Tea: A High-level Language and Runtime System for Automating Statistical Analyses

Eunice Jun, Maureen Daum, Jared Roesch
University of Washington

Sarah E. Chasins
University of California, Berkeley

Emery D. Berger
University of Massachusetts Amherst

Rene Just, Katharina Reinecke
University of Washington

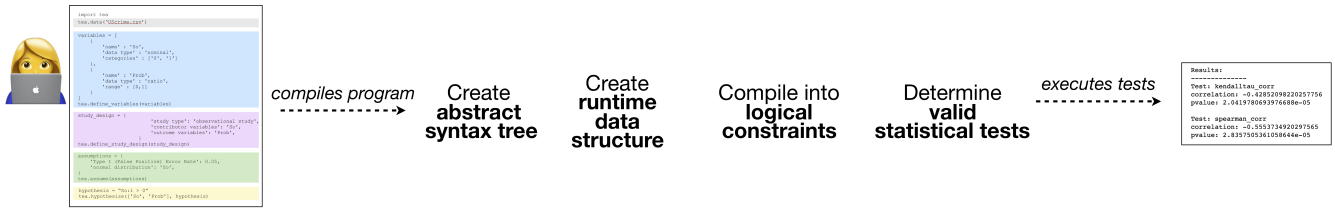

Figure 1: Overview of Tea's language constructs and its runtime system's compilation process. The colors correspond to the annotations provided in Figure 2.

ABSTRACT

Though statistical analyses are centered on research questions and hypotheses, current statistical analysis tools are not. Users must first translate their hypotheses into specific statistical tests and then perform API calls with functions and parameters. To do so accurately requires that users have statistical expertise. To lower this barrier to valid, replicable statistical analysis, we introduce Tea[1], a high-level declarative language and runtime system. In Tea, users express their study design, any parametric assumptions, and their hypotheses. Tea compiles these high-level specifications into a constraint satisfaction problem that determines the set of valid statistical tests, and then executes them to test the hypothesis. We evaluate Tea using a suite of statistical analyses drawn from popular tutorials. We show that Tea generally matches the choices of experts while automatically switching to non-parametric tests when parametric assumptions are not met. We simulate the effect of mistakes made by non-expert users and show that Tea automatically avoids both false negatives and false positives that could be produced by the application of incorrect statistical tests.

[1]named after Fisher's "Lady Tasting Tea" experiment [11]

arxiv, April 2019, on-line
2019. https://doi.org/0000001.0000001

KEYWORDS

statistical analysis, declarative programming language

1 INTRODUCTION

> *The enormous variety of modern quantitative methods leaves researchers with the nontrivial task of matching analysis and design to the research question. - Ronald Fisher [11]*

Since the development of modern statistical methods (e.g., Student's t-test, ANOVA, etc.), statisticians have acknowledged the difficulty of identifying which statistical tests people should use to answer their specific research questions. Almost a century later, choosing appropriate statistical tests for evaluating a hypothesis remains a challenge. As a consequence, errors in statistical analyses are common [20], especially given that data analysis has become a common task for people with little to no statistical expertise.

A wide variety of tools (such as SPSS [46], SAS [45], and JMP [43]), programming languages (R [44]), and libraries (including numpy [32], scipy [17], and statsmodels [37]), enable people to perform specific statistical tests, but they do not address the fundamental problem that users may not know which statistical test to perform and how to verify that specific assumptions about their data hold.

In fact, all of these tools place the burden of valid, replicable statistical analyses on the user and demand deep knowledge of statistics. Users not only have to identify their research questions, hypotheses, and domain assumptions, but also must select statistical tests for their hypotheses (e.g., Student's t-test or one-way ANOVA). For each statistical test, users must be aware of the statistical assumptions each test makes about the data (e.g., normality or equal variance between groups) and how to check for them, which requires additional statistical tests (e.g., Levene's test for equal variance), which themselves may demand further assumptions about the data. This entire process requires significant knowledge about statistical tests and their preconditions, as well as the ability to perform the tests and verify their preconditions. This cognitively demanding process can easily lead to mistakes.

This paper presents Tea, a high-level declarative language for automating statistical test selection and execution that abstracts the details of statistical analysis from the users. Tea captures users' hypotheses and domain knowledge, translates this information into a constraint satisfaction problem, identifies all valid statistical tests to evaluate a hypothesis, and executes the tests. Figure 1 illustrates Tea's compilation process. Tea's higher-level, declarative nature aims to lower the barrier to valid, replicable analyses.

We have designed Tea to integrate directly into common data analysis workflows for users who have minimal programming experience. Tea is implemented as an open-source Python library, so programmers can use Tea wherever they use Python, including within Python notebooks.

In addition, Tea is flexible. Its abstraction of the analysis process and use of a constraint solver to select tests is designed to support its extension to emerging statistical methods, such as Bayesian analysis. Currently, Tea supports frequentist Null Hypothesis Significance Testing (NHST).

The paper makes the following contributions:

- Tea, a novel domain-specific language (DSL) for automatically selecting and executing statistical analyses based on users' hypotheses and domain knowledge (Section 4),
- the Tea runtime system, which formulates statistical test selection as a maximum constraint satisfaction problem (Section 5), and
- an initial evaluation showing that Tea can express and execute common NHST statistical tests (Section 6).

We start with a usage scenario that provides an overview of Tea (Section 2). We discuss the concerns about statistics in the HCI community that shaped Tea's design (Section 3), the implementation of Tea's programming language (Section 4), the implementation of Tea's runtime system (Section 5), and the evaluation of Tea as a whole (Section 6). We then discuss limitations and future work and how Tea is different from related work. We conclude by providing information on how to use Tea.

2 USAGE SCENARIO

This section describes how an analyst who has no statistical background can use Tea to answer their research questions. We use as an example analyst a historical criminologist who wants to determine how imprisonment differed across regions of the US in 1960[2]. Figure 2 shows the Tea code for this example.

The analyst specifies the data file's path in Tea. Tea handles loading and storing the data set for the duration of the analysis session. The analyst does not have to worry about transforming the data in any way.

The analyst asks if the probability of imprisonment was higher in southern states than in non-southern states. The analyst identifies two variables that could help them answer this question: the probability of imprisonment ('Prob') and geographic location ('So'). for non-southern. Using Tea, the analyst defines the geographic location as a dichotomous nominal variable where '1' indicates a southern state and '0' indicates a non-southern state, and indicates that the probability of imprisonment is a numeric data type (ratio) with a range between 0 and 1.

The analyst then specifies their study design, defining the study type to be 'observational study' (rather than experimental study) and defining the independent variable to be the geographic location and the outcome (dependent) variable to be the probability of imprisonment.

Based on their prior research, the analyst knows that the probability of imprisonment in southern and non-southern states is normally distributed. The analyst provides an assumptions clause to Tea in which they specify this domain knowledge. They also specify an acceptable Type I error rate (probability of finding a false positive result), more colloquially known as the 'significance threshold' ($\alpha = .05$) that is acceptable in criminology. If the analyst does not have assumptions or forgets to provide assumptions, Tea will use the default of $\alpha = .05$.

The analyst hypothesizes that southern states will have a higher probability of imprisonment than non-southern states. The analyst directly expresses this hypothesis in Tea. *Note that at no point does the analyst indicate which statistical tests should be performed.*

From this point on, Tea operates entirely automatically. When the analyst runs their Tea program, Tea checks properties of the data and finds that Student's t-test is appropriate.

[2]The example is taken from [8] and [39]. The data set comes as part of the MASS package in R.

Tea executes the Student's t-test and non-parametric alternatives, such as the Mann-Whitney U test, which provide alternative, consistent results.

Tea generates a table of results from executing the tests, ordered by their power (i.e., results from the parametric t-test will be listed first given that it has higher power than the non-parametric equivalent). Based on this output, the analyst concludes that their hypothesis—that the probability of imprisonment was higher in southern states than in non-southern states in 1960—is supported. The results from alternative statistical tests support this conclusion, so the analyst can be confident in their assessment.

The analyst can now share their Tea program with colleagues. Other researchers can easily see what assumptions the analyst made and what the intended hypothesis was (since these are explicitly stated in the Tea program), and reproduce the exact results using Tea.

3 DESIGN CONSIDERATIONS

In designing Tea's language and runtime system, we considered best practices for conducting statistical analyses and derived our own insights on improving the interaction between users and statistical tools.

We identified five key recommendations for statistical analysis from Cairns' report on common statistical errors in HCI [4], which echoes many concerns articulated by Wilkinson Wilkinson [47], and from the American Psychological Association's Task Force on Statistical Inference [1]:

- Users should make their assumptions about the data explicit [1].
- Users should check assumptions statistical tests make about the data and variables and report on the results from conducting tests to check these assumptions [1, 4].
- Users should account for multiple comparisons [1, 4].
- When possible, users should consider alternative analyses that test their hypothesis and select the simplest one [1].
- Users should contextualize results from statistical tests using effect sizes and confidence intervals [1].

An additional practice we wanted to simplify in Tea was *reproducing analyses*. Table 1 shows how Tea compares to current tools in supporting these best practices.

Based on these guidelines, we identified two key interaction principles for Tea:

(1) *Users should be able to express their expertise, assumptions, and intentions for analysis.* Users have domain knowledge and goals that cannot be expressed with the low-level API calls to the specific statistical tests required by the majority of current tools. A higher level of abstraction that focuses on the goals and context of analysis is likely to appeal to users who may not have statistical expertise (Section 4).

(2) *Users should not be burdened with statistical details to conduct valid analyses.* Currently, users must not only remember their hypotheses but also identify possibly appropriate tests and manually check the preconditions for all the tests. Simplifying the user's procedure by automating the test selection process can help reduce cognitive demand (Section 5).

```
import tea
tea.data('UScrime.csv')                                                    1

variables = [
    {
        'name' : 'So',
        'data type' : 'nominal',
        'categories' : ['0', '1']
    },
    {
        'name' : 'Prob',
        'data type' : 'ratio',
        'range' : [0,1]
    }
]
tea.define_variables(variables)                                            2

study_design = {
                        'study type': 'observational study',
                        'independent variables': 'So',
                        'dependent variables': 'Prob',
                    }
tea.define_study_design(study_design)                                      3

assumptions = {
    'normal distribution': 'So',
    'Type I (False Positive) Error Rate': 0.05,
}
tea.assume(assumptions)                                                    4

hypothesis = "So:1 > 0"
tea.hypothesize(['So', 'Prob'], hypothesis)                                5
```

Figure 2: Sample Tea program specification that outlines an experiment to analyze the relationship between geographic location ('So') and probability of imprisonment ('Prob') in a common USCrime dataset [18, 40]. See Section 2 for an explanation of the code. Tea programs specify 1) data, 2) variables, 3) study design, 4) assumptions, and 5) hypotheses.

While there are calls to incorporate other methods of statistical analysis [20, 21], Null Hypothesis Significance Testing (NHST) remains the norm in HCI and other disciplines. Therefore, Tea currently implements a module for NHST with the tests found to be most common by Wacharamanotham et al. [41] (see Table 2 for a list of tests). We believe that Tea's abstraction and modularity will enable the incorporation of other statistical analysis approaches as they move into the mainstream.

4 TEA'S PROGRAMMING LANGUAGE

Tea is a domain-specific language embedded in Python. It takes advantage of existing Python data structures (e.g., classes, dictionaries, and enums). We chose Python because

Table 1: Comparison of Tea to other tools. Despite the published best practices for statistical analyses, most tools do not help users select appropriate tests. Tea not only addresses the best practices but also supports reproducing analyses.

Best practices	SAS	SPSS	JMP	R	Statsplorer	Tea
Explicit statement of user assumptions	–	–	–	–	–	✓
Automatic verification of test preconditions	–	–	sometimes	sometimes	✓	✓
Automatic accounting of multiple comparisons	–	–	–	–	✓	✓
Surface alternative analyses	–	–	–	–	–	✓
Contextualize results	✓	sometimes	✓	sometimes	✓	✓
Easy to reproduce analysis	✓	✓	–	✓	–	✓

of its widespread adoption in data science. Tea is itself implemented as a Python library.

A key challenge in describing studies is determining the level of granularity necessary to produce an accurate analysis. In Tea programs, users describe their studies in five ways: (1) providing a data set, (2) describing the variables of interest in that data set, (3) describing their study design, (4) explicitly stating their assumptions about the variables, and (5) formulating hypotheses about the relationships between variables.

Data

Data is required for executing statistical analyses. One challenge in managing data for analysis is minimizing both duplicated data and user intervention.

To reduce the need for user intervention for data manipulation, Tea requires the data to be a CSV in long format. CSVs are a common output format for data storage and cleaning tools. Long format (sometimes called "tidy data" [42]) is a denormalized format that is widely used for collecting and storing data, especially for within-subjects studies.

Unlike R and Python libraries such as numpy [32], Tea only requires one instance of the data. Users do not have to duplicate the data or subsets of it for analyses that require the data to be in slightly different forms. Minimizing data duplication or segmentation is also important to avoid user confusion about where some data exist or which subsets of data pertain to specific statistical tests.

Optionally, users can also indicate a column in the data set that acts as a relational (or primary) key, or an attribute that uniquely identifies rows of data. For example, this key could be a participant identification number in a behavioral experiment. A key is useful for verifying a study design, described below. Without a key, Tea's default is that all rows in the dataset comprise independent observations (that is, all variables are between subjects).

Variables

Variables represent columns of interest in the data set. Variables have a name, a data type (*nominal*, *ordinal*, *interval*, or *ratio*), and, when appropriate, valid categories. Users (naturally) refer to variables through a Tea program using their names. Only nominal and ordinal variables have a list of possible categories. For ordinal variables, the categories are also ordered from left to right.

Variables encapsulate queries. The queries represent the index of the variable's column in the original data set and any filtering operations applied to the variable. For instance, it is common to filter by category for nominal variables in statistical tests.

Study Design

Three aspects of study design are important for conducting statistical analyses: (1) the type of study (observational study vs. randomized experiment), (2) the independent and dependent variables, and (3) the number of observations per participant (e.g., between-subjects variables vs. within-subjects variables).

For semantic precision, Tea uses different terms for independent and dependent variables for observational studies and experiments. In experiments, variables are described as either "independent" or "dependent" variables. In observational studies, variables are either "contributor" (independent) or "outcome" (dependent) variables. If variables are neither independent nor dependent, they are treated as covariates.

Assumptions

Users' assumptions based on domain knowledge are critical for conducting and contextualizing studies and analyses. Often, users' assumptions are particular to variables and specific properties (e.g., equal variances across different groups). Current tools generally do not require that users encode these assumptions, leaving them implicit.

Tea takes the opposite approach to contextualize and increase the transparency of analyses. It requires that users be explicit about assumptions and statistical properties pertaining to the analysis as a whole (e.g., acceptable Type I error rate/significance threshold) and the data.

```
# One-sided comparisons between groups
hypothesis = 'Region: Southern > Northern'
hypothesis = 'Region: Northern < Southern'

#Partial orders
hypothesis = 'Region: Southern > Southwest,
             Region: Northeast > Midwest'

# Two-sided comparisons
hypothesis = 'Region: Southern != Northern'

# Positive linear relationships
hypothesis = 'Imprisonment ~ Region'
hypothesis = 'Imprisonment ~ +Region'

# Negative linear relationships
hypothesis = 'Imprisonment ~ -Region'

# Under development
hypothesis = 'Region: Southern > 1.5 * Northern'

tea.hypothesize(['Region', 'Imprisonment'], hypothesis)
```

Figure 3: Hypotheses that users can express in Tea.

Hypotheses

Hypotheses drive the statistical analysis process. Users often have hypotheses that are technically alternative hypotheses.

Tea focuses on capturing users' alternative hypotheses about the relationship between two or more variables. Tea uses the alternate hypothesis to conduct either a two-sided or one-sided statistical test. By default, Tea uses the null hypothesis that there is no relationship between variables.

Figure 3 exemplifies the range of hypotheses Tea supports.

5 TEA'S RUNTIME SYSTEM

Tea compiles programs into logical constraints about the data and variables, which it resolves using a constraint solver. A significant benefit of using a constraint solver is extensibility. Adding new statistical tests does not require modifying the core of Tea's runtime system. Instead, defining a new test requires expressing a single new logical relationship between a test and its preconditions.

At runtime, Tea invokes a solver that operates on the logical constraints it computes to produce a list of valid statistical tests to conduct. This process presents three key technical challenges: (1) incorporating statistical knowledge as constraints, (2) expressing user assumptions as constraints, and (3) recursively selecting statistical tests to verify preconditions of other statistical tests.

SMT Solver

As its constraint solver, Tea uses Z3 [6], a Satisfiability Modulo Theory (SMT) solver.

Satisfiability is the process of finding an assignment to variables that makes a logical formula true. For example, given the logical rules $0 < x < 100$ and $y < x$, $\{x = 1, y = 0\}$, $\{x = 10, y = 5\}$, and $\{x = 99, y = -100\}$ would all be valid assignments that satisfy the rules. SMT solvers determine the satisfiability of logical formulas, which can encode boolean, integer, real number, and uninterpreted function constraints over variables. SMT solvers can also be used to encode constraint systems, as we use them here. SMT solvers have been employed in a wide variety of applications ranging from the synthesis of novel interface designs [38], the verification of website accessibility [33], and the synthesis of data structures [26].

Logical Encodings

The first challenge of framing statistical test selection as a constraint satisfaction problem is defining a logical formulation of statistical knowledge.

Tea encodes the applicability of a statistical test based on its preconditions. A statistical test is applicable if and only if all of its preconditions (which are properties about variables) hold. We derived preconditions for tests from courses [22], statistics textbooks [10], and publicly available data science resources from universities [3, 25].

Tea represents each precondition for a statistical test an uninterpreted function representing a property over one or more variables. Each property is assigned `true` if the property holds for the variable/s; similarly, if the property does not hold, the property function is assigned `false`.

Tea also encodes statistical knowledge about variable types and properties that are essential to statistical analysis as axioms, such as the constraint that only a continuous variable can be normally distributed.

Algorithm

Tea frames the problem of finding a set of valid statistical tests as a maximum satisfiability (MaxSAT) problem that is seeded with user assumptions.

Tea translates each user assumption into an axiom about a property and variable. For each new statistical test Tea tries to satisfy, Tea verifies if any precondition of the test violates users' assumptions. If the test's preconditions do not violate users' assumptions, Tea checks to see if the precondition holds. For each precondition checked, Tea adds the property and variable checked as an axiom to observe as future tests are checked. The constraint solver then prunes the search space.

As a result, Tea does not compute all the properties for all variables, which represents a significant optimization when analyzing very large datasets.

At the end of this process, Tea finds a set of valid statistical tests to execute. If this set is empty, Tea defaults to its implementations of bootstrapping [7]. Otherwise, Tea proceeds and executes all valid statistical tests. Tea returns a table of results to users, applying multiple comparison corrections [16] and calculating effect sizes when appropriate.

Table 2: Statistical tests supported in Tea's Null Hypothesis Significance Testing module

Class of tests	Parametric	Non-parametric
Correlation	Pearson's r	Kendall's τ
	Pointbiserial	Spearman's ρ
Bivariate mean comparison	Student's t-test	Welch's
		Mann-Whitney U
		(a.k.a. Wilcoxon rank sum)
	Paired t-test	Wilcoxon signed rank
Multivariate mean comparison	F-test	Kruskal Wallis
	Repeated measures one way ANOVA	Friedman
	Two-way ANOVA	
	Factorial ANOVA	
Proportions: Chi Square, Fisher's Exact		
Others: Bootstrapping (with confidence intervals)		

Optimization: Recursive Queries

When Tea verifies a property holds for a variable, it often must invoke another statistical test. For example, to check that two groups have equal variance, Tea must execute Levene's test. The statistical test used for verification may then itself have a precondition, such as a minimum sample size.

Such recursive queries are inefficient for SMT solvers like Z3 to reason about. To eliminate recursion, Tea lifts some statistical tests to properties. For instance, Tea does not encode the Levene's test as a statistical test. Instead, Tea encodes the property of having equal variance between groups and executes the Levene's test for two groups when verifying that property for particular variables.

6 INITIAL EVALUATION

We assessed the benefits of Tea in two ways. First, we compared Tea's suggestions of statistical tests to suggestions in textbook tutorials. We use these tutorials as a proxy for expert test selection. Second, for each tutorial, we compared the analysis results of the test(s) suggested by Tea to those of the test suggested in the textbook as well as all other candidate tests. We use the set of all candidate tests as as a proxy for non-expert test selection.

We differentiate between candidate tests and valid tests. A candidate test can be computed on the data, when ignoring any preconditions regarding the data types or distributions. A valid test is a candidate test for which all preconditions are satisfied.

How does Tea compare to textbook tutorials?

Our goal was to assess how Tea's test selection compared to tests experts would recommend.

We sampled 12 data sets and examples from R tutorials ([18] and [10]). These included eight parametric tests, four non-parametric tests, and one Chi-square test. We chose these tutorials because they appeared in two of the top 20 statistical textbooks on Amazon and had publicly available data sets, which did not require extensive data wrangling.

For nine out of the 12 tutorials, Tea suggested the same statistical test (see Table 3). For three out of 12 tutorials, which used a parametric test, Tea suggested using a non-parametric alternative instead. The reason for Tea suggesting a non-parametric test was non-normality of the data. Tea's recommendation of using a non-parametric test instead of a parametric one did not change the statistical significance of the result at the .05 level.

For the two-way ANOVA tutorial from [10], which studied how gender and drug usage of individuals affected their perception of attractiveness, a precondition of the two-way ANOVA is that the dependent measure is normally distributed in each category. This precondition was violated. As a result, Tea defaulted to bootstrapping the means for each group and reported the means and confidence intervals. For the point-biserial correlation tutorial from [10], Tea also defaulted to bootstrap for two reasons. First, the precondition of normality is violated. Second, the data uses a dichotomous (nominal) variable, which renders both Spearman's ρ and Kendall's τ as invalid.

How does Tea compare to non-expert users?

Our goal was to assess whether any of the tests suggested by Tea (i.e., valid candidate tests) or any of the invalid candidate tests would lead to a different conclusion than the one drawn in the tutorial. Table 3 shows the results. Specifically, highlighted p-values indicate instances for which the

Table 3: Results of applying Tea to 12 textbook tutorials.
Tea can prevent false positive and false negative results by suggesting only tests that satisfy all assumptions. *Tutorial* gives the test described in the textbook; *Candidate tests (p-value)* gives all tests a user could run on the provided data with corresponding p-values; *Assumptions* gives all satisfied and violated assumptions; *Tea suggests* indicates which tests Tea suggests based on their assumptions. Highlighted p-values indicate instances where a candidate test leads to a wrong conclusion about statistical significance.

Tutorial	Candidate tests (p-value)		Assumptions*	Tea suggests
Pearson	Pearson's r	(6.96925e-06)	② ④ ⑤	–
[18]	Kendall's τ	(2.04198e-05)	② ④	✓
	Spearman's ρ	(2.83575e-05)	② ④	✓
Spearman's ρ	Spearman's ρ	(.00172)	② ④	✓
[10]	Pearson's r	(.01115)	② ④	–
	Kendall's τ	(.00126)	② ④	✓
Kendall's τ	Kendall's τ	(.00126)	② ④	✓
[10]	Pearson's r	(.01115)	② ④	–
	Spearman's ρ	(.00172)	② ④	✓
Pointbiserial	Pointbiserial (Pearson's r)	(.00287)	② ④ ⑤	–
[10]	Spearman's ρ	(.00477)	② ④	–
	Kendall's τ	(.00574)	② ④	–
	Bootstrap	(<0.05)		✓
Student's t-test	Student's t-test	(.00012)	② ④ ⑤ ⑥ ⑦ ⑧	✓
[18]	Mann-Whitney U	(9.27319e-05)	② ④ ⑦ ⑧	✓
	Welch's t-test	(.00065)	② ④ ⑤ ⑦ ⑧	✓
Paired t-test	Paired t-test	(.03098)	② ④ ⑤ ⑦ ⑧	✓
[10]	Student's t-test	(**.10684**)	② ④ ⑤ ⑦	–
	Mann-Whitney U	(**.06861**)	② ④ ⑦	–
	Wilcoxon signed rank	(.04586)	② ④ ⑦ ⑧	✓
	Welch's t-test	(**.10724**)	② ⑦	–
Wilcoxon signed rank	Wilcoxon signed rank	(.04657)	② ④ ⑦ ⑧	✓
[10]	Student's t-test	(.02690)	② ④ ⑦	–
	Paired t-test	(.01488)	② ④ ⑤ ⑦ ⑧	–
	Mann-Whitney U	(.00560)	② ④ ⑦	–
	Welch's t-test	(.03572)	② ④ ⑦	–
F-test	F-test	(9.81852e-13)	② ④ ⑤ ⑥ ⑨	✓
[10]	Kruskal Wallis	(2.23813e-07)	② ④ ⑨	✓
	Friedman	(8.66714e-07)	② ⑦	–
	Factorial ANOVA	(9.81852e-13)	② ④ ⑤ ⑥ ⑨	✓
Kruskal Wallis	Kruskal Wallis	(.03419)	② ④ ⑨	✓
[10]	F-test	(**.05578**)	② ④ ⑤ ⑨	–
	Friedman	(3.02610e-08)	② ⑦	–
	Factorial ANOVA	(**.05578**)	② ④ ⑤ ⑨	–
Repeated measures one way ANOVA	Repeated measures one way ANOVA (.0000)		② ④ ⑤ ⑥ ⑦ ⑨	✓
[10]	Kruskal Wallis	(4.51825e-06)	② ④ ⑦ ⑨	–
	F-test	(1.24278e-07)	② ④ ⑤ ⑥ ⑦ ⑨	–
	Friedman	(5.23589e-11)	② ④ ⑦ ⑨	✓
	Factorial ANOVA	(1.24278e-07)	② ④ ⑤ ⑥ ⑨	✓
Two-way ANOVA	Two-way ANOVA	(3.70282e-17)	② ④ ⑤ ⑨	–
[10]	Bootstrap	(<0.05)		✓
Chi Square	Chi Square	(4.76743e-07)	② ④ ⑨	✓
[10]	Fisher's Exact	(4.76743e-07)	② ④ ⑨	✓

*① one variable, ② two variables, ③ two or more variables, ④ continuous vs. categorical vs. ordinal data, ⑤ normality, ⑥ equal variance, ⑦ dependent vs. independent observations, ⑧ exactly two groups, ⑨ two or more groups

result of a test differs from the tutorial in terms of statistical significance at the .05 level.

For all of the 12 tutorials, Tea's suggested tests led to the same conclusion about statistical significance. For two out of the 12 tutorials, two or more candidate tests led to a different conclusion. These candidate tests were invalid due to violations of independence or normality.

7 LIMITATIONS AND FUTURE WORK

The goal of this paper was to design and assess Tea's high-level DSL and constraint-based runtime system. Here, we identify limitations of the current work that suggest opportunities for future work.

Empirical evaluation of usability. While we believe that abstracting away statistical tests—thus obviating the need for detailed statistical knowledge—will make Tea substantially easier to use than conventional statistical tools, an empirical evaluation with non-statistical expert users will be required to establish this. A study comparing its use with conventional statistical analysis tools such as SPSS or R would be of particular interest.

Relaxing Tea's conservatism. Tea is conservative in its test selection because Tea's runtime system will execute a statistical test only when all the preconditions are met. In practice, some preconditions may be more important than others. For instance, Tea could allow some degree of deviation from absolute normality. Further evaluation with statistical and domain experts could help refine Tea's decision making procedure.

Expanding beyond NHST. Tea's architecture is designed to be flexible and support extension. Currently, Tea provides a module for Null Hypothesis Significance Testing because NHST is the most common paradigm in HCI. As statistics norms change, it will be important for Tea to support a broader range of analyses, including regression and Bayesian inference.

Extending Tea's architecture and language to Bayesian inference presents several key research challenges: (1) easing the process of choosing and expressing priors, (2) easing the process of choosing and expressing models, and (3) suggesting appropriate statistical tests. A variety of probabilistic programming languages emphasize language abstractions that let programmers succinctly express priors and models— BUGS [27], BLOG [30], Stan [5], Church [12], and Figaro [34] are a few prominent examples. Some existing work suggests appropriate statistical tests for a researcher's goals [23, 24, 29], but these suggestions are generally not embodied in a tool, language, or programming environment; we look forward to developing ways to encode these into Tea.

8 DISCUSSION

This paper introduces Tea, a high-level programming language that supports users in formalizing and automating statistical analysis.

Towards Task-Appropriate Analyses. Our evaluation shows that Tea's constraint-based system to find suitable statistical tests generally matches the choices of experts. In particular, it automatically switches to non-parametric tests when parametric assumptions are not met. When assumptions are not met, Tea will always default to tests with fewer assumptions, all the way to the bootstrap [7]. Tea prevents conducting statistical analyses that rely on unfounded assumptions. Given Tea's automated test selection and assumption checking, analyses are more likely to be sound than is currently the case [4].

Towards Reproducible Analyses. Researchers have suggested automation as an opportunity to increase the transparency and reproducibility of scientific experiments and findings [35]. Tea programs are relatively straightforward to write and read and therefore could serve as a way for researchers to share their analysis for others to reproduce and to extend. While almost all previous tools place the burden on users to select suitable statistical tests and check their assumptions, most users conducting data analysis are not statistical experts.

Towards Trustworthy Analyses. Pre-registration holds the promise of promoting trustworthy analyses—e.g., by eliminating HARKing, p-hacking, and cherry picking— but progress towards mainstream pre-registration has stalled without a standard format for expressing study design, hypotheses, and researcher assumptions. Since Tea programs express variables of interest, study design, assumptions, and hypotheses, Tea constitutes a potential standard format for pre-registering studies and hypotheses.

Fine-Tuning the Division of Labor. Tea provides what Heer refers to as "shared representations," representations that support both human agency and system automation [14] in statistical analysis. Users are in ultimate control with Tea. Tea's language empowers users to represent their knowledge and intent in conducting analyses (i.e., to test a hypothesis). Users convey their experimental designs, assumptions, and hypotheses, the high-level goals and domain knowledge that only the user can provide. Tea takes on the laborious and error-prone task of searching the space of all possible statistical tests to evaluate a user-defined hypothesis. Thus, Tea plays a complementary role to users in their efforts to conduct valid statistical analyses.

9 RELATED WORK

Tea extends prior work on domain-specific languages for the data lifecycle, tools for statistical analysis, and constraint-based approaches in HCI.

Domain-specific Languages for the Data Lifecycle

Prior domain-specific languages (DSLs) has focused on several different stages of data exploration, experiment design, and data cleaning to shift the burden of accurate processing from users to systems. To support data exploration, Vega-lite [36] is a high-level declarative language that supports users in developing interactive data visualizations without writing functional reactive components. PlanOut [2] is a DSL for expressing and coordinating online field experiments. More niche than PlanOut, Touchstone2 provides the Touchstone Language for specifying condition randomization in experiments (e.g., Latin Squares) [9].To support rapid data cleaning, Wrangler [19] combines a mixed-initiative interface with a declarative transformation language. Tea can be integrated with tools such as Wrangler that produce cleaned CSV files ready for analysis.

In comparison to these previous DSLs, Tea provides a language to support another crucial step in the data lifecycle: statistical analysis.

Tools for Statistical Analysis

Research has also introduced tools support statistical analysis in diverse domains. ExperiScope [13] supports users in analyzing complex data logs for interaction techniques. ExperiScope surfaces patterns in the data that would be difficult to detect manually and enables researchers to collect noisier data in the wild that have greater external validity. Touchstone [28] is a comprehensive tool that supports the design and launch of online experiments. Touchstone provides suggestions for data analysis based on experimental design. Touchstone2 [9] builds upon Touchstone and provides more extensive guidance for evaluating the impact of experimental design on statistical power. Statsplorer [41] is an educational web application for novices learning about statistics. While more focused on visualizing various alternatives for statistical tests, Statsplorer also automates test selection (for a limited number of statistical tests and by executing simple switch statements) and the checking of assumptions (though it is currently limited to tests of normality and equal variance). Wacharamanotham et al. [41] found that Statsplorer helps HCI students perform better in a subsequent statistics lecture.

In comparison to Statsplorer, Tea is specifically designed to integrate into existing workflows (e.g., it can be executed in any Python notebook). It enables reproducing and extending analyses by being script-based, and the analyses are focused on hypotheses that analysts specify.

Constraint-based Systems in HCI

Languages provide semantic structure and meaning that can be reasoned about automatically. For domains with well defined goals, constraint solvers can be a promising technique. Some of the previous constraint-based systems in HCI have been Draco [31] and SetCoLa [15], which formalize visualization constraints for graphs. Whereas SetCoLa is specifically focused on graph layout, Draco formalizes visualization best practices as logical constraints to synthesize new visualizations. With additional logical constraints, the knowledge base can grow, supporting the continual evolution of design recommendations.

Another constraint-based system is Scout [38], a mixed-initiative system that supports interface designers in rapid prototyping. Designers specify high-level constraints based on design concepts (e.g., a profile picture should be more emphasized than the name), and Scout synthesizes novel interfaces. Scout also uses Z3's theories of booleans and integer linear arithmetic.

We extend this prior work by providing the first constraint-based system for statistical analysis.

10 CONCLUSION

Tea is a high-level domain-specific language and runtime system that automates statistical test selection and execution. Tea achieves these by applying techniques and ideas from human-computer interaction, programming languages, and software engineering to statistical analysis. Our hope is that Tea opens up possibilities for new tools for statistical analysis, helps researchers in diverse empirical fields, and resolves a century-old question: "Which test should I use to test my hypothesis?"

11 USING TEA

Tea is an open-source Python package that users can download using Pip, a Python package manager. Tea can be used in iPython notebooks. The source code can be accessed at http://tea-lang.org.

REFERENCES

[1] American Psychological Association. 1996. Task Force on Statistical Inference. https://www.apa.org/science/leadership/bsa/statistical/

[2] Eytan Bakshy, Dean Eckles, and Michael S Bernstein. 2014. Designing and deploying online field experiments. In *Proceedings of the 23rd international conference on World wide web*. ACM, 283–292.

[3] J. Bruin. 2019. Choosing the Correct Statistical Test in SAS, Stata, SPSS and R. https://stats.idre.ucla.edu/other/mult-pkg/whatstat/

[4] Paul Cairns. 2007. HCI... not as it should be: inferential statistics in HCI research. In *Proceedings of the 21st British HCI Group Annual Conference*

on People and Computers: HCI... but not as we know it-Volume 1. British Computer Society, 195–201.

[5] Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. Stan : A Probabilistic Programming Language. *Journal of Statistical Software* 76 (01 2017). https://doi.org/10.18637/jss.v076.i01

[6] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 337–340.

[7] Bradley Efron. 1992. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*. Springer, 569–593.

[8] Isaac Ehrlich. 1973. Participation in illegitimate activities: A theoretical and empirical investigation. *Journal of political Economy* 81, 3 (1973), 521–565.

[9] Alexander Eiselmayer, Chatchavan Wacharamanotham, Michel Beaudouin-Lafon, and Wendy Mackay. 2019. Touchstone2: An Interactive Environment for Exploring Trade-offs in HCI Experiment Design. (2019).

[10] Andy Field, Jeremy Miles, and Zoë Field. 2012. *Discovering statistics using R*. Sage publications.

[11] Ronald Aylmer Fisher. 1937. *The design of experiments*. Oliver And Boyd; Edinburgh; London.

[12] N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. 2008. Church: a language for generative models. *Uncertainty in Artificial Intelligence*.

[13] François Guimbretière, Morgan Dixon, and Ken Hinckley. 2007. ExperiScope: an analysis tool for interaction data. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1333–1342.

[14] Jeffrey Heer. 2019. Agency plus automation: Designing artificial intelligence into interactive systems. *Proceedings of the National Academy of Sciences* 116, 6 (2019), 1844–1850.

[15] Jane Hoffswell, Alan Borning, and Jeffrey Heer. 2018. SetCoLa: High-Level Constraints for Graph Layout. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 537–548.

[16] Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* (1979), 65–70.

[17] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001–2019. SciPy: Open source scientific tools for Python. http://www.scipy.org/

[18] Robert I Kabacoff. 2011. R: In Action. (2011).

[19] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3363–3372.

[20] Maurits Kaptein and Judy Robertson. 2012. Rethinking statistical analysis methods for CHI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1105–1114.

[21] Matthew Kay, Gregory L Nelson, and Eric B Hekler. 2016. Researcher-centered design of statistics: Why Bayesian statistics better fit the culture and incentives of HCI. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 4521–4532.

[22] Scott Klemmer and Jacob Wobbrock. 2019. Designing, Running, and Analyzing Experiments. https://www.coursera.org/learn/designexperiments

[23] John K. Kruschke. 2010. *Doing Bayesian Data Analysis: A Tutorial with R and BUGS* (1st ed.). Academic Press, Inc., Orlando, FL, USA.

[24] John K. Kruschke and Torrin M. Liddell. 2018. The Bayesian New Statistics: Hypothesis testing, estimation, meta-analysis, and power analysis from a Bayesian perspective. *Psychonomic Bulletin & Review* 25, 1 (01 Feb 2018), 178–206. https://doi.org/10.3758/s13423-016-1221-4

[25] Kent State University Libraries. 2019. SPSS Tutorials: Analyzing Data. https://libguides.library.kent.edu/SPSS/AnalyzeData

[26] Calvin Loncaric, Emina Torlak, and Michael D Ernst. 2016. Fast synthesis of fast collections. *ACM SIGPLAN Notices* 51, 6 (2016), 355–368.

[27] David J. Lunn, Andrew Thomas, Nicky Best, and David Spiegelhalter. 2000. WinBUGS - A Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing* 10, 4 (01 Oct 2000), 325–337. https://doi.org/10.1023/A:1008929526011

[28] Wendy E Mackay, Caroline Appert, Michel Beaudouin-Lafon, Olivier Chapuis, Yangzhou Du, Jean-Daniel Fekete, and Yves Guiard. 2007. Touchstone: exploratory design of experiments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1425–1434.

[29] Michael E. J. Masson. 2011. A tutorial on a practical Bayesian alternative to null-hypothesis significance testing. *Behavior Research Methods* 43, 3 (Sept. 2011), 679–690. https://doi.org/10.3758/s13428-010-0049-5

[30] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. 2005. BLOG: Probabilistic Models with Unknown Objects. In *Proc. 19th International Joint Conference on Artificial Intelligence*. 1352–1359. http://sites.google.com/site/bmilch/papers/blog-ijcai05.pdf

[31] Dominik Moritz, Chenglong Wang, Greg L Nelson, Halden Lin, Adam M Smith, Bill Howe, and Jeffrey Heer. 2019. Formalizing visualization design knowledge as constraints: Actionable and extensible models in Draco. *IEEE transactions on visualization and computer graphics* 25, 1 (2019), 438–448.

[32] Travis E Oliphant. 2006. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA.

[33] Pavel Panchekha, Adam T Geller, Michael D Ernst, Zachary Tatlock, and Shoaib Kamil. 2018. Verifying that web pages have accessible layout. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 1–14.

[34] Avi Pfeffer. 2011. Practical Probabilistic Programming. In *Inductive Logic Programming*, Paolo Frasconi and Francesca A. Lisi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 2–3.

[35] Alex Reinhart. 2015. *Statistics done wrong: The woefully complete guide*. No starch press.

[36] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics* 23, 1 (2017), 341–350.

[37] Skipper Seabold and Josef Perktold. 2010. Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference*, Vol. 57. Scipy, 61.

[38] Amanda Swearngin, Andrew J Ko, and James Fogarty. 2018. Scout: Mixed-Initiative Exploration of Design Variations through High-Level Design Constraints. In *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings*. ACM, 134–136.

[39] Walter Vandaele. 1987. *Participation in illegitimate activities: Ehrlich revisited, 1960*. Vol. 8677. Inter-university Consortium for Political and Social Research.

[40] William N Venables and Brian D Ripley. 2013. *Modern applied statistics with S-PLUS*. Springer Science & Business Media.

[41] Chat Wacharamanotham, Krishna Subramanian, Sarah Theres Volkel, and Jan Borchers. 2015. Statsplorer: Guiding novices in statistical analysis. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2693–2702.

[42] Hadley Wickham et al. 2014. Tidy data. *Journal of Statistical Software* 59, 10 (2014), 1–23.

[43] Wikipedia contributors. 2019. JMP (statistical software) — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=

JMP_(statistical_software)&oldid=887217350. [Online; accessed 5-April-2019].

[44] Wikipedia contributors. 2019. R (programming language) — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=R_(programming_language)&oldid=890657071. [Online; accessed 5-April-2019].

[45] Wikipedia contributors. 2019. SAS (software) — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=SAS_(software)&oldid=890451452. [Online; accessed 5-April-2019].

[46] Wikipedia contributors. 2019. SPSS — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=SPSS&oldid=888470477. [Online; accessed 5-April-2019].

[47] Leland Wilkinson. 1999. Statistical methods in psychology journals: Guidelines and explanations. *American psychologist* 54, 8 (1999), 594.