Boba: Authoring and Visualizing Multiverse Analyses

Yang Liu, Alex Kale, Tim Althoff, and Jeffrey Heer

1 INTRODUCTION

The last decade saw widespread failure to replicate findings in published literature across multiple scientific fields [2][3][31][36]. As the replication crisis emerged [1], scholars began to re-examine how data analysis practices might lead to spurious findings. An important contributing factor is the flexibility in making analytic decisions [13][44][42]. Drawing inferences from data often involves many decisions: what are the cutoffs for outliers? What covariates should one include in the statistical models? Different combinations of choices might lead to diverging results and conflicting conclusions. The flexibility in making decisions might inflate false-positive rates when researchers explore multiple alternatives and selectively report desired outcomes [42], a practice known as p-hacking [30]. Even without exploring multiple paths, fixating on a single analytic path might be less rigorous, as multiple justifiable alternatives might exist and picking one would be arbitrary [43]. For example, a crowdsourced study [41] show that well-intentioned experts still produce large variations in analysis outcomes when analyzing the same dataset independently.

In response, prior work proposes multiverse analysis, an approach to outline all “reasonable” alternatives a-priori, exhaust all possible combinations between them, execute the end-to-end analysis per combination, and interpret the outcomes collectively [43][44]. A multiverse analysis demonstrates the extent to which conclusions are robust to sometimes arbitrary analytic decisions [43][44]. Furthermore, reporting the full range of possible results, not just those which fit a particular hypothesis or narrative, helps increase the transparency of a study [59].

However, researchers face a series of barriers when performing multiverse analyses. Authoring a multiverse is tedious, as researchers are no longer dealing with a single analysis, but hundreds of forking paths resulting from possible combinations of analytic decisions. Without proper scaffolding, researchers might resort to multiple, largely redundant analysis scripts [23], or rely on error-prone control flows involving nested for-loops and if-statements [44]. Interpreting the outcomes of a vast number of analyses is also challenging. Besides gauging the overall robustness of the findings, researchers often seek to understand what decisions are critical in obtaining particular outcomes (e.g., [43][44][51]). As multiple decisions might interact, understanding the nuances in how decisions affect robustness will require a comprehensive exploration, suggesting a need for an interactive interface.

Fig. 1. Authoring and visualizing multiverse analyses with Boba. Users start by annotating a script with analytic decisions (a), from which Boba synthesizes a multiplex of possible analysis variants (b). To interpret the results from all analyses, users start with a graph of analytic decisions (c), where sensitive decisions are highlighted in darker blues. Clicking a decision node allows users to compare point estimates (d, blue dots) and uncertainty distributions (d, gray area) between different alternatives. Users may further drill down to assess the fit quality of individual models (e) by comparing observed data (orange) with model predictions (teal).

Abstract—Multiverse analysis is an approach to data analysis in which all “reasonable” analytic decisions are evaluated in parallel and interpreted collectively, in order to foster robustness and transparency. However, specifying a multiverse is demanding because analysts must manage myriad variants from a cross-product of analytic decisions, and the results require nuanced interpretation. We contribute Boba: an integrated domain-specific language (DSL) and visual analysis system for authoring and reviewing multiverse analyses. With the Boba DSL, analysts write the shared portion of analysis code only once, alongside local variations defining alternative decisions, from which the compiler generates a multiplex of scripts representing all possible analysis paths. The Boba Visualizer provides linked views of model results and the multiverse decision space to enable rapid, systematic assessment of consequential decisions and robustness, including sampling uncertainty and model fit. We demonstrate Boba’s utility through two data analysis case studies, and reflect on challenges and design opportunities for multiverse analysis software.

Index Terms—Multiverse Analysis, Statistical Analysis, Analytic Decisions, Reproducibility

---

• All authors are with the University of Washington. E-mails: yliu0, kalea, althoff, jheer@uw.edu.

Manuscript received xx xxx, 201x; accepted xx xxx, 201x. Date of Publication xx xxx, 201x; date of current version xx xxx, 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxx
To lower these barriers, we present Boba, an integrated domain-specific language (DSL) and visualization system for multiverse authoring and interpretation. Rather than managing myriad analysis versions in parallel, the Boba DSL allows users to specify the shared portion of the analysis code only once, alongside local variations defining alternative analysis decisions. The compiler enumerates all compatible combinations of decisions and synthesizes individual analysis scripts for each path. As a meta-language, the Boba DSL is agnostic to the underlying programming language of the analysis script (e.g., Python or R), thereby supporting a wide range of data science use cases.

The Boba Visualizer facilitates assessment of the output of all analysis paths. We support a workflow where users view the results, refine the analysis based on model quality, and commit the final choices to making inference. In the initial stage, the system provides linked views of both analysis results and the multiverse decision space to enable a systematic exploration of how decisions do (or do not) impact outcomes. In addition to revealing decision sensitivity, the visualizer allows users to take into account sampling uncertainty and model fit by comparing observed data with model predictions [11]. Users can exclude models poorly suited for inference by adjusting a model fit threshold, or adopt a principled approach based on model averaging to incorporate model fit in inference. We discuss the implications of post-hoc refinement, along with other challenges in multiverse analysis, in our design reflections.

We evaluate Boba in a code comparison example and two data analysis case studies. We first demonstrate how the Boba DSL eliminates custom control-flows when implementing a real-world multiverse of considerable complexity. Then, in two multiverses replicated from prior work [43,51], we show how the Boba Visualizer affords multiverse interpretation, enabling a richer understanding of robustness, decision patterns, and model fit quality via visual inspection. In both case studies, model fit visualizations surface previously overlooked issues and change what one can reasonably take away from these multiverses.

2 Related Work

We draw from prior work on visualizing and authoring multiverse analyses, and strategies for authoring alternative programs and designs.

2.1 Multiverse Analysis

Analysts begin a multiverse analysis by identifying reasonable analytic decisions a-priori [33,44]. Prior work defines reasonable decisions as those with firm theoretical and statistical support [43], and decisions can span the entire analysis pipeline from data collection and wrangling to statistical modeling and inference [27,48]. While general guidelines such as a decision checklist [48] exist, defining what decisions are reasonable still involves a high degree of researcher subjectivity.

The next step in multiverse analyses is to exhaust all compatible decision combinations and execute the analysis variants (we call a variant a universe). Despite the growing interest in performing multiverse analysis (e.g., [6,18,32,38]), few tools currently exist to aid authoring. Young and Holsteen [51] developed a STATA module that simplifies multimodel analysis into a single command, but it only works for simple variable substitution. Rdjfanalysis [10], an R package, supports more complex alternative scenarios beyond simple value substitution, but the architecture assumes a linear sequential relationship between decisions. Our DSL similarly provides scaffolding for specifying a multiverse, but it has a simpler syntax, extends to other languages, and handles procedural dependencies between decisions.

After running all universes to obtain a set of results, the next task is to interpret these results collectively. Some prior studies visualize results from individual universes by either juxtaposition [37,43,44] or animation [9]. Visualizations in other studies apply aggregation [8,35], for example showing a histogram of effect sizes. The primary issue with juxtaposing or animating individual outcomes is scalability, though the issue might be circumvented by sampling [43]. Our visualizer shows individual outcomes, but overlays or aggregates outcomes in larger multiverses to provide scalability.

In addition to gauging overall robustness, many studies also investigate which analytic decisions are most consequential. The simplest approach is a table [5,7,37,44] where rows and columns map to decisions, and cells represents outcomes from individual universes. Simonsohn et al. [43] extend this idea, visualizing the decision space as a matrix beneath a plot of sorted effect sizes. These solutions might not scale as they juxtapose individual outcomes, and the patterns of how outcomes vary might be difficult to identify depending on the spatial arrangements of rows and columns. Another approach slices the aggregated distribution of outcomes along a decision dimension to create a trellis plot [35]. The trellis plot shows how results vary given a decision, but does not convey whether decisions are prominent given certain results. Our visualizer uses trellis plots and supplements it with brushing to show how decisions contribute to particular results.

Finally, prior work relies on various strategies to perform inference: given a multiverse, does a hypothesized effect indeed occur? The simplest approach is counting, for example reporting the fraction of universes having a significant p-value [5,44] and/or having an effect with the same sign [6]. Young and Holsteen [51] calculate a robustness ratio analogous to the t-statistic. Simonsohn et al. [43] compare the actual multiverse results to a null distribution obtained from randomly shuffling the variable of interest. We build upon Simonsohn’s approach and use weighted model averaging based on model fit quality [50] to aggregate uncertainty across universes.

2.2 Authoring Alternative Programs and Designs

Prior work observes that analysts often manage alternative solutions from exploratory work by making duplicate code snippets and files, but these ad-hoc variants can be messy and difficult to keep track of [15,23]. Provenance tracking tools, especially those with enhanced history interactions [23,24], provide a mechanism to track and restore alternative versions. In Variolite [23], users can select a chunk of code directly in an editor to create and version alternatives. Our DSL similarly allows users to insert local alternatives in code, but instead of assuming that users interact with one version at a time, we generate multiple variants mapping to possible combinations of alternatives.

A related line of work supports manipulating multiple alternatives simultaneously. Techniques like subjunctive interfaces [28,29] and Parallel Pies [46] embed and visualize multiple design variants in the same space, and Parallel Pies allows users to edit multiple variants in parallel. Juxtapose [16] extends the mechanism to software development, enabling users to author program alternatives as separate files and edit code duplicates simultaneously with linked editing. A visualization authoring tool for responsive design [17] also enables simultaneous editing across variants. Our DSL uses a centralized template such that edits in the shared portion of code affect all variants simultaneously.

3 Design Requirements

Our overarching goal is to make it easier for researchers to conduct multiverse analyses. From prior literature and our past experiences, we identify barriers in authoring a multiverse and visualizing its results, and subsequently identify tasks that our tools should support.

3.1 Requirements for Authoring Tools

As noted in prior work [9,27], specifying a multiverse is tedious. This is primarily because a multiverse is composed of many forking paths, yet non-linear program structures are not well supported in conventional tools [40]. One could use a separate script per analytic path, such that it is easy to reason with an individual variant, but these largely redundant variants are difficult to keep track of, let alone update [23]. Alternatively, one could rely on control flows in a single script to simulate nonlinear execution of the forking paths, but it is hard to selectively inspect and rerun a single path because useful snippets are embedded in convoluted control flow structures. Instead, a tool should eliminate the need to write redundant code and custom control flows, while at the same time allowing analysts to simultaneously update variants and reason with a single variant.

Compared to arbitrary non-linear paths from an iterative exploratory analysis, the forking paths in multiverses are usually highly systematic, as they are a cross-product of analytic decisions. We take advantage of this characteristic, and account for other scenarios common in existing multiverse analyses. We distill the following design requirements:
R1: **Multiplexing.** Users should be able to specify a multiverse by writing the shared portion of the analysis source code along with analytic decisions, while the tool creates the forking paths for them. Users should also be able to reason about a single universe and update all universes simultaneously.

R2: **Decision Complexity.** Decisions come in varying degrees of complexity, from simple value replacements (e.g., cutoffs for excluding outliers) to elaborate logic requiring multiple lines of code to implement. The tool should allow succinct ways to express simple value replacements while at the same time support more complex decisions.

R3: **Procedural Dependency.** Existing multiverses [6, 44] contain procedural dependencies [27], in which a downstream decision only exists if a particular upstream choice is made. For example, researchers do not need to choose priors if using a Frequentist model instead of a Bayesian model. The tool should support procedural dependencies.

R4: **Decision Idiosyncrasies.** Due to the idiosyncrasies in implementation, the same conceptual decision can manifest in multiple forms. For example, the same set of parameters can appear in different formats to comply with different function APIs. Users should be able to specify different implementations of a high-level decision.

R5: **Language Agnostic.** Users should be able to author their analysis in any programming languages, as potential users are from various disciplines adopting different workflows and programming languages.

### 3.2 Task Analysis for Visual Analysis System
After researchers execute all analytic paths in a multiverse and obtain corresponding results, they face additional challenges in interpreting the results collectively. The primary task in prior work (Sect. 2) is to understand the overall robustness of outcomes across all reasonable specifications. If the robustness test indicates conflicting conclusions, a natural follow-up task is to identify consequential decisions: what decisions are critical to obtaining a particular conclusion? What decisions produce large variations in results?

Besides these common tasks among prior multiverse analyses, we also propose additional tasks to cover potential blind spots in the literature. First, besides point estimates, a tool should convey appropriate uncertainty information. Such information would help users gauge the end-to-end uncertainty caused by both sampling and decision variations, and compare the variance between conditions. Second, it is important to assess the model fit quality to distinguish trustworthy models from the ones producing questionable estimates.

Uncertainty information and fit issues become particularly important during statistical inference. Users should be able to propagate uncertainty in the multiverse to support judgments about the overall reliability of effects, and they should be able to refine the multiverse to exclude models with fit issues before proceeding to make inferences.

To summarize, we identify the following main tasks that our visual analysis interface should support:

- **T1:** Decision Overview – gain an overview of the decision space to understand the multiverse and contextualize subsequent tasks.
- **T2:** Robustness Overview – gauge the overall robustness of findings obtained through all reasonable specifications.
- **T3:** Decision Impacts – identify what combinations of decisions lead to large variations in outcomes, and what combinations of decisions are critical in obtaining specific outcomes.
- **T4:** Uncertainty – assess the end-to-end uncertainty as well as uncertainty associated with individual universes.
- **T5:** Model Fit – assess the model fit quality of individual universes to distinguish trustworthy models from questionable ones.
- **T6:** Inference – perform statistical inference to judge how reliable the hypothesized effect is, while accounting for model quality.

Besides the tasks, our system should be compatible with the following data characteristics (D1): the visual encoding should be scalable to large multiverses and large input datasets.

### 4 The Boba DSL
We design a domain-specific language (DSL) to simplify the authoring of multiverse analyses. With the DSL, users annotate the source code of their analysis to indicate decision points and alternatives, and provide additional information for procedural dependencies between decisions. The compiler takes as input an annotated script and accompanying metadata, and produces a collection of universe scripts, each containing the code to execute one analytic path in the multiverse (Fig. 1, R1). An example Boba specification for a small multiverse is shown in Fig. 2.

#### 4.1 Language Constructs
The basic language primitives in the Boba DSL consist of source code, placeholder variables, blocks, constraints, and code graphs.

##### 4.1.1 Source Code
The most basic ingredient of an annotated script is the source code (Fig. 2, black text). The compiler treats the source code as a string of text, which according to further language rules will be synthesized into text in the output files. As the compiler is agnostic about the semantics of the source code, users are free to write the source code in any programming language (R5).

##### 4.1.2 Placeholder Variables
Placeholder variables are useful to specify decisions points consisting of simple value substitution (R2). To define a placeholder variable, users provide an identifier and a set of possible alternative values that the variable can take up (Fig. 2, blue text). To use the variable, users insert the identifier into any position in the source code. During synthesis, the compiler removes the identifier and replaces it with one of its alternative values. Variable definition may occur at the same place as its usage (Fig. 2a) or ahead of time inside the config block (supplemental Fig. 2).

##### 4.1.3 Code Blocks
The mechanism for code blocks (Fig. 2, pink text) divides the source code into multiple snippets of one or more lines of code, akin to cells in a computational notebook. A block can be a normal block (Fig. 2, block A), or a decision block (Fig. 2, block M) with multiple versions. The content of a normal block will be shared by all universes, whereas only one version of the decision block will appear in a universe. Decision blocks allow users to specify alternatives that require more elaborate logic to define (R2). In the remainder of Sect. 4, decision points refers to placeholder variables and decision blocks.

With the constructs introduced so far, a natural way to express procedural dependency (R3) is to insert a placeholder variable in some, but not all versions of a decision block. For example, in Fig. 2 the variable `brm_family` only exists when `bayesian` of block M is chosen.
4.1.4 Constraints

By default, Boba assumes all combinations between decision points are valid. Constraints allow users to express dependencies between decisions. For example, infeasible combinations, which will restrict the universes to a smaller set. Boba supports two types of constraints: procedural dependencies (R3) and linked decisions (R4).

A procedural dependency constraint is attached to a decision point or one of its alternatives, and has a conditional expression to determine when the decision/alternative should exist (Fig. 3, orange text). Variables within the scope of the conditional expression are declared decision points, and the values are the alternatives that the decision points have taken up. For example, the first constraint in Fig. 3 indicates that ECL computed is not compatible with NMO reported.

The second type of constraint allows users to link multiple decision points, indicating that these decision points are different manifestations of a single conceptual decision (R4, see supplemental Fig. 2). Linked decisions have one-to-one mappings between their alternatives, such that the i-th alternatives are chosen together in the same universe. One-to-one mappings can also be expressed using multiple procedural dependencies, but linked decisions make them easier to specify.

4.1.5 Code Graph

Users may further specify the execution order between code blocks as a directed acyclic graph (DAG), where a parent block executes before its child. To create a universe, the compiler selects a linear path from the start to the end, and concatenates the source code of the blocks along the path. Branches in the graph represent alternative paths that will appear in different universes. With the graph, users can flexibly express complex dependencies between blocks, including procedural dependencies (R3). For example, to indicate that block Prior should only appear after block Bayesian but not block Frequentist, the user may simply make Prior a descendant of Bayesian but not Frequentist.

4.2 Compilation and Runtime

The compiler parses the input script, computes compatible combinations between decisions, and generates output scripts. During parsing, the compiler extracts the corresponding language primitives from the input file. To enumerate compatible combinations, the compiler proceeds in the following steps. First, it obtains the DAG specifying the execution relationships between code blocks. If users omit the DAG, the compiler creates a default graph which is a linear path of all code blocks depending on their order in the input script. The compiler modifies the DAG to incorporate decision blocks and constraints. Then, it computes all possible paths from any source node with no input edges to any destination node with no output edges. For each path, the compiler further expands placeholder variables (i.e., it enumerates all possible combinations between them). Finally, for each path and variable combination, the compiler concatenates source code along the path, replaces placeholder variables with corresponding values, and outputs a universe script. It also outputs a summary table that keeps track of all the decisions made in each universe, along with other intermediate data that can be ingested into the Boba Visualizer.

Boba infers the language of the input script based on its file extension and uses the same extension for output scripts. These output scripts might be run with the corresponding script language interpreter. Universe scripts log the results into separate files, which will be merged together after all scripts finish execution. As the universe scripts are responsible for computation such as extracting point estimates and computing uncertainty, we provide language-specific utilities for a common set of model types to generate these visualizer-friendly outputs.

We provide a command-line tool for users to (1) invoke the compiler, (2) execute the generated universe scripts, (3) merge the universe outputs, and (4) invoke the visualizer as a local server reading the output files. The compiler and the command-line tool are implemented in Python and are available on GitHub as open source software.

4.3 Example: Replicating a Real-World Multiverse

We use a real-world multiverse example [44] to illustrate how the Boba DSL eliminates the need for custom control flows otherwise required for authoring a multiverse in a single script. The multiverse, originally proposed by Steegen et al. [44], contains five decisions and a procedural dependency. Fig. 3 shows a markup of the R code implemented by the original authors (we modified the lines in purple to avoid computing infeasible paths). The script starts with five nested for-loops (yellow highlight) to repeat the analysis for every possible combination of the five decisions. Then, depending on the indices of the current decisions, the authors either index into an array, or use if-statements to define alternative program behaviors (blue highlight). Finally, to implement a procedural dependency, it is necessary to skip the current iteration when incompatible combinations occur (purple highlight).

The resulting script has multiple issues. First, the useful snippets defining multiverse behavior are embedded in the custom control flow structure, making them hard to find and read. Second, it is difficult to keep track of which option is which. Third, without additional error-handling and checkpoint mechanisms, an error in the middle will terminate the program before any results are saved.

The corresponding specification in the Boba DSL is shown in Fig. 3b. The script starts directly with the prepossessing code shared by all universes. It then uses decision code blocks to define alternative snippets in decision NMO and ECL, and uses a placeholder variable to simulate the value array for a simpler decision F. It additionally specifies constraints (orange text) to signal infeasible paths. Compared to Fig. 3, this script reduces the amount of boilerplate code needed for control-flows, and useful snippets are no longer embedded in convoluted structures. The specification compiles to 120 separate files. Errors in one universe no longer affect the successful completion of others due to distributed execution, and users can selectively review and debug a single analysis.

5 The Boba Visualizer

Next, we introduce Boba Visualizer, a visual analysis system for interpreting the outputs from all analysis paths. We present the system features and design choices in a fictional usage scenario where Emma, an HCI researcher, uses the visualizer to explore a multiverse on data collected in her experiment. We construct the multiverse based on how the authors of a published research article [45] might analyze their data, but the name “Emma” and her workflow are fictional.

Emma runs an experiment to understand whether a webpage reader view improves reading speed. To ensure that her conclusion does not depend on idiosyncratic specifications, Emma identifies six analytic decisions in her analysis pipeline, including choices in data preprocessing and statistical modeling. She then writes a multiverse specification in the Boba DSL, compiles it to 216 analysis scripts, and runs all scripts to obtain a collection of outputs. Now that Emma has hundreds of effect sizes and confidence intervals, she wants to gauge the robustness of the analysis: do the effect sizes support her hypothesis that reader view improves reading speed? Which decisions lead to conflicting results? Are the estimates coming from trustworthy models? To answer these questions, she loads the outputs into the Boba Visualizer.
As the overall outcome distribution suggests conflicting conclusions, Emma wants to investigate what decisions lead to changes in results. She first familiarizes herself with the available decisions.

Then, we scale the height of the area chart such that the total area viewed again has a one-to-one mapping between a visual element and a visualization, by showing a collection of probability density functions (PDFs) or cumulative distribution functions (CDFs) (Fig. 6). In both views, we draw a cubic basis spline for the corresponding distribution per universe, and reduce the opacity of the curves to visually “merge” the curves within the same space. The PDFs and CDFs views again have a one-to-one mapping between a visual element and a universe to afford interactions. Users can switch between aggregated uncertainty, PDFs, and CDFs from a dropdown menu. To help con-
What decisions lead to negative estimates? Emma brushes negative estimates in a subplot (Fig. 5c) and inspects option ratios (Fig. 5d). When the user brushes a range of results, the bars are updated accordingly and uncertainty views to compare the variance between conditions. The interaction allows users to systematically examine the empirical distribution of results following the $i$-th alternative, and $S = \{1, 2, \ldots, k\}$ where $k$ is the number of alternatives in $D$. We then take the median of $K$ as the sensitivity score $S_D$. In both methods, we map $S_D$ to a single-hue color ramp of blue shades. As the F-Test relies on variance, which is not a reasonable measure for dispersion of some distributions, Boba uses the nonparametric K–S statistic by default. Users can override the default in the config file.

5.3 Facet and Brushing

Seeing that the decision “Device” has a large impact, Emma clicks on the node to further examine how results vary (Fig. 3). She finds that one condition exclusively produces point estimates around zero (Fig. 3a) and it also has a much larger variance (Fig. 3b).

Clicking a node in the decision graph divides the outcome distribution into a trellis plot, where each subplot shows the subset of universes adopting an alternative. The interaction allows users to systematically examine the trends and patterns caused by a decision (T3). Akin to the overall outcome distribution, users can toggle between point estimates and uncertainty views to compare the variance between conditions. The trellis plot can be further divided on a second decision via shift-clicking a second node to show the interaction between two decisions. With faceting, users may comprehensively explore the data to sanity-check and assess unexpected patterns, by viewing all univariate and bivariate plots. Alternatively, as sensitive decisions are automatically highlighted, users might quickly locate and examine consequential decisions.

What decisions lead to negative estimates? Emma brushes negative estimates in a subplot (Fig. 5) and inspects option ratios (Fig. 5a).

Users may brush a region in the outcome view to investigate responsible decisions in the option ratio view. By showing the relative percentages of alternatives, the option ratio view reveals dominating alternatives that produce specific results (T3).

This view visualizes each decision as a stacked bar chart to illustrate the percentages of results coming from different alternatives. When the user brushes a range of results, the bars are updated accordingly to reflect changes, and dominating alternatives (those having a higher percentage than default) are highlighted. For example, Emma notices that the lmer model and two sets of fixed effects are particularly responsible for the negative outcomes in Fig. 5. We color the bar segments using a categorical color scale to make bars visually distinguishable.

5.4 Model Fit View

Now that Emma understands what decisions lead to null effects, she wonders if these results are from trustworthy models. She changes the color-by field to get an overview of model fit quality (Fig. 7a) and sees that the universes around zero have a poorer fit. She then uses a slider to remove universes that fail to meet a quality threshold (Fig. 7b).

Boba enables an overview of model fit quality across all universes (T5) by coloring the outcome view with a model quality metric (Fig. 7). We use $k$-fold cross validation [47] to compute model quality, as metrics such as Akaike Information Criterion cannot be used to compare model fit across classes of models (e.g., hierarchical vs. linear) [12]. Prior work shows that cross validation performs better in estimating predictive density for a new dataset than information criteria [47], suggesting that it is a better approximation of out-of-sample predictive validity.

We map the normalized root mean squared error (NRMSE) to a single-hue colormap of blue shades where a darker blue indicates a better fit. To obtain NRMSE, we first compute the overall mean squared prediction error (MSE) from a $k$-fold cross validation

$$MSE = \frac{1}{k} \sum_{j=1}^{k} \frac{1}{n_j} \sum_{i=1}^{n_j} (y_i - \hat{y}_i)^2$$

where $k$ is the number of folds (we set $k = 5$ in all examples), $n_j$ is the size of the test set in the $j$-th fold, $y_i$ is the observed value, and $\hat{y}_i$ is the predicted value. We then normalize the MSE by the span of the maximum $y_{\text{max}}$ and minimum $y_{\text{min}}$ values of the observed variable:

$$\text{NRMSE} = \sqrt{MSE/(y_{\text{max}} - y_{\text{min}})}$$

To further investigate model quality, Emma drills down to individual universes by clicking a dot in the outcome view. She sees in the model fit view (Fig. 7) that a model only produces one possible estimate.

Clicking a result in the outcome view populates the model fit view with visual predictive checks, which show how well predictions from a given model replicate the empirical distribution of observed data (T1), allowing users to further assess model quality (T5). The model fit visualization juxtaposes violin plots of the observed data and model predictions to facilitate comparison of the two distributions (see Fig. 1). We additionally overlay raw data points within the violin plots using a centered density dot plot layout, to help reveal any discrepancies in approximation due to kernel density estimation in the violin plots.

When the number of observations is large (D1), we quantize the raw data, sampling at evenly spaced percentiles, and plot this representative subset as centered quantile dotplots [22]. As clicking individual universes can be tedious, the model fit view suggests additional universes that have similar point estimates to the selected universe.

5.5 Inference

After an in-depth exploration, Emma proceeds to the final step, asking “given the multiverse, how reliable is the effect?” She confirms a warning message (T6) that the hypothesized effect is (T6). Bona provides an inference view at the end of the analysis workflow, after users have engaged in exploration. Once
in the inference view, all earlier views and interactions are inaccessible to avoid multiple comparison problems arising from repeated inference. The inference view contains different plots depending on the outputs from the authoring step, so that users can choose between robust yet computationally-expensive methods and simpler ones.

A more robust inference uses permutation tests for data with random assignment to generate the expected effect size under the null hypothesis of no effect is true. Specifically, we shuffle the column with the randomly assigned variable (reader view in this case) \( N \) times, run the multiverse of size \( M \) on each of the shuffled datasets, and obtain \( N \times M \) point estimates. As there is no link between reader view and speed in the shuffled datasets, these \( N \times M \) point estimates constitute the null distribution.

When the null distribution is available, the inference view shows an aggregate plot followed by a detail plot (Fig. 8b). The aggregate plot (Fig. 8a) compares the null distribution (red) to possible outcomes of the actual multiverse (blue) across sampling and decision variations. The detail plot (Fig. 8b) depicts individual universes instead, showing the point estimate (colored dot) against the range within the 2.5th and 97.5th percentile of the null distribution (gray line). The point estimate is orange if it is outside the range, or blue otherwise. Underneath both plots, we provide detailed descriptions (supplemental Fig. 1) to guide users in interpretation: For the aggregate plot, we prompt users to compare the distance between the averages of the two densities to the spread. For the detail plot, we count the number of universes with the point estimate outside its corresponding range. If the null distribution is unavailable, Boba shows a simpler aggregate plot (Fig. 8a) where the expected effect size under the null hypothesis is marked with a red line.

In addition, Boba enables users to propagate concerns in model fit quality to the inference view in two possible ways. The first way employs a model averaging technique called stacking to take a weighted combination of the universes according to their model fit quality. The technique learns a simplex of weights, one for each universe model, via optimization that maximizes the log-posterior-fit quality. The technique learns a simplex of weights, one for each universe model, via optimization that maximizes the log-posterior-fit quality. The technique learns a simplex of weights, one for each universe model, via optimization that maximizes the log-posterior-fit quality. The technique learns a simplex of weights, one for each universe model, via optimization that maximizes the log-posterior-fit quality. The technique learns a simplex of weights, one for each universe model, via optimization that maximizes the log-posterior-fit quality. The technique learns a simplex of weights, one for each universe model, via optimization that maximizes the log-posterior-fit quality. The technique learns a simplex of weights, one for each universe model, via optimization that maximizes the log-posterior-fit quality. The technique learns a simplex of weights, one for each universe model, via optimization that maximizes the log-posterior-fit quality. The technique learns a simplex of weights, one for each universe model, via optimization that maximizes the log-posterior-fit quality. The technique learns a simplex of weights, one for each universe model, via optimization that maximizes the log-posterior-fit quality.

6 Case Studies

We evaluate Boba through a pair of analysis case studies, where we implement the multiverse using the Boba DSL and interpret the results using the Boba Visualizer. The supplemental material contains the Boba specifications of both examples, additional figures, and a video demonstrating all the interactions described below.

6.1 Case Study: Mortgage Analysis

The first case study demonstrates how analysts might quickly arrive at insights provided by summary statistics in prior work, while at the same time gaining a richer understanding of robustness patterns. We also show that by incorporating uncertainty and model fit checks, Boba surfaces potential issues that prior work might have neglected.

Young et al. [51] propose a multimodel analysis approach to gauge whether model estimates are robust to alternative model specifications. Akin to the philosophy of multiverse analysis, the approach takes all combinations of possible control variables in a statistical model. The outputs are multiple summary statistics, including (1) an overall robustness ratio, (2) uncertainty measures for sampling and modeling variations, and (3) metrics reflecting the sensitivity of each variable.

As an example, the authors present a case study on mortgage lending, asking "are female applicants more likely to be approved for a mortgage?" They fit a multiverse of linear regression models with a term for female and other control variables capturing demographic and financial information. The resulting summary statistics indicate that the estimate is not robust to modeling decisions with large end-to-end uncertainty, and two control variables, married and black, are highly influential. These summary statistics provide a powerful synopsis, but may fail to convey more nuanced patterns in result distributions. The authors manually create additional visualizations to convey interesting trends in data, for instance the estimates follow a multimodal distribution. These visualizations, though necessary to provide a richer understanding of model robustness, are ad-hoc and not included in the software package.

We replicate the analysis in Boba. The Boba DSL specification simply consists of eight placeholder variables, each indicating whether a control variable is included in the model formula. Then, we compile the specification to 256 scripts, run them all, and start the Boba Visualizer.

We first demonstrate that the default views in the Boba Visualizer afford similar insights on uncertainty, robustness, and decision sensitivity. Upon launching the visualizer, we see a decision graph and an overall outcome distribution (Fig. 9). The decision view (Fig. 9)
we assess model fit quality by clicking a dot in the outcome view and visual check that an alternative model, for example logistic regression, view (Fig. 9b, background gray area) has a wide spread, suggesting that view (Fig. 9b) shows that the point estimates are highly varied with uncertainty and model fit are potential blind spots in prior literature. These observations support our arguments in Sect. 3.2 that uncertainty and model fit are potential blind spots in prior literature.

The patterns revealed by ad-hoc visualizations in previous work are also readily available in the Boba Visualizer, either in the default views or with two clicks guided by prominent visual cues. The default outcome view (Fig. 9b) shows that the point estimates follow a multimodal distribution with three separate peaks. Clicking the two highlighted (most sensitive) nodes in the decision view (Fig. 9c), produces a trellis plot (Fig. 9d), where each subplot contains only one cluster. From the trellis plot, it is evident that the leftmost and rightmost peaks in the overall distribution come from two particular combinations of the influential variables. Alternatively, users might arrive at similar insights by brushing individual clusters in the default outcome view.

Finally, the uncertainty and model fit visualizations in Boba surface potential issues that previous work might have overlooked. First, though the point estimates in Fig. 9 fall into three distinct clusters, the aggregated uncertainty distribution appears unimodal despite a wider spread. The PDF plot (Fig. 9e) shows that sampling distribution from one analysis typically spans the range of multiple peaks, thus explaining why the aggregated uncertainty is unimodal. These observations suggest that the multimodal patterns exhibited by point estimates are not robust when we take sampling variations into account. Second, we assess model fit quality by clicking a dot in the outcome view and examining the model fit view (Fig. 9f). As shown in Fig. 9f, while the observed data only takes two possible values, the linear regression model produces a continuous range of predictions. It is clear from this visual check that an alternative model, for example logistic regression, is more appropriate than the original linear regression models, and we should probably interpret the results with skepticism given the model fit issues. These observations support our arguments in Sect. 3.2 that uncertainty and model fit are potential blind spots in prior literature.

6.2 Case Study: Female Hurricanes Caused More Deaths?

Next, we replicate another multiverse example introduced by Simonsohn et al. [20], where the authors challenged the results of a published research article [20]. The original study [20] reports that hurricanes with female names have caused more deaths, presumably because female names are perceived as less threatening and lead to less preparation. However, the conclusion led to a heated debate on proper ways to conduct the data analysis. To understand if the conclusion is robust to alternative specifications, Simonsen et al. identified seven analytic decisions that appear reasonable, including alternative ways to exclude outliers, operationalize femininity, select the model type, and choose covariates. They then conducted a multiverse analysis and interpreted the results in a visualization called the specification curve.

We build the same multiverse using these seven analytic decisions in Boba. In the Boba DSL specification, we use a decision block to specify two alternative model types: negative binomial regression versus linear regression with log-transformed deaths as the dependent variable. The rest of the analytic decisions are placeholder variables that can be expressed as straightforward value substitutions. However, the two different model types lead to further differences in extracting model estimates. For example, we must invert the log-transformation to model estimates. For example, we must invert the log-transformation in the linear model to obtain predictions in the original units. We create additional placeholder variables for implementation differences related to model types and link them with the model decision block. The specification compiles to 1,728 individual scripts.

We then interpret the results using the Boba Visualizer. As shown in the overview distribution (Fig. 10a), the majority of point estimates support a small, positive effect (female hurricanes lead to more deaths, and the extra deaths are less than 20), while some estimates suggest a larger effect. A small fraction of results have the opposite sign.

What analytic decisions are responsible for the variations in the estimates? The decision view indicates that multiple analytic decisions might be influential (Fig. 10b). We click on the relatively sensitive decisions, outliers, damage and model, to examine their impact. In the corresponding univariate trellis plots (supplemental Fig. 3), certain choices tend to produce larger estimates, such as not excluding any outliers, using raw damage instead of log damage, and using negative binomial regression. However, in each of these conditions, a consider-
able number of universes still support a smaller effect, suggesting that it is not a single analytic decision that leads to larger estimates.

Next, we click on two influential decisions to examine their interaction. In the trellis plot of model and damage (Fig. 10d), one combination (choosing both log damage and negative binomial model) produces mostly varied estimates without a dominating peak next to zero. Brushing the large estimates in another combination (raw damage and linear model) indicates that these results are coming from specifications that additionally exclude no outliers. Removing these decision combinations will eliminate the possibility of obtaining a large effect.

But do we have evidence that certain outcomes are less trustworthy? We toggle the color-by drop-down menu so that each universe is colored appropriately for analysis methods that might be synthesized [19]. An-}

verse analysis, conducting a multiverse analysis still requires statistical knowledge to work with a single script.

users can leverage existing debugging tools for the corresponding language to work with a single script.

and difficulties sampling parameter spaces with complex geometries. The large estimates are almost exclusively coming from models with a poor fit. We further verify the model fit quality by picking example universes and examining the model fit view (Fig. 10f). The visual predictive checks confirm issues in model fit, for example the models fail to generate predictions smaller than 3 deaths, while the observed data contains plenty such cases.

Now that we have reasons to be skeptical of the large estimates, the remaining universes still support a small, positive effect. How reliable is the effect? We proceed to the inference view to compare the possible outcomes in the observed multiverse and the expected distribution under the null hypothesis (Fig. 10i). The two distributions are different in terms of mode and shape, yet they are highly overlapping, which suggests the effect is not reliable. The detail plot depicting individual universes (supplemental Fig. 1) further confirms this observation. Out of the entire multiverse, only 3 universes have point estimates outside the 2.5th and 97.5th percentile of the corresponding null distribution.

7 Discussion

Through the process of designing, building, and using Boba, we gain insights into challenges that multiverse analysis poses for software designers and users. We now reflect on these challenges and additional design opportunities for supporting multiverse analysis.

While Boba is intended to reduce the Gulf of execution for multiverse analysis, conducting a multiverse analysis still requires statistical expertise. Future work might attempt to represent expert statistical knowledge to lower the barriers for less experienced users. One strategy is to represent analysis goals in higher-level abstractions, from which appropriate analysis methods might be synthesized [17]. Another is to guide less experienced users through key decision points and possible alternatives [27], starting from an initial script.

Running all scripts produced by Boba can be computationally expensive due to their sheer number. Boba already leverages parallelism, executing universes across multiple processes. Still, scripts often perform redundant computation and the compiler may produce prohibitively many scripts. Future work should include optimizing multiverse execution, for example caching shared computation across universes, or efficiently exploring decision spaces via adaptive sampling.

As a new programming tool, Boba requires additional support to increase its usability, including code editor plugins, debugging tools, documentation, and community help. In this paper we assess the feasibility of Boba, with the understanding that these additional aspects will need to be subsequently addressed. Currently, as Boba specifications are compiled to individual scripts in a specific programming language, users can leverage existing debugging tools for the corresponding language to work with a single script.

However, debugging analysis scripts becomes difficult at the scale of a multiverse because a change that fixes a bug in one script might not fix bugs in others. When we attempt to run a multiverse of Bayesian regression models, for example, models in multiple universes do not converge for a variety of reasons including problems with identifiability and difficulties sampling parameter spaces with complex geometries. These issues are common in Bayesian modeling workflows and must be resolved by adjusting settings, changing priors, or reparameterizing models entirely. At the scale of multiverse analysis, debugging this kind of model fit issue is particularly difficult because existing tools for diagnostics and model checks (e.g., trace and pairs plots) are designed to assess one model at a time. While this points to a need for better debugging and model diagnostic tools in general, it also suggests that these tools must be built with a multiplexing workflow in mind if they are going to facilitate multiverse analysis.

Analysts must take care when reviewing and summarizing multiverse results, as a multiverse is not a set of randomly drawn, independent specifications. In general, the Boba Visualizer avoids techniques that assume universe results are independent and identically distributed (IID). A possible venue for future work is to explicitly account for statistical dependence among universes to remove potential bias. Boba might also do more to aid the communication of results, for example helping to produce reports that communicate multiverse results [9].

Previous approaches to multiverse analysis have largely overlooked the quality of model fit, focusing instead on how to enumerate analysis decisions and display the results from the entire multiverse. We visualize model fit in two ways: we use color to encode the NRMSE from a k-fold cross validation in the outcome view, and use predictive checks to compare observed data with model predictions in the model fit view. Together these views show that a cross-product of analytic decisions can produce many universes with poor model fits, and many of the results that prior studies include in their overviews may not provide a sound base for subsequent inferences. The prevalence of fit issues, which are immediately apparent in the Boba Visualizer, calls into question the idea that a multiverse analysis should consist of a cross-product of all a-priori “reasonable” decisions. We propose adding a step to the multiverse workflow where analysts must distinguish between what seems reasonable a-priori vs. post-hoc. Boba supports this step in two ways: in the inference view we can use model averaging to produce a weighted combination of universes based on model fit, or we can simply omit universes below a certain model fit threshold chosen by the users. The latter option relies on analysts making a post-hoc subjective decision and might be susceptible to p-hacking. However, one can pre-register a model quality threshold to eliminate this flexibility. Should we enable more elaborate and interactive ways to give users control over pruning? If so, how do we prevent analysts from unintentionally biasing the results? These questions remain future work.

Indeed, a core tension in multiverse analysis is balancing the imperative of transparency with the need for principled reduction of uncertainty. Prior work on researcher degrees of freedom in analysis workflows [21] identifies strategies that analysts use to make decisions (see also [4, 26]), including two which are relevant here: reducing uncertainty in the analysis process by following systematic procedures, and suppressing uncertainty by arbitrarily limiting the space of possible analysis paths. In the context of Boba, design choices which direct the user’s attention toward important information (e.g., highlighting models with good fit and decisions with a large influence on outcomes) and guide the user toward best practices (e.g., visual predictive checks) serve to push the user toward reducing rather than suppressing uncertainty. Allowing users to interact with results as individual dots in the outcome view while showing aggregated uncertainty in the background reduces the amount of information that the user needs to engage with in order to begin exploring universes, while also maintaining a sense of the range of possible outcomes. We believe that guiding users’ attention and workflow based on statistical principles is critical.

8 Conclusion

This paper presents Boba, an integrated DSL and visual analysis system for authoring and interpreting multiverse analyses. With the DSL, users annotate their analysis script to insert local variations, from which the compiler synthesizes executable script variants corresponding to all possible analysis paths. We provide a command line tool for compiling the DSL specification, running the generated scripts, and merging the outputs. We contribute a visual analysis system with linked views between analytic decisions and model estimates to facilitate systematic exploration of how decisions impact robustness, along with views for sampling uncertainty and model fit. We also provide facilities for principled pruning of “unreasonable” specifications, and support inference to assess effect reliability. Using Boba, we replicate two existing multiverse studies, gain a rich understanding of how decisions affect results, and find issues around uncertainty and model fit that
We thank the anonymous reviewers, UW IDL members, Uri Simonsohn, Mike Merril, Ge Zhang, Pierre Dragojevic, Yvonne Jansen, Matthew Kay, Brian Hall, Abbraaneal Sarma, Fanny Chevalier, and Michael Moen for their help. This work was supported by NSF Award 1901386.

**Acknowledgments**

Boba is available as open source software at ANONYMIZED-URL.

The statistical crisis in science. *Nature*, 533(7604):452–454, 2016. doi: 10.1038/533452a

Raise standards for preclinical cancer research. *Nature*, 483(7391):531–533, 2012. doi: 10.1038/483533a

No support for historical candidate gene or candidate gene-by-interaction hypotheses for major depression across multiple large samples. *American Journal of Psychiatry*, 176(5):376–387, 2019. doi: 10.1176/appi.ajp.2018.18070881

Is there evidence of racial disparity in police use of deadly force? analyses of officer-involved fatal shootings in 2015–2016. *Social psychological and personality science*, 10(5):586–595, 2019. doi: 10.1177/1948550618775108

Revisiting the pose pose effect: How robust are the results reported by carney, cuddy, and yap (2010) to data analytic decisions? *Social Psychological and Personality Science*, 8(5):493–499, 2017. doi: 10.1177/1948550617715484

Emotion regulation ability mediates the link between depressive symptoms and affective bipolarity. *Cognition and Emotion*, 33(5):1076–1083, 2019. doi: 10.1080/26993317.2018.1524774

M. Cesario, D. J. Johnson, and W. Terrill. Is there evidence of racial disparity in police use of deadly force? *Social psychological and personality science*, 10(5):586–595, 2019. doi: 10.1177/1948550618775108

J. Cesaroi, E. K. Kalokerinos, B. Bastian, and P. Kuppens. The bipolarity of affect and depressive symptoms. *Journal of personality and social psychology*, 114(2):323, 2018. doi: 10.1037/pspp0000186

Increasing the transparency of research papers with explorable multivariate analyses. In *Proc. ACM Human Factors in Computing Systems*, pp. 65:1–65:15, 2019. doi: 10.1145/3290605.3300295

G. L. Nelson, and E. B. Hekler. Researcher-centered design of computational notebooks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2017 doi: 10.1145/3025453.3025738

E. Dejonckheere, E. K. Kalokerinos, B. Bastian, and P. Kuppens. Poor emotion regulation ability mediates the link between depressive symptoms and affective bipolarity. *Cognition and Emotion*, 33(5):1076–1083, 2019. doi: 10.1080/26993317.2018.1524774

E. Dejonckheere, M. Mestdagh, M. Houben, Y. Erbas, M. Pe, P. Koval, A. Brose, B. Bastian, and P. Kuppens. The bipolarity of affect and depressive symptoms. *Journal of personality and social psychology*, 114(2):323, 2018. doi: 10.1037/pspp0000186

J. M. Rohrer, B. Egloff, and S. C. Schmukle. Probing birth-order effects and affective bipolarity. *Cognition and Emotion*, 33(5):1076–1083, 2019. doi: 10.1080/02699931.2018.1524747

J. K. Reinecke. Tea: A high-level language and runtime system for automating statistical analysis. *CoRR*, abs/1904.05387, 2019.
