The Scone DSL: Smart Sampling for Smarter Statistics

Eunice Jun
University of Washington, USA
Microsoft Research, USA
emjun@cs.washington.edu

Emery Berger
University of Massachusetts Amherst, USA
Microsoft Research, USA
emery@cs.umass.edu

Ben Zorn
Microsoft Research, USA
ben.zorn@microsoft.com

Abstract

Data analysts are interested in rapidly generating and testing hypotheses in their data. Analysts use convenient, non-representative samples they must obtain and manage manually because they want to expedite their analyses and use familiar tools. Unfortunately, this practice runs the risk of finding false positive relationships. During a recently completed formative study, we identified a tension between *rapid, iterative data analysis* and *statistical generalizability*. Based on these findings, we are brainstorming ideas for a DSL Scone. We present preliminary ideas for Scone. Our aim is for Scone to provide a higher level of abstraction than SQL and automatically generate unique, representative samples to test hypotheses, *smart sampling*. Our primary design goals so far are for Scone to fit into analysts' rapid workflows and for Scone increase the statistical generalizability of analysts' findings.

2012 ACM Subject Classification Software and its engineering → Domain specific languages; Human-centered computing → User interface toolkits

Keywords and phrases domain-specific language; data science; data analysis; sampling; statistics; generalizability

Digital Object Identifier 10.4230/OASIcs...

1 Introduction

Best practices in statistics [8] recommend separating statistical data analysis into two distinct stages: exploratory data analysis (EDA) [7] and confirmatory data analysis. During EDA, analysts discover patterns in the data and generate hypotheses to test. During confirmatory data analysis, analysts test their hypotheses and draw conclusions about the data.

In practice, however, the two stages are intermingled and the transition between the two phases more fluid. Data analysts may generate and immediately test a hypothesis while still exploring the data.

Conducting all statistical tests during exploration and confirmation on the same dataset is problematic because each statistical test has a degree of uncertainty, or probability of providing a false positive result. The joint probability of finding false positives increases exponentially with each additional statistical test performed on the same dataset. This problem is often referred to as the Multiple Comparisons Problem.

Analysts may perform multiple comparisons on the same dataset because collecting new data after exploration may be too costly (e.g., clinical trials or natural experiments that cannot be repeated).

Depending on the data size, there are multiple strategies for controlling the number of false discoveries. For small data, analysts can use procedures such as the Bonferroni [3] and Benjamini-Hochberg corrections [1] to stricten their significance tresholds. However, these corrections lower the significance thresholds such that analysts may only be able to detect very strong effects. In other words, by using ad hoc corrections to significance thresholds, data analysts may miss practically significant effects (true positives). With sufficiently large data, analysts can create and use unique subsets for exploration and confirmation without changing the significance thresholds. As a result, analysts can still detect weaker yet practically significant effects.

Based on our formative conversations with data analysts, we found that data analysts working with big data (where there is enough data to use unique datasets for each statistical test) prefer to work with subsets of data that fit in memory on their local machines (i.e., laptops) for (i) speed of analysis development and (ii) familiarity with tools that are designed for local use on smaller scale data analysis (mostly R and Python) rather than at scale (e.g., MapReduce). From a user-centered design perspective, the analysts are engaging in a *rapid, iterative analysis* process and appropriately choosing tools and methods that help them rapidly iterate on analyses. Unfortunately, data analysts use simple sampling methods that are may not lead to representative subsets of data. For example, analysts rely on convenience samples, such as the *first k rows* for time series data, which could lead to analyses that cover a very short period of time. Analyses conducted on the non-representative samples run the risk of not generalizing to the larger dataset or population.

Based on ongoing observations and interviews with data analysts about their sampling practices, we hope to design and develop Scone, a high-level DSL that will fit into analysts' rapid, iterative analysis processes and provide support for generating representative samples to increase the generalizability of statistical results. We just completed the formative study with data scientists and are now beginning to discuss language design. We anticipate incorporating techniques and methods from the database, programming language, and human-computer interaction literature.

In this paper, we briefly discuss background information (section 2), a few design motivations we have already defined (section 3), and technical challenges that Scone will need to solve (section 4). We conclude with our goals for presenting Scone and attending PLATEAU (section 5).

2 Background and Related Work

Our work builds on prior work in the database, human-computer interaction communities, and programming languages communities.

The Multiple Comparisons Problem arises from conducting multiple statistical tests on the same dataset, compounding the probability of finding spurious false positive relationships. Although freely developing and exploring multiple hypotheses is appropriate during exploratory data analysis [7], testing multiple hypotheses during confirmatory data analysis should be more closely monitored and taken into account when interpretting and reporting results.

QUDE [10], a database system, aims to address the multiple comparisons problem through the implementation of novel methods to control the false discovery rate. QUDE focuses on maximizing statistical power and assumes usage scenarios where data analysts do not have enough data to use a unique dataset for each hypothesis test. In contrast to QUDE, Scone targets the scenario where data analysts have more than enough data to use unique datasets for each statistical test and maintain sufficient statistical power. For big data analytics, using smaller subsets does not reduce statistical power in noticeable or practically meaningful degrees.

Using uniform random samples to control for the Multiple Comparisons Problem overlooks important nuances in the data. For example, for time series data, uniform random samples based on time may result in subsamples that represent the overall longitudinal trend, but such a sampling proceudre overlooks any interactions between data attributes, such as any geographic differences in trends.

Techniques in probabilistic programming and program synthesis have addressed related issues in data science. We may draw on these techniques to implement Scone.

Probabilistic programs that perform statistical inferences on a model compute properties over generated samples [9]. Analyzing probabilistic programs to reduce the number of possible program can expedite sample generation and generate data samples that are more evenly distributed across the entire space of possible samples [2]. Because writing probabilistic programs is difficult, we plan to generate samples from users' statistical analysis code and to have the samples cover important properties of the data (e.g., the distribution of the values) rather than the program (e.g., combination of parameters, as done in [2]).

Programming-by-example and program synthesis are also promising techniques to incorporate in Scone. Prior work performs common data transformation tasks by synthesizing R programs using specifications and examples [4]. Scone could use a similar technique by eliciting example samples from users and synthesizing queries for generating tables, which could later be transformed using tools such as Morpheus [4].

3 Design Motivations

Our recently completed formative study informed the design and technical requirements of Scone. We have already identified a tension between *rapid, iterative analysis* and *statistical soundness and generalizability.* Current analysis practices prioritize the former, at the risk of compromising the latter. In Scone, we aim to support the former and guarantee the latter, thereby improving data analysis practices overall.

Towards this goal, we have identified two preliminary design motivations:

1. *Fit into existing workflows.* Tukey described exploratory data analysis as a flexible attitude towards data analysis that may involve many techniques, with visualization playing a key role [5]. As such, a tool for generating new samples during exploration and confirmation should impose as few restrictions on the exploratory process as possible and should easily become a part of existing workflows. As such, Scone should be implemented not as a separate standalone application but rather as a library that can be invoked in common programming and analysis environments.
2. *Respond quickly.* Data analysts work with samples of data to expedite their analysis process. To fit in with existing practices, Scone should provide samples to maintain statistical soundness at interactive speeds, approximately 500ms for visualization [6].

4 The Scone DSL

We are still in the early stages of designing the Scone language. We plan to implement Scone with a Python frontend because of its familiarity and wide adoption in data science (Design Motivation 1) and employ a SQL backend with optimizations (Design Motivation 2).

Scone requires that the data for analysis already exist in a database. In Scone, we make a closed-world assumption. In other words, Scone assumes that the database contains data from an entire population of interest.

Scone will need to address (at least) the following key challenges:

1. Scone needs to be at a high enough level of abstraction to be easier than writing a SQL query directly yet expressive enough for users to specify complex queries.
2. Scone needs to define and support "correct" sampling methods for a statistical analysis. We will likely scope the class of analyses Scone can support.
3. Scone should provide acceptable worst case behavior that maintains statistical soundness, such as when users exhaust all the data of a particular category. Should Scone continue to allow users to get new samples without a category? What are the warnings Scone should issue as users "use up" the data?

The aim is for Scone to take high-level user specifications and analysis code in Python and compile them into SQL queries for generating samples. As mentioned in the Background and Related Work (section 2), we are exploring possible design and implementation strategies, including at the interface level: analyzing user programs, requiring user annotation of their programs, additional specifications, and/or examples for generating constraints for synthesizing SQL queries, at the program generation level: an implementation using a probabilistic language [2] or a conflict-driven synthesis loop with user guidance, and at the query optimization level: generating database sketches offline and/or pre-fetching subsets of data involving specific columns.

We imagine a user will provide

5 Specific Aims for PLATEAU

We are excited to engage in discussion at PLATEAU. We have two broad goals: (i) share and get feedback on Scone and (ii) discuss larger themes and questions around research that is at the intersection of systems, programming languages, and human-computer interaction.

In particular, we hope to discuss the following topics.

- Scone: There are several threat models to statistical soundness. How could Scone be improved to provide *strong guarantees of soundness* even in the face of possible misuse? (What are different classes of guarantees users care about vs. systems vs. statistics?)
- Scone: What would be meaningful, feasible evaluations of Scone? More broadly, how do we evaluate new DSLs such as Scone?
- Scone: How might attendees who conduct statistical analyses regularly use Scone?
- Scone: What other opportunities do we have to take an interdisciplinary approach to *data science* issues?
- Interdisciplinary work: How do we conceptualize, write, and talk about the *design process* for research that spans programming languages and human-computer interaction?
- How do we discuss and assess work that combines multiple methods and approaches, including qualitative studies, building systems, and quantitative measures? Could our communities create more space for hybrid methodologies?
- Interdisciplinary work: What are new areas of systems or programming languages research that human-computer interaction can motivate or inspire?
- Interdisciplinary work: When are tools enough or even a good idea? In software engineering and human-computer interaction, we sometimes focus on teaching or incentivizing users to act in "correct" ways. In programming languages, tools that prevent poor

behavior are often the proposed answer. How do we identify opportunities for improved *practices* vs. *tools*?

References

1 Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

2 Arun Chaganty, Aditya Nori, and Sriram Rajamani. Efficiently sampling probabilistic programs via program analysis. In *Artificial Intelligence and Statistics*, pages 153–160, 2013.

3 Olive Jean Dunn. Multiple comparisons among means. *Journal of the American statistical association*, 56(293):52–64, 1961.

4 Yu Feng, Ruben Martins, Jacob Van Geffen, Isil Dillig, and Swarat Chaudhuri. Component-based synthesis of table consolidation and transformation tasks from examples. In *ACM SIGPLAN Notices*, volume 52, pages 422–436. ACM, 2017.

5 Lyle V Jones. *The Collected Works of John W. Tukey: Philosophy and Principles of Data Analysis 1949-1964*, volume 3. CRC Press, 1986.

6 Zhicheng Liu and Jeffrey Heer. The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics*, 20(12):2122–2131, 2014.

7 John Tukey. W.(1977). exploratory data analysis. *Reading: Addison-Wesley.*

8 John W Tukey. We need both exploratory and confirmatory. *The American Statistician*, 34(1):23–25, 1980.

9 Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*, 2018.

10 Zheguang Zhao, Lorenzo De Stefani, Emanuel Zgraggen, Carsten Binnig, Eli Upfal, and Tim Kraska. Controlling false discoveries during interactive data exploration. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 527–540. ACM, 2017.