InferenceQL is a probabilistic programming system for scalable Bayesian AutoML from database tables. InferenceQL is designed to help make Bayesian approaches to data analysis more accessible to broad audiences and to assist experts in auditing and improving the quality of data, models, and inferences. Unlike traditional probabilistic programming systems, InferenceQL provides automation for learning models using nonparametric Bayesian structure learning of probabilistic programs. Experts can override these models with custom probabilistic programs for specific subsets of variables and conditional distributions. For a broad class of models, InferenceQL can generate realistic synthetic data subject to constraints and can automatically compute exact probabilities and mutual information values. Finally, InferenceQL aims to enable scalable Bayesian model criticism via posterior predictive checks, data quality screening via conditional probability calculation, fairness auditing via conditional probability ratios, and synthetic data generation to enhance privacy. These capabilities are accomplished using constructs that interleave standard database queries with Bayesian inference.

Automated Bayesian inference from databases is important and useful in several ways. For example, many real-world databases have high rates of missing values, more fields than observed records, heterogeneous data types, high rates of data entry error, and other factors that complicate the application of traditional ML-based AutoML techniques [9, App. E]. Furthermore, many real-world applications benefit from uncertainty quantification, interactive model checking and model criticism, and conditional probability estimation for ad-hoc fairness auditing. These problems are naturally formulated in terms of Bayesian inference [11, 28].

InferenceQL is a probabilistic programming system for automated Bayesian inference from database tables. InferenceQL provides a domain-general mechanism for Bayesian structure learning [18] of probabilistic program source code [27], as well as domain-general mechanisms for scalable exact and approximate inference in these probabilistic programs. Users thus do not have to know how to write probabilistic programs in order to use InferenceQL to solve problems. Instead, users rely on automated data modeling techniques to navigate the design choices that might otherwise be handled by experienced modelers. InferenceQL also enables Bayesian inference operations to be interleaved with ordinary SQL operations, yielding complex database-native workflows for Bayesian AutoML. InferenceQL has been used successfully in field tests for a broad range of applications, including AutoML for clinical trial oversight in three real-world clinical trials.

This workshop paper introduces InferenceQL via an exploratory data analysis application. It also briefly reviews the system architecture of InferenceQL and the class of probabilistic programs that deliver its AutoML capabilities. It presents preliminary quantitative results from experiments comparing InferenceQL’s modeling accuracy to GLM, VAE, and CTGAN baselines. Finally, it reviews related work, including both modeling formalisms and ML and database integrations, and discusses some limitations and broader impacts.
InferenceQL

Query Planner

Probabilistic Program Synthesis

Data Table

Program Analysis

Probabilistic Row Generator

Query Plan

InferenceQL Query Engine

Probabilities

Synthetic Data

Figure 1: System architecture of InferenceQL.

Figure 2: Synthesizing probabilistic programs that model heterogeneously typed cross-sectional data.

(a) Subset of satellites data table showing 21 variables and 10 records

(b) Synthesized row generator

1 Example

The InferenceQL system automates data analysis and machine learning tasks by allowing users to input data tables and queries and to automatically generate answers for them (Figure 1). It consists of a probabilistic program synthesis component [27] that creates generative model programs that are called row generators. The InferenceQL query planner and query engine use row generators to answer questions about the data and the domain by querying an underlying probabilistic model.

Figure 2 shows an example of probabilistic program synthesis, which takes a heterogeneously-typed data table of satellites (maintained by the Union of Concerned Scientists [32]) and returns a probabilistic program that models the data. Figure 3(a) shows the high-level interface to creating synthesizing programs. Users can then compare synthetic data (generated from the probabilistic programs) with observed data in order to develop intuition about what the model learned from the data, shown in Figure 3(b). InferenceQL can generate synthetic data from both marginal distributions and conditional distributions given a user-specified predicate (code box in Figure 3(b)). The two plots in Figure 3(b) illustrate a qualitative goodness-of-fit in the sense that distribution of synthetic (orange dots) samples appears to approximately match the observed data (black dots).

Data analysts can use the query language to search for probable anomalies and data-entry errors, shown in Figure 3(c). To find values for the column Period_minutes that the model considers
improbable in light of the data, the query (left code box) produces the result by comparing the probability of the value for Period_minutes marginally and conditionally in the WHERE clause. The only rows returned are those whose conditional probability is lower than the marginal; the probabilistic program beats the default (a further 300x improvement in RMSE).

Finally, users with domain expertise can customize probabilistic programs. Figure 3(d) shows an example custom orbital model from physics. To quantitatively assess the goodness-of-fit, we first split the data into training and test data and build three models: the automatically synthesized default model, a generalized linear model (GLM), and a custom probabilistic program for noisy orbital physics. We then predict a column in the held-out data set. The default model predicts more accurately than the GLM (4x more accurate via root mean square error (RMSE)) and the custom probabilistic program beats the default (a further 300x improvement in RMSE).
2 Experiments

We now report experiments evaluating InferenceQL against statistical and neural baselines.

**Generative modeling benchmark.** Table 1 shows the average Jensen-Shannon divergence between (discretizations of) the observed data and learned generative models, for all pairwise marginals in 10 datasets from the UCI machine learning repository [8], according to simulations from InferenceQL, Gaussian copulas [23], CTGAN [34], and TVAE [34]. The bold entries indicate statistically significant lowest error under a Bonferroni corrected Mann-Whitney $U$ test, which are achieved by InferenceQL in 8 of 10 benchmark problems and zero times by other techniques.

**Anomaly detection benchmark.** Table 2 shows that InferenceQL detects a higher percentage of anomalies than does a GLM baseline on three datasets from the UCI repository. Anomalies were inserted into a target column by flipping the class label in each row with probability 0.05 and detected using a query similar to the one in Figure 3(d).

**Query optimization.** The InferenceQL query planner contains a built-in optimization for querying row generators specified in the SPPL language [28]. Table 3 shows the runtime of InferenceQL queries for computing the conditional probability of all cell values in one target column given all the other values in the same row, for 10 datasets from the UCI repository. The third column shows the runtime using InferenceQL’s independence analysis optimization, which statically eliminates from the query all conditioning variables that are structurally independent of the target variable. The fourth column shows the runtime using InferenceQL without independence analysis and the final column shows the runtime using the Python API to SPPL, which both do not automatically leverage independence analysis and are slower in cases where independencies can be exploited.
3 Related Work

Many AutoML systems have been developed for tabular data; prominent examples include Amazon’s AutoGluon-Tabular [9] and SageMaker Autopilot [7], Google Cloud Platform AutoML Tables [17], Uber’s Ludwig [21], H2O AutoML [16], and a number of earlier systems such as Auto-WEKA [31], auto-sklearn [10], hyperopt-sklearn [2], TPOT [22], autoxgboost [30], ML-Plan [20], OBOE [35], GAMA [12], and Auto-Keras [15]. A survey and comparison of many of these systems can be found in Erickson et al. [9, §3]. In contrast to these systems, which typically emphasize discriminative ML, InferenceQL provides users with generative models that can be queried repeatedly to answer a wide range of questions about the data.

The BayesDB probabilistic programming system [19] is closely related to InferenceQL, but more limited. BayesDB and InferenceQL both provide automatic Bayesian model discovery, suitable for exploratory data analysis, predictive modeling, and inferential statistics from sparse, heterogeneously-typed data tables. But InferenceQL is often more scalable than BayesDB, due in part to its use of sum-product expressions [28] (a class of probabilistic circuits [6]) to implement query plans and enable automated query optimizations. InferenceQL can be used to query custom models in the Gen probabilistic programming language [5], leveraging Gen’s support for pseudo-marginal approximations to the composable generative population model interface [24]. InferenceQL also provides a more compositional and expressive query language than BayesDB, including support for SQL-like inlining of column transformations and predicates. Together, these improvements allow InferenceQL users to interleave SQL and inference operations in complex end-to-end Bayesian AutoML workflows.

Probabilistic databases have been developed for querying noisy or uncertain data [29, 33]; for a survey of several probabilistic database systems, see [33, §6.2]. Other classes of database systems that integrate probability in some form include databases that use probabilistic circuits to improve query performance [13] and database systems extended by functions for imputation [4], time series prediction [1], random data generation [14] and simulation [3]. Unlike InferenceQL, these systems do not provide automated Bayesian model discovery, custom probabilistic programming, or compositional SQL-like queries that interleave SQL with Bayesian inference.

4 Conclusion

Limitations. It is unclear how to tune InferenceQL to compete with traditional non-Bayesian AutoML systems on discriminative ML problems. In principle, InferenceQL can use SPPL encodings of decision-tree classifiers [28] to match the accuracy of typical ML deployments. Currently, users can only achieve this awkwardly, via manual customization of the underlying models. It is unclear when and how InferenceQL should switch from Bayesian to non-Bayesian AutoML methods. Also, the current InferenceQL prototype only supports cross-sectional data tables. It would be conceptually straightforward and worthwhile in practice to integrate domain-general Bayesian structure learning methods for multivariate time series [25] or relational systems [26].

Broader Impact. If maximally successful, InferenceQL could enable typical SQL database users to apply Bayesian inference and probabilistic programming. InferenceQL could significantly reduce the cost and improve the quality of Bayesian data analysis, for both experts and novices, and help to make Bayesian approaches more routinely applicable, potentially improving the quality of data analysis. InferenceQL could also help reduce the risk associated with data breaches and data sharing, by enabling broader use of synthetic data.

Potential harms include reduced cost for invasive, abusive, or manipulative applications of modeling, by governments, corporations, and other actors — which may, in turn, cause people to steal sensitive data or surveil more.
Acknowledgements. This work was funded in part by DARPA under the SD2 and SAIL-ON programs; research contracts with Takeda Pharmaceuticals, Liberty Mutual, and Intel Corporation; and philanthropic gifts from the Aphorism Foundation and the Siegel Family Foundation.

References


