Expressing and Checking Statistical Assumptions

ALEXI TURCOTTE, CISPA Helmholtz Center for Information Security, Germany
ZHEYUAN WU, Saarland University, Germany

Literate programming environments like Jupyter and R Markdown notebooks, coupled with easy-to-use languages like Python and R, put a plethora of statistical methods right at a data analyst's fingertips. *But are these methods being used correctly?* Statistical methods make *statistical assumptions* about samples being analyzed, and in many cases produce reasonable looking results even if assumptions are not met.

We propose an approach that allows library developers to annotate functions with statistical assumptions, phrases them as hypotheses about the data, and inserts hypothesis tests investigating the likelihood that the assumption is met; this way, analysts using these functions will have their data checked automatically. We implement this approach in two tools: *prob-check-py* for Python, and *prob-check-r* for R, and to evaluate them we identify common hypothesis testing and statistical modeling functions, annotate them with the relevant statistical assumptions, and run **128** Kaggle notebooks that use those methods to identify misuses. Our investigation reveals statistically significant evidence against assumptions in **84.38%** of surveyed notebooks, and in **53.36%** of calls to annotated functions. In the case of hypothesis tests, had an equivalent test that *did not make these assumptions* been chosen, a *different conclusion* would have been drawn in **11.51%** of cases.

CCS Concepts: • **Software and its engineering** → *General programming languages*; • **Mathematics of computing** → *Probability and statistics.*

Additional Key Words and Phrases: Python, R, Statistical Assumptions, Annotations

ACM Reference Format:
Alexi Turcotte and Zheyuan Wu. 2025. Expressing and Checking Statistical Assumptions. *Proc. ACM Softw. Eng.* 2, FSE, Article FSE121 (July 2025), 24 pages. https://doi.org/10.1145/3729391

1 Introduction

We live in an era of data. Popular languages like R and Python are replete with libraries of *statistical methods* for performing data analysis, and literate programming environments like Jupyter and R Markdown notebooks make it easy to analyze and share insights about data. And beyond this, these languages and tools are crucial for analyzing huge data sets thanks to the automation of otherwise tedious and laborious data analysis tasks.

That said, statistics is a tricky business. One reason for this is that statistical methods often make a variety of *statistical assumptions* about input data. Consider something as ubiquitous as the *t-test*, a method for testing hypotheses about means of one or two samples: perhaps the most famous t-test is Student's, and Student's two-sample t-test makes *three* assumptions about the samples under analysis: the distribution of samples means should be normal, the samples should have equal variance, and they should be independent. A big part of sound statistics is verifying assumptions and using appropriate methods to analyze data [38, 41, 56, 84].

Part of what makes these statistical assumptions so tricky is that they are probabilistic in nature, and moreover, they will rarely be exactly met. For example, there is a small chance that a sample drawn from a normally distributed population will not appear normal. Also, what are the odds that

Authors' Contact Information: Alexi Turcotte, CISPA Helmholtz Center for Information Security, Saabrücken, Germany, alexi.turcotte@cispa.de; Zheyuan Wu, Saarland University, Saarbrücken, Germany, zhwu00001@stud.uni-saarland.de.

ACM 2994-970X/2025/7-ARTFSE121
https://doi.org/10.1145/3729391

two independent samples have the *exact* same variance? When investigating such assumptions, statisticians will sometimes plot their data to manually and visually verify them, or use other statistical methods to test hypotheses related to the assumptions. In other words, they investigate if there is enough evidence in support of an assumption.

So then the question is: *are data analysts doing any checking at all?* The short answer is: probably not. As part of this paper, we investigate **128** notebooks on the popular Kaggle notebook hosting site, and find that only **11** check or mention all of the assumptions related to the statistical methods they use. This includes popular notebooks, tutorials, and highly regarded Kaggle data analysis competition submissions that have been up-voted and cloned thousands of times.

We found this lack of checking to be disheartening, so in this paper we introduce a simple annotation language to allow popular statistical assumption to be expressed *and have an automated check generated.* The idea is to allow library developers to encode statistical assumptions explicitly and have them be checked, so that analysts using these statistical libraries can receive feedback pertaining to unmet assumptions automatically. To check an assumption, it is phrased as a hypothesis about the data, much like a statistician would do, and an automated hypothesis test is injected into the code of the annotated method. This framework is programming language agnostic, and we implement two proofs of concept: *prob-check-py* for Python, and *prob-check-r* for R. To use *prob-check-py* a developer encodes statistical assumptions in numpy docstrings, and to use *prob-check-r* they encode the assumptions in the standard Roxygen2 documentation framework.

To evaluate these tools and this approach, we annotate functions implementing common hypothesis testing and linear regression methods with the relevant assumptions, find **128** Kaggle notebooks containing references to these functions, and run them using the checked versions of these functions. Of **1,102** calls to annotated functions, **586** had strong evidence against at least one statistical assumption being met, and such cases were present in **108** notebooks. Moreover, strong evidence against assumptions was found in **267** of **360** unique call sites to these functions. When considering hypothesis tests, sometimes the difference between samples is so great that it is detected even when assumptions may not be met, but in **35** calls (representing **11.51%** of calls to hypothesis testing methods), had the analyst used a more appropriate test given the characteristics of the data, *the conclusion would have been different!*

In summary, the primary contributions of this work are:

- an annotation language for expressing assumptions about statistical properties of data;
- two implementations[1] of the annotation language, one for Python (*prob-check-py*) and another for R (*prob-check-r*), showcasing the generality of the language;
- an evaluation revealing many misuses of statistical functions in popular Kaggle notebooks, and an investigation of the severity of these misuses.

The remainder of the paper is organized as follows: Section 2 establishes statistical background; Section 3 presents a glossary of key terms; Section 4 motivates the work with a real example; Section 5 presents the annotation language, checks, and implementation; Section 6 discusses our evaluation; Section 7 treats threats to validity; Section 8 surveys related work; and finally, Section 9 concludes and presents some future work.

2 Statistical Background

Statistics is a fundamental method to understand the extensive and complex data collected in various fields, and to make effective decisions and inference under uncertainty. This section introduces some basic statistical concepts that are essential to understanding this work with a running example.

[1]See https://doi.org/10.5281/zenodo.13756766 for details.

2.1 Basic Terminology and Hypothesis Testing

The primary goal of statistics is to infer characteristics of *populations* from *samples* drawn from those populations. In order to connect facts about a sample to features of a population, one can use *statistical inference* techniques. This could be as complex as fitting a model to the data, but can also be as simple as posing and testing hypotheses about it. This latter methodology, called *hypothesis testing*, is used to determine if there is sufficient evidence in a sample to reject a hypothesis about the population that sample was drawn from. In a hypothesis test, an analyst poses a *null hypothesis*, called H_0, corresponding to a hypothesized property value or relationship between properties of one or more populations. Then, the *alternative hypothesis*, called H_1 or H_a, indicates the opposite.

Running Example. Say a civil servant is investigating how much weekly exercise (in minutes) their town's citizens are getting based on a recent town survey. They might hypothesize that teenagers get more exercise than other age groups (their H_0) and would investigate if the data *from the survey* would support or refute such a hypothesis about *the entire population.*

2.2 Assumptions

Many hypothesis testing methods make assumptions about the characteristics of the samples or populations being analyzed; these are known as *statistical assumptions*, the most common assumptions are listed below.

Distributions. Parametric statistical methods make assumptions about the distributions of populations that were sampled; this allows them to more precisely infer population parameters provided the assumption is met. The *normal distribution* is particularly important in statistics, and methods like the analysis of variance (ANOVA) and t-tests (like Student's and Welch's) assume that the means of the samples being analyzed are normally distributed.

Homoscedasticity. Multiple samples are said to be *homoscedastic* if they have equal variance. Statistical methods that assume homoscedasticity often pool the variance of the samples being analyzed, something that is only really valid if they are homoscedastic. For example, Student's t-test and ANOVA assume homoscedasticity, but Welch's t-test does not.

Independence. Independence is a fundamental requirement in most hypothesis testing methods, and refers to the condition where the outcome of one observation is not affected by others. There are two types of independence:

- *within-group* independence requires that no individual measurement within a sample affects any other, which is often managed by the experimenter (e.g., by sampling with replacement instead of without replacement);
- *between-group* independence ensures that different samples are not interconnected, which is also often managed by the experimenter but can be investigated through statistical tests after data collection to some degree.

Running Example. It is possible that weekly exercise minutes is not normally distributed; probably most people get little-to-no exercise, with outliers that exercise a lot, making for a *long-tailed* distribution. Also, the data may not be homoscedastic: there might be more variability among teenagers than among all non-teenagers. If the survey was given to households, the data might not be independent, since children of parents who exercise more might also exercise more.

2.3 Confirming Assumptions

Confirming assumptions and exploring the characteristics of data is an important first step in statistical analysis, but most assumptions cannot be checked with complete certainty. This is a consequence of the probability theory underlying statistics. For example, there is a (tiny!) chance that a sample drawn from a normally distributed population does not appear normal.

There are two major ways to investigate assumptions and data characteristics. First, plotting and visual checks: statisticians plot histograms to visualize distribution shapes, box plots to see quartiles, outliers, and data spread, or even quantile-quantile (Q-Q) plots to visualize how closely the quantiles of probability distributions align. Beyond visual checks, there are hypothesis testing methods to test hypotheses about sample properties. For instance, the *Shapiro-Wilk* test can be used to test the null hypothesis that a sample is drawn from a normally distributed population. But, and very crucially, one must understand that these hypothesis tests can *incorrectly* reject a null hypothesis when it is in fact true, discussed next.

Running Example. Our civil servant plots a histogram of weekly exercise minutes, confirming a long-tailed distribution. They could use a non-parametric statistical method to investigate their hypothesis, but instead choose to log-scale the data, and perform a Shapiro-Wilk test to assess its normality. They perform Bartlett's test (a homoscedasticity test) to test an assumption of homoscedasticity between the log-scaled groups, confirming it.

2.4 Quantification of Uncertainty

Inferring population characteristics from samples introduces uncertainty, which can lead to two types of errors:

- **Type 1 error**: when the hypothesis is true but is rejected, also called a false positive.
- **Type 2 error**: when the hypothesis is false but is accepted, also called a false negative.

Calculating the probabilities of these two errors can effectively quantify the uncertainty in hypothesis testing. The probability of committing a Type 2 error is related to the *power* of a test against the alternative hypothesis, which often plays a role in choosing which tests to use. In statistics, one test is considered *more powerful* than another if its Type 2 error rate is lower. Intuitively, a more powerful test is more able to detect departures from the null hypothesis, and typically tests that make more assumptions are more powerful.

As for the reliability of statistical results, the probability of committing a Type 1 error (named α) is more relevant. In hypothesis testing, an acceptable threshold of α is pre-set; this is called *significance level.* The significance level should be chosen by the statistician based on the situation, and 0.01 or 0.05 are the standard, commonly used values [56]. A common way to see if a hypothesis test rejects or fails to reject the null hypothesis is to calculate the *p-value*, indicating the degree with which the data contradicts the null hypothesis, and compare it with the chosen significance level. A p-value less than the significance level leads to H_0 being rejected, and represents statistically significant results. The p-value can be thought of as the lowest significance level α at which H_0 could be rejected. A lower p-value signifies stronger evidence against the null hypothesis.

Running Example. Since there is little evidence against the data being normally distributed, and say the survey was designed such that all observations are independent, the civil servant decides to use a t-test to investigate the difference in the mean of weekly exercise minutes. There are several versions of the t-test, two notable ones are Student's, which assumes homoscedasticity, and Welch's, which does not. Given that the civil servant has previously established that the groups are homoscedastic, they decide to use Student's test, as it has a higher power than Welch's.

2.5 Confirming Assumptions with Tests

One consequence of this uncertainty is that it complicates confirming assumptions, and there is no clear consensus on the "best" way to verify them. Particularly with tests for assumptions, some methods are quite sensitive to departures from the assumed properties of the data. Moreover, automated testing can result in false positives as well as false negatives due to the inherent statistical nature of the data collection process. These are the Type 1 and Type 2 errors we discussed in the previous section. Also, many statistical methods are robust, meaning that assumptions not being met does not translate to significant errors in the results.

The statistics community is far from united when it comes to testing assumptions, with plenty of work discouraging testing [54, 58, 75, 91], and plenty recommending testing [6, 12, 26, 29, 38, 41, 42, 80]. In practice, a study [21] where researchers were tasked with investigating a fictitious data set found that they rarely investigated assumptions about data, and when they did, would often use statistical tests. Besides tests, visual methods can help analysts ascertain properties of data, and yet Schoder et al. [58] recommend against using visual techniques to assess the distribution of data, as do Wilk and Gnanadeskain [16], due to subjectivity concerns. But ultimately, being aware of and investigating whether or not assumptions are met is generally regarded as wise, and testing is one of many tools for this purpose.

Even those that are in favor of testing disagree on which statistical tests to use. In particular, the Shapiro-Wilk test has been the standard for many years for testing normality, and it was shown to have a lower Type 2 error rate [35] by Razali and Yap in 2011, but recently the Anderson-Darling test has gained favor. Keselman et al. [26, 27] and Othman et al. [40] suggest using the Anderson-Darling test (as opposed to the Kolmogorov-Smirnov or the Cramer-von Mises tests), although they did not compare with the Shapiro-Wilk test since Shapiro-Wilk specifically tests for adherence to the normal distribution, and these others are more generic.

The software engineering community routinely performs statistical tests on user studies; Kitchenham et al [28] point out several issues in studies in the human-centered software engineering domain. They highlighting three pieces of recent work in statistics performing simulation studies which examine the error rates of statistical tests with and without pre-testing distributional assumptions. While conducted in similar ways, these studies came to conflicting conclusions:

- Rasch et al. [54] find that no pre-test should be used, Welch's test should be used by default, and the Mann-Whitney U test should not be used;
- Rochon et al. [55] find that pre-testing did not cause much harm, at worst beign unnecessary for large samples, and in the case of small samples the Shapiro-Wilk test is not powerful enough to detect deviations from normality, so non-parametric methods should be preferred;
- Lantz et al. [29] conclude that simply using ANOVA or the Kruskal-Wallis test without pre-testing did not perform noticeably better than with pre-test, but that in cases of strong non-normality, pre-testing was much better than simply using ANOVA.

We aim to incorporate the lack of agreed upon best practice into our annotation design. By allowing developers to specify the testing method, as well as the significance level, we will give them the ability to align with their preferred school of thought.

2.6 Statistical Modeling

Hypothesis tests are not the only way to analyze data, and there is a wide array of methods to fit *models* to data. For instance, *linear regression* is a method to estimate the linear relationship between a dependent variable and one or more independent variables. Such relationships are often obtained by minimizing the sum of the squared residuals, i.e., the sum of the differences between the fitted line and the actual data. These techniques *also* make assumptions, such as "the residuals should be normally distributed", and "the residuals should be homoscedastic" (in this case, homoscedasticity of residuals means that the variance of the residuals is constant throughout, i.e., the variance does not depend on any of the independent variables).

Running Example. The civil servant wants to use linear regression to determine if there is a linear relationship between daily exercise minutes and age. They fit a model to the data, find a negative linear relationship between exercise minutes and age, and analyze the residuals: while the residuals seem to be normally distributed, they are not homoscedastic, as the variance in exercise minutes seems to decrease with age. They hypothesize that this is because, as people get older, they are less able to exercise in general.

2.7 Python, R, Kaggle, and Notebooks

Python and R are among the most popular programming languages, particularly for data analysis partly due to how many statistical libraries are available in each language. Kaggle is a website that hosts Python and R data science *notebooks*, which are literate programming environments in which users mix blocks of code with accompanying text, mathematical formulas, plots, etc. Unless otherwise specified, Kaggle notebooks are accompanied by a data set, making results reproducible and robust. These notebooks represent *statistical software*, and as the software engineering community it is our responsibility to develop languages and tools to support the creation of robust software; this is the purpose of the approach presented in this paper.

3 Glossary

In this section, we provide a comprehensive glossary of terms that will appear in the paper.

- a **statistical assumption** is an assumption related to statistical properties of data.
- a **statistical method** is some method or procedure from statistics that tests statistical properties. These statistical methods typically make statistical assumptions, but not always.
- a **statistical function** is a function (in Python, R, etc.) that implements a statistical method. Student's t-test is a statistical method, and Python's `ttest_ind` function (from `scipy.stats` library) is a statistical function.
- a **statistical library** is a library that implements one or more statistical functions.

The above are simple or generally recognized terms in statistics. We propose to augment this with additional terminology:

- a statistical assumption is **unconfirmed** if a statistical test cannot conclude that the the assumption is met (by rejecting the null hypothesis); otherwise, the assumption is **confirmed**.
- a **potential misuse** of a statistical method occurs if one of the statistical assumptions made by the method are unconfirmed.
- a **significant misuse** is a misuse of a statistical method where a different method that tests the same property and makes no unconfirmed statistical assumptions would conclude differently at the same statistical significance level.

Also, in the context of this paper, we will draw an explicit dichotomy between **developers** of statistical libraries and **analysts**, or users of statistical libraries. The developers of Python's `scipy.stats` library are developers, and people who use `scipy.stats` are analysts.

4 Motivation

To help illustrate how analysts can stumble into pitfalls, consider a Kaggle notebook in which an analyst is investigating the factors that contribute to the attrition rate of employees at IBM [77]. We forked this notebook [78] and modified it with the analysis we will present here, and Fig. 1 contains snippets and graphs required for following along in this section.

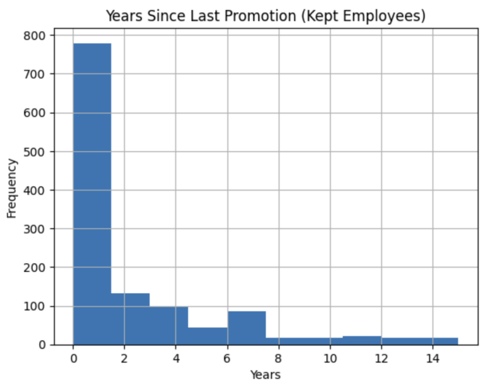
In the notebook, the analyst is working with a data set containing a plethora of information about IBM employees, like salary and hours worked, in addition to their responses to various survey questions and whether or not the employee quit. The analyst wants to determine which features

```
1   def get_p_value(s1, s2):
2       if(len(s1) > 30 & len(s2) > 30):
3           z, p = ztest(s1,s2)
4           return p
5       else:
6           t, p = ttest_ind(s1,s2)
7           return p
8
9   def get_p_values(data, cat, numerics):
10      output = {}
11      for numeric in numerics:
12          s1 = ... # lost employees
13          s2 = ... # retained employees
14          row = {"p-value" : get_p_value(s1,
                        s2)}
15          output[numeric] = row
16      return pandas.DataFrame(data=output).T
17
18  def get_sig_numerics(data, cat, numerics):
19      df = get_p_values(data, cat, numerics)
20      return list(df[df["p-value"] < 0.05].
                    index)
```

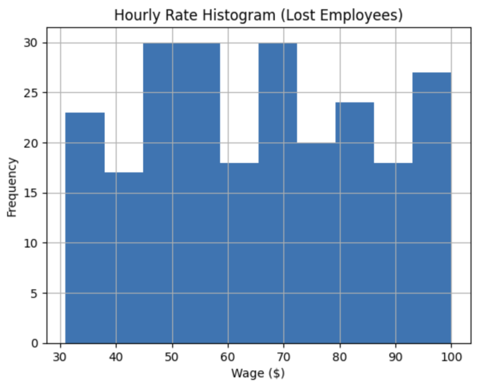
(a) Analyst defines their own automated statistical analysis methods

```
21  import scipy.stats as stats
22
23  yslp = 'YearsSinceLastPromotion'
24
25  lost = data[data['Attrition'] == 'Yes']
26  kept = data[data['Attrition'] == 'No']
27
28  kept.hist(yslp) # Fig. 1(b)
29  lost.hist('HourlyRate') # Fig. 1(d)
30
31  stats.shapiro(lost['HourlyRate'])
32  # -> 1.40e-06
33  stats.shapiro(kept[yslp])
34  # -> 1.08e-41
35
36  p_kept = kept[yslp]
37  p_lost = lost[yslp]
38
39  # the following yields 0.2058
40  stats.ttest_ind(p_lost, p_kept)
41  # the following yields 0.0412
42  stats.mannwhitneyu(p_lost, p_kept)
```

(c) Our own investigation of the characteristics of the data (e.g., assumed properties)



(b) Histogram of years since promotion



(d) Histogram of hourly rate

Fig. 1. IBM Employee Attrition Analysis Example

differ between lost and retained employees, and so they extract all of the numeric data and wish to perform automated statistical analysis on it.

The analyst then defines a function, `get_sig_numerics` on line 18, which identifies all of the numeric features that have different means (with statistical significance) between lost and retained employees. To do so, they call a function, `get_p_values` (line 9), that loops over the data structure, separates the employees into two groups, and performs either a Z-test[2] or t-test (in `get_p_value`, line 1). Then, if the resulting p-value is smaller than the chosen significance level 0.05 (line 20), the feature is deemed significant.

Now, the function `ttest_ind` performs Student's t-test by default, and as we saw in the previous section this version of the t-test makes three assumptions: normality, homoscedasticity,

[2]The Z-test is a test for the mean of normally distributed samples.

and independence. In the snippet just discussed, we see that the analyst is not performing *any* automated checking of these assumptions to accompany their automated analysis, and we found no mention of the assumptions in text blocks or comments and did not find any visual investigation of assumptions, although it may be possible that they were aware of or investigated the assumptions elsewhere (e.g., in another notebook). So, consider Fig. 1c, wherein we perform our own exploratory analysis of a few features of the data. We first split the data into lost and retained employees (since those are the groups that will be passed to the test), then plot two histograms: one for the `'YearsSinceLastPromotion'` feature, and another for the `'HourlyRate'` feature, which can be found in Figs. 1b and 1d respectively. We argue that there is *strong evidence against* these samples being normally distributed, and so that the assumption of normality is *unconfirmed*. To further corroborate this, we perform a Shapiro-Wilk test of normality on both samples on lines 31 and 33, which yields very small p-values, further evidence against normality.

Given this, what should the analyst have done? Even though many statistical methods are quite robust when assumptions do not appear to be met, there is a significant consequence of unconfirmed assumptions in this notebook. First, note that a non-parametric method to investigate the difference between two independent groups is the Mann-Whitney U signed rank test; we compute such a test on line 42, which yields a p-value of 0.0412, compared with the p-value of the t-test obtained on line 40 (0.2058). Had the Mann-Whitney U test been employed, the automated analysis on line 20 would have categorized the `'YearsSinceLastPromotion'` feature as being significant, where it was deemed insignificant using Student's t-test.

In summary, a little checking goes a long way. In this paper, we propose a framework to allow developers of statistical libraries to annotate functions with statistical assumptions and have these assumptions checked automatically by inserting hypothesis tests into the code, which tries to automatically confirm the assumptions when analysts use the statistical functions. This one time effort by developers can benefit analysts at no cost to them; the next section describes our approach.

5 Annotation Language and Checks

We first describe the design of the annotations themselves, and then explain the statistical assumptions we have encoded in our language, and finally discuss the methodology for checking those assumptions before sketching the implementation of this approach.

5.1 Annotation Design

The guiding philosophy behind the annotation language design is to capture *statistical assumptions* for which there exist *statistical tests*. Thus, we propose the following design:

$$b \Rightarrow P : A, T, 1 - \alpha$$

In this annotation, we have the following:

- b is an optional expression that should evaluate to a boolean indicating when the assumption should be checked;
- P indicates which parameter(s) the assumption applies to, and can refer to one or multiple parameters, or can also be an arithmetic expression over parameters, e.g., $P = x - y$ would indicate that the assumption applies to the difference between the parameters x and y;
- A is an assumption, and the assumptions we encode are discussed in the next subsection;
- T is the method to automatically test the assumption, also discussed in the next subsection;
- $\alpha \in [0, 1]$ is the significance level for checking the assumption, i.e., the significance level in the generated test. The expression $1 - \alpha$ is the *confidence* associated with the check.

A note on the "residuals" parameter. In the case that an assumption applies to the residuals of a model, note that the residuals are computed once the model has been fit. To support the expression of assumptions in this case, a parameter p can take the form $residuals = e$, where e is an expression in the target language to compute the residuals of the model in the body of the function that is building it. To sketch, imagine a function `lm` that builds a model object `m` and returns it to the user, and imagine that that the model class has a `resid()` method to compute the residuals; the parameter in this case would be $residuals = m.resid()$.

In the rest of this section we will progressively apply our approach to a running example of a hypothetical developer implementing a two-sample version of the t-test that gives the user the option to specify if Welch's method should be used (with Student's method as the default). We will work with the following snippet:

```
#' Uses Student's t-test or Welch's t-test.
def my_t_test(x, y, use_student = True):
    # ...
```

5.2 Statistical Assumptions

To determine the most common assumptions used in statistical methods, we consulted documentation from Python's `scipy.stats` [71] package, the R `stats` package [50], as well as several statistics textbooks [33, 56, 84, 86]. As it is now, our approach currently supports checks related to the distribution of data, homoscedasticity, and sample independence—to the best of our knowledge, these are the most common assumed statistical properties. Moreover, these correspond to the assumptions made by the most commonly used statistical methods in a variety of fields [34, 42, 76, 84, 87]. Concretely, our criteria for including an assumption is that *the assumption must have an associated statistical test*, so that it may be inserted into the code.

Distributions. A common assumption is that data is drawn from a population that follows a particular distribution. There is a widely accepted nomenclature for writing distributions, e.g., the normal distribution is written like $N(\bar{x}, \sigma^2)$ where $\bar{x}$ indicates the mean, and σ^2 the variance. In our survey of the literature, there are not many cases where specific values of distribution parameters (like $\bar{x}$) were assumed, mostly cases where a particular distribution was assumed, so the distribution family alone can be encoded, but the parameters can also be optionally specified.

To express this in our framework, one can write A as *dist* $<D>$, indicating that a parameter is *distributed according to distribution D*. We deliberately include *dist* as part of the annotation as, without it, the annotation would read as though it is suggesting that the parameter *is* that distribution, e.g., $x : N$ suggests that x is a normal distribution.

In our running example, we would add the following annotation to express that `x` and `y` should be normally distributed. Note the confidence $1 - \alpha$, here 0.95:

```
#' Uses Student's t-test or Welch's t-test.
#' x, y : dist<N>, 0.95
def my_t_test(x, y, use_student = True):
    # ...
```

If a developer wanted to specify that a quantity *resid* be normally distributed with mean 0 and arbitrary standard deviation with confidence 0.95, they can express further parameters along with the distribution family `resid : dist<N(0, _)>, 0.95`.

Homoscedasticity. Many statistical methods investigating multiple samples assume that all samples have equal variance, and methods investigating a single sample assumes that the sample has constant variance. In our framework, we encode this assumption with A = *homoscedastic*. E.g., $p_0, ..., p_n$: *homoscedastic*, 0.90 means "parameters p_0 through p_n should have equal variance". In our running example, Student's t-test assumes homoscedasticity of its two input samples, but Welch's test does not; as such, we can make use of the $b \Rightarrow \ldots$ component of our annotation and write:

```
#' Uses Student's t-test or Welch's t-test.
#' x, y : dist<N>, 0.95
#' use_student => x, y : homoscedastic, 0.95
def my_t_test(x, y, use_student = True):
    # ...
```

Sample Independence. Many statistical methods require that samples be *independent*, and specialized alternative methods exist for paired (or dependent) data (e.g., the samples are from the same set of students but for different tests). Independence is expressed with $A = \textit{independent}$, so, e.g., an annotation like $x, y : \textit{independent}, 0.99$ expresses that x and y should be independent.

5.3 Checking Assumptions

Conceptually, these statistical assumptions are checked by posing them as hypotheses, and testing them using an appropriate hypothesis testing method, comparing the p-value of the test with the provided significance level α in the annotation.

Note on Hypothesis Testing. Technically, a "positive" hypothesis test does not accept the null hypothesis (H_0), it can either fail to reject the null hypothesis, or reject it in favor of the alternative (H_1). We are not accepting H_0, we are saying that there is not enough evidence to reject it in favor of H_1; i.e., the data supports H_0 more than H_1.

Distributions. For a parameter p, we pose:

- H_0: p sampled from distribution D;
- H_1: p not sampled from distribution D.

To test this, we utilize the *Kolmogorov-Smirnov* test, which compares the empirical cumulative distribution function of the sample with the cumulative distribution function of the reference distribution by computing the distance between the two. If D is the normal distribution, we instead utilize the *Shapiro-Wilk* test for normality, as it is designed specifically for that distribution and has been shown to have a lower Type 2 error than alternative tests [35]. Our annotation also allows the developer to specify which test is run, bypassing these rules, and also allowing for alternative testing methods. E.g., the Anderson-Darling test is an alternative for the Kolmogorov-Smirnov test.

Homoscedasticity. For a set of parameters P, we pose, for each pair (p_i, p_j) of parameters $\in P$:

- H_0: p_i and p_j have equal variance;
- H_1: p_i and p_j have unequal variance.

We use *Levene's test* which, at a high level, compares the variance of each sample with the pooled variance of all samples. If normality is also assumed, we instead use *Bartlett's test.* A low p-value from these tests indicates significant evidence to *reject* H_0.

In the single parameter p case, we pose:

- H_0: p has constant variance;
- H_1: p does not have constant variance.

In this case, we use the non-parametric version of *Goldfeld-Quandt*'s test of homoscedasticity.

Independence. In general, independence cannot be tested for. A related notion is correlation, which can be tested for (e.g., correlation has been used in the context of testing for independence, e.g., Badea and Vlad [3], Quessy [44]). Intuitively, two independent samples are highly unlikely to be correlated, and if we consider the contrapositive, correlated samples are highly unlikely to be independent. The idea is to identify potential misuses rather than correct uses, and formulating a test by looking at correlation is useful in this respect. Given this, to test the independence of a set of parameters P, for each pair $(p_i, p_j) \in P$, we pose the following:

- H_0: there is no significant correlation between p_i and p_j;
- H_1: there is significant correlation between p_i and p_j.

We test this by computing the Spearman rank correlation coefficient—this also yields a *p-value* which indicates the probability of unrelated populations producing the same correlation. A small *p-value* is strong indication of a correlation between populations, and evidence against independence.

The Nuance of Confirming Assumptions. In Section 2.5, we outlined statistical literature with vastly differing opinions on the matter of testing statistical assumptions to highlight the fact that there is no universally agreed upon best practice in the community. For instance, while the Shapiro-Wilk test has been the standard for many years for testing normality, and it was shown to have a lower Type 2 error rate [35] by Razali and Yap in 2011, recently the Anderson-Darling test has gained favor (see Keselman et al. [26, 27] and Othman et al. [40]). Statisticians routinely publish papers on simulation studies investigating the relative effectiveness of equivalent testing methods, e.g., see Yazici et al. [90] for a comparison of normality tests. Morris et al. [37] describe good practice for conducting such studies and study *100* simulation studies that were undertaken in a recent volume of Statistics in Medicine, and there are many other simulation studies [9, 14, 19, 36].

But ultimately, being aware of statistical assumptions is generally regarded as being wise. The goal of our approach is to empower library developers to specify statistical assumptions made by their functions, to help ensure that users (e.g., analysts) are not misusing them; these annotations are turned into *automatic* and *inexpensive* statistical tests.

Now, analysts might be plotting data to verify assumptions, so when turning these annotations into checks our framework also injects new function parameters to allow users to skip the checks. The idea is, if a user is unaware of the assumptions, the checks will run and they will be notified if any assumptions are not confirmed, and if they are aware of them, they can choose to skip them.

As we have mentioned, there are also several statistical tests that test the same assumptions. To reflect this in our framework, the optional T parameter allows annotators to specify the method they want to be run to confirm the assumption. Our approach currently supports the following, though this list is extensible and can support more tests:

- for *distributional* assumptions, we support the Kolmogorov-Smirnov test (default), Shapiro-Wilk test (default for normal distribution, most powerful test for normality [35, 89]), and Anderson-Darling (recent alternative to the Kolmogorov-Smirnov test, advocated for in recent work [26, 27, 40]). See Yazici et al. [90] for a comparison of normality tests.
- for *homoscedasticity*, we support Levene's test (default), Bartlett's test (default if normality is also assumed), and Goldfeld-Quandt's test (default if testing single sample homoscedasticity). See Katsileros et al. [25] for a comparative study of the effectiveness of various multi-sample homoscedasticity tests (they find Levene's to be favorable in most cases).
- for the assumption of *independence*, which cannot be tested for in general, we instead test for correlation, supporting the Spearman's rank correlation coefficient; also, users can specify that no test should be run for this assumption. Spearman's rank coefficient has been used in the context of testing for independence, e.g., Badea and Vlad [3], Quessy [44]. A comparison of correlation testing methods was performed by Fujita et al. [13] in the context of DNA similarity. Some recent work investigated permuting samples and checking various characteristics about the original and permuted samples, but this is not a full-fledged statistical test [81]. The chi-square independence test [32] is only designed for categorical data.

5.4 Implementation

To illustrate the effectiveness and generalizability of this technique, we implemented *prob-check-py* for the Python programming language, and also implemented *prob-check-r* for the R programming language. We defined the annotation language using a simple grammar; in *prob-check-py*, a developer can write annotations directly using numpy docstrings, and in *prob-check-r* a developer writes annotations in the `roxygen2` framework using the `@probcheck` tag. This documentation is then parsed and checks are inserted into the function being documented:

- **Distribution**: for the general case, in *prob-check-py* a call to `kstest` [63] is inserted, and in *prob-check-r* a call to `ks.test` [48] is inserted. In the case of the normal distribution, for *prob-check-py* a call to `shapiro` [66] is inserted, and *prob-check-r* a call to `shapiro.test` [51] is inserted. Annotators can also specify alternative testing methods; the tool currently supports the Anderson-Darling test, using `anderson` from `scipy.stats` [60] in *prob-check-py* and `ad.test` from the `nortest` [39] package in *prob-check-r*.
- **Homoscedasticity**: for *prob-check-py*, we insert a call to `bartlett` [61] if the samples are assumed normal, and `levene` [64] otherwise. For *prob-check-r*, a call to `bartlett.test` [46] is inserted if the samples are also assumed to be normally distributed, and to `leveneTest` [5] from the `car` package otherwise. (Note: we insert the code implementing `leveneTest` from the `car` package directly to avoid introducing additional package dependencies.) In the single sample case, we insert a call to `het_goldfeldquandt` [79] from the `statsmodels` package, and in R `gqtest` [31] from the `lmtest` package.
- **Independence**: in *prob-check-py*, we insert a call to `spearmanr` [67], and in *prob-check-r* we call `cor.test` [47] specifying `method = "spearman"`.

R is an incredibly dynamic programming language: it is vectorized, dynamically typed, lazy and functional with limited side-effects, and has many retrofitted object systems [82]. A particular example of R's dynamism is in its *formula* data type: in R, users can define formulas like `x ~ y` in which the "terms" `x` and `y` refer (dynamically) either to lists in scope, or to columns in an accompanying data frame. We expand the annotation language with a special parameter term $P = \mathit{terms}(p_f\,[, p_d])$ to capture this, with p_f specifying the formula parameter and p_d the optional data parameter. This annotation would look like: `terms(formula, data) : dist<N>, 0.95`.

Who Writes the Annotations? These annotations are designed to accompany statistical functions, and so the *developers of statistical libraries* write them. If analysts use the checked version of the library, no further effort will be required on their part, and their data will be automatically checked.

6 Evaluation

In order to evaluate our approach, we selected some of the most common parametric hypothesis testing functions from Python's `scipy.stats` [71] library and R's `stats` [50] package, as well as linear regression methods (two from Python, one from R); Table 1 lists each function that we annotated (**Function**), as well as the assumptions we encoded (**Assumption**). We determined which assumptions to encode by reading the documentation and consulting statistical literature.

Then, we pose the following research questions:

(RQ1) How many annotations are required to be written by developers of statistical libraries?
(RQ2) How many unconfirmed statistical assumptions are detected by the checks?
(RQ3) How many potential/significant misuses of statistical functions are detected by our tests?
(RQ4) How often do analysts manually test assumptions?
(RQ5) What is the overhead associated with the checks?

Table 1. Which statistical functions were annotated with which assumptions, and how many of those assumptions were detected as unconfirmed in our evaluation. The first row of the table reads as: `ttest_1samp` *from Python was annotated with an assumption of normality of the input sample, there were 29 call sites (53) calls to it, and the assumption was unconfirmed in 21 call sites (44 calls).*

Function (Language)	**Assumption**	**Call Sites (Calls)**	**Call Sites (Calls) w/ Unconfirmed**
`ttest_1samp` [68] (py)	Normality	29 (53)	21 (44)
`ttest_ind` [69] (py)	Normality	42 (107)	37 (93)
`ttest_ind` [69] (py)	Homoscedasticity	42 (107)	13 (43)
`ttest_ind` [69] (py)	Independence	42 (107)	2 (5)
`ttest_rel` [70] (py)	Normality of differences	22 (22)	5 (5)
`f_oneway` [62] (py)	Normality	38 (94)	35 (87)
`f_oneway` [62] (py)	Homoscedasticity	38 (94)	33 (33)
`f_oneway` [62] (py)	Independence	38 (94)	2 (2)
`linregress` [65] (py)	Residual normality	28 (464)	22 (165)
`linregress` [65] (py)	Residual homoscedasticity	28 (464)	9 (48)
`LinearRegression.fit` [59] (py)	Residual normality	17 (35)	13 (31)
`LinearRegression.fit` [59] (py)	Residual homoscedasticity	17 (35)	5 (16)
`t.test` [52] (R)	Normality	38 (38)	18 (18)
`t.test` [52] (R)	Homoscedasticity	38 (38)	0 (0)
`t.test` [52] (R)	Independence	38 (38)	2 (2)
`aov` [45] (R)	Normality	47 (47)	34 (34)
`aov` [45] (R)	Homoscedasticity	47 (47)	14 (14)
`aov` [45] (R)	Independence	47 (47)	0 (0)
`lm` [49] (R)	Residual normality	99 (242)	63 (88)
`lm` [49] (R)	Residual homoscedasticity	99 (242)	34 (48)

The primary motivation behind **RQ1** is to establish the annotation burden on statistical library developers, noting that analysts using these libraries write no annotations. **RQs 2** and **3** investigate the prevalence of unconfirmed assumptions, and explores how these unconfirmed assumptions translate into potential misuses of statistical methods. We pose **RQ4** to determine if the assumptions encoded by our annotations are confirmed in any other ways by analysts in notebooks. And finally **RQ5** aims to investigate the overhead of the dynamic tests inserted by our tool.

6.1 Evaluation Corpus

To help answer these research questions, we collected a corpus of **128** notebooks from Kaggle that used at least one statistical function for which we added annotations. We focused on Kaggle notebooks due to their reproducibility, as Kaggle notebooks are provided along with their data sources in most cases. To find these, we used the Kaggle CLI to list notebooks that contained at least one textual reference to one of the functions we annotated. Once this list was obtained, we had to manually validate the listed notebooks:

- we ensured that the notebook contained an actual call;
- we discard notebooks that reference a private data source;
- we discard notebooks that take more than 10 minutes to run;
- we cloned the notebook manually so the correct execution environment was maintained and ran the notebook once, discarding cases where the notebook contained a non-trivial error.

We did this until we found **15** notebooks for each function. Many of these notebooks are either highly up-voted, cloned by many other uses, or both. Some notebooks contained calls to several functions under analysis, hence why we have fewer than **135** total.

Table 2. A selection of annotations for *prob-check-py* (written in numpy docstrings) and *prob-check-r* (written in Roxygen2 comments, here the "@probcheck" tag is omitted).

Function	Assumption	Annotation
`ttest_1samp` [68]	Normality	`a : dist<N>, 0.95`
`ttest_ind` [69]	Homoscedasticity	`equal_var => a, b : homoscedastic, 0.95`
`t.test` [45]	Normality of Difference	`!is.null(y) && paired => x - y : dist<N>, 0.95`
`t.test` [45]	Independence	`! "paired" %in% names(list(...)) || "paired" %in% names(list(...)) &&`
		`! list(...)$paired => terms(formula, data) : independent, 0.95`
`linregress` [65]	Residual normality	`residuals = y - (result.intercept + result.slope*x) : d<N(0, _)>, 0.05`

6.2 Methodology

For **RQ1**, we count the annotations we wrote. **RQ2** requires running notebooks, so we redefine the relevant functions at the top of a notebook, run it, and count the warnings emitted by our checks when an assumption is unconfirmed. **RQ3** requires more finesse. We first want to determine how many of the unconfirmed assumptions detected in **RQ2** translate to potential misuses, but more importantly, in the case of hypothesis tests we want to determine how many of these potential misuses would result in *different outcomes* if a more appropriate statistical method was employed. I.e., how many potential misuses translate to significant misuses. When a statistician performs a hypothesis test, they select a significance level α that determines when to reject the null hypothesis H_0. The most common significance levels are 0.01 and 0.05 [56], and typically a statistician will compare the p-value of the test with that significance level, rejecting H_0 if the p-value is less than α. So, for this research question, if a statistical function had at least one unconfirmed assumption, we identify the statistical function that should have been used given the assumptions that were confirmed, insert a call to it, and compare the p-value of the results. Let p_a denote the p-value of the appropriate test, p_i denote the p-value of the inappropriate test: if both p_a and p_i are *greater* or *smaller* than the chosen α, the outcome of the test would be the same, otherwise using a different method would result in a different outcome, which we classify as a significant misuse.

Then, for **RQ4**, we manually read through the notebooks to identify other methods to check assumptions, e.g., if a user plotted a histogram of the data to view the distribution, or if they mentioned assumptions in comments or text blocks. And finally for **RQ5**, we conduct a small synthetic experiment comparing the mean run time of functions with and without checks on random samples of length 100.

6.3 Research Questions

RQ1. *How many annotations are required to be written by developers of statistical libraries?*

Table 1 lists which assumptions were encoded in which functions. Some of these assumptions needed multiple annotations to realize: for example, R's `t.test` function is highly dynamic, and implements a 1 sample t-test, 2 sample t-test, and even a paired t-test depending on the value of flag parameters, and thus three annotations were required to capture just the assumption of normality over all these cases. Thus, in total we introduced **12** annotations in Python, and **14** in R. R requires more annotations relative to Python due to its aforementioned dynamicity, since `t.test` effectively implements Python's `ttest_1samp`, `ttest_ind`, and `ttest_rel`. In principle, if the API features one function for each statistical method, one annotation per assumption should suffice; the 8 annotations for 8 assumptions in the Python case provide evidence of this.

To get a feel for the annotations, Table 2 shows a selection of five annotations ranging in complexity from both Python and R. Most of the annotations are like the first two cases, but the annotations for R do have a higher complexity on average. For instance, functions with a `...` parameter are variadic in R, and (perhaps shockingly) users can pass named extra parameters to these

Table 3. Summary of misuses of statistical functions. Functions below the double line are regression functions, which we did not test for significant misuses since alternate methods are highly subjective. The first row of the table reads as: *for* `ttest_1samp` *from Python, we found 29 call sites (53 calls) total, 21 call sites (44 calls) with an unconfirmed assumption (i.e., potential misuse), and 2 call sites (6 calls) where a more appropriate statistical method would have yielded a different conclusion (i.e., a significant misuse).*

Function (Language)	**Total Call Sites (Calls)**	**Call Sites (Calls) w/ Pot. Misuse**	**Call Sites (Calls) w/ Sig. Misuse**
`ttest_1samp` (py)	29 (53)	21 (44)	2 (6)
`ttest_ind` (py)	42 (107)	39 (95)	10 (10)
`ttest_rel` (py)	22 (22)	5 (5)	1 (1)
`f_oneway` (py)	38 (94)	35 (89)	5 (8)
`ttest` (R)	38 (38)	23 (23)	6 (6)
`aov` (R)	47 (47)	46 (46)	4 (4)
`linregress` (py)	28 (464)	22 (165)	- (-)
`LinearRegression.fit` (py)	17 (35)	13 (31)	- (-)
`lm` (R)	99 (242)	63 (88)	- (-)
Σ	**360 (1,102)**	**267 (586)**	**28 (35)**

functions, and the names of those extra parameters can be accessed with `names(list(...))`. So, the long and complex annotation for independence on `t.test` is really saying: check independence if the string `"paired"` is not specified as the name of an extra parameter, or if `"paired"` is the name of an extra parameter, the value should be false. This is an R expression, so we imagine that an R developer should be comfortable writing that portion of the check.

Takeaway. A total of 26 annotations were required to annotate 18 assumptions.

RQ2. *How many unconfirmed statistical assumptions are detected by the checks?*

We ran each Kaggle notebook in our corpus with checked versions of the functions listed in Table 1 that emit warnings if annotated assumptions are unconfirmed. Again in Table 1, the column **Call Sites (Calls)** indicates how many call sites (resp. calls) to the annotated function are found in our corpus, and the column **Call Sites (Calls) w/ Unconfirmed** indicates how many of those call sites (resp. calls) had at least one unconfirmed assumption. Note that in this experiment the tests are performed at a significance level of 0.05, the default for the annotations.

Judging by the results, it would appear that data analysts are using statistical methods when there is strong evidence suggesting that underlying statistical assumptions are not met. The assumption of normality, a distributional assumption, is the most frequent unconfirmed assumption; not only is it the easiest assumption to check, many non-parametric methods exist for the (apparently, many) cases where populations do not appear to be normally distributed.

Takeaway. We find strong evidence against assumptions very often, and especially evidence against the assumption of normality.

RQ3. *How many potential/significant misuses of statistical functions are detected by our tests?*

When there is strong evidence against a statistical assumption from Table 1 being met, the analyst should probably have used a different statistical method that assumes less. Table 3 breaks down calls and call sites where assumptions were unconfirmed.

Potential Misuses. In total, **267** call sites and **586** calls had at least one unconfirmed assumption, and so are classified as *potential misuses* (column **Call Sites (Calls) w/ Pot. Misuse** in Table 3).

Notably, `ttest_ind` has many such calls due to its use in a few loops, like in the motivating example. Analysis of variance methods like `aov` and `f_oneway` process multiple samples at a time, increasing the likelihood that an assumption is unconfirmed (as, e.g., all samples should be normally distributed). In all of the R notebooks we looked at, each call site of `t.test` and `aov` only had one associated invocation, indicating perhaps that R users are less likely to call these functions in loops or such structures.

Significant Misuses. Now, *what is the significance of these misuses?* To answer, let us focus on hypothesis tests. If analysts are unsure about their data meeting assumptions, there are myriad other statistical methods that make fewer assumptions that can easily be substituted in. When performing a hypothesis test, an analyst will typically compare the p-value obtained with a significance level, commonly 0.05 or 0.01, and if the p-value falls below that threshold, they will reject their null hypothesis. So, to investigate the consequences of unconfirmed assumptions in this context, we inserted a call to an equivalent hypothesis test that was appropriate given the assumptions that *were* confirmed at call sites in which at least one assumption was unconfirmed, and compared the p-value obtained in both cases. Here, we deem a potential misuse *significant* if the p-value obtained by the two tests fall on different sides of the significance level chosen by the analyst.

Specific to our evaluation, whenever the assumption of normality was unconfirmed, we used the non-parametric alternative: for multi-sample independent t-tests (`ttest_ind`, `t.test`), we ran a Mann-Whitney U test; for single-sample and paired t-tests (`ttest_1samp`, `ttest_rel`, `t.test`), we ran a Wilcoxon signed-rank test; for one-way analyses of variance (`f_oneway`, `aov`), we ran a Kruskal-Wallis test. If homoscedasticity was unconfirmed but not normality, we use Welch's t-test (`ttest_ind` with `equal_var=False` in Python, or `t.test` with `var.equal = FALSE` in R) for two-sample t-tests, and a Kruskal-Wallis test for one-way analysis of variance. If independence appeared unconfirmed (because the samples appeared to be correlated), we would expect the analyst to confirm that the variables are indeed independent and that the correlation was due to chance or some other factor. In cases where the samples were indeed dependent, they should use a paired test instead. We did not proceed further with these cases as it was not possible to tell if the samples were actually independent (which in general would require the domain knowledge of the analyst, or a deeper understanding of the data set).

Column **Call Sites (Calls) w/ Sig. Misuse** summarizes these results. Overall, at **16.47%** of call sites (**11.51%** of calls) where an assumption was unconfirmed, the more appropriate test would have resulted in a different conclusion being drawn. Here are some examples:

- In a notebook exploring post-university salaries in the US [8], the analyst concludes that students of state schools that are also "party" schools tend to have a higher mid-career salary than students that attend non-party state schools. In the notebook, the analyst even writes "I'm pretty surprised that there is some significance here", and they would have found no statistical significance if they used the appropriate test.
- In a COVID-19 data notebook from India [15], the analyst finds that there is not a significant difference in the COVID-19 "cure rate" between the states of Tamil Nadu and Kerala, when they would have found a significant difference had they used the appropriate test.

Conceptually, these significant misuses occur when the evidence against the null hypothesis is neither extremely strong nor weak; in these cases, a discerning analyst should pick the most *powerful* test given the characteristics of the data and of the population to ensure the most appropriate conclusion. (Recall that the power of a test is a technical term in statistics quantifying the probability that the test commits a Type 2 error, i.e. the probability of falsely accepting the null hypothesis.)

We focused on hypothesis tests here because there are clear non-parametric alternatives that can be very easily substituted when an assumption is unconfirmed. In contrast, in the context of statistical modeling and linear regression, there are many options available to developers to correct a mistake, e.g., fitting a different model, considering different dependent variables, using a weighted method in case of a lack of homoscedasticity, transforming the data, etc. To avoid bias, we focus on hypothesis tests since the procedure for selecting an alternative is clear.

Takeaway. Unconfirmed assumptions result in **267** potential misuses in call sites (and in **586** calls). Furthermore, **28** call sites calling hypothesis testing functions (representing **35** calls) would result in different conclusions being drawn if more appropriate methods were used; this corresponds to **16.47%** call sites (**11.51%** calls) to hypothesis testing functions.

RQ4. *How often do analysts manually test assumptions?*

Given the inexact nature of statistical assumptions, there are many ways to investigate them: as discussed in Section 2, statisticians will plot histograms, box plots, Q-Q plots, etc., or will conduct hypothesis tests about assumptions, or a combination. So the question is: *how often do they do this?* We manually checked the notebooks in our corpus for evidence of assumption checking, and found that only **11** (or **8.59%**) notebooks discuss and check *all* assumptions. Interestingly, in one of those cases, they find evidence against normality and homoscedasticity and yet proceed with the test anyway without mentioning robustness—perhaps they were ignorant of alternative methods.

On the other hand, **62** notebooks performed no checking. This leaves **55** notebooks with partial or imperfect checking, in which we identified the following trends:

- many notebooks check distributional assumptions, but not homoscedasticity;
- few notebooks discuss independence, and when they do, they either assume it based on the subject of the sample, or they check by investigating correlation, as we do in our checks;
- in many cases, assumptions are incorrectly checked on the grouped data before it is divided;
- residuals were only explicitly investigated in 3 notebooks;
- plotting and visual checking are the most common methods of verifying assumptions.

In exploring this research question, a few notebooks stood out to us. In notebook "AB_Testing" [2], the user created an automated assumption checking method, but an incorrectly written `if` statement results in `ttest_ind` being called with `equal_var=True` if the samples do not appear to be homoscedastic; we believe this is a bug. Also, several tutorial notebooks [7, 73, 74] with 471 up-votes and 809 forks or clones between them have no discussion of statistical assumptions.

Takeaway. Of the **128** notebooks we surveyed, **11** check all assumptions, and **55** check a subset imperfectly. Data visualization is commonly used to investigate assumptions.

RQ5. *What is the overhead associated with the checks?*

Table 4 shows the results of a synthetic experiment comparing the run time of the functions with and without inserted checks: we report the mean and standard deviation of running the functions with and without checks on synthetic samples of length 100. Note the table reports milliseconds; none of these functions take long to run. While the overhead factor is high (column **Factor**), the additional cost of checking these calls in milliseconds is small, and we do not imagine it would interfere with a data analyst using the functions in a notebook.

Takeaway. The overhead of the checks appears low.

Table 4. Summary of experiments investigating run time. The first row of the table reads: *the average run time of* `ttest_ind` *on a sample of length 100 is 0.133ms without checks, with a standard deviation of 0.015ms, and is 0.183ms with checks, with a standard deviation of 0.099ms. The factor overhead in this case is 1.37x.*

	RT Before (ms)		**RT After (ms)**		
Function	**Avg.**	**St. Dev.**	**Avg.**	**St. Dev.**	**Factor**
`ttest_1samp`	0.133	0.015	0.183	0.099	1.37x
`ttest_ind`	0.203	0.089	0.412	0.205	2.03x
`ttest_rel`	0.134	0.016	0.317	0.133	2.37x
`f_oneway`	0.044	0.005	1.286	1.162	29.2x
`linregress`	0.027	0.003	0.269	0.008	9.96x
`LinearRegression.fit`	0.133	0.014	0.400	0.051	3.01x
`t.test`	0.046	0.005	0.668	5.183	14.5x
`aov`	0.333	0.222	5.789	5.103	17.4x
`lm`	0.353	0.117	0.672	0.140	1.90x

7 Threats to Validity

The principle threat to validity of this work is the statistical nature of the approach, since tests to confirm statistical assumptions can commit Type 1 and Type 2 errors. Type 1 errors occur when the null hypothesis is rejected when it is true, and Type 2 errors occur when the null hypothesis is accepted when it is false. The null hypothesis of the assumption checks is that the assumption is met, so a Type 1 error would result in a check failing when it should have passed, and vice-versa for a Type 2 error. These are risks inherent to statistics, which we mitigate by choosing appropriate testing methods (to mitigate Type 2 error) and a reasonable confidence $\alpha = 0.05$; the probability that the statistical checks make a Type 1 error is α.

Our findings may not generalize beyond the Kaggle notebooks we studied, but we took measures to minimize potential bias. Mainly, we randomly selected notebooks from those that contained a reference to a statistical function we had annotations for. Our corpus contains a variety of styles of notebooks, e.g., "first notebooks", tutorials, and submissions to Kaggle competitions, among others. The notebooks in our corpus have a wide range of up-votes and forks, hail from a diverse set of author accounts, and none of them are themselves forks, indicating a mix in quality and provenance. Separately, notebooks hosted on Kaggle offer a very important advantage as they very often contain the dataset used by the notebook, unlike notebooks hosted on, e.g., Github.

Finally, in our approach an unconfirmed assumption results in a warning to the analyst, and there is a real possibility that too many warnings will cause analysts to ignore them. This could be addressed by annotation writers choosing a strict α for the checks. For instance, an α of 0.01 rather than 0.05 would generate fewer warnings, and so likely result in a smaller Type 1 error (at the cost of a higher Type 2 error). Also, some statistical methods are robust under certain conditions, e.g., if the sample has low variance or if sample size is high, and are not overly affected by unconfirmed assumptions in those cases. Investigating how to incorporate robustness into the annotations is an interesting avenue of future work.

8 Related Work

8.1 Contracts

In the context of programming languages and software engineering, contracts are essentially a specified desired property for data, and are typically checked at runtime with assertions. Contracts appear in many domains, and are a well-studied area of research [20]. Option contracts [10] are contracts where the server gives the client the *option* of having the contract checked. The concept of *blame* [11, 30, 88] is integral to contracts: essentially, when a contract is found to be violated, the party responsible for the violation is "blamed", a boon for debugging. Some work proposes checking certain properties with hypothesis tests, e.g., in the context of stochastic algorithms [72]. Joshi et al. [22] perform statistical tests on execution profiles of approximate programs to investigate if a user-supplied accuracy formula holds. Gopinathan and Sergey [17] go a step further towards verification and propose a set of mechanised theorems for proving properties of abstract approximate query structures, which often use randomization to achieve time- and space-efficiency.

8.2 Probabilistic Programming

Probabilistic programming is a tool for statistical modeling, which, at face value, seems a very related topic. In actuality, the field is more concerned with building statistical modeling methods, and not to statistical data analysis. Work by Bornholt et al. [4] presents a type for uncertain data, e.g., position from a sensor. In their scheme, an "uncertain" value is accompanied by the distribution it is known to follow, and the distribution is sampled at conditionals as part of a probabilistic check, in contrast to our approach that inserts checks for the raw values without requiring sampling. Separately, work by Sampson et al. [57] presents probabilistic assertions for probabilistic programs. They allow programmers to express assertions about probabilistic programs, and they propose a tool to verify statically that those assertions pass, up to some specified probability and with specified confidence. Unlike their approach, ours is concerned with a particular execution of a program, and testing the likelihood that a raw input has a particular statistical property.

8.3 Statistical Languages

The annotations proposed in this work can be viewed as a step towards types for statistical languages. There has been some work in this area: for R, work by Turcotte and Vitek [83] discusses challenges associated with developing a static type system for R, and Turcotte et al. [82] propose a set of type annotations that are translated to dynamic checks. Neither of these pieces of work mentions statistical assumptions. On the side of Python, the language has added support for type hints [43], which has led to further research: Rak et al. [53] investigate how they are used, Allamanis et al. [1] proposed a framework for inferring such types with a graph neural network, and separate work by Guo et al. [18] investigate the state-of-the-art in type inference. If the annotations proposed in this paper are extended, adapted, or incorporated into type annotations, inferring them will be interesting due to their probabilistic nature: e.g., a list of values drawn from a normal distribution may not look normal, or "pass" a Shapiro-Wilk test due to the randomness of sampling.

Separately, a number of systems have been proposed to facilitate statistical analysis, such as Statsplorer [85], Tea [23], and Tisane [24]. Broadly, these approaches offer opt-in frameworks to help analysts perform sound model building and analysis, either as standalone software or libraries. In contrast, our approach does not ask anything from the analyst: a developer of functions making statistical assumptions can encode those assumptions and have checks inserted for them, so that an analyst using those functions will have their data checked for compliance. Our approach can also be more easily integrated into existing tools; e.g., our prototype applies our technique to Python's scipy and R's stats library, so analysts can keep their notebooks and continue using these languages, and has already revealed significant misuses in many popular Kaggle notebooks.

9 Conclusion

Statistical methods like hypothesis tests and modeling make assumptions about the data or populations that the data was drawn from. These assumptions, being statistical in nature, are not trivial to check, and in this work we propose a language to allow developers of statistical libraries to annotate methods with common statistical assumptions that those methods are predicated on. Our framework translates these assumptions into checks that account for the nuance in verifying them: in short, we generate hypothesis tests investigating the likelihood that the assumption is met.

We implemented prototypes[3] of this approach for Python and R, annotated common hypothesis testing and regression methods, and surveyed **128** Kaggle notebooks that use them. These annotated methods are some of the most widely used statistical techniques in a variety of fields [34, 42, 76, 84, 87]. We found misuses of statistical methods in **108** notebooks, with evidence of unconfirmed assumptions in **74.44%** of call sites (**53.36%** of calls). Specifically for hypothesis tests, while many are robust to unconfirmed assumptions we nevertheless found that at **16.47%** call sites (and in **11.51%** of calls), the use of more appropriate methods would have resulted in a different conclusion!

Future Work. The approach presented in this paper generalizes to support any statistical assumption for which there is an associated test. For instance, there exist statistical tests for multivariate normality, co-linearity, and conditional independence; many of these more complex assumptions are widely used in the machine learning domain, which is an exciting future direction for this work. Given the scale of the data involved and the complexity of the tests, we hypothesize that the performance of the tests will be a challenge in this domain. Moreover, in many cases the model needs to be built already in order to confirm certain properties about it (akin to how regression needs to be performed before residuals can be tested); there is an interesting dynamic here, where perhaps there is a subset of data on which assumptions can be investigated before proceeding.

There are also interesting direct extensions to this work within the domain of statistics, e.g., we view this work as a step towards a true, purpose-built, modern programming language for statistics and data analysis. One technical challenge there is developing a static technique which can precisely determine where and on exactly what data assumptions are checked; if one were to implement such a system in Python or R, precise type checking is difficult due to language dynamicity. Beyond that, in this paper we have extensively discussed the nuance associated with testing assumptions, and incorporating this nuance into a static type system will also be conceptually challenging; e.g., what would it mean for a notebook to be well-typed w.r.t. the rules of statistics?

Acknowledgments

We would like to thank the reviewers for their thorough and helpful feedback, as well as Simon Lenau, Rudolf Siegel, Ellen Arteca, and Andreas Zeller. 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.

References

[1] Miltiadis Allamanis, Earl T. Barr, Soline Ducousso, and Zheng Gao. 2020. Typilus: neural type hints. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation* (London, UK) *(PLDI 2020)*. Association for Computing Machinery, New York, NY, USA, 91–105. doi:10.1145/3385412.3385997

[2] arkaradeniz. 2024. AB_Testing. See URL: https://www.kaggle.com/code/arkaradeniz/ab-testing. Accessed Mar 21 2024.

[3] Bogdan Badea and Adriana Vlad. 2006. Revealing statistical independence of two experimental data sets: an improvement on Spearman's algorithm. In *International Conference on Computational Science and Its Applications*. Springer, 1166–1176. doi:10.1007/11751540_127

[4] James Bornholt, Todd Mytkowicz, and Kathryn S McKinley. 2014. Uncertain<T>: a first-order type for uncertain data. In *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*. 51–66. doi:10.1145/2644865.2541958

[3]See https://doi.org/10.5281/zenodo.13756766 for details.

[5] car Team. 2024. leveneTest. See URL: https://rdrr.io/cran/car/man/leveneTest.html. Version 3.1-2, accessed March 12 2024.
[6] IR Cardoso de Oliveira and DF Ferreira. 2010. Multivariate extension of chi-squared univariate normality test. *Journal of Statistical Computation and Simulation* 80, 5 (2010), 513–526. doi:10.1080/00949650902731377
[7] carlolepelaars. 2019. Statistical Tutorial. See URL: https://www.kaggle.com/code/carlolepelaars/statistics-tutorial. Accessed Mar 21 2024.
[8] cbhyphen. 2024. Post College Salaries Exploration in R. See URL: https://www.kaggle.com/code/cbhyphen/post-college-salaries-exploration-in-r. Accessed Mar 21 2024.
[9] Rafael Rodrigues de Souza, Marcos Toebe, Anderson Chuquel Mello, and Karina Chertok Bittencourt. 2023. Sample size and Shapiro-Wilk test: An analysis for soybean grain yield. *European Journal of Agronomy* 142 (2023), 126666. doi:10.1016/j.eja.2022.126666
[10] Christos Dimoulas, Robert Bruce Findler, and Matthias Felleisen. 2013. Option contracts. *ACM SIGPLAN Notices* 48, 10 (2013), 475–494. doi:10.1145/2544173.2509548
[11] Christos Dimoulas, Robert Bruce Findler, Cormac Flanagan, and Matthias Felleisen. 2011. Correct blame for contracts: no more scapegoating. *SIGPLAN Not.* 46, 1 (jan 2011), 215–226. doi:10.1145/1925844.1926410
[12] Jurgen A Doornik and Henrik Hansen. 2008. An omnibus test for univariate and multivariate normality. *Oxford bulletin of economics and statistics* 70 (2008), 927–939. doi:10.1111/j.1468-0084.2008.00537.x
[13] André Fujita, João Ricardo Sato, Marcos Angelo Almeida Demasi, Mari Cleide Sogayar, Carlos Eduardo Ferreira, and Satoru Miyano. 2009. Comparing Pearson, Spearman and Hoeffding's D measure for gene expression association analysis. *Journal of bioinformatics and computational biology* 7, 04 (2009), 663–684. doi:10.1142/S0219720009004230
[14] Joseph L Gastwirth, Yulia R Gel, and Weiwen Miao. 2009. The impact of Levene's test of equality of variances on statistical theory and practice. *Statist. Sci.* 24, 3 (2009), 343–360. doi:10.1214/09-STS301
[15] gcmadhan. 2024. COVID Analysis Prediction. See URL: https://www.kaggle.com/code/gcmadhan/covid-analysis-prediction. Accessed Mar 21 2024.
[16] Ramanathan Gnanadesikan and Martin B Wilk. 1968. Probability plotting methods for the analysis of data. *Biometrika* 55, 1 (1968), 1–17.
[17] Kiran Gopinathan and Ilya Sergey. 2020. Certifying certainty and uncertainty in approximate membership query structures. In *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part II 32*. Springer, 279–303. doi:10.1007/978-3-030-53291-8_16
[18] Yimeng Guo, Zhifei Chen, Lin Chen, Wenjie Xu, Yanhui Li, Yuming Zhou, and Baowen Xu. 2024. Generating Python Type Annotations from Type Inference: How Far Are We? *ACM Transactions on Software Engineering and Methodology* (2024). doi:10.1145/3652153
[19] Zofia Hanusz and Joanna TarasiŃska. 2014. Simulation study on improved Shapiro–Wilk tests for normality. *Communications in Statistics-Simulation and Computation* 43, 9 (2014), 2093–2105. doi:10.1080/03610918.2013.844835
[20] Ralf Hinze, Johan Jeuring, and Andres Löh. 2006. Typed contracts for functional programming. In *Functional and Logic Programming: 8th International Symposium, FLOPS 2006, Fuji-Susono, Japan, April 24-26, 2006. Proceedings 8*. Springer, 208–225. doi:10.1007/11737414_15
[21] Rink Hoekstra, Henk AL Kiers, and Addie Johnson. 2012. Are assumptions of well-known statistical techniques checked, and why (not)? *Frontiers in psychology* 3 (2012), 137. doi:10.3389/fpsyg.2012.00137
[22] Keyur Joshi, Vimuth Fernando, and Sasa Misailovic. 2019. Statistical algorithmic profiling for randomized approximate programs. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 608–618. doi:10.1109/ICSE.2019.00071
[23] Eunice Jun, Maureen Daum, Jared Roesch, Sarah Chasins, Emery Berger, Rene Just, and Katharina Reinecke. 2019. 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*. 591–603. doi:10.1145/3332165.3347940
[24] Eunice Jun, Audrey Seo, Jeffrey Heer, and René Just. 2022. 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*. 1–16. doi:10.1145/3491102.3501888
[25] Anastasios Katsileros, Nikolaos Antonetsis, Paschalis Mouzaidis, Eleni Tania, Penelope J Bebeli, and Alex Karagrigoriou. 2024. A comparison of tests for homoscedasticity using simulation and empirical data. *CSAM (Communications for Statistical Applications and Methods)* 31, 1 (2024), 1–35. doi:10.29220/CSAM.2024.31.1.001
[26] HJ Keselman, Abdul R Othman, and Rand R Wilcox. 2013. Preliminary testing for normality: Is this a good practice? *Journal of Modern Applied Statistical Methods* 12 (2013), 2–19. doi:10.22237/jmasm/1383278460
[27] Harvey J Keselman, Abdul R Othman, and Rand R Wilcox. 2014. Testing for normality in the multi-group problem: Is this a good practice. *Clinical Dermatology* 2, 1 (2014), 29–43. doi:10.11138/cderm/2014.2.1.029
[28] Barbara Kitchenham, Lech Madeyski, and Pearl Brereton. 2019. Problems with statistical practice in human-centric software engineering experiments. In *Proceedings of the 23rd International Conference on Evaluation and Assessment in*

Software Engineering. 134–143. doi:10.1145/3319008.3319009

[29] Björn Lantz, Roy Andersson, and Peter Manfredsson. 2016. Preliminary tests of normality when comparing three independent samples. *Journal of Modern Applied Statistical Methods* 15 (2016), 135–148. doi:10.22237/jmasm/1478002140

[30] Lukas Lazarek, Alexis King, Samanvitha Sundar, Robert Bruce Findler, and Christos Dimoulas. 2019. Does blame shifting work? *Proc. ACM Program. Lang.* 4, POPL, Article 65 (dec 2019), 29 pages. doi:10.1145/3371133

[31] lmtest. 2024. gqtest. See URL: https://www.rdocumentation.org/packages/lmtest/versions/0.9-40/topics/gqtest. Version 0.9-40, accessed Aug 16 2024.

[32] Mary L McHugh. 2013. The chi-square test of independence. *Biochemia medica* 23, 2 (2013), 143–149. doi:10.11613/bm.2013.018

[33] Irwin Miller and Marylees Miller. 1999. *John E. Freund's mathematical statistics*. Vol. 6. Prentice Hall Upper Saddle River, NJ.

[34] Prabhaker Mishra, Chandra Mani Pandey, Uttam Singh, Amit Keshri, and Mayilvaganan Sabaretnam. 2019. Selection of appropriate statistical methods for data analysis. *Annals of cardiac anaesthesia* 22, 3 (2019), 297–301. doi:10.4103/aca.ACA_248_18

[35] Nornadiah Mohd Razali and Bee Yap. 2011. Power Comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling Tests. *J. Stat. Model. Analytics* 2 (01 2011).

[36] Aurora Monter-Pozos and Elizabeth González-Estrada. 2024. On testing the skew normal distribution by using Shapiro–Wilk test. *J. Comput. Appl. Math.* 440 (2024), 115649. doi:10.1016/j.cam.2023.115649

[37] Tim P Morris, Ian R White, and Michael J Crowther. 2019. Using simulation studies to evaluate statistical methods. *Statistics in medicine* 38, 11 (2019), 2074–2102. doi:10.1002/sim.8086

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

[39] nortest. 2025. ad.test. See URL: https://www.rdocumentation.org/packages/nortest/versions/1.0-4/topics/ad.test. Version 1.0-4, accessed Feb 3 2025.

[40] Abdul R Othman, HJ Keselman, and Rand Wilcox. 2015. Assessing Normality: Applications in Multi-Group Designs. *Malaysian Journal of Mathematical Sciences* 9, 1 (2015).

[41] Cecilia Maria Patino and Juliana Carvalho Ferreira. 2018. 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. doi:10.1590/S1806-37562018000000303

[42] Farzana Perveen and Zahid Hussain. 2012. Use of statistical techniques in analysis of biological data. *Basic Research Journal of Agricultural Science and Review* 1, 1 (2012), 1–10.

[43] Python Team. 2020. Type Hints for Python. https://docs.python.org/3/library/typing.html.

[44] Jean-François Quessy. 2009. Theoretical efficiency comparisons of independence tests based on multivariate versions of Spearman's rho. *Metrika* 70, 3 (2009), 315–338. doi:10.1007/s00184-008-0194-3

[45] R Core Team. 2024. aov. See URL: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/aov. Version 3.6.2, accessed March 12 2024.

[46] R Core Team. 2024. bartlett.test. See URL: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/bartlett.test. Version 3.6.2, accessed March 12 2024.

[47] R Core Team. 2024. cor.test. See URL: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/cor.test. Version 3.6.2, accessed March 12 2024.

[48] R Core Team. 2024. ks.test. See URL: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/ks.test. Version 3.6.2, accessed March 12 2024.

[49] R Core Team. 2024. lm. See URL: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/lm. Version 3.6.2, accessed Aug 4 2024.

[50] R Core Team. 2024. The R Stats Package. See URL: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/00Index.html. Version 4.4.0, accessed 21/11/2023..

[51] R Core Team. 2024. shapiro.test. See URL: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/shapiro.test. Version 3.6.2, accessed March 12 2024.

[52] R Core Team. 2024. t.test. See URL: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/t.test. Version 3.6.2, accessed March 12 2024.

[53] Ingkarat Rak-Amnouykit, Daniel McCrevan, Ana Milanova, Martin Hirzel, and Julian Dolby. 2020. Python 3 types in the wild: a tale of two type systems. In *Proceedings of the 16th ACM SIGPLAN International Symposium on Dynamic Languages*. 57–70. doi:10.1145/3426422.3426981

[54] Dieter Rasch, Klaus D Kubinger, and Karl Moder. 2011. The two-sample t test: pre-testing its assumptions does not pay off. *Statistical papers* 52 (2011), 219–231. doi:10.1007/s00362-009-0224-x

[55] Justine Rochon, Matthias Gondan, and Meinhard Kieser. 2012. To test or not to test: Preliminary assessment of normality when comparing two independent samples. *BMC medical research methodology* 12 (2012), 1–11. doi:10.1186/1471-2288-

12-81
[56] Joseph P Romano. 1986. *Testing statistical hypotheses*. Vol. 3. Springer. 62–63 pages.
[57] Adrian Sampson, Pavel Panchekha, Todd Mytkowicz, Kathryn S McKinley, Dan Grossman, and Luis Ceze. 2014. Expressing and verifying probabilistic assertions. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 112–122. doi:10.1145/2594291.2594294
[58] V Schoder, A Himmelmann, and KP Wilhelm. 2006. Preliminary testing for normality: some statistical aspects of a common concept. *Clinical and experimental dermatology* 31, 6 (2006), 757–761. doi:10.1111/j.1365-2230.2006.02206.x
[59] scikit. 2024. sklearn.linear_model.LinearRegression. See URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html. Version 1.5.2, accessed Aug 2 2024.
[60] SciPy Community. 2024. scipy.stats.anderson. See URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.anderson.html. Version 1.15.1, accessed Feb 3 2025.
[61] SciPy Community. 2024. scipy.stats.bartlett. See URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.bartlett.html. Version 1.12.0, accessed Feb 6 2024.
[62] SciPy Community. 2024. scipy.stats.f_oneway. See URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html. Version 1.12.0, accessed Feb 6 2024.
[63] SciPy Community. 2024. scipy.stats.kstest. See URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kstest.html. Version 1.12.0, accessed Feb 6 2024.
[64] SciPy Community. 2024. scipy.stats.levene. See URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.levene.html. Version 1.12.0, accessed Feb 6 2024.
[65] SciPy Community. 2024. scipy.stats.linregress. See URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.linregress.html. Version 1.12.0, accessed Aug 2 2024.
[66] SciPy Community. 2024. scipy.stats.shapiro. See URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html. Version 1.12.0, accessed Feb 6 2024.
[67] SciPy Community. 2024. scipy.stats.spearmanr. See URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html. Version 1.12.0, accessed Feb 6 2024.
[68] SciPy Community. 2024. scipy.stats.ttest_1samp. See URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_1samp.html. Version 1.12.0, accessed Feb 6 2024.
[69] SciPy Community. 2024. scipy.stats.ttest_ind. See URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html. Version 1.12.0, accessed Feb 6 2024.
[70] SciPy Community. 2024. scipy.stats.ttest_rel. See URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html. Version 1.12.0, accessed Feb 6 2024.
[71] SciPy Community. 2024. Statistical functions (scipy.stats). See URL: https://docs.scipy.org/doc/scipy/reference/stats.html. Version 1.11.4, accessed 21/11/2023..
[72] Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2004. Statistical model checking of black-box probabilistic systems. In *Computer Aided Verification: 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004. Proceedings 16*. Springer, 202–215. doi:10.1007/978-3-540-27813-9_16
[73] shashwatwork. 2021. Guide to Statistical Hypothesis Testing in Python. See URL: https://www.kaggle.com/code/shashwatwork/guide-to-statistical-hypothesis-tests-in-python. Accessed Mar 21 2024.
[74] shilongzhuang. 2023. Statistical Testing Guide for Beginners. See URL: https://www.kaggle.com/code/shilongzhuang/statistical-testing-guide-for-beginners. Accessed Mar 21 2024.
[75] Jonathan J Shuster. 2005. Diagnostics for assumptions in moderate to large simple clinical trials: do they really help? *Statistics in medicine* 24, 16 (2005), 2431–2438. doi:10.1002/sim.2175
[76] Andrew F Siegel. 2016. *Practical business statistics*. Academic Press.
[77] slamnz. 2024. IBM Employee Attrition Analysis by Numerics. See URL: https://www.kaggle.com/code/slamnz/ibm-employee-attrition-analysis-by-numerics. Accessed Mar 2 2024.
[78] StatLint. 2024. IBM Employee Attrition Analysis by Numerics. See URL: https://www.kaggle.com/code/statlint/ibm-employee-attrition-analysis-by-numerics. Accessed Mar 2 2024.
[79] statsmodels. 2024. het_goldfeldquandt. See URL: https://www.statsmodels.org/dev/generated/statsmodels.stats.diagnostic.het_goldfeldquandt.html. Version 0.15.0, accessed Sept 1 2024.
[80] Bariş Sürücü. 2006. Goodness-of-fit tests for multivariate distributions. *Communications in Statistics-Theory and Methods* 35, 7 (2006), 1319–1331. doi:10.1080/03610920600628999
[81] Meltem Sonmez Turan. 2016. IID Testing in SP 800 90B. See URL: https://csrc.nist.gov/csrc/media/events/random-bit-generation-workshop-2016/documents/presentations/sessionii-3-meltem-sonmez-turan-presentation.pdf. Accessed 18/04/2025..
[82] Alexi Turcotte, Aviral Goel, Filip Křikava, and Jan Vitek. 2020. Designing types for R, empirically. *Proc. ACM Program. Lang.* 4, OOPSLA, Article 181 (nov 2020), 25 pages. doi:10.1145/3428249

[83] Alexi Turcotte and Jan Vitek. 2019. Towards a Type System for R. In *Proceedings of the 14th Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems* (London, United Kingdom) *(ICOOOLPS '19)*. Association for Computing Machinery, New York, NY, USA, Article 4, 5 pages. doi:10.1145/3340670.3342426

[84] JP Verma and Abdel-Salam G Abdel-Salam. 2019. *Testing statistical assumptions in research*. John Wiley & Sons.

[85] Chat Wacharamanotham, Krishna Subramanian, Sarah Theres Völkel, and Jan Borchers. 2015. Statsplorer: Guiding novices in statistical analysis. In *Proceedings of the 33rd annual acm conference on human factors in computing systems*. 2693–2702. doi:10.1145/2702123.2702347

[86] Larry Wasserman. 2004. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media.

[87] Rand R Wilcox. 2003. *Applying contemporary statistical techniques*. Elsevier.

[88] Jack Williams, J Garrett Morris, and Philip Wadler. 2018. The root cause of blame: contracts for intersection and union types. *Proceedings of the ACM on Programming Languages* 2, OOPSLA (2018), 1–29. doi:10.1145/3276504

[89] Bee Wah Yap and Chiaw Hock Sim. 2011. Comparisons of various types of normality tests. *Journal of Statistical Computation and Simulation* 81, 12 (2011), 2141–2155.

[90] Berna Yazici and Senay Yolacan. 2007. A comparison of various tests of normality. *Journal of statistical computation and simulation* 77, 2 (2007), 175–183. doi:10.1080/00949655.2010.520163

[91] Donald W Zimmerman. 1996. Some properties of preliminary tests of equality of variances in the two-sample location problem. *The Journal of General Psychology* 123, 3 (1996), 217–231. doi:10.1080/00221309.1996.9921274

Received 2024-09-13; accepted 2025-04-01