Modeling Sampling Workflows in Empirical Software Engineering

GDR GPL
2025

Romain Lefeuvre
University of Rennes
France
romain.lefeuvre@irisa.fr

Maïwenn Le goasteller
Insa
France
maiwenn.le-goasteller@irisa.fr

Jessie Galasso
Mcgill University
Canada
jessie.galasso-carbonnel@mcgill.ca

Quentin Perez
INSA Rennes
France
quentin.perez@irisa.fr

Benoit Combemale
Inria
France
benoit.combemale@inria.fr

Houari Sahraoui
DIRO, Université de Montréal
Canada
sahraouh@iro.umontreal.ca

Empirical research in Software Engineering

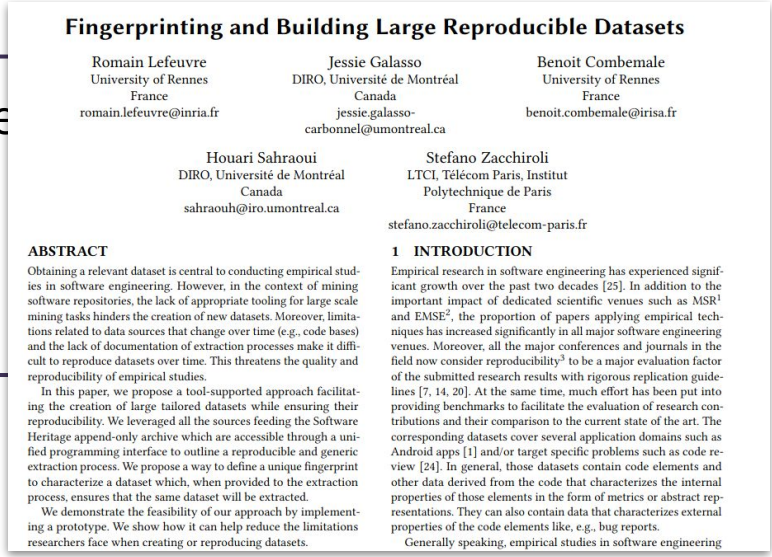
- Research question(s) on a population of software artefacts of interest
- Different kinds of artefacts: source code, binaries, etc.
- Often not feasible to study the entire population

Focus on Mining Software Repository (MSR) studies

Inspired from : M. Vidoni. 2022. A systematic process for Mining Software Repositories: Results from a systematic literature review. Inf. Softw. Technol. 144, C (Apr 2022). https://doi.org/10.1016/j.infsof.2021.106791

Key properties for Mining Software Repository studies

- **Reproducibility** of dataset / sample extraction
- **Representativeness** of studied sample
- **Generalization** of findings



Fingerprinting and Building Large Reproducible Datasets

Romain Lefeuvre
University of Rennes
France
romain.lefeuvre@inria.fr

Jessie Galasso
DIRO, Université de Montréal
Canada
jessie.galasso-carbonnel@umontreal.ca

Benoit Combemale
University of Rennes
France
benoit.combemale@irisa.fr

Houari Sahraoui
DIRO, Université de Montréal
Canada
sahraouh@iro.umontreal.ca

Stefano Zacchiroli
LTCI, Télécom Paris, Institut Polytechnique de Paris
France
stefano.zacchiroli@telecom-paris.fr

ABSTRACT

Obtaining a relevant dataset is central to conducting empirical studies in software engineering. However, in the context of mining software repositories, the lack of appropriate tooling for large scale mining tasks hinders the creation of new datasets. Moreover, limitations related to data sources that change over time (e.g., code bases) and the lack of documentation of extraction processes make it difficult to reproduce datasets over time. This threatens the quality and reproducibility of empirical studies.

In this paper, we propose a tool-supported approach facilitating the creation of large tailored datasets while ensuring their reproducibility. We leveraged all the sources feeding the Software Heritage append-only archive which are accessible through a unified programming interface to outline a reproducible and generic extraction process. We propose a way to define a unique fingerprint to characterize a dataset which, when provided to the extraction process, ensures that the same dataset will be extracted.

We demonstrate the feasibility of our approach by implementing a prototype. We show how it can help reduce the limitations researchers face when creating or reproducing datasets.

1 INTRODUCTION

Empirical research in software engineering has experienced significant growth over the past two decades [25]. In addition to the important impact of dedicated scientific venues such as MSR[1] and EMSE[2], the proportion of papers applying empirical techniques has increased significantly in all major software engineering venues. Moreover, all the major conferences and journals in the field now consider reproducibility[3] to be a major evaluation factor of the submitted research results with rigorous replication guidelines [7, 14, 20]. At the same time, much effort has been put into providing benchmarks to facilitate the evaluation of research contributions and their comparison to the current state of the art. The corresponding datasets cover several application domains such as Android apps [1] and/or target specific problems such as code review [24]. In general, those datasets contain code elements and other data derived from the code that characterizes the internal properties of those elements in the form of metrics or abstract representations. They can also contain data that characterizes external properties of the code elements like, e.g., bug reports.

Generally speaking, empirical studies in software engineering

Sample representativeness

- Well-documented concern across scientific fields[1], including software engineering [2]
- Defined as the extent to which **“a sample’s properties of interest resemble those of the target population"**
- **Dimension specific**
- Representativeness can be supported with different arguments :
    - Large and random sample
    - Breadth of a sample
    - Similar distributions

[1] William Kruskal and Frederick Mosteller. 1979. Representative Sampling, II: Scientific Literature, Excluding Statistics. International Statistical Review / Revue Internationale de Statistique 47, 2 (1979), 111–127

[2] Sebastian Baltes and Paul Ralph. 2022. Sampling in software engineering research: a critical review and guidelines. 27, 4 (2022), 94. doi:10.1007/s10664-021-10072-8

Sampling strategies

Approach	Capsule Description
Convenience	Select items based on expediency
Purposive	Select items most useful for study's objective
Referral-chain	Select items based on relationship to existing items
Respondent-driven	Bias-mitigating variant of referral-chain
Whole frame	Select the entire sampling frame
Simple random	Select items entirely by chance
Systematic random	Select every xth item from a random start
Stratified	Select items from different groups randomly but in equal proportion
Quota	Select items from different groups purposively but in equal proportion
Cluser	Select items in stages, where each stage is a subset of the previous

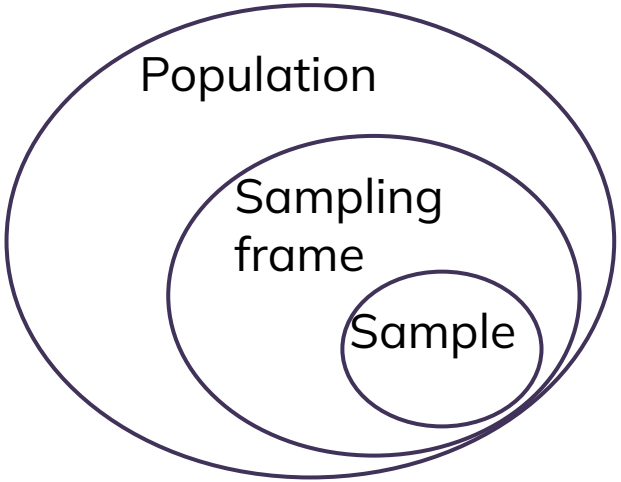
Probabilistic and non probabilistic techniques.

Argument of representativeness dependent on sampling type

Sebastian Baltes and Paul Ralph. 2022. Sampling in software engineering research: a critical review and guidelines. Empirical Softw. Engg. 27, 4 (Jul 2022). https://doi.org/10.1007/s10664-021-10072-8

Challenge related to sampling approach design

Traditional 3-tiers framework



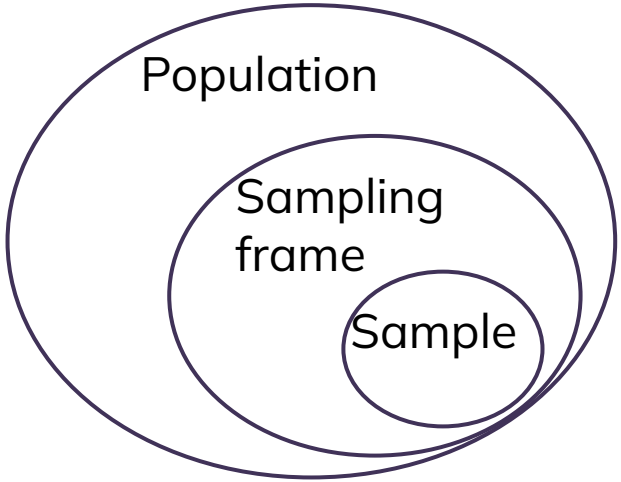
Software projects contains **numerous artefact types** (code, documentation issue reports, tests etc..) connected in complex ways

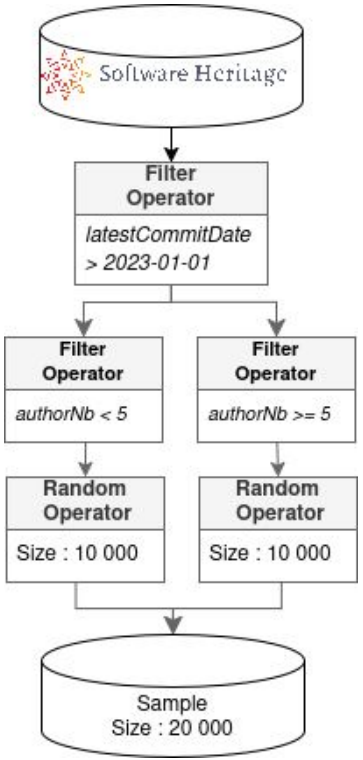
Sampling requires multiple phases of refinement and selection

The 3-tier framework is not adapted to represent sampling workflow

Challenge related to sampling approach design

Traditional 3-tiers framework





The 3-tier framework is not adapted to represent sampling workflow

Challenge related to sampling approach design

- No formalism : Incomplete textual description of methodology

- Lack of probabilistic sampling
  - only 8% of analysed study use random sampling [1]
  - “Generalisability crisis” [1]

Need for a framework, to explicitly model multi-stage sampling strategies

[1] Sebastian Baltes and Paul Ralph. 2022. Sampling in software engineering research: a critical review and guidelines. Empirical Softw. Engg. 27, 4 (Jul 2022). https://doi.org/10.1007/s10664-021-10072-8

Challenge related to reasoning on the sampling strategies

- Representativeness of a sample needs to be discussed
- In practice, study use **vague descriptors** like “real-world”, “diverse,” or “representative,” but rarely provide supporting evidence [1]
- Argument for representativity require **quantitative argument**
  - “Large and random sample” ⇒ sample size / statistical test
  - “Similar distribution” ⇒ comparison of key property distributions

[2]

Modelling sampling workflow can support generalizability reasoning through automated analysis.

[1] Sebastian Baltes and Paul Ralph. 2022. Sampling in software engineering research: a critical review and guidelines. Empirical Softw. Engg. 27, 4 (Jul 2022). https://doi.org/10.1007/s10664-021-10072-8
[2] Kruskal, W., & Mosteller, F. (1980). Representative sampling, IV: The history of the concept in statistics, 1895-1939. International Statistical Review/Revue Internationale de Statistique, 169-195.

Two levels of generalisation

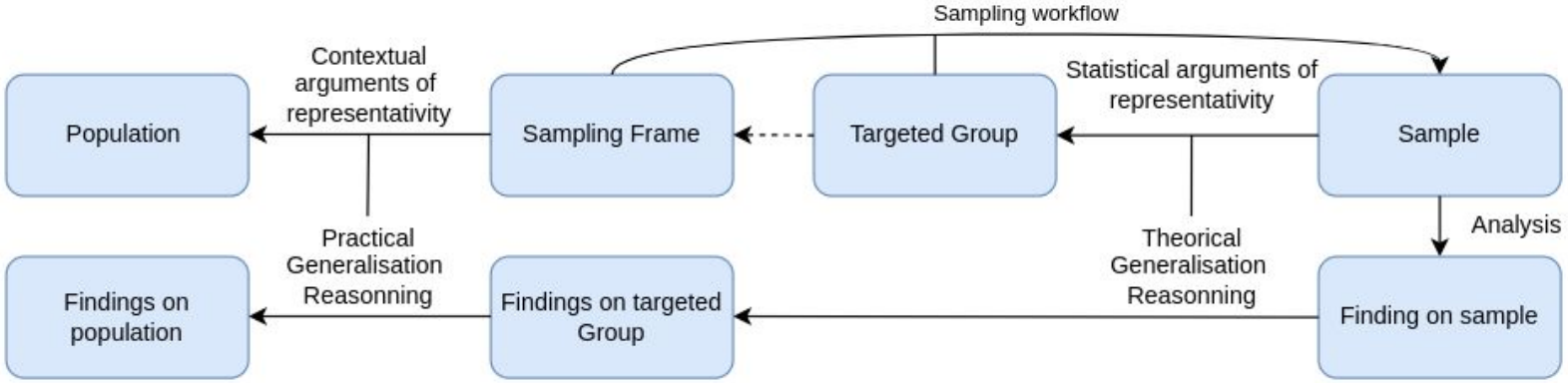
Active Open source software

Software heritage

Active software repositories archived by Software Heritage

Union of two random sample of :

- 10k repos with <5 contributors
- 10k repos with >= contributors

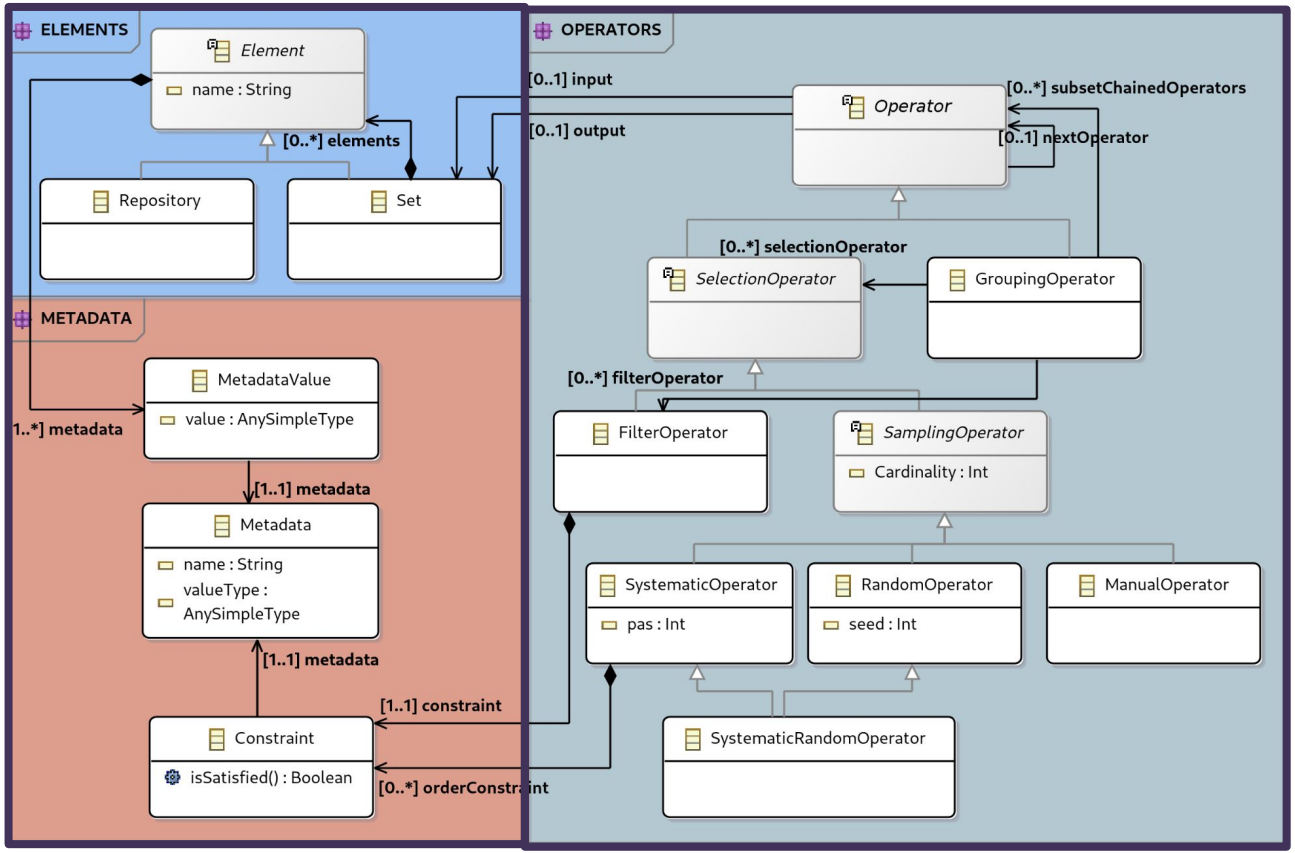


Research Questions :

RQ1: Does a DSL combining basic sampling operators can model complex sampling workflows in SE?

RQ2: To what extent does formally modeling sampling workflows support representativeness reasoning?

Modelling sampling workflow



- Formalize the sampling workflow with no ambiguity
- Executable and reproducible
- Provide an analysable representation
- Use a generic data model

Java internal DSL

```
1  void main(){
2      //Selection of active repository : Filter by date
3      filterOperator(latestCommitDate.boolConstraint(
4                              x->x>time(2023, 1, 1)))
5      //Stratified Sampling by author number
6      .chain(groupingOperator(
7          //First strata : projects with less than 5 authors
8          filterOperator(authorNb.boolConstraint(x -> x<5))
9                      //Random sampling of 10k repo
10                       .chain(randomSelectionOperator(10000)),
11         //Second strata : projects with 5 or more authors
12         filterOperator(authorNb.boolConstraint(x -> x>=5))
13                     //Random sampling of 10k repo
14                       .chain(randomSelectionOperator(10000)))
15     )
16     .input(swhLoader("2024-05-16-history-hosting"))
17     .executeWorkflow();
18 }
```

Figure 3: Modelled workflow of our running example

- Java fluent API
- Connector to SWH

Evaluation of the expressivity of the DSL

```
void main()  {
 var url = Metadata.ofString("url");
 var lang = Metadata.ofString("lang");
 var id = Metadata.ofString("id");
 var commitNb = Metadata.ofDouble("commitNb");

 //Cluster Operator
 groupingOperator(
   filterOperator(commitNb
        .boolConstraint(x -> x<100 )),
   filterOperator(commitNb
        .boolConstraint(x -> x>=100
                        && x<1000 )),
   filterOperator(commitNb
        .boolConstraint(x -> x>=1000)))
   .chain(randomSelectionOperator(2))
   .input(jsonLoader("input.json",id,commitNb
                                ,url,lang))
   .output(jsonWritter("cluster.json"))
   .execute();
```

Cluster Sampling

```
//Stratified Random Operator
groupingOperator(
 filterOperator(commitNb
                .boolConstraint(x -> x<100))
 .chain(randomSelectionOperator(100)),
 filterOperator(commitNb
                .boolConstraint(x -> x>=100
                                && x<1000))
 .chain(randomSelectionOperator(100)),
 filterOperator(commitNb
                .boolConstraint(x-> x>=1000 ))
 .chain(randomSelectionOperator(100)))
 .input(jsonLoader("input.json",id,commitNb
                                ,url,lang))
 .output(jsonWritter("stratified_random.json"))
 .execute();
```

Stratified Random Sampling

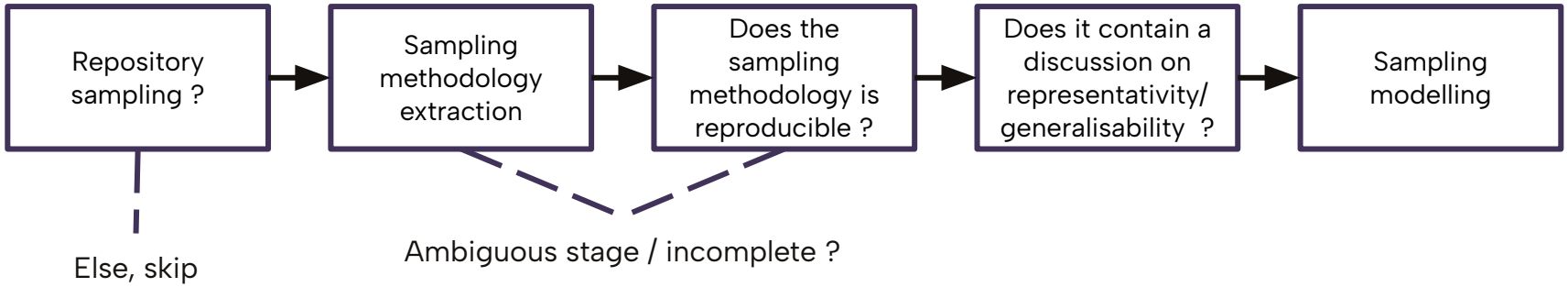
```
//Quota Operator
groupingOperator(
 filterOperator(commitNb
                .boolConstraint(x -> x<100))
 .chain(manualSamplingOperator(1,10,54,76,38)),
 filterOperator(commitNb
                .boolConstraint(x -> x>=100
                                && x<1000))
 .chain(manualSamplingOperator(6,8,14)),
 filterOperator(commitNb
                .boolConstraint(x -> x>=1000))
 .chain(manualSamplingOperator(53,54,2,5)))
 .input(jsonLoader("input.json",id,commitNb
                                ,url,lang))
 .output(jsonWritter("quota.json"))
 .execute();
;
```

Quota Sampling

Common multi-stage sampling approaches captured

Evaluation of the expressivity of the DSL

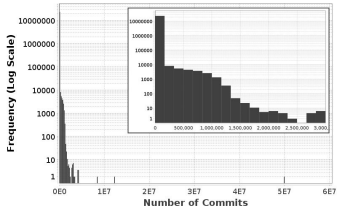
Review of Mining Software Repository (MSR) 2023 and 2024 papers

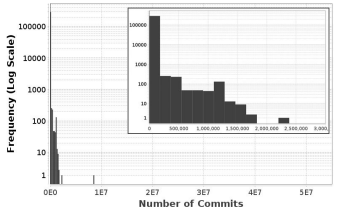


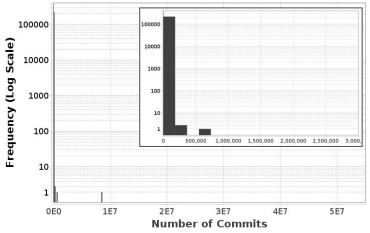
Objectives :

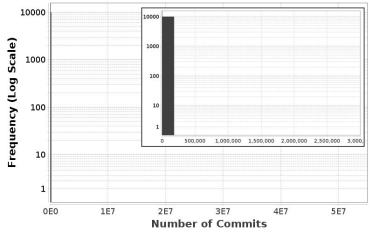
- Evaluate the expressivity of our DSL
- Evaluate the opportunity of sampling modelling in MSR community

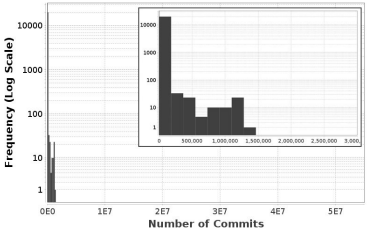
Supporting representativeness : distribution analysis



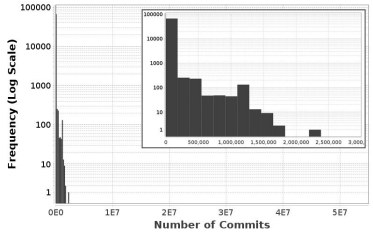


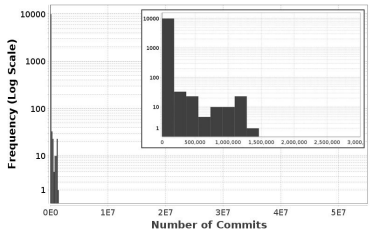






INPUT SET	Filter Operator	SET #1	Grouping Operator	Filter Operator	SET #2	Random Operator	SET #4	SAMPLE
Size : 23 601 029	latestCommitDate > 2023-01-01	Size : 289 923		authorNb < 5	Size : 222 715	10 000	Size : 10 000	Size : 20 000
				Filter Operator	SET #3	Random Operator	SET #5	
				authorNb >= 5	Size : 67 208	10 000	Size : 10 000	





Supporting representativeness : distribution analysis

Automatic execution of statistical test :

- Chi-square test goodness of fit (category)
- **Kolmogorov-Smirnov** (continuous metadata)
  - null hypothesis = same distribution
  - (0.05 significance level)

p-value = 0.18

p-value = 0.99

Null hypothesis accepted, same distribution

Supporting representativeness : random sample size

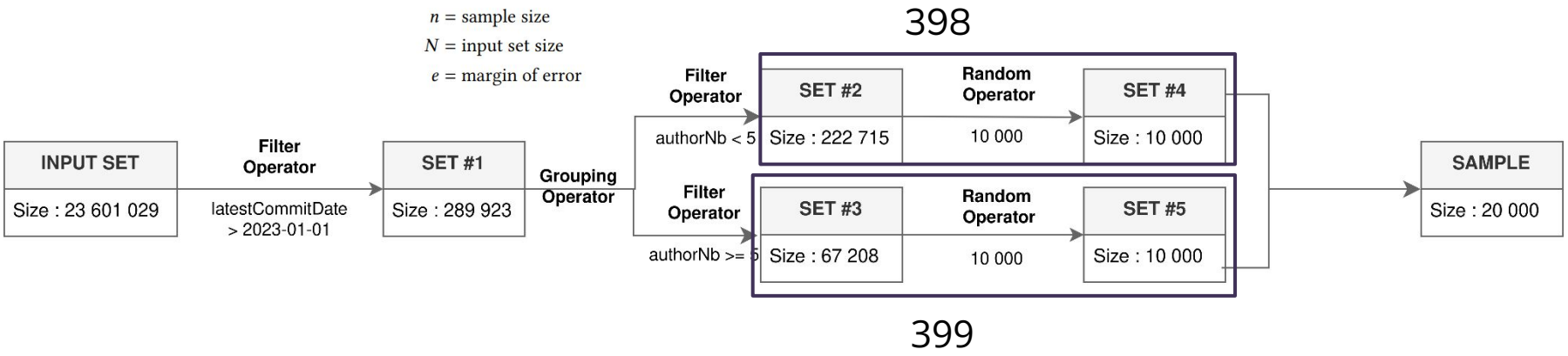
"Large and random sample" representativeness argument, based on statistical test

- Cochran's formula (normality assumption)
- **Yamane**

$$n = \frac{N}{1 + N \cdot e^2}$$

where:

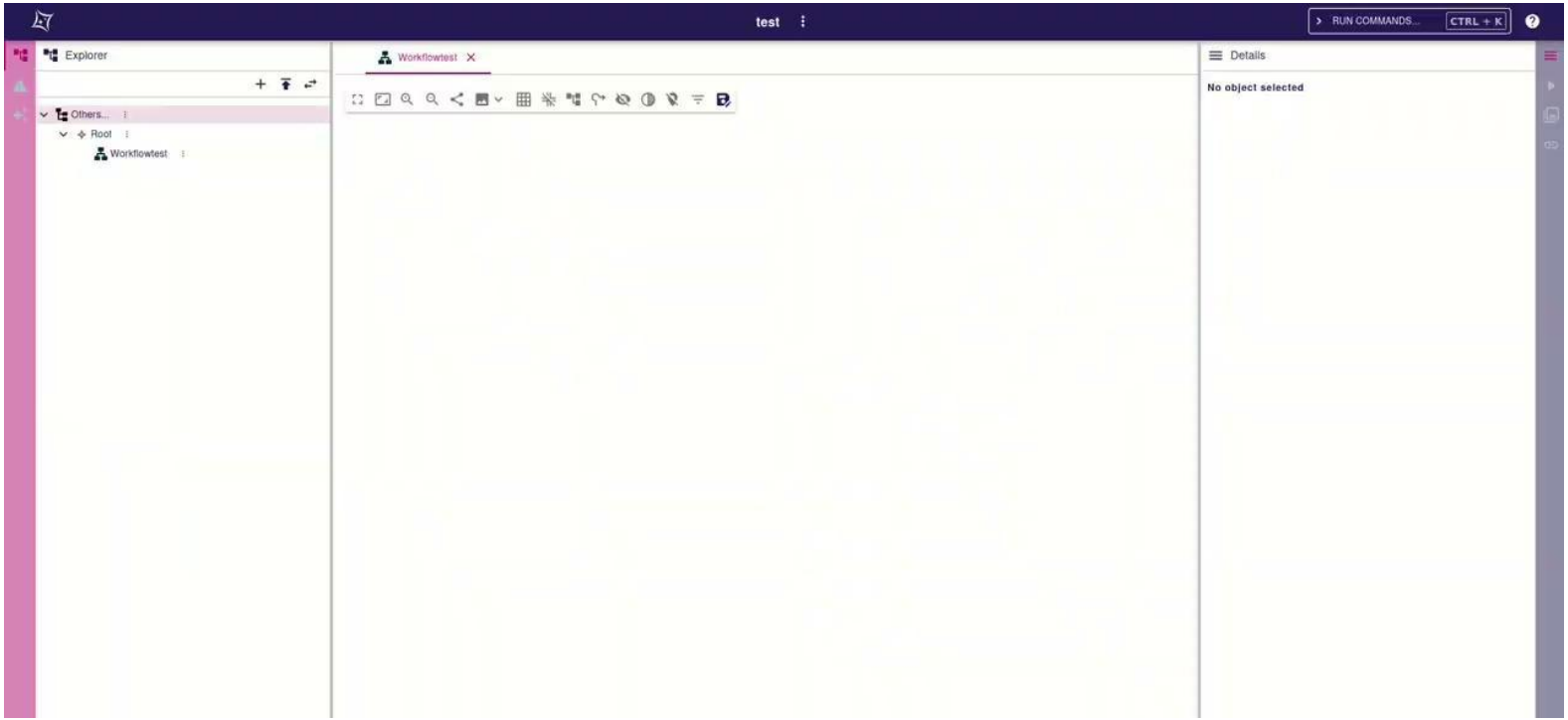
n = sample size
N = input set size
e = margin of error



William G Cochran. 1977. Sampling Techniques. John Wiley & Sons, Nashville, TN.
Taro Yamane. 1973. Statistics: An introductory analysis. Harper & Row New York.

Work in progress

Graphical DSL



Key points

- We propose a Domain Specific Language to model Sampling workflow
- This DSL is executable, and support multiple data sources
- The formalisation support representativity reasoning
  - Explicit definition of sampling workflow
  - Enable to perform analysis on the workflow and its execution