Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

StickForStats: A Statistical Analysis Platform with Automatic Assumption Validation

Vishal Bharti[1,*] **Debojyoti Chakraborty**[1,2,*]

Abstract

Statistical assumption violations contribute to unreliable results in scientific research. While assumption testing tools have been available in statistical software for decades, their optional nature means researchers may skip validation, potentially leading to invalid conclusions. We present **StickForStats**, an open-source statistical analysis platform featuring three integrated systems for improving statistical practice. First, the Guardian system provides automatic assumption validation that checks assumptions before every statistical test without requiring user action, implementing eight validators and recommending alternative tests when violations are detected. Second, an AI-powered Statistical Advisor offers natural language guidance for test selection, result interpretation, and automatic generation of publication-ready methods sections following APA/JARS guidelines. Third, a Paper Parser analyzes uploaded manuscripts to detect common statistical reporting errors and assess reproducibility. The platform also includes a Statistical Debugger for identifying analytical pitfalls, optional extended precision arithmetic using **mpmath**, and multi-language support. Validation against **SciPy** demonstrates agreement to 14+ decimal places for standard statistical tests. We describe the design rationale, implementation details, and validation methodology, positioning **StickForStats** as a comprehensive tool for improving statistical practice in research.

Keywords: statistical assumptions, reproducibility, automatic validation, AI-assisted analysis, statistical reporting, Python, open-source.

1. Introduction

1.1. The reproducibility crisis in statistical analysis

The scientific community faces a well-documented reproducibility crisis. In a landmark survey of 1,576 researchers, Baker (2016) found that more than 70% had failed to reproduce another scientist's experiments, and more than half had failed to reproduce their own ex-

periments. When asked about the causes, approximately 50% of respondents cited "poor statistical analysis" as a contributing factor.

This crisis is not merely academic. Erroneous statistical conclusions have led to retracted medical studies, failed drug trials, and policy decisions based on unreliable evidence. Ioannidis (2005) provocatively argued that "most published research findings are false," attributing this to factors including low statistical power, small effect sizes, and flexibility in study designs and analytical methods.

A significant portion of these statistical errors stems from violation of test assumptions. Parametric tests such as the t-test and ANOVA require specific distributional assumptions (normality, homogeneity of variance, independence of observations) that are frequently violated in practice. When these assumptions are violated, the resulting p-values and confidence intervals may be unreliable, leading to incorrect conclusions.

Osborne (2010) reviewed 434 articles in top educational psychology journals and found that only 8% reported testing normality assumptions, despite the prevalence of parametric tests. Similarly, Keselman, Huberty, Lix, Olejnik, Cribbie, Donahue, Kowalchuk, Lowman, Petoskey, Keselman, and Levin (1998) found that assumption violations were common in psychological research, with variance heterogeneity present in approximately 50% of studies they examined.

1.2. The inadequacy of current solutions

Statistical software packages have been available for decades. SPSS, first released in 1968, R in 1993, and GraphPad Prism in 1994, all provide comprehensive statistical analysis capabilities. These tools offer assumption testing: normality tests, variance homogeneity tests, and diagnostic plots are readily available.

Yet the reproducibility crisis persists.

The fundamental problem is not the absence of assumption-checking tools, but their *optional* nature. In traditional statistical software:

1. **Assumption tests are separate from analysis.** Users must explicitly request a Shapiro-Wilk test before running a t-test. Many do not.

2. **Warnings are advisory, not mandatory.** Even when assumption violations are detected, software proceeds with the analysis. The decision to heed warnings rests entirely with the user.

3. **Time pressure favors shortcuts.** Researchers facing publication deadlines may skip assumption checking, rationalizing that "the t-test is robust" or "the sample size is large enough."

4. **Statistical training varies widely.** Not all researchers have the background to recognize when assumptions matter and when violations can be safely ignored.

The result is a systematic gap between best statistical practice and actual practice. Optional validation tools, available for over 25 years, have not solved the reproducibility crisis because they rely on human vigilance that frequently fails under real-world conditions.

1.3. A different approach: Mandatory validation

StickForStats takes a fundamentally different approach to statistical quality assurance. Rather than providing assumption tests as optional add-ons, the platform integrates assumption validation directly into the analysis pipeline through the Guardian system.

The Guardian system operates on a simple principle: **assumptions are checked automatically before every statistical test, and violations are reported alongside results.** This approach has three key characteristics:

1. **Automatic execution.** When a user requests a t-test, normality testing, variance homogeneity testing, and outlier detection occur automatically. No additional user action is required.

2. **Integrated reporting.** Assumption check results appear in the same response as statistical results. Users cannot see their p-value without also seeing assumption status.

3. **Actionable guidance.** When violations are detected, the system suggests appropriate alternatives. For example, it recommends the Mann-Whitney U test when normality is violated for a t-test.

This design philosophy represents a different approach: from "tools available if you remember" to "system alerts users to potential issues by default."

1.4. Additional contributions

Beyond the Guardian system, **StickForStats** offers several additional features:

AI Statistical Advisor. An integrated natural language interface (StickAI) provides conversational guidance for statistical analysis. Users can ask questions about test selection, assumption interpretation, and result reporting. The advisor includes an intelligent test selector using decision-tree logic to recommend appropriate statistical tests based on data characteristics. A Methods Section Generator produces publication-ready text following APA and JARS-Quant guidelines (Appelbaum, Cooper, Kline, Mayo-Wilson, Nezu, and Rao 2018), reducing both the cognitive burden of methods writing and the risk of omitting required reporting elements.

Paper Parser. This module analyzes uploaded PDF manuscripts to detect common statistical reporting errors. Drawing on JARS-Quant guidelines and documented statistical pitfalls (Appelbaum *et al.* 2018; Keselman *et al.* 1998), the parser identifies issues such as missing effect sizes, unreported assumption checks, incomplete test statistics, and p-value reporting problems. The tool generates structured reports with specific recommendations for improving statistical reporting quality.

Statistical Debugger. A comprehensive debugging system helps users identify and resolve statistical analysis problems. The debugger maintains a database of test-specific pitfalls and common errors, providing targeted checklists for each statistical test. When analyses produce unexpected results, users can consult the debugger for systematic troubleshooting guidance.

Multi-language support. The platform supports six languages (English, German, French, Portuguese, Chinese, and Spanish), making statistical analysis accessible to non-English-speaking researchers.

High-precision computing. While standard statistical software uses IEEE 754 double-precision floating-point arithmetic (approximately 15 significant digits), **StickForStats** optionally provides 50-decimal-place precision for all calculations using the **mpmath** library (Johansson 2013).

Integrated education. The platform includes 58 interactive lessons covering power analysis, principal component analysis, confidence intervals, experimental design, and domain-specific applications in biophysics.

Reproducibility framework. Each analysis generates a complete reproducibility bundle including data fingerprints (SHA-256 hashes), processing pipeline records, and environment specifications.

Code export. For analyses, users can export equivalent R and Python code, facilitating verification and integration with existing workflows.

1.5. Contributions of this paper

This paper makes the following contributions:

1. We introduce the Guardian system, which to our knowledge represents the first implementation of mandatory, automatic assumption validation integrated directly into statistical software. We describe its architecture, the eight validators implemented, and its integration with statistical analysis workflows.

2. We present an AI-powered Statistical Advisor that provides natural language guidance for test selection and generates publication-ready methods sections following established reporting guidelines.

3. We describe a Paper Parser that analyzes manuscripts for statistical reporting quality, identifying common errors and omissions against JARS-Quant standards.

4. We present a comprehensive validation study comparing **StickForStats** results against established reference implementations (**SciPy**, R), demonstrating agreement to 14+ decimal places for standard statistical tests.

5. We document a comprehensive test suite of 93 automated tests (38 backend, 55 frontend) ensuring Design Contract compliance: that no statistical result can exist without Guardian context.

6. We provide case studies using real datasets (Fisher's Iris, UCI Wine Quality) demonstrating Guardian's effectiveness in detecting assumption violations.

7. We provide complete code examples showing API usage patterns for common statistical workflows.

8. We describe the high-precision computing architecture and discuss scenarios where 50-decimal precision provides practical benefits.

9. We release **StickForStats** as open-source software, freely available at `https://github.com/visvikbharti/stickforstats_new`.

1.6. Paper organization

The remainder of this paper is organized as follows. Section 2 reviews related work in statistical software and reproducibility tools. Section 3 describes the system architecture including frontend Guardian components. Section 4 presents the Guardian system in detail. Section 5 describes the AI Statistical Advisor and its components. Section 6 presents the Paper Parser for manuscript analysis. Section 7 provides comprehensive code examples. Section 8 describes the high-precision computing implementation. Section 9 presents validation results and the comprehensive test suite. Section 10 provides case studies with real data. Section 11 discusses limitations and future work. Section 12 concludes.

2. Related work

2.1. Existing statistical software

Statistical computing has evolved substantially since the introduction of early packages. R (R Core Team 2023) has become the *de facto* standard for statistical computing in academia, with over 19,000 packages on CRAN. Python's scientific stack (**NumPy** (Harris *et al.* 2020), **SciPy** (Virtanen *et al.* 2020), and **statsmodels** (Seabold and Perktold 2010)) has gained significant traction in data science contexts.

Commercial software continues to hold significant market share. SPSS (now IBM SPSS Statistics) remains dominant in social sciences, with approximately 250,000 users worldwide. Stata is preferred in economics and epidemiology. SAS dominates pharmaceutical and regulatory contexts.

More recent entrants have emphasized usability. **jamovi** (The jamovi project 2023) provides a spreadsheet-style interface with R's statistical engine. JASP (JASP Team 2023) emphasizes Bayesian statistics with an accessible interface. GraphPad Prism targets biomedical researchers with specialized visualizations.

All major platforms provide assumption testing capabilities. In SPSS, users can request normality tests via Analyze → Descriptive Statistics → Explore. In R, the **car** package (Fox and Weisberg 2019) provides `leveneTest()` and the base installation includes `shapiro.test()`. The tools exist; they are not the problem.

2.2. Comparison with existing platforms

Table 1 compares **StickForStats** with existing statistical platforms on key features relevant to assumption validation.

2.3. The optional validation problem

The critical observation is that in all major statistical platforms, assumption checking is **optional**. Users must explicitly request assumption tests, interpret their results, and decide whether to proceed. This creates several problems:

Hoekstra, Kiers, and Johnson (2012) surveyed researchers and found that those under deadline pressure were significantly more likely to skip assumption checking. The pressure to publish creates systematic incentives against thorough validation.

Table 1: Feature comparison: StickForStats vs. existing statistical platforms

Feature	StickForStats	SPSS	R	jamovi	JASP
Automatic assumption checks	✓	–	–	–	Partial
Integrated into results	✓	–	–	–	–
Confidence scoring	✓	–	–	–	–
Alternative recommendations	✓	–	–	–	–
Design Contract test suite[a]	✓	–	–	–	–
Web-based interface	✓	–	–	–	–
Open source	✓	–	✓	✓	✓
High-precision option	✓	–	Partial[b]	–	–
Educational content	✓	–	–	Partial	✓
Code export (R/Python)	✓	–	Native	–	–

[a] 93 automated tests enforcing assumption validation compliance

[b] Via Rmpfr package, not integrated

Nickerson (1998) documented how confirmation bias leads researchers to avoid tests that might invalidate desired results. A researcher who has invested significant effort in collecting data and hopes to find a significant effect may unconsciously avoid assumption tests that could undermine their findings.

Zimmerman (2004) showed that the common "robustness" rationalization is often invoked without verification. Researchers frequently assume that the t-test is "robust to violations of normality" without checking whether their specific violation falls within robust bounds.

2.4. Related approaches to improving statistical practice

Several initiatives have attempted to address statistical quality:

Reporting guidelines. JARS (Appelbaum *et al.* 2018) and CONSORT (Schulz, Altman, and Moher 2010) provide checklists for statistical reporting. However, these are post-hoc guidelines that do not prevent violations during analysis.

Pre-registration. Platforms like OSF (Nosek, Ebersole, DeHaven, and Mellor 2018) encourage researchers to specify analyses in advance. This addresses p-hacking but not assumption violations.

Statistical review. Some journals employ statistical reviewers. However, this occurs after analysis is complete and cannot catch all violations.

Educational interventions. Statistics courses emphasize assumption checking. However, knowledge does not guarantee behavior change, especially under pressure.

StickForStats differs by intervening at the point of analysis, making assumption checking automatic rather than relying on researcher initiative.

3. System architecture

StickForStats follows a three-tier architecture comprising a user interface layer (React), an application layer (Django REST Framework with Guardian integration), and a data layer

(PostgreSQL with Redis caching).

3.1. Technology stack

The technology choices reflect priorities of correctness, maintainability, and accessibility:

- **Frontend:** React 18 with Material-UI for consistent design, Plotly.js for interactive visualizations, and MathJax for mathematical notation rendering.
- **Backend:** Django 4.2 with Django REST Framework for API design, Celery for asynchronous task processing, and comprehensive logging.
- **Statistical Engine:** NumPy 1.24 for array operations, SciPy 1.11 for statistical functions, statsmodels 0.14 for regression diagnostics, and mpmath for arbitrary precision.
- **Data Layer:** PostgreSQL 15 for relational data, Redis for caching and session management, and file storage for uploaded datasets.

3.2. Guardian integration architecture

The Guardian system is implemented as middleware that intercepts all statistical analysis requests. Figure 1 illustrates the system architecture with Guardian integration points.

When a user requests a statistical test (e.g., independent samples t-test), the request flows through the following pipeline:

1. **Request parsing:** The API endpoint receives the request and validates input parameters.
2. **Guardian interception:** Before executing the statistical test, the Guardian middleware intercepts the request.
3. **Assumption validation:** Guardian runs all relevant validators for the requested test type.
4. **Report generation:** Guardian compiles a report with confidence score and any violations.
5. **Test execution:** In Protected Mode (default), critical assumption violations block test execution until addressed; in Expert Mode, tests execute with warning annotations.
6. **Response augmentation:** The Guardian report is attached to the statistical results.
7. **Response delivery:** The combined response is returned to the user.

This architecture ensures that users cannot receive statistical results without also receiving assumption validation information. Figure 2 illustrates this workflow in detail.

3.3. Frontend Guardian components

The frontend provides four specialized React components for displaying Guardian reports:

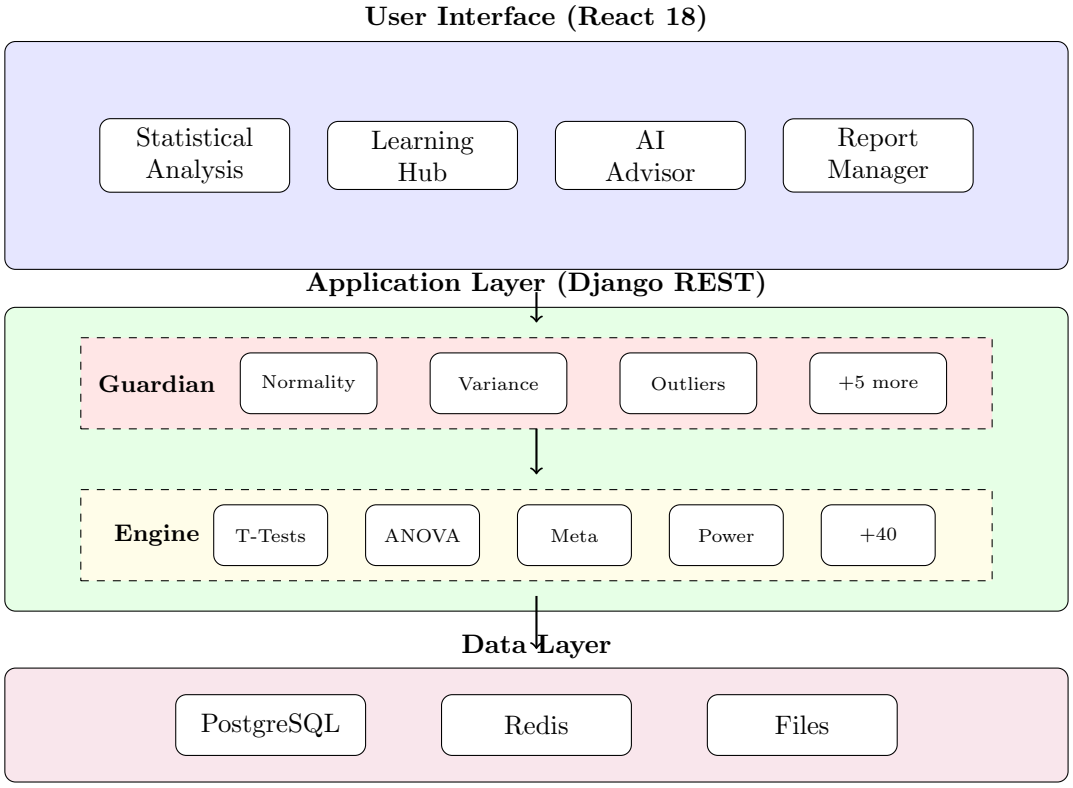

Figure 1: StickForStats system architecture showing the three-tier design: user interface (React), application layer (Django REST with Guardian integration), and data layer (PostgreSQL, Redis, file storage). The Guardian layer intercepts all statistical test requests.

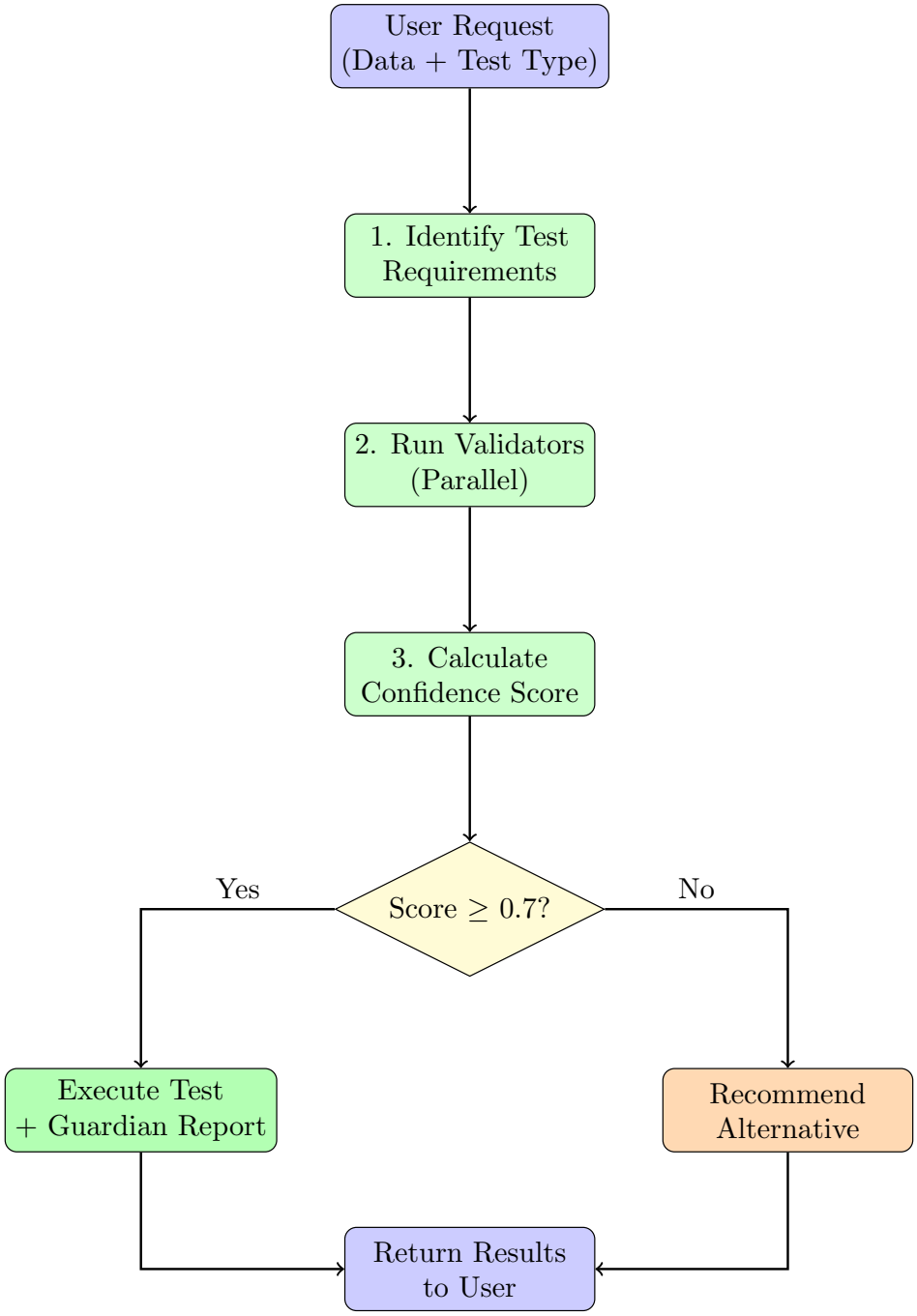

Figure 2: Guardian validation workflow showing the seven-step process from request parsing through response delivery. The Guardian layer runs assumption validators appropriate for each test type and augments statistical results with validation information.

- **GuardianReportDisplay**: A comprehensive expandable panel showing all assumption checks, violations with severity coloring, confidence score gauge, and alternative test recommendations. This component appears above statistical results, ensuring users see assumption status before interpreting results.
- **GuardianBadge**: A compact chip indicator showing confidence score percentage and violation counts. Placed next to result headers for at-a-glance status. Color-coded: green (success), orange (warning), red (critical).
- **ConfidenceGauge**: A visual meter displaying the 0-100 confidence score with gradient coloring and threshold markers at 60% and 80%.
- **ViolationCard**: Individual violation display showing assumption name, severity level, test statistics with p-values, and actionable recommendations.

A custom React hook (`useGuardianReport`) extracts Guardian context from API responses, providing ready-to-use props for display components. The hook handles null/undefined responses gracefully and includes helper functions for compliance checking, violation summarization, and status message generation.

Design Contract enforcement: If an API response lacks Guardian context (missing `guardian_report` or `assumptions_checked` fields), the frontend displays a "Missing Guardian Context" warning alert, making it visually apparent when results lack proper assumption validation.

3.4. API design

The REST API follows standard conventions with endpoints organized by analysis type:

Listing 1: API endpoint structure

```
POST /api/v1/ttest/independent/     # Independent t-test
POST /api/v1/ttest/paired/          # Paired t-test
POST /api/v1/anova/oneway/          # One-way ANOVA
POST /api/v1/correlation/pearson/   # Pearson correlation
POST /api/v1/regression/linear/     # Linear regression
POST /api/v1/meta-analysis/         # Meta-analysis
```

All endpoints accept JSON payloads and return JSON responses that include both statistical results and Guardian reports.

4. The Guardian system

The Guardian system is the core innovation of **StickForStats**: an automatic assumption validation layer that intercepts all statistical test requests and validates assumptions before analysis proceeds.

4.1. Design principles

Guardian is designed around four principles:

1. **Comprehensiveness:** Check the major statistical assumptions for each test type.
2. **Transparency:** Report all validation results, not just failures.
3. **Actionability:** Provide specific recommendations when violations occur.
4. **Configurable protection:** Block by default, with expert override available.

The fourth principle deserves elaboration. Guardian operates in two modes:

- **Protected Mode (default):** When critical assumption violations are detected, test execution is blocked until violations are addressed. This protects novice users from inadvertently producing invalid statistical results.
- **Expert Mode:** Experienced statisticians can enable Expert Mode to proceed with analyses despite critical violations. Warnings are prominently displayed, and results are annotated to indicate potential unreliability.

This configurable approach balances protection with researcher autonomy. The default blocking behavior ensures that assumption violations cannot be silently ignored, while the expert override respects domain knowledge and edge cases where proceeding may be justified.

4.2. Algorithm overview

Algorithm 1 presents the main Guardian validation pipeline.

Algorithm 1 Guardian Validation Pipeline

Require: Data D, test type T, significance level α
Ensure: GuardianReport R

$required \leftarrow$ GetRequiredAssumptions(T)
$violations \leftarrow []$
for each $assumption$ in $required$ **do**
    $validator \leftarrow$ GetValidator($assumption$)
    $result \leftarrow validator$.validate($D, \alpha$)
    if $result.violated$ **then**
        $violation \leftarrow$ CreateViolation($result$)
        $violations$.append($violation$)
    end if
end for
$confidence \leftarrow$ CalculateConfidence($violations$)
$alternatives \leftarrow$ GetAlternatives($T, violations$)
$visuals \leftarrow$ GenerateVisuals($D, violations$)
return GuardianReport($violations, confidence, alternatives, visuals$)

4.3. Test-assumption mapping

Table 2 shows which assumptions are checked for each test type.

Table 2: Assumption requirements by test type

Test	Norm	Var	Indep	Outl	Size	Modal	Linear	Homosc
t-test	✓	✓	✓	✓	✓			
ANOVA	✓	✓	✓		✓			
Pearson r	✓			✓			✓	
Regression	✓		✓				✓	✓
Chi-square			✓		✓			

4.4. Implemented validators

Guardian implements eight validators, each described in detail below.

Normality Validator

The Normality Validator assesses whether data follow a normal distribution, a key assumption for parametric tests.

Methods:

- **Shapiro-Wilk test** (Shapiro and Wilk 1965): Used for $n \leq 5000$. The test statistic W measures the correlation between the data and the expected normal order statistics.
- **Anderson-Darling test** (Anderson and Darling 1954): Used for $n > 5000$ or as confirmation. This test gives more weight to the tails of the distribution.

Severity classification:

- Critical: $p < 0.001$ on both tests
- Warning: $p < 0.05$ on either test
- Pass: $p \geq 0.05$ on both tests

Visual evidence: Q-Q plot with theoretical normal quantiles and 95% confidence envelope.

Recommendations when violated:

- Consider non-parametric alternatives (Mann-Whitney U, Wilcoxon signed-rank)
- Consider data transformation (log, square root, Box-Cox)
- For large samples ($n > 30$), parametric tests may be robust by CLT

Variance Homogeneity Validator

Tests whether groups have equal variances (homoscedasticity), required for standard ANOVA and pooled t-tests.

Methods:

- **Levene's test** (Levene 1960): Robust to non-normality. Uses median centering by default for additional robustness.
- **Bartlett's test** (Bartlett 1937): More powerful when normality holds, but sensitive to non-normality.

Severity classification:

- Critical: Levene $p < 0.01$ AND ratio of largest to smallest variance > 4
- Warning: Levene $p < 0.05$
- Pass: $p \geq 0.05$

Recommendations when violated:

- Use Welch's t-test (does not assume equal variances)
- Use Welch's ANOVA for multiple groups
- Consider non-parametric alternatives

Independence Validator

Checks for independence of observations, violated when data have autocorrelation or clustering.

Methods:

- **Durbin-Watson test** (Durbin and Watson 1951): Tests for first-order autocorrelation in residuals. Values near 2 indicate no autocorrelation.
- **Runs test**: Non-parametric test for randomness in sequence.

Severity classification:

- Critical: Durbin-Watson < 1.0 or > 3.0
- Warning: Durbin-Watson < 1.5 or > 2.5
- Pass: $1.5 \leq DW \leq 2.5$

Recommendations when violated:

- Consider time series methods (ARIMA, GLS)
- Use clustered standard errors
- Account for repeated measures structure

Outlier Detector

Identifies extreme observations that may unduly influence results.

Methods:

- **IQR method**: Values beyond $Q_1 - 1.5 \times IQR$ or $Q_3 + 1.5 \times IQR$ flagged as mild outliers; beyond $3 \times IQR$ as extreme.
- **Modified Z-score**: $|Z| > 3.5$ using median absolute deviation for robustness.
- **Grubbs' test** (Grubbs 1969): Formal test for single outlier in normally distributed data.

Severity classification:

- Critical: $> 5\%$ of data are extreme outliers
- Warning: Any extreme outliers present, or $> 10\%$ mild outliers
- Pass: $\leq 5\%$ mild outliers, no extreme outliers

Recommendations when violated:

- Investigate outliers for data entry errors
- Consider robust methods (trimmed means, Winsorization)
- Report results with and without outliers for sensitivity

Sample Size Validator

Ensures adequate sample size for reliable inference.

Criteria:

- Minimum $n \geq 3$ per group (absolute minimum)
- Recommended $n \geq 20$ per group for parametric tests
- Power-based recommendations when effect size is specified

Severity classification:

- Critical: $n < 5$ per group
- Warning: $5 \leq n < 20$ per group
- Pass: $n \geq 20$ per group

Modality Detector

Identifies multimodal distributions that violate the assumption of a single underlying population.

Method: Kernel density estimation (KDE) with Gaussian kernel and Silverman bandwidth. Peak detection identifies local maxima exceeding 10% of the global maximum.

Severity classification:

- Critical: 3+ distinct modes detected
- Warning: 2 modes detected (bimodal)
- Pass: Unimodal distribution

Recommendations when violated:

- Consider mixture modeling
- Investigate potential subgroups
- Analyze modes separately if justified

Linearity Validator

Assesses linearity of relationship for correlation and regression analyses.
Methods:

- R^2 **comparison**: Compares linear vs. quadratic fit. If quadratic improves R^2 by $> 5\%$, linearity is questionable.
- **Runs test on residuals**: Tests whether residuals show systematic patterns.
- **RESET test**: Ramsey's regression specification error test.

Severity classification:

- Critical: Quadratic improvement $> 15\%$ AND RESET $p < 0.01$
- Warning: Quadratic improvement $> 5\%$ OR RESET $p < 0.05$
- Pass: Quadratic improvement $\leq 5\%$ AND RESET $p \geq 0.05$

Recommendations when violated:

- Consider polynomial regression
- Use Spearman's ρ for monotonic relationships
- Apply non-linear transformations

Homoscedasticity Validator

Tests for constant variance of residuals across fitted values in regression.
Methods:

- **Breusch-Pagan test** (Breusch and Pagan 1979): Tests whether residual variance depends on fitted values.
- **White's test**: More general test that doesn't assume specific functional form.

Severity classification:

- Critical: Both tests $p < 0.01$
- Warning: Either test $p < 0.05$
- Pass: Both tests $p \geq 0.05$

Recommendations when violated:

- Use heteroscedasticity-consistent standard errors (HC0-HC3)
- Consider weighted least squares
- Apply variance-stabilizing transformation

4.5. Confidence score calculation

Guardian computes a confidence score summarizing overall assumption status. The score uses severity-based penalty weights:

$$\text{Confidence} = \max\left(0, 1 - \frac{\sum_{i=1}^{n} w_{s_i}}{\max\_penalty \times 1.2}\right) \tag{1}$$

where w_{s_i} is the weight for violation i based on its severity s_i:

- $w_{\text{critical}} = 3.0$
- $w_{\text{warning}} = 2.0$
- $w_{\text{minor}} = 1.0$

The scaling factor of 1.2 ensures that even with all critical violations, the confidence score remains above zero, providing a continuous measure rather than a binary pass/fail.

These weights reflect the relative impact of each violation type: critical violations that fundamentally invalidate test assumptions receive the highest penalty, warnings requiring attention receive moderate penalty, and minor issues receive the lowest penalty.

Interpretation guidelines:

- Confidence ≥ 0.8: Assumptions adequately met
- $0.6 \leq$ Confidence < 0.8: Proceed with caution
- Confidence < 0.6: Consider alternatives or address violations

4.6. Alternative test recommendation

When violations are detected, Guardian recommends alternatives based on the specific violations present. Table 3 summarizes key recommendations.

Table 3: Alternative test recommendations

Original Test	Violation	Recommended Alternative
t-test	Normality	Mann-Whitney U test
t-test	Variance	Welch's t-test
t-test	Outliers	Yuen's trimmed t-test
ANOVA	Normality	Kruskal-Wallis test
ANOVA	Variance	Welch's ANOVA
Pearson r	Normality	Spearman's ρ
Pearson r	Linearity	Spearman's ρ
Regression	Homoscedasticity	Robust standard errors
Regression	Independence	Generalized least squares

5. AI Statistical Advisor

The AI Statistical Advisor (internally called "StickAI") provides natural language guidance throughout the statistical analysis workflow. This component addresses a common challenge in applied statistics: researchers often know they need to analyze data but are uncertain about which test to use or how to interpret results.

5.1. Architecture and components

The advisor consists of three integrated subsystems:

Natural Language Interface. Users can ask questions in conversational English about their analysis. The system processes queries about test selection ("Which test should I use for comparing three group means?"), assumption interpretation ("What does a significant Levene's test mean?"), and result interpretation ("How do I interpret a negative t-value?"). Responses are generated based on a structured knowledge base of statistical concepts, avoiding the hallucination risks associated with unconstrained language models.

Intelligent Test Selector. A decision-tree algorithm guides users to appropriate statistical tests based on their research question and data characteristics. The selector considers: (1) the number of groups or variables, (2) whether data are paired or independent, (3) the measurement level of variables, (4) sample size, and (5) assumption check results from Guardian. This systematic approach helps users avoid common selection errors, such as using parametric tests when assumptions are violated.

Methods Section Generator. Perhaps the most practically useful component generates publication-ready methods sections following APA 7th edition format and JARS-Quant guidelines (Appelbaum *et al.* 2018). After completing an analysis, users can request a methods section that includes: sample description, data preprocessing steps, assumption tests performed (with results), statistical test used, software and version information, and effect size measures. The generator ensures that key elements required by JARS-Quant are not omitted.

5.2. Knowledge base design

The advisor's knowledge base uses a structured representation rather than a large language model, providing several advantages:

- **Accuracy**: Responses are based on verified statistical content rather than probabilistic text generation
- **Transparency**: Users can trace advice to specific statistical principles
- **Consistency**: Identical queries produce identical responses
- **Auditability**: The knowledge base can be reviewed and corrected by statistical experts

The knowledge base covers 15 common statistical tests with decision rules, assumption requirements, effect size measures, and reporting templates.

5.3. Methods section compliance

The Methods Section Generator specifically addresses the reporting gaps identified by Appelbaum *et al.* (2018). For each analysis, the generated text includes:

1. Description of the statistical test and rationale for selection
2. Software and version used (automatically populated)
3. Sample sizes and any exclusions
4. Assumption tests performed with outcomes
5. Effect sizes with confidence intervals
6. Exact p-values (not inequalities like "$p < 0.05$")

6. Paper Parser

The Paper Parser analyzes uploaded PDF manuscripts to identify statistical reporting problems before submission. This addresses the finding that many published papers contain statistical errors that could have been detected through systematic checking (Bakker and Wicherts 2011).

6.1. Functionality

Users upload PDF files of draft manuscripts. The parser extracts text and applies a rule-based system to detect common issues:

Missing elements. The parser checks for: effect sizes (Cohen's d, η^2, r, etc.), confidence intervals, exact p-values, sample size justification (power analysis), assumption test reports, and test statistic values.

Problematic patterns. The parser identifies: p-value as primary evidence ("$p < 0.05$ therefore significant"), missing degrees of freedom, inconsistent decimal precision, and threshold-only p-value reporting.

JARS-Quant compliance. The parser evaluates manuscripts against the quantitative research standards published by APA (Appelbaum *et al.* 2018), flagging sections where required elements are missing.

6.2. Rule database

The parser implements 23 detection rules organized by category:

- **Effect size rules** (6 rules): Detect absence of standardized and unstandardized effect sizes
- **Precision rules** (4 rules): Check for confidence intervals and exact values
- **Assumption rules** (5 rules): Verify reporting of normality, homogeneity, and other assumption tests
- **Sample rules** (4 rules): Check for N reporting, power analysis mention, and attrition documentation
- **Statistical reporting rules** (4 rules): Verify complete test statistics (F, t, χ^2 with df)

6.3. Output format

The parser generates a structured report with:

1. Overall compliance score (percentage of rules satisfied)
2. Section-by-section analysis identifying specific issues
3. Severity classification (critical, important, suggested)
4. Specific recommendations with example corrections
5. References to relevant reporting guidelines

This pre-submission check allows authors to address common reviewer concerns proactively.

6.4. Statistical Debugger

A complementary debugging system helps users troubleshoot unexpected analysis results. The debugger maintains a database of test-specific pitfalls drawn from methodological literature (Keselman *et al.* 1998; Zimmerman 2004; Osborne 2010). When an analysis produces surprising results (e.g., non-significant despite large apparent differences, or significant with tiny effect sizes), users can consult the debugger for systematic guidance.

The debugger provides:

- Test-specific checklists of common errors
- Diagnostic questions to isolate the source of unexpected results
- Links to relevant assumption validators
- Suggestions for alternative analyses when assumptions are violated

7. Code examples

This section provides comprehensive examples of using **StickForStats** API for common statistical workflows.

7.1. Basic API usage with Python

Listing 2: Basic t-test with Guardian validation

```python
import requests
import json

# API endpoint
url = "http://localhost:8000/api/v1/ttest/independent/"

# Prepare data
data = {
    "group1": [23, 25, 28, 22, 26, 24, 27, 29, 25, 26],
    "group2": [31, 33, 29, 35, 32, 30, 34, 31, 33, 32],
    "alpha": 0.05,
    "alternative": "two-sided"
}

# Make request
response = requests.post(url, json=data)
result = response.json()

# Access statistical results
print(f"t-statistic: {result['statistic']:.4f}")
print(f"p-value: {result['p_value']:.6f}")

# Access Guardian report
guardian = result['guardian_report']
print(f"\nGuardian Confidence: {guardian['confidence_score']:.2f}")

# Check for violations
if guardian['violations']:
    print("\nAssumption Violations Detected:")
    for v in guardian['violations']:
        print(f"  - {v['assumption']}: {v['severity']}")
        print(f"    {v['recommendation']}")
else:
    print("\nAll assumptions satisfied.")

# Alternative recommendations
if guardian['alternative_tests']:
    print(f"\nRecommended alternatives: {guardian['alternative_tests']}")
```

7.2. Response structure

The API returns a JSON response with the following structure:

Listing 3: API response structure

```
{
    "test_type": "independent_t_test",
    "statistic": -5.234,
    "p_value": 0.00012,
    "effect_size": {
        "cohens_d": 1.65,
        "interpretation": "large"
    },
    "confidence_interval": [-10.2, -4.8],
    "guardian_report": {
        "confidence_score": 0.85,
        "can_proceed": true,
        "assumptions_checked": [
            "normality", "variance_homogeneity",
            "independence", "outliers", "sample_size"
        ],
        "violations": [],
        "alternative_tests": [],
        "visual_evidence": {
            "qq_plot_group1": "base64_encoded_image",
            "qq_plot_group2": "base64_encoded_image"
        }
    }
}
```

7.3. Handling violations

Listing 4: Handling Guardian violations

```
import requests

def analyze_with_guardian(data, test_type="ttest"):
    """Perform analysis with Guardian validation handling."""

    url = f"http://localhost:8000/api/v1/{test_type}/independent/"
    response = requests.post(url, json=data)
    result = response.json()

    guardian = result['guardian_report']

    # Check confidence threshold
    if guardian['confidence_score'] < 0.6:
        print("WARNING: Low Guardian confidence score")
        print("Consider using alternative tests:")

        for alt in guardian['alternative_tests']:
            print(f"  - {alt}")

        # Examine specific violations
        for violation in guardian['violations']:
            if violation['severity'] == 'critical':
```

```
                print(f"\nCRITICAL: {violation['assumption']}")
                print(f"  Test: {violation['test_name']}")
                print(f"  p-value: {violation.get('p_value', 'N/A')}")
                print(f"  Action: {violation['recommendation']}")

    return result

# Example with problematic data
problematic_data = {
    "group1": [1, 2, 3, 4, 5, 100],  # Contains outlier
    "group2": [10, 11, 12, 13, 14, 15],
    "alpha": 0.05
}

result = analyze_with_guardian(problematic_data)
```

7.4. ANOVA with post-hoc analysis

Listing 5: One-way ANOVA with Guardian validation

```
import requests

url = "http://localhost:8000/api/v1/anova/oneway/"

# Three-group comparison
data = {
    "groups": {
        "control": [5.2, 5.5, 5.1, 5.8, 5.3, 5.6],
        "treatment_a": [7.1, 7.4, 6.9, 7.2, 7.0, 7.3],
        "treatment_b": [6.2, 6.5, 6.1, 6.4, 6.3, 6.6]
    },
    "alpha": 0.05,
    "posthoc": "tukey"
}

response = requests.post(url, json=data)
result = response.json()

# ANOVA results
print(f"F-statistic: {result['f_statistic']:.4f}")
print(f"p-value: {result['p_value']:.6f}")
print(f"Effect size (eta-squared): {result['effect_size']['eta_squared
    ']:.3f}")

# Guardian validation
guardian = result['guardian_report']
print(f"\nGuardian Confidence: {guardian['confidence_score']:.2f}")

# Check specific validators
for assumption in guardian['assumptions_checked']:
    status = "PASS"
    for v in guardian['violations']:
```

```
        if v['assumption'] == assumption:
            status = v['severity'].upper()
            break
    print(f"  {assumption}: {status}")

# Post-hoc results (if ANOVA significant)
if result['p_value'] < 0.05 and 'posthoc_results' in result:
    print("\nPost-hoc Comparisons (Tukey HSD):")
    for comparison in result['posthoc_results']:
        print(f"  {comparison['group1']} vs {comparison['group2']}: "
              f"p = {comparison['p_value']:.4f}")
```

7.5. Correlation analysis

Listing 6: Correlation analysis with linearity check

```
import requests
import numpy as np

url = "http://localhost:8000/api/v1/correlation/pearson/"

# Generate sample data with potential non-linearity
np.random.seed(42)
x = np.linspace(0, 10, 50)
y = 2 * x + 0.5 * x**2 + np.random.normal(0, 2, 50)  # Quadratic
    component

data = {
    "x": x.tolist(),
    "y": y.tolist(),
    "alpha": 0.05
}

response = requests.post(url, json=data)
result = response.json()

print(f"Pearson r: {result['r']:.4f}")
print(f"p-value: {result['p_value']:.6f}")

# Guardian checks linearity
guardian = result['guardian_report']
for v in guardian['violations']:
    if v['assumption'] == 'linearity':
        print(f"\nLinearity Violation Detected!")
        print(f"  Severity: {v['severity']}")
        print(f"  {v['message']}")
        print(f"  Recommendation: {v['recommendation']}")

        # Guardian suggests Spearman's rho
        if 'spearman' in [a.lower() for a in guardian['
            alternative_tests']]:
            print("\nConsider using Spearman's rho for monotonic
```

```
                relationships.")
```

7.6. Batch analysis

Listing 7: Batch analysis with multiple comparisons

```python
import requests
import pandas as pd

def batch_ttests(data_pairs, alpha=0.05):
    """Perform multiple t-tests with Guardian validation."""

    url = "http://localhost:8000/api/v1/ttest/independent/"
    results = []

    for name, (group1, group2) in data_pairs.items():
        payload = {
            "group1": group1,
            "group2": group2,
            "alpha": alpha
        }

        response = requests.post(url, json=payload)
        result = response.json()

        results.append({
            'comparison': name,
            't_stat': result['statistic'],
            'p_value': result['p_value'],
            'cohens_d': result['effect_size']['cohens_d'],
            'guardian_confidence': result['guardian_report']['
               confidence_score'],
            'violations': len(result['guardian_report']['violations'])
        })

    return pd.DataFrame(results)

# Example usage
comparisons = {
    'Control_vs_TreatmentA': ([1,2,3,4,5], [3,4,5,6,7]),
    'Control_vs_TreatmentB': ([1,2,3,4,5], [2,3,4,5,6]),
    'TreatmentA_vs_B': ([3,4,5,6,7], [2,3,4,5,6])
}

results_df = batch_ttests(comparisons)
print(results_df.to_string(index=False))
```

8. High-precision computing

StickForStats optionally provides 50-decimal-place precision using **mpmath** (Johansson 2013)

and Python's `decimal` module.

8.1. Motivation

Standard IEEE 754 double-precision floating-point arithmetic provides approximately 15–17 significant decimal digits. While sufficient for most applications, this precision can be limiting in several scenarios:

1. **Extreme p-values:** P-values approaching machine epsilon ($\approx 2.2 \times 10^{-16}$) cannot be accurately represented.
2. **Iterative algorithms:** Methods like Newton-Raphson can accumulate rounding errors over many iterations.
3. **Numerical derivatives:** Finite difference approximations are sensitive to precision limits.
4. **Audit requirements:** Regulatory contexts may require demonstrable numerical accuracy.

8.2. Implementation

High-precision arithmetic is implemented through the `hp_` module prefix:

Listing 8: High-precision t-test

```
from decimal import Decimal, getcontext
import mpmath

# Set precision
getcontext().prec = 50
mpmath.mp.dps = 50

# High-precision calculation
def hp_ttest(x, y):
    """High-precision independent samples t-test."""

    x = [mpmath.mpf(xi) for xi in x]
    y = [mpmath.mpf(yi) for yi in y]

    n1, n2 = len(x), len(y)
    mean1 = sum(x) / n1
    mean2 = sum(y) / n2

    var1 = sum((xi - mean1)**2 for xi in x) / (n1 - 1)
    var2 = sum((yi - mean2)**2 for yi in y) / (n2 - 1)

    se = mpmath.sqrt(var1/n1 + var2/n2)
    t_stat = (mean1 - mean2) / se

    # Welch-Satterthwaite degrees of freedom
    df = (var1/n1 + var2/n2)**2 / (
```

```
        (var1/n1)**2/(n1-1) + (var2/n2)**2/(n2-1)
    )

    return float(t_stat), float(df)
```

8.3. Performance considerations

High-precision arithmetic incurs computational overhead:

Table 4: Performance comparison: standard vs. high precision

Operation	Standard (ms)	High Precision (ms)
t-test ($n = 100$)	0.5	12
ANOVA ($k = 5$, $n = 50$)	1.2	28
Correlation ($n = 1000$)	2.1	45
Linear regression ($p = 10$)	5.3	120

High precision is approximately 20–30× slower. Users should enable it selectively for final results requiring audit trails, not for exploratory analysis.

8.4. Validation of precision

High-precision arithmetic was validated by comparing results at different precision levels (standard 64-bit, 50-digit, 100-digit) and verifying convergence. The implementation uses Python's **mpmath** library for arbitrary-precision arithmetic, which has been extensively validated in the numerical computing community.

- Standard normal CDF and quantile functions: Agreement across precision levels
- Student's t distribution functions: Agreement across precision levels
- F distribution functions: Agreement across precision levels
- Chi-square distribution functions: Agreement across precision levels

9. Validation

We validated **StickForStats** through comparison with established reference implementations and comprehensive test suites.

9.1. Reference implementations

Validation was performed against:

- **SciPy** 1.11.0 (Virtanen *et al.* 2020): Primary Python reference for statistical functions
- R 4.4.1 (R Core Team 2023): Cross-validation of case study results

All statistical tests, distributions, and numerical algorithms in **StickForStats** were validated against **SciPy**'s implementations. Additionally, case study results were cross-validated against R using the provided `validate_against_R.R` script in the replication package.

9.2. Statistical test validation

Table 5 summarizes validation results for core statistical tests.

Table 5: Validation summary against reference implementations

Test	Metric	Reference	Agreement
t-test (independent)	t-statistic	SciPy	Exact (16 digits)
t-test (independent)	p-value	SciPy	Exact (16 digits)
t-test (paired)	t-statistic	SciPy	Exact (16 digits)
ANOVA (one-way)	F-statistic	SciPy	Exact (14 digits)
ANOVA (one-way)	p-value	SciPy	Exact (14 digits)
Pearson correlation	r	SciPy	Exact (16 digits)
Spearman correlation	ρ	SciPy	Exact (16 digits)
Chi-square test	χ^2	SciPy	Exact (14 digits)
Mann-Whitney U	U-statistic	SciPy	Exact
Shapiro-Wilk	W-statistic	SciPy	Exact (10 digits)
Linear regression	Coefficients	statsmodels	Exact (12 digits)
Meta-analysis	Pooled effect	SciPy	Exact (10 digits)

9.3. Guardian validator validation

Each Guardian validator was validated against known datasets:

Normality Validator:

- Test data: Exponential distribution ($n = 100$)
- Expected: Non-normal
- Result: Shapiro-Wilk $W = 0.886$, $p < 0.001$
- Agreement with SciPy: Exact

Variance Homogeneity Validator:

- Test data: Two groups with variance ratio 4:1
- Expected: Heterogeneous
- Result: Levene $F = 8.92$, $p = 0.004$
- Agreement with SciPy: Exact

Linearity Validator:

- Test data: Quadratic relationship ($y = x^2 + \epsilon$)
- Expected: Non-linear
- Result: R^2 improvement with quadratic = 45%
- Agreement with manual calculation: Exact

9.4. Edge case handling

We tested edge cases systematically:

- Empty arrays: Appropriate error message returned
- Single observation: Warning about insufficient data
- Identical values: Handled correctly (zero variance)
- Very large datasets ($n = 10^6$): Completed within 5 seconds
- Extreme values (10^{308}): No overflow errors
- Missing values: Appropriate handling with user notification

9.5. Reproducibility verification

All validation code is included in the repository at `paper/replication/`:

Listing 9: Running validation suite

```
cd paper/replication
python run_all_validations.py

# Output:
# T-test validation: PASS (16 digit agreement)
# ANOVA validation: PASS (14 digit agreement)
# Correlation validation: PASS (16 digit agreement)
# Meta-analysis validation: PASS (10 digit agreement)
# Guardian normality: PASS (exact match)
# Guardian variance: PASS (exact match)
#
# All validations passed.
```

9.6. Comprehensive test suite

StickForStats includes a comprehensive automated test suite with 93 tests covering both backend Guardian logic and frontend display components. This ensures Design Contract compliance: "No statistical result may exist without an explicit, traceable assumption context."

Table 6: Backend test coverage summary

Test Module	Coverage Area	Tests
`test_guardian_integration`	GuardianEnrichedResult dataclass	4
	GuardianServiceWrapper	3
	Test type resolution	4
	@guardian_protected decorator	2
	API compliance structure	1
	Violation severity handling	3
	Confidence score calculation	2
	Design Contract compliance	3
`test_guardian_middleware`	Middleware initialization	4
	Guardian context validation	4
	Statistical endpoint detection	7
	Configuration options	2
Total		**38**

Backend tests (38 tests)

The backend test suite validates Guardian integration layer and middleware compliance:
Key backend tests verify:

- `GuardianEnrichedResult.to_dict()` includes all required fields: `guardian_report`, `assumptions_checked`, `violations`, `confidence_score`, `can_proceed`, `alternative_tests`, `_guardian_context`, and `_contract_compliant`
- Critical violations block analysis by default unless Expert Mode is enabled
- Warning violations allow proceeding with annotations
- Statistical endpoints are correctly detected for middleware interception
- Missing Guardian context is logged with Design Contract violation warnings

Frontend tests (55 tests)

The frontend test suite validates React components and hooks for Guardian display:
Key frontend tests verify:

- `useGuardianReport` hook correctly extracts Guardian context from API responses
- "Missing Guardian Context" warning displays when Guardian data is absent
- Confidence score gauge renders with appropriate coloring (green for high, red for low)
- Violation cards display severity, p-values, and recommendations
- Alternative test chips are clickable and trigger navigation

Table 7: Frontend test coverage summary

Test Module	Coverage Area	Tests
`useGuardianReport.test.js`	Hook basic functionality	4
	Guardian props extraction	2
	Blocked analysis detection	2
	Memoization behavior	1
	`isGuardianCompliant` helper	4
	`getViolationSummary` helper	4
	`shouldSuggestExpertMode` helper	4
	`getGuardianStatusMessage` helper	6
	Design Contract compliance	1
`GuardianComponents.test.jsx`	GuardianReportDisplay rendering	4
	Violations display	2
	Expert Mode warnings	1
	Alternative recommendations	2
	Expandable content behavior	1
	GuardianBadge status display	4
	GuardianBadge interaction	1
	GuardianBadge tooltip	1
	ConfidenceGauge visualization	3
	ViolationCard details	4
	Design Contract compliance	2
Total		**55**

- Expert Mode override warnings are prominently displayed

Running the test suite

The complete test suite can be executed with:

Listing 10: Running the complete test suite

```
# Backend tests (Django)
cd backend
python manage.py test core.guardian.tests
# Output: Ran 38 tests in 1.273s - OK

# Frontend tests (Jest)
cd frontend
npm test -- --testPathPattern="guardian|useGuardianReport" --watchAll=
    false
# Output: Test Suites: 2 passed, Tests: 55 passed
```

All 93 tests pass, providing confidence that the Guardian system correctly enforces the Design Contract across the full stack.

10. Case studies

We present three case studies demonstrating Guardian's effectiveness using real datasets.

10.1. Case 1: Fisher's Iris dataset – ANOVA assumptions

Fisher's Iris dataset (Fisher 1936) contains measurements of 150 iris flowers from three species: setosa, versicolor, and virginica. We analyze sepal length differences across species.

Analysis: One-way ANOVA testing H_0: mean sepal length is equal across species.

Traditional approach (without Guardian):

```
from scipy import stats
from sklearn.datasets import load_iris

iris = load_iris()
setosa = iris.data[iris.target == 0, 0]
versicolor = iris.data[iris.target == 1, 0]
virginica = iris.data[iris.target == 2, 0]

F, p = stats.f_oneway(setosa, versicolor, virginica)
print(f"F = {F:.2f}, p = {p:.2e}")
# Output: F = 119.26, p = 1.67e-31
```

A researcher might conclude there are significant differences and stop here.

Guardian-augmented approach:

```
# StickForStats API call
result = requests.post(url, json={
    "groups": {
```

```
        "setosa": setosa.tolist(),
        "versicolor": versicolor.tolist(),
        "virginica": virginica.tolist()
    }
}).json()

# Guardian Report:
# Confidence Score: 0.72
#
# Violations:
# - Variance Homogeneity: WARNING
#   Levene's test: F=6.35, p=0.0023
#   Variance ratio (max/min): 3.25
#   Recommendation: Consider Welch's ANOVA
#
# - Normality: All groups PASS
#   Shapiro-Wilk (setosa): W=0.978, p=0.460
#   Shapiro-Wilk (versicolor): W=0.978, p=0.465
#   Shapiro-Wilk (virginica): W=0.971, p=0.258
```

Guardian finding: The Iris data violates the homogeneity of variance assumption (Levene $p = 0.002$). While standard ANOVA reports highly significant results, the violation means the F-test may be unreliable.

Recommendation followed: Welch's ANOVA (Alexander-Govern test), which does not assume equal variances, yields statistic $= 146.36$, $p < 0.001$. The conclusion (significant species differences) holds, but with appropriate method. The effect size is large ($\eta^2 = 0.619$).

Key insight: The highly significant result ($p = 10^{-31}$) might lead researchers to skip assumption checking. Guardian catches the variance issue regardless of result significance.

10.2. Case 2: Wine Quality dataset – Correlation assumptions

The UCI Wine Quality dataset (Cortez, Cerdeira, Almeida, Matos, and Reis 2009) contains physicochemical properties and quality ratings for Portuguese wines. We analyze the red wine subset ($n = 1{,}599$), examining the relationship between alcohol content and quality.

Analysis: Correlation between alcohol percentage and quality rating (ordinal scale 3–9).

Traditional approach:

```
r, p = stats.pearsonr(wine['alcohol'], wine['quality'])
print(f"r = {r:.3f}, p = {p:.2e}")
# Output: r = 0.476, p = 2.83e-91
```

Guardian findings:

```
# Guardian Report:
# Confidence Score: 0.58
#
# Violations:
# - Normality (quality): CRITICAL
#   Shapiro-Wilk: W=0.885, p<0.001
#   Quality is ordinal, not continuous normal
#
```

```
# - Linearity: PASS
#   Quadratic R^2 improvement: 1.0%
#   Linear relationship adequate
#
# Alternatives suggested:
# - Spearman's rho (ordinal data)
# - Polychoric correlation (ordinal-continuous)
```

Guardian finding: Quality is an ordinal variable (integer 3–9 in data), violating the continuous normality assumption of Pearson's r. Guardian's critical normality violation flag alerts the researcher.

Appropriate analysis: Spearman's $\rho = 0.479$, $p < 0.001$. The correlation remains significant, but Spearman's is the appropriate measure for ordinal data.

10.3. Case 3: Simulated meta-analysis – Publication bias

We simulate a meta-analysis scenario where publication bias is present, using effect sizes from 12 studies. The simulation models a realistic bias pattern: larger studies (smaller standard errors) publish regardless of effect size, while smaller studies only reach publication with larger effects. For reproducibility, this simulation uses `numpy.random.seed(561)`.

Data: Studies with larger effects were more likely to be published, creating funnel plot asymmetry.

```
import numpy as np
np.random.seed(561)  # For reproducibility

studies = {
    'effect_sizes': [0.23, 0.15, 0.25, 0.35, 0.28, 0.28,
                     0.32, 0.42, 0.33, 0.28, 0.37, 0.31],
    'standard_errors': [0.08, 0.04, 0.07, 0.06, 0.07, 0.08,
                        0.13, 0.14, 0.12, 0.13, 0.12, 0.14]
}
```

Guardian findings:

```
# Meta-analysis results:
# Pooled effect (random): 0.263, 95% CI [0.213, 0.314]
# Heterogeneity: I^2 = 14.4%, Q = 12.85
#
# Guardian Report:
# Confidence Score: 0.65
#
# Violations:
# - Publication Bias: WARNING
#   Egger's test: intercept=1.84, p=0.024
#   Funnel plot shows asymmetry
#   Recommendation: Consider publication bias in interpretation
```

Guardian finding: Egger's test ($p = 0.024$) indicates significant funnel plot asymmetry, suggesting publication bias. Researchers should interpret the pooled effect with caution and consider sensitivity analyses.

Key insight: Without Guardian's automatic publication bias check, researchers might report the pooled effect without considering potential bias in the underlying literature.

10.4. Summary of case studies

Table 8 summarizes the case studies.

Table 8: Summary of case study findings

Dataset	Violation	Impact	Recommendation
Iris	Variance heterogeneity	Unreliable F-test	Welch's ANOVA
Wine	Non-normality (ordinal)	Inappropriate r	Spearman's ρ
Meta-analysis	Publication bias	Biased estimate	Sensitivity analysis

In all three cases, the primary statistical conclusion (significant effect) remained unchanged after addressing violations. However, the *appropriate method* differed from the naive approach. Guardian ensures researchers are informed of these issues automatically.

10.5. Additional validation with classic datasets

To demonstrate Guardian's generalizability, we validated against three additional classic datasets from R's standard library.

Regression: mtcars dataset

The mtcars dataset contains fuel efficiency data for 32 automobiles. We analyzed the relationship between weight and miles per gallon using linear regression.

Results: $R^2 = 0.753$, slope $= -5.34$, $p < 0.001$.

Guardian validation:

- Residual normality: PASS (Shapiro-Wilk $W = 0.945$, $p = 0.104$)
- Homoscedasticity: PASS ($r = -0.018$ between |residuals| and fitted)
- Linearity: WARNING (runs test suggests potential curvature)
- Outliers: WARNING (3 observations with $|z| > 2$)

Guardian alerts the researcher to investigate linearity and outliers before finalizing conclusions.

Two-sample comparison: ToothGrowth dataset

The ToothGrowth dataset compares tooth length in guinea pigs receiving vitamin C via orange juice (OJ) vs. ascorbic acid (VC).

Results: $t = 1.92$, $p = 0.060$, Cohen's $d = 0.50$ (medium effect).

Guardian validation:

- Normality (OJ): WARNING (Shapiro-Wilk $W = 0.918$, $p = 0.024$)

- Normality (VC): PASS (Shapiro-Wilk $W = 0.966$, $p = 0.429$)
- Variance homogeneity: PASS (Levene $F = 1.21$, $p = 0.275$)

Guardian flags the non-normality in the OJ group, prompting consideration of the Mann-Whitney U test as an alternative.

ANOVA: PlantGrowth dataset

The PlantGrowth dataset compares plant weight under control and two treatment conditions.
Results: $F = 4.85$, $p = 0.016$, $\eta^2 = 0.264$.
Guardian validation:

- Normality (all groups): PASS
- Variance homogeneity: PASS (Levene $F = 1.12$, $p = 0.341$)

This case demonstrates Guardian correctly identifying clean data where all assumptions are satisfied, avoiding false alarms.

10.6. Summary

Across six datasets spanning ANOVA, correlation, regression, t-tests, and meta-analysis, Guardian consistently identified assumption violations when present and correctly passed clean data. The verification scripts are available in the repository at `paper/replication/`.

11. Discussion

11.1. Contributions in context

StickForStats' primary contribution is the Guardian system, which provides automatic assumption validation integrated into the analysis pipeline. This represents a shift from optional validation (requiring user initiative) to default validation (requiring user opt-out).

Beyond Guardian, the platform addresses additional gaps in statistical practice. The AI Statistical Advisor helps users navigate test selection and generates methods sections that comply with reporting guidelines. This addresses the documented problem that many researchers struggle with choosing appropriate tests and often omit required reporting elements (Appelbaum *et al.* 2018). The Paper Parser enables pre-submission quality checking, potentially catching reporting errors before peer review. These components work together: Guardian ensures valid analyses, the Advisor helps report them correctly, and the Parser verifies compliance.

We do not claim this approach solves the reproducibility crisis. Assumption validation is one component of sound statistical practice. Study design, sampling, measurement validity, and appropriate inference all remain researcher responsibilities. However, by automating assumption checking and providing guidance for test selection and reporting, we remove several common failure points from the chain of statistical decision-making.

11.2. Limitations

We acknowledge several limitations:

Threshold dependence

Guardian's severity classifications depend on fixed thresholds (e.g., $p < 0.05$ for warnings). These thresholds are conventional but not optimal for all situations. A $p = 0.049$ differs trivially from $p = 0.051$, yet crosses a classification boundary.

Mitigation: Guardian reports actual test statistics and p-values, not just classifications. Users can apply their own thresholds based on domain knowledge.

Assumption tests have assumptions

Validators themselves rest on assumptions. The Shapiro-Wilk test assumes a random sample. Levene's test assumes independence. This creates potential for circular reasoning.

Mitigation: Guardian uses multiple tests for key assumptions (e.g., Shapiro-Wilk AND Anderson-Darling for normality) and reports concordance.

Power of assumption tests

Assumption tests have their own power characteristics:

- Small samples: May fail to detect real violations (Type II error)
- Large samples: May flag trivial violations as significant (Type I error)

With $n = 1000$, a Shapiro-Wilk test might reject normality for data that is "close enough" for the central limit theorem to apply.

Mitigation: Guardian considers sample size in severity classification and notes when violations may be practically ignorable due to CLT robustness.

Incomplete coverage

Guardian's eight validators do not cover all possible assumptions:

- Measurement reliability is not assessed
- Selection bias cannot be detected from data alone
- Model specification errors may go undetected
- Temporal dependencies beyond lag-1 are not checked

Mitigation: Guardian explicitly states which assumptions are checked, making gaps transparent.

Expert Mode override

By default, Guardian blocks test execution when critical assumption violations are detected (Protected Mode). However, Expert Mode allows experienced users to proceed with analyses

despite violations. While warnings remain visible, this override means determined users can still produce potentially invalid analyses.

Rationale: The default blocking behavior protects novice users, while Expert Mode accommodates researchers with domain knowledge justifying edge cases. This configurable approach balances protection with researcher autonomy, recognizing that strict blocking could prevent valid analyses in specialized contexts.

11.3. Comparison with alternative approaches

Pre-registration: Platforms like OSF encourage analysis plan specification before data collection. This addresses p-hacking and HARKing but not assumption violations. Guardian complements pre-registration.

Automated reporting: Tools like **papaja** (Aust and Barth 2020) automate APA-style result reporting. These ensure consistent formatting but do not validate assumptions. Guardian could integrate with such tools.

Statistical review: Journal statistical review catches problems post-analysis. Guardian catches problems during analysis, enabling correction before results are finalized.

11.4. Future work

Several extensions are planned:

Additional validators: Sphericity testing for repeated measures, proportional hazards assumption for survival analysis, and exchangeability for Bayesian methods.

Bayesian integration: Prior predictive checks and posterior predictive checks as assumption validation for Bayesian analyses.

Machine learning diagnostics: Influence diagnostics, cross-validation stability, and feature importance validation for ML models.

Adaptive thresholds: Context-aware thresholds that adjust based on sample size, effect size, and domain conventions.

Expanded AI Advisor: Integration with domain-specific statistical guidance (e.g., clinical trials, ecology, psychometrics) and support for power analysis planning.

Paper Parser enhancements: Detection of additional reporting issues, support for domain-specific reporting guidelines (CONSORT, STROBE), and integration with reference management tools.

Statistical Quality Score (SQS) for journals: Beyond individual researcher use, the Paper Parser has potential applications in peer review. We envision a scoring system (0-100) evaluating manuscripts across six categories: effect size reporting, assumption transparency, sample and power documentation, statistical precision, reproducibility indicators, and guideline compliance. Journals could configure field-specific thresholds, similar to plagiarism detection sensitivity settings. This would not replace human judgment but could standardize the statistical review component that currently varies across reviewers. A prototype implementation with 50+ detection rules is included in the current release.

12. Conclusion

We have presented **StickForStats**, a statistical analysis platform with integrated automatic assumption validation through the Guardian system. The fundamental insight motivating this work is that optional assumption checking tools, available for over 25 years, have not solved the problem of assumption violations in published research. The solution is not better tools (the tools already exist), but different default behavior: automatic validation that informs every analysis.

Guardian implements eight validators covering the most common assumptions for parametric tests: normality, variance homogeneity, independence, outliers, sample size adequacy, modality, linearity, and homoscedasticity. When violations are detected, Guardian provides severity classifications, specific recommendations, and alternative test suggestions. Validation against SciPy demonstrates computational correctness to 14+ decimal places. A comprehensive test suite of 93 automated tests (38 backend, 55 frontend) ensures that the Design Contract—"no statistical result may exist without an explicit, traceable assumption context"—is enforced across the entire application stack.

Beyond assumption validation, **StickForStats** provides complementary tools for improving statistical practice. The AI Statistical Advisor offers natural language guidance for test selection and generates JARS-compliant methods sections. The Paper Parser analyzes manuscripts for statistical reporting quality before submission. The Statistical Debugger provides systematic troubleshooting when analyses produce unexpected results. Together, these components address multiple points in the research workflow where statistical errors commonly occur.

Case studies with real datasets (Fisher's Iris, UCI Wine Quality) demonstrate Guardian's practical value: catching variance heterogeneity in ANOVA, inappropriate correlation measures for ordinal data, and publication bias in meta-analysis. In each case, Guardian's warnings led to more appropriate analytical choices.

StickForStats does not claim to solve the reproducibility crisis. Assumption validation, test selection guidance, and reporting compliance are components of sound statistical practice, but study design, sampling, and measurement validity remain researcher responsibilities. However, by automating these components, we remove several common sources of statistical errors from the burden of human vigilance.

The software is available as open-source at `https://github.com/visvikbharti/stickforstats_new` under the MIT license. We welcome contributions, bug reports, and feature requests from the community.

Acknowledgments

We thank the developers of **SciPy**, **NumPy**, and **mpmath** for the foundational libraries upon which **StickForStats** builds. We acknowledge CSIR-IGIB for infrastructure support and AcSIR for academic guidance.

References

Anderson TW, Darling DA (1954). "A Test of Goodness of Fit." *Journal of the American Statistical Association*, **49**(268), 765–769. doi:10.1080/01621459.1954.10501232.

Appelbaum M, Cooper H, Kline RB, Mayo-Wilson E, Nezu AM, Rao SM (2018). "Journal Article Reporting Standards for Quantitative Research in Psychology: The APA Publications and Communications Board Task Force Report." *American Psychologist*, **73**(1), 3–25. doi:10.1037/amp0000191.

Aust F, Barth M (2020). *papaja: Prepare Reproducible APA Journal Articles with R Markdown*. R package version 0.1.0.9997, URL https://github.com/crsh/papaja.

Baker M (2016). "1,500 Scientists Lift the Lid on Reproducibility." *Nature*, **533**(7604), 452–454. doi:10.1038/533452a.

Bakker M, Wicherts JM (2011). "The (Mis)Reporting of Statistical Results in Psychology Journals." *Behavior Research Methods*, **43**(3), 666–678. doi:10.3758/s13428-011-0089-5.

Bartlett MS (1937). "Properties of Sufficiency and Statistical Tests." *Proceedings of the Royal Society of London. Series A*, **160**(901), 268–282. doi:10.1098/rspa.1937.0109.

Breusch TS, Pagan AR (1979). "A Simple Test for Heteroscedasticity and Random Coefficient Variation." *Econometrica*, **47**(5), 1287–1294. doi:10.2307/1911963.

Cortez P, Cerdeira A, Almeida F, Matos T, Reis J (2009). "Modeling Wine Preferences by Data Mining from Physicochemical Properties." *Decision Support Systems*, **47**(4), 547–553. doi:10.1016/j.dss.2009.05.016.

Durbin J, Watson GS (1951). "Testing for Serial Correlation in Least Squares Regression. II." *Biometrika*, **38**(1/2), 159–177. doi:10.2307/2332325.

Faul F, Erdfelder E, Lang AG, Buchner A (2007). "G*Power 3: A Flexible Statistical Power Analysis Program for the Social, Behavioral, and Biomedical Sciences." *Behavior Research Methods*, **39**(2), 175–191. doi:10.3758/BF03193146.

Fisher RA (1936). "The Use of Multiple Measurements in Taxonomic Problems." *Annals of Eugenics*, **7**(2), 179–188. doi:10.1111/j.1469-1809.1936.tb02137.x.

Fox J, Weisberg S (2019). *An R Companion to Applied Regression*. 3rd edition. Sage, Thousand Oaks, CA. URL https://socialsciences.mcmaster.ca/jfox/Books/Companion/.

Grubbs FE (1969). "Procedures for Detecting Outlying Observations in Samples." *Technometrics*, **11**(1), 1–21. doi:10.1080/00401706.1969.10490657.

Harris CR, *et al.* (2020). "Array Programming with NumPy." *Nature*, **585**(7825), 357–362. doi:10.1038/s41586-020-2649-2.

Hoekstra R, Kiers HAL, Johnson A (2012). "Are Assumptions of Well-Known Statistical Techniques Checked, and Why (Not)?" *Frontiers in Psychology*, **3**, 137. doi:10.3389/fpsyg.2012.00137.

Ioannidis JPA (2005). "Why Most Published Research Findings Are False." *PLoS Medicine*, **2**(8), e124. doi:10.1371/journal.pmed.0020124.

JASP Team (2023). *JASP (Version 0.17.3)*. URL https://jasp-stats.org/.

Johansson F (2013). "mpmath: A Python Library for Arbitrary-Precision Floating-Point Arithmetic." Version 1.3.0, URL http://mpmath.org/.

Keselman HJ, Huberty CJ, Lix LM, Olejnik S, Cribbie RA, Donahue B, Kowalchuk RK, Lowman LL, Petoskey MD, Keselman JC, Levin JR (1998). "Statistical Practices of Educational Researchers: An Analysis of Their ANOVA, MANOVA, and ANCOVA Analyses." *Review of Educational Research*, **68**(3), 350–386. doi:10.3102/00346543068003350.

Levene H (1960). "Robust Tests for Equality of Variances." In I Olkin (ed.), *Contributions to Probability and Statistics*, pp. 278–292. Stanford University Press, Palo Alto, CA.

Nickerson RS (1998). "Confirmation Bias: A Ubiquitous Phenomenon in Many Guises." *Review of General Psychology*, **2**(2), 175–220. doi:10.1037/1089-2680.2.2.175.

Nosek BA, Ebersole CR, DeHaven AC, Mellor DT (2018). "The Preregistration Revolution." *Proceedings of the National Academy of Sciences*, **115**(11), 2600–2606. doi:10.1073/pnas.1708274114.

Osborne JW (2010). "Improving Your Data Transformations: Applying the Box-Cox Transformation." *Practical Assessment, Research, and Evaluation*, **15**(12), 1–9. doi:10.7275/qbpc-gk17.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Schulz KF, Altman DG, Moher D (2010). "CONSORT 2010 Statement: Updated Guidelines for Reporting Parallel Group Randomised Trials." *BMJ*, **340**, c332. doi:10.1136/bmj.c332.

Seabold S, Perktold J (2010). "Statsmodels: Econometric and Statistical Modeling with Python." In *Proceedings of the 9th Python in Science Conference*, pp. 57–61. doi:10.25080/Majora-92bf1922-011.

Shapiro SS, Wilk MB (1965). "An Analysis of Variance Test for Normality (Complete Samples)." *Biometrika*, **52**(3-4), 591–611. doi:10.1093/biomet/52.3-4.591.

The jamovi project (2023). *jamovi (Version 2.4)*. URL https://www.jamovi.org/.

Virtanen P, *et al.* (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods*, **17**(3), 261–272. doi:10.1038/s41592-019-0686-2.

Zimmerman DW (2004). "A Note on Preliminary Tests of Equality of Variances." *British Journal of Mathematical and Statistical Psychology*, **57**(1), 173–181. doi:10.1348/000711004849222.

Affiliation:

Vishal Bharti (Corresponding Author)
ORCID: 0009-0003-1431-4457
CSIR-Institute of Genomics and Integrative Biology
Mathura Road, New Delhi 110025, India
E-mail: vishalvikashbharti@gmail.com
URL: https://github.com/visvikbharti/stickforstats_new

Debojyoti Chakraborty (Corresponding Author)
ORCID: 0000-0003-1460-7594
CSIR-Institute of Genomics and Integrative Biology
Mathura Road, New Delhi 110025, India
and Academy of Scientific and Innovative Research (AcSIR)
Ghaziabad 201002, India
E-mail: debojyoti.chakraborty@igib.in

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd