The Fault in our Stats

Alexi Turcotte
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
alexi.turcotte@cispa.de

Neev Nirav Mehta
Saarland University
Saarbrücken, Germany
neme00002@stud.uni-saarland.de

***Abstract*—Data analysts need to be careful when they apply statistical inference techniques to data, as misuse of statistical inference methods can lead an analyst to draw the wrong conclusions. They need to be careful because, in the general case, misuse of statistics does not result in obvious problems; the numbers returned often look reasonable, and programs with misuses of statistics do not crash. In this work, we propose a technique to quickly and statically check data science programs for compliance with statistics best practice rules, including *checking* all assumptions made by statistical methods, as well as *correcting* for the multiple comparison problem, or "data dredging". This technique is predicated on a novel *statistics intermediate representation*, called SIR, that encodes the details most salient to statistics. We implement this technique in a tool called STAT-LINT, the first statistics linter, and evaluate STAT-LINT on 90 Python data science notebooks, finding that only 14 fully check all obligations, only two apply any correction for multiple comparisons, none validate model residuals, and over two thirds of obligations go unchecked.**

***Index Terms*—statistics, Python, static analysis, data science**

I. Introduction

Data scientists, or data analysts, often use *statistics* to analyze and answer questions about data, and do so using statistical libraries in languages like Python, R, and Julia. So-called "literate programming" environments, i.e., *notebooks*, interleave code blocks with text discussing data and figures and are a popular method for creating and sharing data analysis. As of April 2025, there are over 1.3 million public notebooks and 23 million users on the Kaggle website, a popular place to store notebooks in a reproducible environment that includes any datasets used by the notebook.

Notebooks are not without traditional bugs [10], [28]. Also, there are myriad ways to misuse statistics [13], [32], [23], [9]. Recent work has identified several bugs related to misuses of statistics in notebooks [55], specifically related to the misuse of *parametric statistical methods*, i.e., statistical methods that make assumptions about the data they analyze. The thing is, if a parametric method is used even when an assumption does not appear to be met, *programs do not crash, and reasonable looking results are still produced.* For example, consider Student's t-test, which is used to find if there is a statistically significant difference between two samples. This test makes three assumptions about the data: the sample mean should be normally distributed, the variance of the populations the samples are drawn from should be equal, and the samples should be independent. If any of these assumptions are false for the samples, Student's t-test still yields a result that looks reasonable. This is, in part, a good thing, because the assumptions underlying correct use of statistical methods often cannot be checked with complete certainty, and moreover some statistical methods are "robust" in that they still work well if assumptions they make do not appear to be met.

Now, analysts are not without recourse, and an important aspect of statistical best practice is to discuss and investigate the assumed properties of data [45], [49], [47], [35], [33], [37]. There are also many *non-parametric* statistical methods that *do not make assumptions* about data, but are relatively less well known than ubiquitous methods like Student's t-test, analysis of variance, and linear regression. If an analyst wants to check if they are using statistical methods correctly, they can visualize data, run automated tests to ascertain the degree to which data adheres to assumed properties, or assume a property based on some prior knowledge. Recently, Turcotte and Wu [55] proposed a technique whereby developers of statistical libraries can annotate functions with statistical assumptions, and have those assumptions translated into automated tests that run and check data for compliance. To check data, calls to assumption testing methods (e.g., the Shapiro-Wilk test for normality) are injected into library code, but these tests can be very sensitive to departures from assumed properties [25], [24], [1], [14]. This can result in many spurious warnings for analysts.

In this work, we propose a set of statistics best practice rules that should be obeyed in a program using statistical methods. For best practice as it relates to parametric statistical methods, the rules require that any *obligations* imposed by statistical methods are *checked* by the analyst; e.g., the assumption of normality imposes an obligation to check data for normality. Checking can be done either by *testing* data with the appropriate assumption testing method or by *visualizing* data with an appropriate data visualization method. We also include a rule to prevent the multiple comparison problem ("data dredging"), where spurious inferences can be drawn from repeatedly testing the same data. And finally, we include rules related to statistical modeling, in particular linear regression, namely that the *residuals* of a model must be checked for normality and homoscedasticity.

To implement these rules, we developed a *statistics intermediate representation* called SIR that encodes the details most salient to statistics: data flow, statistical tests, statistical models, and plots. We then developed a tool, STAT-LINT, which compiles Python data analysis programs into SIR, and verifies the degree to which all statistics best practice rules were

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00205

obeyed. Our approach also includes a *quality score* indicating how well a check fulfills an obligation. We evaluated STAT-LINT on **90** notebooks from the Kaggle website, and found that only **14** notebooks fully check all obligations related to correct use of parametric statistics, only **2** notebooks correct for multiple comparisons, and *none* of the notebooks we studied investigated the residuals of fitted linear regression models. Some notebooks contained a *significant statistical misuse*; i.e., a case where correcting the improper use of a statistical method (such as using a non-parametric test in place of a parametric test) would result in a *different* conclusion being drawn. We used STAT-LINT to investigate if any checks were made in such cases, and found complete checks in only **8** cases (of **23** total). Finally, when we repaired the notebooks that did not correct for multiple comparisons, we found **17** additional significant misuses related to the multiple comparisons.

In summary, the contributions of this paper are:

- a statistics IR, SIR, encoding the details of a program most salient to statistics;
- a set of statistics best practice rules for validating use of statistical methods;
- a tool, STAT-LINT, which analyzes data science code for compliance with the aforementioned best practice rules (available in an artifact[1]);
- an evaluation of STAT-LINT on **90** Kaggle notebooks, revealing novel statistical misuses and shedding light on how analysts test their data.

The remainder of the paper is organized as follows: Section II presents statistics background, Section III presents examples of misuses of statistics in notebooks, then Section IV presents our approach, Section V describes our evaluation of this approach, Section VI presents threats to validity and limitations, Section VII sketches related software engineering and statistics research, and finally Section VIII concludes.

II. Background

A. Statistics Background

Statistics is a discipline about inferring characteristics of a *population* from *samples* of that population. An effective mathematical tool to characterize populations is the *probabilistic distribution*, like the normal distribution, which describes how samples drawn from a population will look (most of the time). Distributions also encode characteristics of the population, like the mean and variance. The normal distribution is particularly important, and many statistical methods are designed in anticipation of data being normally distributed.

Statistical Inference: Hypothesis tests are statistical methods that answer questions about populations from samples. E.g., *given two samples from two different populations, can we conclude that the two populations have the same mean?* Examples include Student's t-test and the analysis of variance (ANOVA). Technically, these hypothesis tests pose a null hypothesis (e.g., the means are equal), and a calculation is performed to determine to what degree the data supports the hypothesis; the null hypothesis can then fail to be rejected, or rejected in favor of an alternative hypothesis.

Parametric and Non-Parametric Statistics: Statistical methods like hypothesis tests are said to be *parametric* if they assume characteristics about the samples and populations being analyzed. The aforementioned Student's t-test and ANOVA are parametric methods that assume, among other properties, that the sample mean is normally distributed. There also exist a plethora of *non-parametric* statistical methods which make no assumptions, and many parametric methods have non-parametric "equivalents" that essentially answer the same question about analyzed data. For instance, the Mann-Whitney U test is a non-parametric equivalent of Student's t-test for two independent samples.

Statistical Errors: In the context of statistical inference, there are two important types of errors. A *Type 1* error occurs when a true null hypothesis is falsely rejected by a test; this is a false positive in statistics. A *Type 2* error occurs when a false null hypothesis is not rejected; this is a false negative in statistics. To control Type 1 errors, analysts choose a confidence level α when conducting inference, and α actually is the Type 1 error rate. They can then conduct a test and compare the p-value with α, and if the p-value falls below the "threshold" α then the null hypothesis is rejected. To control the Type 2 error rate (also called β), analysts should choose the most *powerful* method given the characteristics of their data. Statistical power is a measure of how well a method detects departures from the null hypothesis, and generally parametric methods are more powerful than non-parametric methods.

Investigating Assumptions: When an analyst wants to perform some parametric statistics, they (should) first establish if the assumptions are appropriate given the data. To do so, analysts can assume the property based on some prior knowledge, or check the assumption by (1) performing other statistical tests, or (2) visualizing the data and confirm.

For (1), the idea is to pose as hypothesis that the assumption is met, and perform a test to see the degree to which the data supports the hypothesis. E.g., the Shapiro-Wilk [48] test poses as the null hypothesis that the population is distributed normally, and the sample is analyzed to see how likely it is that it was drawn from a normally distributed population. There are many such tests, such as the Anderson-Darling distribution test [2], Levene's [26] and Bartlett's [5] tests of homoscedasticity, and Pearson's [36] correlation coefficient.

For (2), some assumptions can be gleaned from visual inspection of the data. For instance, one can plot a histogram of a sample to see how closely it matches the normal distribution. Moreover, one can plot a quantile-quantile (Q-Q) plot which plots the quantiles of two sets of data, which is also used to see if data follows a distribution. These plots also reveal details about variance, and can reveal relationships between variables. That being said, visual inspection is a subjective process.

Statistical Modeling: Analysts can also fit models to data to try to ascertain trends, understand the relationship between data, and predict values. *Linear regression* is such a modeling method that attempts to fit a linear model to the data. To check

[1]See https://zenodo.org/records/15560705.

the quality of the linear model, an analyst should examine the *residuals* of the model [6], [7], [39], i.e., the difference between the fitted model and the actual observed data.

B. Misuses of Statistical Methods

When assumptions are not met, statistical tests still yield reasonable looking results. For instance, if two small samples that do not appear to be normally distributed are compared with Student's t-test, it will still yield a number. This makes it especially difficult to detect misuses of statistics. To mitigate the risk of misuse, analysts should be aware of and investigate assumptions in order to choose the most appropriate method.

Another important statistical misuse is the *multiple comparison problem*, known also as *data dredging*. In statistics, due to the random nature of the sampling process, there is always a chance that a sample does not represent the entire population. Statistical methods are designed to control and account for this, but this is an inherently random process, and so errors are possible. When multiple simultaneous statistical inferences are performed on the same data set, the chances that spurious differences are detected increases. E.g., when we run a t-test at 95% confidence ($\alpha = 0.05$), there is a 5% chance of a Type I error, or statistical false positive; so if we run 20 tests on the same data, chances are that we find one spurious difference.

The primary way to mitigate this risk is to perform *multiple test correction*, i.e., increasing the threshold required to draw inference from data. The most conservative approach is to apply a *Bonferroni Correction* [12]; if an analyst wants a total Type-1 error rate of α, every test should be conducted with α/n where n is the total number of tests being performed. There are other similar approaches [3], [18], [17].

C. Glossary

In their paper, Turcotte and Wu [55] propose a glossary of terms to connect the nomenclatures of statistics and software engineering. The essential terms to this work are:

- **statistical methods** are hypothesis testing or other methods from statistics, and **statistical functions** are functions implementing such methods;
- **statistical assumptions** are assumptions made by statistical methods that cannot be checked with total certainty;
- a **potential statistical misuse**, or potential misuse, occurs when a statistical method is used when there is demonstrated evidence against its assumptions being met;
- a **significant statistical misuse**, or significant misuse, is a potential misuse where an equivalent statistical method that makes fewer assumptions would yield a different result if run on the same data.

We propose to extend the definition of statistical misuses to include situations where multiple tests were run on the same data and a correction (e.g., Bonferroni correction) was not applied; these *potential* misuses become *significant* if the corrected confidence level would result in a different conclusion being drawn. Additionally, we propose:

- statistical assumptions can be **tested**, **visualized**, **assumed**, or **unchecked**;
- a **checked** statistical assumption (either tested, visualized, or assumed) can be **improperly** checked if a super- or subset of the data subject to the assumption was checked, or **properly** checked otherwise.

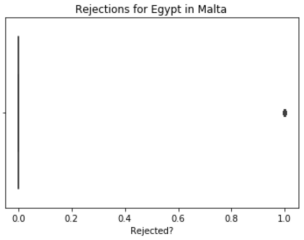


Fig. 1: Box plot of visa application rejections for Egyptians applying for a Maltese visa (see [30]). This hardly looks like a box plot, and the data is very clearly *not* normally distributed.

III. Motivation

Part of the challenge of sound and correct data analysis is that current programming languages, environments, and tools do not give analysts any feedback as to potential misuses of statistical methods.

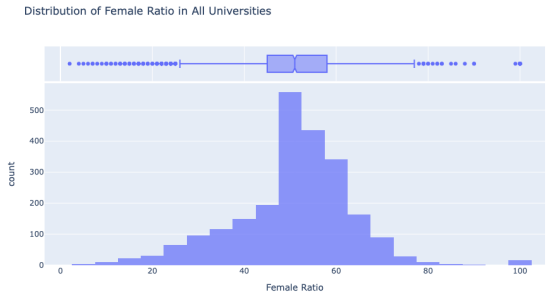
A. Misuses of Parametric Statistical Methods

For example, let us consider a Kaggle notebook that investigates which embassies give out visas at a higher rate (*"The Visa Shopper Guide"* [30]). The dataset contains information regarding Schengen State consulates (e.g., which country and city they are located in), the kinds of visas the consulates received applications for as well as how many, and how many visas were issued or not issued. There are 19 columns in total.

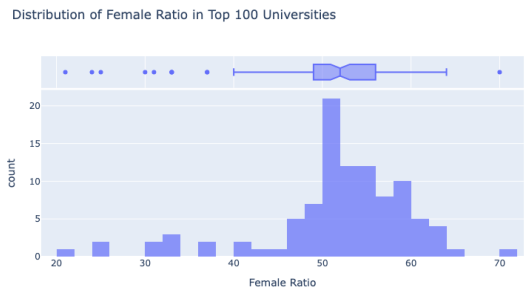
The analyst chooses to focus on Egypt, and plots pie and bar charts to visualize overall how many applications were received from Egypt for each Schengen state. Then, they split the data into the 15 most and least generous consulates w.r.t. the visa rejection rate. After printing out the data frames and looking at the raw data, they pose and investigate a question: *Could these rejection rates be due to random chance?*

To answer this, they use hypothesis testing methods. First, they assume that each consulate is receiving applications from the same "pool" or *population* of applicants. Then, they loop over the split data and conduct a univariate t-test to compare the average rejection rate of each of the most and least generous consulates with the total mean rejection rate to determine if there is a statistically significant difference between each consulate and the global mean, using a strict significance level of 0.001 for these tests.

The univariate t-test assumes that the sample mean is normally distributed, which would be true if the population being sampled was itself normally distributed. To investigate this assumption, we inserted tests for normality using the Shapiro-Wilk test and Anderson-Darling test, as well as generated box plots of the data subject to the t-test. In each of the most and least generous cases, both the Shapiro-Wilk and Anderson-Darling tests revealed significant departures from normality



(a) Female ratio across all universities. Note the symmetric, normal-looking distribution.



(b) Female ratio across top 100 universities. Note the skewed distribution, with outliers especially on the left.

Fig. 2: Plots from notebook investigating the male and female ratios at global universities. The analyst created the plot in (a), but performed a parametric test assuming normality with the data in (b). In this case, we found a significant statistical misuse.

(very high significance levels, e.g., the Shapiro-Wilk p-values were all near 0 indicating high confidence). Moreover, the box plots painted a very clear picture of non-normal data (see Fig. 1 for an example). In the pictured case and other similar cases in this notebook, the analyst concluded that there *was not* a significant difference between the rejection rate of these consulates and the global mean; had they used a more appropriate method (e.g., a Wilcoxon signed-rank test), they would actually have found the opposite, namely that these consulates do differ significantly from the global mean.

B. Imperfect Checks

In some cases, an analyst will plot or test their data, but they can miss the mark, as is the case in the notebook titled "UniRank 2023 Insights" [61] investigating the ratio of males to females in universities worldwide. The notebook contains many plots which clearly show data distributions.

For example, the analyst created the plot in Fig. 2a showing the distribution of the ratio of females to males across all universities. However, the analyst *does not test with exactly this data*. Instead, they run a t-test on the female ratio *only at the top 100 universities*, but there is no way to tell from the plot which subset of the data this constitutes. We generated the plot in Fig. 2b to show that the distribution of the data does not appear to be completely normal. In this case, the analyst performed a t-test to see if there was a significant difference in the gender ratios at the top 100 universities, and the t-test found *no* significant difference. We performed a more appropriate Mann-Whitney U test instead, and concluded that there *was in fact* a significant difference in ratios.

C. The Multiple Comparisons Problem ("Data Dredging")

Let's say an analyst was diligent about testing or plotting to test assumptions, as was the case in the "PASSNYC: The Magic of Data Science" notebook [19] investigating the impact of PASSNYC outreach services (these services are designed to improve the chances that New York City minority or underserved middle school students score well on a standardized advanced high school placement exam called the SHSAT). The dataset is split into two groups: one group is middle schools that have attached high schools where at least one student took the SHSAT, and the other group is schools where the highest grade is middle school but no students took the SHSAT.

At face value, this is quite a good notebook, as the analyst generates plots of almost all the data subject to test. However, they do run *a lot* of tests; the data set is subject to 12 different tests, such as:

- "is the % of Asian students different in the two groups?"
- "is the % of White students different in the two groups?"
- "is there a difference in the rigor of instruction between the two groups?"

The recommended process to reduce risk of multiple comparison problems is to choose stricter confidence thresholds. To see the effect of this, we introduced a Bonferroni correction in this notebook, where we divided the significance level chosen by the analyst ($\alpha = 0.05$) by the total number of tests (12). This actually reveals a false positive: with correction applied, we cannot conclude that the % of Black students differs in a statistically significant way between groups.

IV. Approach

In this paper, we present an approach to automatically identify situations where statistics best practice was not followed. To achieve this, our approach compiles notebooks into an intermediate representation (IR) that encodes only the details salient to statistics. We encode best practice rules based on this IR, and the compiled notebooks are checked to see the degree to which they adhere to these rules.

A. Compile to Intermediate Representation

The first stage of the approach takes a program performing some statistical data analysis (e.g., a Python Jupyter notebook) and compiles the program into our *Statistics Intermediate Representation* (SIR). The cornerstones of this intermediate language are the *structure of data*, any uses of *statistical methods*, as well as any calls to *data visualization methods*.

1) Recording Data Structure: Data frames are the essential data structure for data analysis, and in SIR we represent data frames as an object which records:

- the identifier for the base data frame;
- the set of columns referenced;
- any filters or slices;
- any group-by clauses and aggregator functions.

For example, the Python expression:

```
df[df['age'] > 18]['salary'][50:150]
```

would be translated in SIR as:

$$\{df, cols : [\text{`}salary\text{'}], filters : [\text{`}age\text{'} > 18], slice : (50, 150)\}$$

If data analysts explicitly *sample* a data frame (e.g., with `df.sample()`), we record "sample" as the slice.

2) Statistical Methods: We also record calls to statistical functions. These take the form:

$$test\ D\ N_T$$

where D is the data being tested (there can be one or more) and N_T specifies the name of the statistical method. Recall: a statistical function implements a statistical method, but it's the methods themselves that impose the assumptions.

For example, the following line of Python:

```
ttest_ind(df[df['Color'] == 'Blue'],
  df[df['Color'] == 'Red'],
  equal_var=False)
```

will be translated in SIR as:

$$\begin{aligned} test\ \{df, filters : [\text{`}Color\text{'} == \text{`}Blue\text{'}]\}, \\ \{df, filters : [\text{`}Color\text{'} == \text{`}Red\text{'}]\}, \\ \textit{Welch's t-test} \end{aligned}$$

In this case, the statistical function `ttest_ind` implements Welch's t-test if the `equal_var` parameter is false. If the result of a statistical function is saved in a variable, the identifier is translated as a special *test result* identifier which we track to see if they are involved in multiple comparisons.

3) Statistical Modeling Methods: Methods building statistical models are compiled as well. The shape of these is:

$$model\ P\ D\ N_M$$

where D is the data the model is built from, P are the parameters of the model, and N_M is the name of the statistical model that was fit. For example, the following line of Python:

```
m, b, _, _, _ = linregress(X, Y)
```

will be translated as:

$$model\ [m, b]\ X, Y\ linregress$$

Here, the data is passed into the modeling method by variable; SIR has an alias resolution phase that will replace these instances of X and Y with the appropriate data.

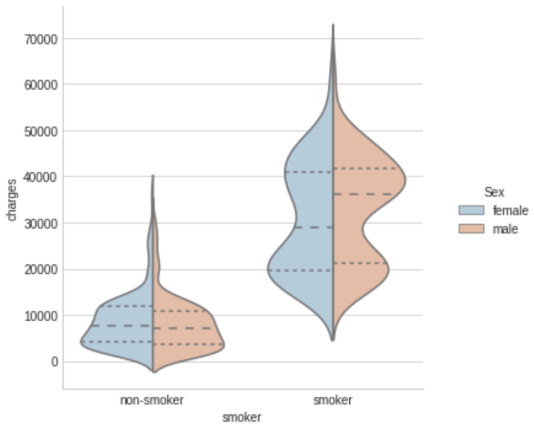


Fig. 3: Example of data visualized by a violin plot with filters.

4) Data Visualization Methods: Calls to data visualization methods are handled in much the same way as tests. Consider:

$$plot\ D\ N_V$$

Here, D represents the data being visualized and N_V the name of the visualization method. E.g., the following line of Python:

```
sns.catplot(x="smoker", y="charges",
  hue="sex", kind="violin", data=df)
```

will be translated in SIR as:

$$\begin{aligned} plot\ \{df, cols : [\text{`}charges\text{'}], \\ filters : [\text{`}smoker\text{'}, \text{`}sex\text{'}]\}\ violin \end{aligned}$$

The idea with the plot SIR expression is to capture precisely what data is displayed in the plot. For instance here, the violin plot will plot `"charges"` for each `"smoker"` value, and additionally display data filtered by the `"sex"` of the subject. We have included an example plot for reference in Fig. 3. From the plot, an analyst can view the distribution and variance of the charges for each combination of smoker, non-smoker and female, male; this is what we capture in SIR.

5) Models: In order for our compiler to accurately compile statistical and data visualization functions from Python into SIR, we have built a set of *models* that specify the translation. We developed these through careful study of Python data visualization libraries, statistical libraries, and open-source code that utilizes them in order to determine what they test/plot, and how to translate that into the IR.

For example, consider the call to `catplot` shown above. This is a visualization function in the *seaborn* library for drawing categorical plots. From studying the API[2], we can see that the kind of plot is specified by the "kind" argument, the data is either the "data" argument with "x" and "y" arguments specifying columns, or if no "data" argument is passed then "x" and "y" form pairs of points that can be plotted in,

[2]See https://seaborn.pydata.org/generated/seaborn.catplot.html.

e.g., a scatter or line plot. Additionally, arguments "hue" and "weights" can take additional columns to color or size the elements of the visualization. We can see this in Fig. 3. We also developed models for testing methods; e.g., a model for `ttest_ind` specifies that the `equal_var` parameter value differentiates between Student's and Welch's t-test, and overall the test models are simpler than the plot models.

Our compiler is modular, and adding support for more plots or tests involves defining a model, but the models are quite small (the `catplot` model is just a Python dictionary with 7 elements, and is included in the artifact for interested readers).

6) Functions, Conditionals, and Loops: Data analysis programs, being programs typically written in general-purpose programming languages, have access to more than just statistical APIs. Analysts might loop over the columns of a data frame, guard certain methods behind conditions, or define functions that implement some common data analysis procedures that they apply many times. To this end, SIR supports function definitions and calls, loops, condition expressions, and aliasing. After a data analysis program is first translated into SIR, we perform two passes over the initial IR to deal with functions and aliases.

First, we perform *function call resolution*, where calls to functions are replaced with:

- aliases for function arguments to ensure proper data flow;
- the in-lined function body;
- "un-alias" expressions to restore state.

Next, we perform *alias resolution*, where identifiers are replaced with their values. This handles all regular aliases, as well as the function call aliases introduced by the previous pass. At the end, the only identifiers left in the SIR program are data frame identifiers and so-called "loop identifiers", or "over-approximate identifiers"; these represent loop iterator variables. We make only limited attempts to reason about loop iterations, and so the presence of these loop identifiers is an indicator for unsoundness in subsequent analysis. Generally speaking, if a loop identifier is present in the column, filter, or slice component of a SIR data structure, we assume that all possible column, filter, or slice values can be taken.

Summary: The goal of SIR is to standardize data analysis code into a "normal form" more amenable to analysis. Our current prototype, STAT-LINT, supports Python, the language of choice for most statistical notebooks, and a very expressive language with many ways to do the same thing. Beyond having several libraries each implementing similar methods, data frame expressions are highly varied, and Python has a lot of dynamism and syntactic sugar that make direct static analysis of the code complicated. As a concrete example, recall the Python data frame expression mentioned earlier; the order of the filters, column access, and slice are fully interchangeable, so translating those varied expressions into the same SIR expression greatly simplifies further analysis.

B. Analysis of Intermediate Representation

By now, we have compiled a data analysis program into SIR, and have resolved calls and aliases. This SIR program encodes the statistical and data visualization methods that were applied to data, and now we add the *checks* and *obligations* imposed by the tests and visualizations to the SIR program.

1) Add Checks and Obligations: A set of models maps *test*, *plot*, and *model* SIR expressions into the *checks* they make and *obligations* they impose. These take the form:

$$plot\ \textbf{or}\ test\ \textbf{or}\ model \longmapsto check\ \textbf{or}\ obligation$$

For instance, our t-test model imposes obligations of normality, homoscedasticity, and independence on the data being tested, and our scatterplot model states that the correlation between data sets is shown if more than one is plotted. Not all tests impose obligations, e.g., Levene's test checks homoscedasticity of the inputs, and so does Bartlett's test but it also assumes normality. Examples of our mapping models are given below:

$$\begin{aligned} test\ d_1, d_2\ \textit{Student's t-test} &\longmapsto \textit{obl}\ d_1, d_2\ normal \\ &\longmapsto \textit{obl}\ d_1, d_2\ homoscedastic \\ &\longmapsto \textit{obl}\ d_1, d_2\ independent \\ plot\ d_1, d_2\ scatter &\longmapsto vis\ d_1, d_2\ correlation \\ test\ d_1, d_2\ Bartlett &\longmapsto test\ d_1, d_2\ homoscedastic \\ &\longmapsto \textit{obl}\ d_1, d_2\ normal \\ plot\ d_1\ histogram &\longmapsto vis\ d_1\ normal \end{aligned}$$

The *model* SIR expressions are slightly more involved, as residuals are *computed* based on model parameters as well as the data involved in the model. For linear regression:

$$\begin{aligned} model\ [m, b]\ X, Y\ lregress &\longmapsto \textit{obl}\ b + m * X - Y\ normal \\ &\longmapsto \textit{obl}\ b + m * X - Y\ hmsced \end{aligned}$$

The obligations are imposed on the residuals, the difference between the fitted model ($b + m * X$) and the data (Y).

In our framework, we support two styles of checks: *test* and *visualize* (or *vis*), which discriminates between assumptions that were tested with methods like the Shapiro-Wilk test, and that were examined via plots like a histogram. In practice, STAT-LINT does not distinguish between these two cases, but the rules are configurable in case someone disagrees with our assessment (e.g., some work in statistics discourages checking assumptions with visualizations [47], [15]). The full set of models is available in the artifact.

2) Resolve Obligations: This final translation results in a set of checks and obligations, and so we are finally in a position to validate if all obligations are satisfied in the program. We define a set of declarative *statistical best practice rules* in the following style:

$$\frac{necessary\ checks}{satisfied\ obligation, Q_C} \quad \text{(RULE-NAME)}$$

Here, the *necessary checks* must be present for the *obligation* to be *satisfied*. We additionally note a check quality Q_C which represents the overall quality of the match between the checks and in the obligation. In exact matches $Q_C = 1$, but

lower Q_C scores can occur if data does not exactly match, e.g., when checks are performed on a sub- or super-set of the data with an obligation; essentially, if any of the filters, slice, or groupby aspects of the data are not the same, we record a partial match of $Q_C = 0.5$. Lower scores can also occur if not all the data involved in an obligation is checked.

As an example, consider a t-test on two samples. Both samples should be normally distributed, and an analyst can check for normality of both individually:

$$\frac{test\ d_1\ normal \qquad test\ d_2\ normal}{obligation\ d_1, d_2\ normal, 1.0}\ \text{(NORM-DBL-EACH)}$$

If they fail to check both data, we have a partial match:

$$\frac{test\ d_1\ normal}{obligation\ d_1, d_2\ normal, 0.5}\ \text{(NORM-DBL-LEFT)}$$

Note that when checking obligations, all possible options are considered, and as long as one match applies with $Q_C = 1$ then the obligation is said to be met, or *fully checked*.

Perhaps an analyst plotted a subset of the data subject to an obligation; in such a case, we have a partial match:

$$\frac{vis\ d'\ normal \qquad d \approx d'}{obligation\ d\ normal, 0.5}\ \text{(NORMAL-IMP-VIS)}$$

In this and other rules, $d \approx d'$ denotes a partial match between d and d'. $d \approx d'$ iff both refer to the same data frame, same columns, but the filters, slice, or grouping do not match.

Note on Correlation and Independence: Technically, checking for correlation is not the same as checking for independence. That being said, data that appears correlated is unlikely to be independent; it is not uncommon [4], [41] to use correlation tests to investigate independence, and many notebooks (particularly exploratory data analysis notebooks) check for correlations to find relationships between data (**16.7%** of notebooks studied in our evaluation did so). As such, we include this rule, which can be disabled if desired:

$$\frac{test\ d_1, ..., d_n\ correlation \qquad n > 1}{obligation\ d_1, ..., d_n\ independent, 1.0}\ \text{(CORR-IND)}$$

Independence of a single sample cannot be tested or visualized, and must be assumed; this is not supported in STAT-LINT.

3) Multiple Comparisons: The multiple comparison problem in statistics occurs when an analyst performs many tests on the same piece of data. This is unwise as it increases the likelihood of spurious statistical significance. To capture this, we define a few additional rules:

$$\frac{|test\_result\ \ p_T\ \ d\ \ n_T| \leq n \qquad n_T \in I_N}{p_T\ comp\_op\ \frac{\alpha}{n}}\ \text{(MULT-COMP-SURE)}$$

Here, $|\ IR\ statement\ |$ represents the number of occurrences of that IR statement in the SIR program. The rule states: when there are at most n test results p_T from statistical inference tests $n_T \in I_N$ (I_N is a list of statistical inference test names, including Student's t-test but excluding assumption tests like Shapiro-Wilk) performed on the same data d, the test results must be compared against a significance level α that is divided by n; i.e., a Bonferroni correction. This represents cases where we are *absolutely sure* that a correction factor was applied. There are situations where an analyst will not explicitly compare the p-value (i.e., test result) of a test with a confidence, e.g., they might just print the results and interpret them themselves. In such cases we report a warning.

4) Model Obligations: A discerning analyst will check the residuals, but many analysts instead plot the model against the data to visually judge the fit. This is less precise, so we consider this a partial check (similar for homoscedasticity):

$$\frac{vis\ b + m * X\ line}{obl\ b + m * X - Y\ normal, 0.5}\ \text{(PLOT-LINE-PARTIAL)}$$

Summary: Given a SIR program compiled from a data analysis program, we first translate all test, visualization, and modeling methods into the checks they perform and obligations they make, and then check these for adherence to a set of statistical best practice rules, finding obligations that are fully met (with a quality score $Q_C = 1.0$), partially met (with a $Q_C \in (0, 1)$) or unmet ($Q_C = 0.0$).

Important Note: In notebooks, it is not uncommon for an analyst to perform some check in a cell, look at it, and decide what subsequent analysis to perform based on the result. In other words, they do not necessarily guard the application of statistical methods behind checks for assumptions. As such, we also do not require this in our statistical rules; we give the analyst the benefit of the doubt, and are satisfied if checks were performed regardless of where they were performed. This is important for checks related to model residuals, as the model has to be fit before it can be validated.

C. Implementation

We developed an interpreter for SIR in Python, and a compiler from Python into SIR also written in Python. We also provide models for data visualization functions, statistical functions, as well as a set of checks and obligations and statistics best practice rules as Python dictionaries that are consumed by our SIR interpreter. These rules are fully customizable. All of these pieces together make up a prototype implementation called STAT-LINT, the first linter for statistical misuses. All of this is available in our artifact.

D. A Note on Soundness

Python, a dynamic language, is not amenable to precise static analysis [63]. Particularly with data manipulation code, there are many dynamic methods that are infeasible to model, e.g., analysts can apply arbitrary functions to columns of data frames. Moreover, notebooks can have a complex execution order between cells [51], [59], and STAT-LINT treats the notebook as if it executed from the first cell to the last; this is

particularly relevant to alias resolution. That being said, checks do not have to happen before obligations, and can happen after. Moreover, since data is often modified through the course of execution, accounting for every possible execution order would likely result in many false positives. Our compiler from Python into SIR is a best effort, and has been thoroughly tested and validated on the notebooks discussed in the next section, as well as a battery of "unit test" notebooks to ensure that STAT-LINT can correctly and precisely capture most behavior.

E. Extending SIR and STAT-LINT

There are myriad other best practice rules that one could add to STAT-LINT. On the topic of validating assumptions, one could add rules related to multi-variate statistical analysis (e.g., ensuring that the variance-covariance matrix is computed), or rules related to time series analysis (e.g., ensuring that time series are checked for seasonality or autocorrelation). To achieve this, one mainly needs to define the model for the methods of interest (so that it can be compiled into SIR), and then define the best practice rules for that method. These would be the easiest to add, but there are also best practice rules outside validating assumptions; e.g., Wasserstein and Lazar [60] conclude that "Good statistical practice emphasizes ... a variety of numerical and graphical summaries of data ..." and that "no single index should substitute for scientific reasoning"; such rules could be incorporated into STAT-LINT, with a rule ensuring that data be passed to multiple plotting or summary methods, or a rule ensuring that data is not only passed to methods that compute a p-value, resp. It is also possible to extend STAT-LINT to handle different multiple comparison corrections, e.g., by adding a rule like MULT-COMP-SURE but with Šidák's correction [50] applied instead.

V. EVALUATION

To evaluate STAT-LINT, we pose these research questions:

RQ1) How many instances each of fully, partially, and unchecked obligations are discovered by STAT-LINT? *To get a sense for how often analysts check assumptions.*

RQ2) How many significant misuses have fully or partially checked obligations? *To get a sense for how much checking prevents significant misuses.*

RQ3) What would the results of assumption tests and visualizations show in these cases? *To get a sense for the clarity provided by checking assumptions.*

RQ4) How many potential and significant misuses are the result of multiple comparisons? *To collect data on the prevalence of a novel statistical misuse and assess the impact of multiple comparisons.*

RQ5) Does STAT-LINT report false positives or false negatives? *To investigate the impact of unsoundness.*

RQ6) What is the running time of STAT-LINT? *To get a sense for how usable the tool is in its current state.*

A. Experimental Setup

STAT-LINT analyzes Python data science applications, and we focus this evaluation on Kaggle notebooks. Notebooks interweave code with data visualization and discussion, making them a popular format for data analysis. Kaggle in particular offers a reproducible environment for actually running the notebooks alongside the data they require. For this evaluation we selected the 60 Kaggle notebooks analyzed in prior work [55] that used parametric statistical methods and were written in Python and supplemented this with 30 additional notebooks from Kaggle; this made up **90** notebooks. All notebooks we studied are available in our artifact.

For **RQ1** we run STAT-LINT on each notebook and report and discuss the results. **RQ2-RQ4** focus on *significant misuses*; recall that a *potential statistical misuse* is a call to a parametric statistical method for which a test of at least one assumed property yields significant evidence against that assumption being met. Then, a *significant misuse* occurs when the p-value of a more appropriate but equivalent method falls on a different side of the chosen confidence threshold. So for **RQ2** we investigate the significant misuses in our studied notebooks and see if analysts were performing any checks. For **RQ3**, we investigate the significance of the failed assumption tests and the plots of the data. For **RQ4**, we investigate a novel statistical misuse: multiple comparisons without correction. For this we use STAT-LINT to find data that was subject to multiple tests, and compare the p-value of tests with corrected significance levels to see if the p-value falls on a different side of the corrected significance threshold. For **RQ5**, we manually investigated all obligations to identify cases where unsoundness might have caused a check to be missed or incorrectly detected. Finally, for **RQ6** we time how long it takes to run STAT-LINT on each notebook. All experiments run on an Apple MacBook Pro M2 Max with 64 GB RAM.

B. Baselines

To the best of our knowledge, ours is the first approach to *statically* detect violations of statistics best practice; as such, there is no baseline to compare against beyond one we construct ourselves. That said, the *prob-check* [55] tool from prior work is thematically similar; *prob-check* allows *library developers* to annotate statistical APIs with statistical assumptions, and inserts checks into the library code to "guard" API use by testing data for compliance. Our tool, in contrast, investigates the degree to which statistics code adheres to statistics best practice. Put differently, *prob-check* asks "does *data* comply with statistical assumptions?", while we ask "do *analysts* follow statistics best practice?"

In the context of **RQ2-3**, we empirically compare *prob-check* with STAT-LINT in that we investigate the overlap between unchecked obligations detected by STAT-LINT and significant misuses detected by *prob-check*; all significant misuses discussed here were found with *prob-check*. To build a baseline for **RQs 4-5**, we read each notebook, taking note of any called statistical methods, which plots were made, and what data was passed to these methods and plotting functions; we then matched up obligations made by methods with checks made by either tests or plots. For **RQ4**, we counted how often the same data frame was passed to statistical methods.

RQ1) *How many instances each of fully, partially, and unchecked obligations are discovered by* STAT-LINT*?*

Of **90** notebooks, only **14** have all obligations checked. This is mostly achieved by plotting the data subject to test, and we found that testing assumptions explicitly is comparatively uncommon (only **20** notebooks have such tests, while **57** have plots). In total, STAT-LINT found **996** obligations spanning these **90** notebooks. Of these, **319** were checked (**91** fully, and **195** partially), which leaves **677** unchecked obligations. Most checks were done by visualizing data, and few were done by testing data. This includes independence between multiple samples, which can be partially investigated by testing or plotting possible correlations between data.

STAT-LINT did not detect any checks of residuals of linear models in the notebooks we studied (we confirmed manually that there were indeed no checks of residuals), though STAT-LINT found **50%** check the fit by plotting the model. We manually inserted plots of residuals into the notebooks, and judged that only **22.2%** of these linear models were a good fit for the data. We include in Fig. 4 an example of how only looking at the model fit can be deceptive; here, a linear fit looks good, but the residuals reveal that perhaps a time series model would be more appropriate. Investigating the residuals of a model highlights the differences between the model and the data, i.e., the ways in which the model is deficient, and analysts can make more informed decisions based on this information.

> *Takeaway:* **68%** of obligations are unchecked, few notebooks tests all assumptions, and none investigate residuals. Visualizing fits and data is most common.

RQ2) *How many sig. misuses have checked obligations?*

Here, we investigate what STAT-LINT reports about the significant misuses detected by *prob-check*. In total, **13** notebooks had at least one significant misuse, and there were **23** total such misuses. Of these, STAT-LINT found that **2** had their assumptions tested, and **10** had their assumptions visualized. Of these visualizations, **8** had the exact same data plotted as was passed to the statistical method. **7** were in a single notebook, and when discussing the test results in text they expressed surprise that the hypothesis tests found no differences between the tested samples; a non-parametric test would have found the difference. The other case was in a notebook where the analyst was plotting and testing all in the same function, without interpreting the plot at all. The other, partially checked cases were: one analyst plotted the data before transforming it, and did not examine the transformed data before performing a t-test; another analyst plotted the whole data but tested only the first 100 elements.

> *Takeaway:* Only **8** of **23** significant misuses were fully visualized and none were tested, suggesting that explicitly investigating assumptions reduces the risk of committing significant statistical misuses.

RQ3) *What would the results of assumption tests and visualizations show in these cases?*

There are two primary ways for analysts to check assumptions: testing, and visualizing. As these are all potential statistical misuses, at least one test in each case must have indicated statistically significant evidence *against* assumptions. In all, the median p-value of these failed tests was on the order of 10^{-17}, indicating a *very* high confidence in the data not meeting the assumption. As for visualization, we generated box plots of the data subject to testing. We created **40** such plots, and the majority (**32**) did not look normal; **24** were clearly long-tailed, **6** looked severely abnormal, and finally **2** were "borderline". All of these plots are available in the artifact, and we included an example each of long-tail, abnormal, and borderline in Fig. 5.

As for the **8** cases where plots "looked reasonable", the tests also had a relatively low p-value (in the 10^{-3}-10^{-5} range). The reason for significant errors in these cases is that the data was quite similar, which is a situation where a careful choice of parametric or non-parametric method is most critical. In these cases, an analyst could perhaps apply some transformations to the data to bring it more clearly in line with assumptions if they choose to use parametric methods.

> *Takeaway:* Most of the data involved in significant misuses of parametric statistical methods appeared quite clearly to not satisfy assumptions.

RQ4) *How many potential and significant misuses are the result of multiple comparisons?*

In total, **7** notebooks had a significant misuse related to the lack of a correction for multiple comparisons; there were **17** such significant misuses. Only **two** notebooks identified and applied a Bonferroni correction, and both were when the parametric method was applied in a loop, computing the correction term based on the loop bounds. Here are some examples of the significant misuses: the difference between total Spotify streams on Friday and Saturday is *not* significant; there is *no* significant difference in student academic performance if they identify as being in a relationship.

> *Takeaway:* Significant misuses resulting from multiple comparisons were present in **8.6%** of notebooks, and analysts rarely apply corrections.

RQ5) *Does* STAT-LINT *report false positives/negatives?*

For this research question, we manually determined all checks and obligations made in each notebook, and compared the output of STAT-LINT with this baseline. For *false positives*, i.e., obligations with missed checks, we found only **4** cases, which were all in a single notebook investigating Spotify trends [31]. The analyst built a function that performs several statistical tests, and that function takes song titles as a string like `'artist - song'`, and splits the string inside the

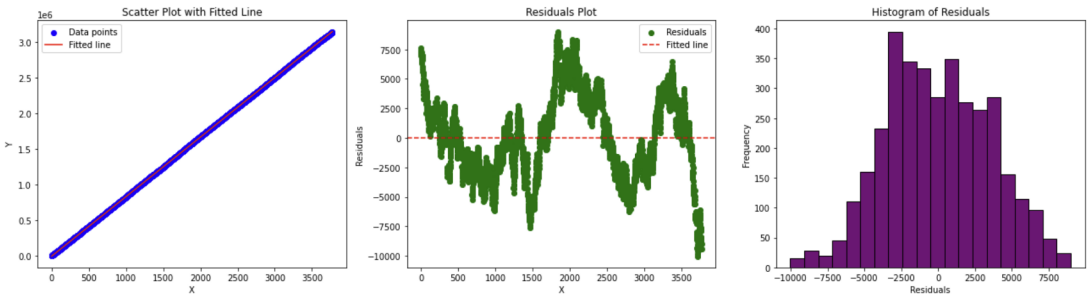
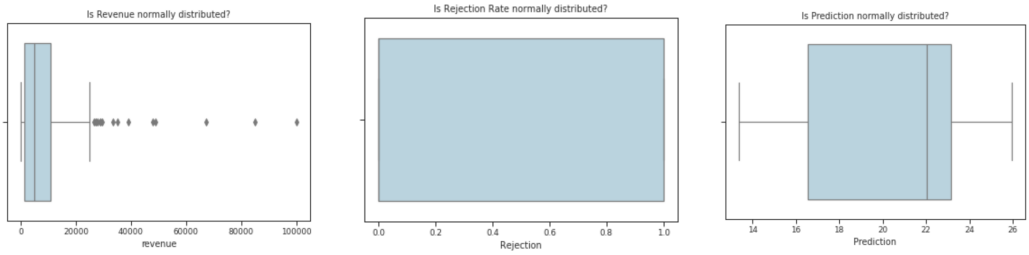


Fig. 4: An example of the pitfalls of plotting only the model fit, drawn from the `simpleeda` notebook in our evaluation. The analyst produced the plot on the left, which suggests a good fit, however had they plotted the residuals (middle and right plots) they would see that the relationship is non-linear.



(a) Long-tail box plot. (b) Abnormal box plot. (c) Borderline box plot.

Fig. 5: Examples of plotting data related to significant misuses. We judge (c) to be borderline as the mean is not quite near the middle of the figure. The other two box plots look highly un-normal.

function to extract the artist and song title. The data frame in the notebook has columns for artist and song, and while the analyst does build the appropriate plot, STAT-LINT does not model string manipulation functions (like split), and thus cannot find what data is subject to analysis. We found no *false negatives* (i.e., obligations incorrectly deemed as checked) in our manual investigation of all checks and obligations.

> *Takeaway:* Unsoundness in STAT-LINT does not appear to result in many missed checks, or obligations incorrectly identified as being checked.

RQ6) *What is the running time of* STAT-LINT*?*

The approach itself is quite lightweight, and the median run time on the studied notebooks is **3.72s**, and the average is **13.98s** namely because of some outliers. The PASSNYC notebook [19] takes almost five minutes to analyze because there are so many checks and obligations, as the phase for finding if obligations are checked takes the longest. The median time to check a single obligation is **0.04s**, and the average is **0.82s** again due to the PASSNYC notebook.

> *Takeaway:* STAT-LINT has a modest run time, and checking single obligations in particular is quite quick.

VI. Threats to Validity and Limitation

The main threat to validity is that *the presence of an assumption check does not necessarily mean that the method was used correctly*. To count a visualization as a check, an analyst should actually interpret a plot and judge whether the plot supports an assumption being true. Some statistics literature even discourages visualizing assumptions and deem the process too subjective [47], [15]. In this way, one can think of STAT-LINT as a charitable statistical linter that gives the analyst the benefit of the doubt. If they have not performed sufficient checks, STAT-LINT's feedback will inform them of the assumptions that they may not have been aware of.

It is also possible that our data set is not representative of real-world statistics notebooks. That said, some of the notebooks used in our evaluation were drawn from prior work [55], and contain a wide range of notebooks: statistics tutorials, exploratory data analysis of large datasets, Kaggle competition notebooks, and also contains notebooks in a mix of styles, including some that perform larger-scale automated testing of large datasets. We supplemented these with additional notebooks randomly sampled from Kaggle (so long as they contained at least one call to the methods we considered in this paper). Several notebooks have many "up votes" and clones, suggesting high visibility in the Kaggle community. Even though STAT-LINT is a static analysis and does not require the notebook to be executed, execution is an important

part of validation and is necessary to detect statistical misuses.

One limitation of our analysis is that it is *unsound* in that we do not precisely track data flow; thus, it is possible that we miss checks for obligations if dynamic language features are used, or only find a partial match between data involved in a check and obligation. To mitigate this threat, we manually created a baseline by carefully examining each notebook and noting the checks and obligations made by methods therein. Sound, scalable, and precise static analysis of Python is currently beyond the state of the art [63].

VII. Related Work

Notebook Studies: The research community is gaining interest in literate programs like notebooks. De Santana et al. [10] study bugs in Jupyter notebooks. Liu et al. [29] study how notebooks are refactored. Kery et al. [22] interview data scientists to glean their development practice. Chattopadhyay et al. [8] present pain points in building computational notebooks by interviewing and surveying data scientists. Pimentel et al. [38] study the reproducibility of notebooks. Wang et al. [59] propose a technique to recover the execution order of a notebook. Wang et al. [58] study how to repair notebooks that no longer execute. These are but a few of many [57], [44], [42] studies of data science notebook development practice. Our work complements this literature, as none of these studies are concerned with misuses of statistical methods.

Statistical Languages: This work proposes a static analysis to validate statistical analyses, a step towards *types for statistical languages*. The two programming languages most widely used for statistics are Python and R. Python has added support for type hints [40], which has led to further research investigating their use [43] and studying their evolution [11]. In R, Turcotte and Vitek [54] discuss the difficulty of developing a type system for R, and Turcotte et al. [53] propose type annotations that are translated to dynamic checks.

A few systems have been proposed to facilitate statistical analysis, such as Statsplorer [56], Tea [20], and Tisane [21]. Unlike our approach, these each represent significant departures from traditional programming languages, and are either stand-alone systems or present domain-specific languages for describing statistical methodology. Turcotte and Wu [55] find many potential misuses using a dynamic approach; in contrast, this work proposes a static approach that instead finds if analysts are checking assumptions themselves, and additionally identifies multiple comparison problems.

Python Static Analysis: As Python is a dynamic language, it is not readily amenable to precise and scalable static analysis [63]. That said, there have been a number of static analysis frameworks proposed for python, e.g., Scalpel [27]. Ruohonen et al. [46] use static analysis to identify security vulnerabilities in PyPI packages. Oh and Oh [34] use static analysis to detect type errors in Python. Gulabovska et al. [16] survey Python static analysis tools. Subotić et al. [51] propose a static analysis framework for data science notebooks that targets a subset of the Python language. Liblit et al. [28] present a preliminary effort to detect defects related to machine learning in notebooks, finding an average of one issue per seven notebooks. Yang et al. [62] generate documentation from data wrangling code using program synthesis and test case selection to help programmers debug their data transformation pipelines. None of these frameworks deal specifically with statistics, which is one of the reasons we developed SIR and STAT-LINT separately from existing work. In the future, we will explore building compilers from R and Julia into SIR.

Misuses of Statistics: There is ample statistics research on misusing statistical methodology [13] (dating as far back as 1938 [9]), e.g., in the biomedical field [52], pharmacology [32], medical research [23]. While misuses are known, the best way to detect and remediate potential misuses of statistics is still a matter of debate. E.g., when it comes to automated testing of statistical assumptions, plenty of work discourages testing [45], [49], [47], and plenty recommends testing [35], [33], [37] (these are but a few of many examples). Some even recommend against visual inspection of data due to subjectivity concerns [47], [15]. (This is particularly interesting given the cases in our evaluation where analysts plotted data but still committed a significant statistical misuse.) We incorporate this diversity of opinions in STAT-LINT by allowing analysts to check assumptions in a variety of ways.

VIII. Conclusion

An important aspect of statistics best practice is to check and validate assumptions about data, as even when assumptions are not met, statistical methods depending on them do not crash and yield normal-looking results. We proposed an approach allowing data analysts to validate their notebooks *quickly* and *statically* to ensure that all obligations imposed by statistical methods they used are checked, and that they are free of multiple comparison problems. We implemented this technique in STAT-LINT, the first statistics linter, and evaluated it on **90** Kaggle data science notebooks, finding that only **14** notebooks fully check all obligations, all but **2** suffer from multiple comparison problems, and **over two thirds** of obligations go unchecked. None of the notebooks we studied validated model residuals, and they should; we show that plotting only the fit of a model can be deceptive. In addition to the **17** new significant misuses related to multiple comparisons that STAT-LINT found, in the **23** cases of significant misuse of parametric statistical methods, we found that analysts *never* tested all assumptions, and only **8** misuses had assumptions validated via plot. Funnily enough, in these cases analysts expressed surprise in the results of their analysis; unbeknownst to them, more appropriate methods would yield more expected results.

Acknowledgements

This research was partially funded by the European Union (ERC "Semantics of Software System", S3, 101093186). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Huge thanks to Andreas Zeller for his support.

REFERENCES

[1] AHAD, N. A., YIN, T. S., OTHMAN, A. R., AND YAACOB, C. R. Sensitivity of normality tests to non-normal data. *Sains Malaysiana 40*, 6 (2011), 637–641.

[2] ANDERSON, T. W., AND DARLING, D. A. Asymptotic theory of certain" goodness of fit" criteria based on stochastic processes. *The annals of mathematical statistics* (1952), 193–212.

[3] ARCURI, A., AND BRIAND, L. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verification and Reliability 24*, 3 (2014), 219–250.

[4] BADEA, B., AND VLAD, A. Revealing statistical independence of two experimental data sets: an improvement on spearman's algorithm. In *International Conference on Computational Science and Its Applications* (2006), Springer, pp. 1166–1176.

[5] BARTLETT, M. S. Properties of sufficiency and statistical tests. *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences 160*, 901 (1937), 268–282.

[6] BELLOTO, J., AND SOKOLOVSKI, T. Residual analysis in regression. *American Journal of Pharmaceutical Education 49*, 3 (1985), 295–303.

[7] CASSON, R. J., AND FARMER, L. D. Understanding and checking the assumptions of linear regression: a primer for medical researchers. *Clinical & experimental ophthalmology 42*, 6 (2014), 590–596.

[8] CHATTOPADHYAY, S., PRASAD, I., HENLEY, A. Z., SARMA, A., AND BARIK, T. What's wrong with computational notebooks? pain points, needs, and design opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2020), CHI '20, Association for Computing Machinery, p. 112.

[9] COHEN, J. B. The misuse of statistics. *Journal of the American Statistical Association 33*, 204 (1938), 657–674.

[10] DE SANTANA, T. L., NETO, P. A. D. M. S., DE ALMEIDA, E. S., AND AHMED, I. Bug analysis in jupyter notebook projects: An empirical study. *ACM Trans. Softw. Eng. Methodol. 33*, 4 (Apr. 2024).

[11] DI GRAZIA, L., AND PRADEL, M. The evolution of type annotations in python: an empirical study. In *Proceedings of the ACM International Conference on the Foundations of Software Engineering* (New York, NY, USA, 2022), ESEC/FSE 2022, Association for Computing Machinery, p. 209220.

[12] DUNN, O. J. Multiple comparisons among means. *Journal of the American statistical association 56*, 293 (1961), 52–64.

[13] GARDENIER, J., AND AND, D. R. The misuse of statistics: Concepts, tools, and a research agenda. *Accountability in Research 9*, 2 (2002), 65–74. PMID: 12625352.

[14] GASTWIRTH, J. L., GEL, Y. R., AND MIAO, W. The impact of Levene's test of equality of variances on statistical theory and practice. *Statistical Science 24*, 3 (2009), 343–360.

[15] GNANADESIKAN, R., AND WILK, M. B. Probability plotting methods for the analysis of data. *Biometrika 55*, 1 (1968), 1–17.

[16] GULABOVSKA, H., AND PORKOLÁB, Z. Survey on static analysis tools of python programs. In *SQAMIA* (2019).

[17] HOCHBERG, Y. A sharper bonferroni procedure for multiple tests of significance. *Biometrika 75*, 4 (1988), 800–802.

[18] HOLM, S. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* (1979), 65–70.

[19] HUNGFEI. PASSNYC: The magic of data science, 2025. See URL: https://www.kaggle.com/hungfei/passnyc-the-magic-of-data-science. Accessed 10/05/2025.

[20] JUN, E., DAUM, M., ROESCH, J., CHASINS, S., BERGER, E., JUST, R., AND REINECKE, K. Tea: A high-level language and runtime system for automating statistical analysis. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (2019), pp. 591–603.

[21] JUN, E., SEO, A., HEER, J., AND JUST, R. Tisane: Authoring statistical models via formal reasoning from conceptual and data relationships. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (2022), pp. 1–16.

[22] KERY, M. B., RADENSKY, M., ARYA, M., JOHN, B. E., AND MYERS, B. A. The story in the notebook: Exploratory data science using a literate programming tool. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2018), CHI '18, Association for Computing Machinery, p. 111.

[23] KIM, J. S., KIM, D.-K., AND HONG, S. J. Assessment of errors and misused statistics in dental research. *International dental journal 61*, 3 (2011), 163–167.

[24] KUNDU, M., MISHRA, S., AND KHARE, D. Specificity and sensitivity of normality tests. In *Proceedings of VI International Symposium on Optimisation and Statistics. Anamaya Publisher* (2011).

[25] LE BOEDEC, K. Sensitivity and specificity of normality tests and consequences on reference interval accuracy at small sample size: a computer-simulation study. *Veterinary clinical pathology 45*, 4 (2016), 648–656.

[26] LEVENE, H. Robust tests for equality of variances. *Contributions to probability and statistics* (1960), 278–292.

[27] LI, L., WANG, J., AND QUAN, H. Scalpel: The python static analysis framework, 2022.

[28] LIBLIT, B., LUO, L., MOLINA, A., MUKHERJEE, R., PATTERSON, Z., PISKACHEV, G., SCHÄF, M., TRIPP, O., AND VISSER, W. Shifting left for early detection of machine-learning bugs. In *International Symposium on Formal Methods* (2023), Springer, pp. 584–597.

[29] LIU, E. S., LUKES, D. A., AND GRISWOLD, W. G. Refactoring in computational notebooks. *ACM Trans. Softw. Eng. Methodol. 32*, 3 (Apr. 2023).

[30] MA7555. The visa shopper guide, 2025. See URL: https://www.kaggle.com/ma7555/getting-started-the-visa-shopper-guide-eg. Accessed 10/05/2025.

[31] MAJICKDAVE. Spotify 2017 analysis, 2025. See URL: https://www.kaggle.com/code/majickdave/spotify-2017-analysis. Accessed 10/04/2025.

[32] MARINO, M. J. The use and misuse of statistical methodologies in pharmacology research. *Biochemical pharmacology 87*, 1 (2014), 78–92.

[33] NIMON, K. F. Statistical assumptions of substantive analyses across the general linear model: a mini-review. *Frontiers in psychology 3* (2012), 322.

[34] OH, W., AND OH, H. Towards effective static type-error detection for python. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering* (2024), pp. 1808–1820.

[35] PATINO, C. M., AND FERREIRA, J. C. Meeting the assumptions of statistical tests: an important and often forgotten step to reporting valid results. *Jornal Brasileiro de Pneumologia 44*, 05 (2018), 353–353.

[36] PEARSON, K., AND GALTON, F. Vii. note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London 58*, 347-352 (1895), 240–242.

[37] PERVEEN, F., AND HUSSAIN, Z. Use of statistical techniques in analysis of biological data. *Basic Research Journal of Agricultural Science and Review 1*, 1 (2012), 1–10.

[38] PIMENTEL, J. F., MURTA, L., BRAGANHOLO, V., AND FREIRE, J. A large-scale study about quality and reproducibility of jupyter notebooks. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)* (2019), pp. 507–517.

[39] POOLE, M. A., AND O'FARRELL, P. N. The assumptions of the linear regression model. *Transactions of the Institute of British Geographers* (1971), 145–158.

[40] PYTHON TEAM. Type Hints for Python. https://docs.python.org/3/library/typing.html, 2020.

[41] QUESSY, J.-F. Theoretical efficiency comparisons of independence tests based on multivariate versions of spearman's rho. *Metrika 70*, 3 (2009), 315–338.

[42] RAGHUNANDAN, D., ROY, A., SHI, S., ELMQVIST, N., AND BATTLE, L. Code code evolution: Understanding how people change data science notebooks over time. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (2023), pp. 1–12.

[43] RAK-AMNOUYKIT, I., MCCREVAN, D., MILANOVA, A., HIRZEL, M., AND DOLBY, J. Python 3 types in the wild: a tale of two type systems. In *Proceedings of the 16th ACM SIGPLAN International Symposium on Dynamic Languages* (2020), pp. 57–70.

[44] RAMASAMY, D., SARASUA, C., BACCHELLI, A., AND BERNSTEIN, A. Visualising data science workflows to support third-party notebook comprehension: an empirical study. *Empirical Software Engineering 28*, 3 (2023), 58.

[45] RASCH, D., KUBINGER, K. D., AND MODER, K. The two-sample t test: pre-testing its assumptions does not pay off. *Statistical papers 52* (2011), 219–231.

[46] RUOHONEN, J., HJERPPE, K., AND RINDELL, K. A large-scale security-oriented static analysis of python packages in pypi. In *2021 18th International Conference on Privacy, Security and Trust (PST)* (2021), pp. 1–10.

[47] SCHODER, V., HIMMELMANN, A., AND WILHELM, K. Preliminary testing for normality: some statistical aspects of a common concept. *Clinical and experimental dermatology 31*, 6 (2006), 757–761.

[48] SHAPIRO, S. S., AND WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika 52*, 3-4 (1965), 591–611.

[49] SHUSTER, J. J. Diagnostics for assumptions in moderate to large simple clinical trials: do they really help? *Statistics in medicine 24*, 16 (2005), 2431–2438.

[50] ŠIDÁK, Z. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American statistical association 62*, 318 (1967), 626–633.

[51] SUBOTIĆ, P., MILIKIĆ, L., AND STOJIĆ, M. A static analysis framework for data science notebooks. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice* (2022), pp. 13–22.

[52] THIESE, M. S., ARNOLD, Z. C., AND WALKER, S. D. The misuse and abuse of statistics in biomedical research. *Biochemia medica 25*, 1 (2015), 5–11.

[53] TURCOTTE, A., GOEL, A., KŘIKAVA, F., AND VITEK, J. Designing types for R, empirically. *Proc. ACM Program. Lang. 4*, OOPSLA (nov 2020).

[54] TURCOTTE, A., AND VITEK, J. Towards a type system for R. In *Proceedings of the 14th Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems* (New York, NY, USA, 2019), ICOOOLPS '19, Association for Computing Machinery.

[55] TURCOTTE, A., AND WU, Z. Expressing and checking statistical assumptions. In *Proceedings of the ACM International Conference on the Foundations of Software Engineering* (2025).

[56] WACHARAMANOTHAM, C., SUBRAMANIAN, K., VÖLKEL, S. T., AND BORCHERS, J. Statsplorer: Guiding novices in statistical analysis. In *Proceedings of the 33rd annual acm conference on human factors in computing systems* (2015), pp. 2693–2702.

[57] WANG, A. Y., MITTAL, A., BROOKS, C., AND ONEY, S. How data scientists use computational notebooks for real-time collaboration. *Proceedings of the ACM on Human-Computer Interaction 3*, CSCW (2019), 1–30.

[58] WANG, J., KUO, T.-Y., LI, L., AND ZELLER, A. Assessing and restoring reproducibility of jupyter notebooks. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering* (New York, NY, USA, 2021), ASE '20, Association for Computing Machinery, p. 138149.

[59] WANG, J., LI, L., AND ZELLER, A. Restoring execution environments of jupyter notebooks. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (2021), IEEE, pp. 1622–1633.

[60] WASSERSTEIN, R. L., AND LAZAR, N. A. The asa statement on p-values: context, process, and purpose, 2016.

[61] XREINA8. UniRank 2023 insights, 2025. See URL: https://www.kaggle.com/xreina8/unirank-2023-insights. Accessed 10/05/2025.

[62] YANG, C., ZHOU, S., GUO, J. L., AND KÄSTNER, C. Subtle bugs everywhere: Generating documentation for data wrangling code. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (2021), IEEE, pp. 304–316.

[63] YANG, Y., MILANOVA, A., AND HIRZEL, M. Complex python features in the wild. In *Proceedings of the 19th International Conference on Mining Software Repositories* (2022), pp. 282–293.