language or application domain.

In the short term, this means that rejecting a software engineering study for lacking random sampling is capricious. Sample size permitting, researchers using nonprobability sampling should consider bootstrapping their analyses to assess stability.

In the long term, we can develop more and better surrogate populations. For instance, the Qualitas corpus is "a large curated collection of open source Java systems" [4]. Similar corpora could be developed for diverse artifacts used in software projects, including design specifications, requirements documents, user stories, scenarios, personas, various diagrams, unit tests, test cases, systems in languages other than Java, and end-user documentation. Creating any one of these corpora is a major undertaking and should be recognized as a significant research contribution in itself.

REFERENCES

[1] Jorge Echeverría, Francisca Pérez, Andrés Abellanas, Jose Ignacio Panach, Carlos Cetina, and Óscar Pastor. 2016. Evaluating Bug-Fixing in Software Product Lines: an Industrial Case Study. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 24.

[2] Barbara Kitchenham and Stuart Charters. 2007. Guidelines for performing systematic literature reviews in software engineering. *Technical report* (2007).

[3] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic Mapping Studies in Software Engineering.. In *Proceedings of the Evaluation and Assessment in Software Engineering Conference (EASE)* (Bari, Italy). ACM, 68–77.

[4] Ewan Tempero, Craig Anslow, Jens Dietrich, Ted Han, Jing Li, Markus Lumpe, Hayden Melton, and James Noble. 2010. The Qualitas Corpus: A Curated Collection of Java Code for Empirical Studies. In *Proceedings of the 17th Asia Pacific Software Engineering Conference*. IEEE, Sydney, Australia, 336–345. DOI: http://dx.doi.org/10.1109/APSEC.2010.46