POLITÉCNICO DE LEIRIA | ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Inferential Statistics with Python

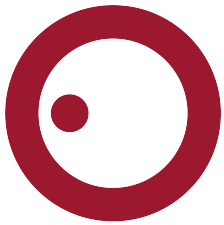
Learning Statistics Through Interactive Python Tools

Daniel Bedoya

School of Technology and Management

Master in Data Science

Leiria, July 2026



escola superior de tecnologia e gestão
instituto politécnico de leiria

Inferential Statistics with Python

Learning Statistics Through Interactive Python Tools

Daniel Bedoya

Supervisor: Miguel Felgueiras
Coordinator, Polytechnic of Leiria

Co-supervisor: Rui Santos
Coordinator, Polytechnic of Leiria

School of Technology and Management

Master in Data Science

Project

Leiria, July 2026

Inferential Statistics with Python

Acknowledgements

No individual achieves success without the backing of a community. I express my gratitude to Professors Miguel Felgueiras and Rui Santos for their mentorship throughout this project, as their guidance proved essential to bring this work to a successful conclusion. I extend my appreciation to my family, whose unwavering support provided me the fuel required to complete this journey. I also recognize the friends who accompanied me along this educational path, sharing both the challenges and the growth of academic life. And above all, I thank God, whose light makes every achievement possible.

Resumo

Na era dos dados, a aprendizagem da estatística inferencial continua a ser fundamental. Este projeto apresenta uma aplicação de *software* de código aberto desenvolvida para apoiar o ensino de conceitos estatísticos fundamentais através de `Python`. O sistema organiza ferramentas essenciais em grandes módulos analíticos: diagnósticos de normalidade, testes de médias, medianas e proporções para uma ou múltiplas populações, tabelas de contingência e análise de correlação.

Desenvolvido com uma Interface Gráfica de Utilizador (GUI) intuitiva, o *software* permite aos utilizadores avaliar conjuntos de dados `CSV` sem experiência em programação. Para reforçar a aprendizagem, cada interação exibe o código subjacente em tempo real, o que permite aos estudantes replicar a análise nos seus próprios ambientes. A plataforma é distribuída como um executável pré-compilado para uma implementação direta, sem necessidade de instalação. Este ambiente reduz as barreiras técnicas enquanto mantém o rigor metodológico, oferecendo uma estrutura focada que liga a teoria abstrata à prática computacional reprodutível.

Palavras-Chave: Estatística inferencial, `Python` no ensino da Estatística, Interface Gráfica de Utilizador, Software de código aberto, Aprendizagem da estatística.

Abstract

In the data era, learning inferential statistics remains fundamental. This project presents an open-source software application designed to support the education of core statistical concepts through `Python`. The system organizes key tools into major analytical modules: normality diagnostics, tests for means, medians, and proportions across single or multiple populations, contingency tables, and correlation analysis.

Built with an intuitive Graphical User Interface (GUI), the software enables users to evaluate `CSV` datasets without programming experience. To reinforce learning, every interaction displays the underlying code in real time, allowing students to replicate the analysis in their own environments. The platform is distributed as a pre-compiled executable for direct, installation-free deployment. This environment reduces technical barriers while maintaining methodological rigor, offering a focused framework that connects abstract theory to reproducible computational practice.

Keywords: Inferential statistics, `Python` in statistics education, Graphical User Interface, Open-source software, Learning statistics.

Contents

List of Figures

1 Introduction

Unlike the previous century, where the primary challenge was the scarcity of data, the contemporary dilemma lies in its proper application and robust interpretation.

The dawn of the Big Data era has conclusively demonstrated that inflating the volume of data does not inherently translate into more accurate statistical inferences.

A manifestation of this phenomenon is represented by `Google Flu Trends (GFT)` in 2013. `GFT` attempted to infer real-world influenza prevalence by relying on a massive corpus of user search queries. However, the model suffered from overforecasting by an astronomical 140% because it confused historical correlation with stable causal mechanisms. Statistically, `GFT` fell prey to endemic non-stationarity and the dynamics of changing search engine algorithms, which artificially inflated the frequency of target terms. It proved that passive, high-volume data collection cannot substitute for rigorous experimental design and epidemiological controls [1, 2].

Concurrently, there is a widespread misconception that complex architectures introduced by modern Machine Learning can navigate this vast ocean of data more effectively without human intervention. Yet, recent history confirms that critical, principle-based statistical analysis remains indispensable.

Amazon's automated recruitment algorithm learned to systematically penalize female candidates because it was trained on historical data from a decade where the tech industry was predominantly male. Statistically, the model simply optimized for a distorted historical distribution, mapping a systemic sociological bias as an objective performance metric [3, 4].

Similarly, the sudden collapse of the digital real estate platform Zillow Offers in 2021 illustrates the dangers of deploying high-dimensional models in unpredictable environments. Zillow's predictive algorithms suffered from severe algorithmic overfitting; they treated short-term macroeconomic noise and post-pandemic market volatility as stable, long-term trends [5, 6].

Both cases serve as stark warnings of what occurs when massive datasets are fed into complex mathematical frameworks without a grounding in statistical validation.

As the classical adage dictates: *garbage in, garbage out*.

Ultimately, mathematical and statistical literacy has transformed from a technical skill into an essential safeguard in an era dominated by intricate models, where fewer and fewer truly understand the mechanics operating inside the black box.

That is why statistical literacy has become a fundamental competency in the modern world. It is imperative to teach individuals how to interpret and leverage data to make informed decisions.

A professional well-grounded in statistics is better equipped to navigate complex situations across a wide range of fields, including public health, engineering, business management, and finance. Furthermore, a solid foundation in statistical reasoning fosters scientific skepticism, empowering individuals to critically evaluate claims and avoid manipulation by misrepresented data [7].

To achieve this level of literacy, educational tools must empower students to apply grounded statistical techniques to real-world projects without being hindered by the initial complexity of mathematical formulations. A proficient decision-maker must be able to grasp the "big picture", understanding how different components interact without necessarily calculating all the minor details by hand.

Historically, proprietary computing platforms such as `IBM SPSS`, `Stata`, and `MATLAB` have addressed this need by abstracting mathematical complexities. However, these tools often confine students within closed ecosystems, each governed by its own proprietary syntax and distinct operational workflow.

While some of these platforms do require users to write code, thereby exposing a portion of the statistical logic, that knowledge remains tightly coupled to the specific software environment. Consequently, students face steep and isolated learning curves, acquiring programming skills that are largely non-transferable. This fragmentation makes transitioning between different analytical tools highly inefficient, as the specialized command structure of one platform rarely applies to another, ultimately limiting the student's adaptability in modern professional landscapes where open-source flexibility is crucial.

In this context, the transition toward open-source environments is primarily driven by two dominant programming languages: `R` and `Python`.

On one hand, `R` remains highly utilized within academic research and traditional scientific disciplines, offering exceptionally reliable packages specifically engineered for rigorous statistical inference and advanced data visualization. On the other hand, `Python` has emerged as a distinctively versatile alternative due to its symbiotic relationship with modern artificial intelligence and machine learning [8].

While `R` excels within its specialized analytical boundaries, `Python` connects students directly to the broader contemporary technological ecosystem. Consequently, for students learning mathematical and statistical techniques, `Python` offers utility that extends far beyond the classroom. Developing foundational skills in this language empowers students not only to grasp core statistical concepts but also to create, manage, and scale complex projects within modern, AI-driven professional landscapes.

That is why the purpose of this project is to develop a `Python`-based educational tool designed to support the teaching of statistical inference. By utilizing a visual interface that simultaneously displays the underlying code required to execute each technique, the tool effectively bridges the gap between conceptual learning and programming practice. This duality allows users to extract a fully functional script that replicates the exact workflow performed within the graphical interface, thereby guaranteeing the reproducibility of the entire analytical process.

Consequently, students not only gain an intuitive understanding of statistical mechanics, but also obtain an adaptable code pipeline that they can readily copy, paste, and modify, providing a practical launchpad to explore `Python` and customize their own independent data workflows.

Considering this, the main objectives of the project are:

- To develop a `Python`-based tool that implements the most relevant statistical inference techniques required in a standard college-level curriculum.
- To provide a visual interface that allows students to explore and apply statistical methods without needing prior, in-depth programming experience.
- To expose the necessary `Python` code alongside the visual interface, enabling users to understand the mechanics, personalize the execution, and eventually transition to writing their own code.
- To establish an open-access repository, ensuring this project is freely available for integration into diverse teaching environments.

The tool focuses on the core components of inferential statistics. This framework includes normality testing as a prerequisite step, followed by one-sample, two-sample, and k-sample hypothesis tests for evaluating variances, means, and medians. Additionally, the tool implements proportion tests, tests of independence, and correlation analyses, thereby covering the essential inference techniques required by standard college-level programs.

Conversely, certain limitations have been established to preserve the pedagogical focus of the project. This tool does not aim to replace standard, production-grade programming libraries. Instead, it serves as an instructional interface designed to help students understand how to utilize established packages such as `statsmodels`. Furthermore, the code exposed by the application is not intended to teach software engineering principles or code optimization techniques. The priority remains strictly on demonstrating how to deploy `Python` syntax as a transparent tool to achieve and interpret the expected statistical outcomes, preventing computational complexity from overshadowing statistical comprehension.

The final deliverable of the project consists of a fully functional, standalone executable application made publicly available alongside its source code through an open-access `GitHub` repository. Students can launch the interface and explore statistical methods immediately, eliminating the initial barrier of installing or configuring a local `Python` environment.

To achieve these objectives within the defined scope, the remainder of this report is organized as follows. Chapter 2 contextualizes the project within the current statistical ecosystem, reviewing enterprise analytical software, prominent `R` packages, and existing `Python`-based initiatives aimed at teaching statistics and data science. Chapter 3 details the software architecture and delineates the specific statistical tools selected for integration into the application. Chapter 4 presents a comprehensive demonstration of the system through three distinct practical scenarios. And Chapter 5 outlines directions for future work and discusses potential developments that could expand upon this project.

2

Computational Ecosystems for Inferential Statistics

In modern data science, statistical analysis is intrinsically tied to computational systems. The evolution of technology has streamlined the execution of robust statistical methods, allowing practitioners to focus on theoretical interpretation while abstracting the complexity of manual, exhaustive calculations.

Consequently, data analysts require tools that facilitate the seamless application of mathematical techniques. To meet this demand, a diverse software ecosystem has emerged, designed to lower technical barriers and streamline statistical workflows.

These solutions range from highly intuitive, visual low-code interfaces to specialized libraries within advanced programming languages.

This chapter provides a comprehensive overview of these tools, establishing the state of the art in the application of inferential statistics through modern software ecosystems.

2.1 Conventional Analytical Paradigms: Enterprise Analytics and Academic Software

Given the contemporary significance of data-driven decision-making, a diverse array of corporate tools has emerged to leverage statistical methods within business environments. Consequently, well-established enterprise software plays a critical role in academic education, training future analysts to interpret data within industry-standard frameworks.

Within this ecosystem, `Microsoft Power BI` stands out as a dominant analytical suite, backed by one of the world's leading software corporations [9, 10]. A primary advantage of this platform is the availability of a free desktop license, which democratizes access to robust, industry-grade tools for individual users [11]. The interface allows users to integrate disparate data sources into a cohesive environment designed

for intuitive and visual data representation [12].

While its core functionality is descriptive, Power BI incorporates the Data Analysis Expressions (DAX) language to perform advanced analytical operations, such as dynamic aggregations and complex time-intelligence calculations. However, DAX lacks native functionality for classical statistical inference. Performing tasks such as executing hypothesis tests is not inherently supported through built-in functions. To achieve true inferential capabilities, Power BI must instead rely on its integration with external languages like R and Python, which handle the underlying statistical computing. [13, 14].

Another dominant platform in this space is Tableau, which relies on an intuitive drag-and-drop visual paradigm that streamlines descriptive data exploration [15]. It offers a free tier for individual users enabling foundational descriptive analytics and data visualization, albeit with data privacy limitations for non-commercial software deployment [16, 17]. Although it incorporates built-in statistical diagnostic features, such as the automatic computation of coefficients of determination (R^2) and p-values within trend lines, its core architecture remains constrained regarding comprehensive inferential modeling [18, 19].

When talking about cloud-native environments, Google Looker Studio is a prominent option [20]. The platform is integrated into the Google Cloud ecosystem and it operates fully in a cloud environment, facilitating real-time collaborative reporting and seamless data integration with native cloud databases, such as Google BigQuery [21]. However, its functional scope is overwhelmingly localized to descriptive dashboards and operational metrics. The platform features minimal built-in support for inferential computing, requiring external integration with advanced analytical components or pre-processed data structures to execute rigid statistical workflows [22, 23].

Expanding the scope of workplace analytical tools, Microsoft Excel remains one of the most widely deployed applications globally. Although fundamentally designed as a spreadsheet processor, it has historically been utilized across both scientific research and corporate environments [24, 25]. Due to its diverse functionality and built-in graphing capabilities, Excel serves as an entry-point data tool within organizational ecosystems where stakeholders often lack advanced technical backgrounds.

However, Excel presents severe limitations regarding data integrity and robust lifecycle management. Because its cell-based architecture lacks a strict database schema layout, it fails to enforce consistent or standardized data transformation. Consequently, institutional data ingestion pipelines cannot safely rely on Excel due to the high risk of accidental data alteration or human-induced corruption [26, 27].

Furthermore, while the platform's Analysis ToolPak add-in provides core inferential capabilities, such as t-tests, analysis of variance (ANOVA), and linear regressions, its execution remains constrained. These analytical operations are bound to a static, non-reproducible workflow that lacks the programmatic automation, auditability, and scalability required for modern data science architectures [28, 29].

Shifting from enterprise Business Intelligence (BI) platforms toward specialized scientific paradigms, several legacy software packages remain the benchmark for rigorous

statistical computation within academia and research-driven industries.

IBM SPSS represents one of the most enduring platforms in this category, particularly within the social sciences. Its primary architecture relies on a Graphical User Interface (GUI) structured around data and variable matrices, allowing researchers to execute complex inferential tests, such as multivariate analysis of variance (MANOVA) and non-parametric evaluations, without requiring programmatic syntax [30, 31].

While its point-and-click design lowers the entry barrier for specialized researchers, manual interface operations can be difficult to audit unless paired with its native syntax language [32, 33].

In contrast, Stata is highly prominent within economics, public policy, and biostatistics due to its command-driven architecture. Unlike BI applications, Stata emphasizes a reproducible workflow where every analytical step is recorded via command scripts (do-files). Its core strength lies in its exceptionally robust econometric capabilities, such as advanced panel data analysis, survival modeling, and complex survey design tracking [34, 35].

While Stata ensures a high degree of workflow reproducibility through its dedicated internal scripting environment, this approach introduces significant rigidity since it relies entirely on a proprietary syntax. Consequently, transitioning these codebases to alternative analytic environments requires substantial refactoring, which ultimately exacerbates vendor lock-in. This closed ecosystem limits integration with contemporary production pipelines, cloud architectures, and open-source data infrastructures, rendering Stata less flexible than general-purpose programming frameworks when deployed outside of strictly academic environments [36, 37].

Other proprietary platforms justify their closed ecosystems by catering to highly regulated corporate industries rather than purely academic research. The Statistical Analysis System (SAS) remains a vital institutional standard within clinical trials, pharmaceuticals, and large-scale financial risk modeling.

SAS distinguishes itself through its rigid data step and procedure architecture, which ensures absolute consistency and compliance with stringent regulatory frameworks, such as the Food and Drug Administration (FDA) [38–40].

However, its steep learning curve and high enterprise licensing costs have driven a gradual paradigm shift toward open-source environments in non-regulated data sectors [41, 42].

In the domain of econometrics, the Econometric Views (EViews) platform represents a prominent institutional standard within central banking, commercial forecasting, and complex time-series modeling. EViews organizes data into specialized time-series containers called workfiles, a structure that automatically aligns date frequencies and enables immediate execution of unit root tests, vector autoregressions, and variance decompositions. Nonetheless, the closed ecosystem of EViews introduces prominent constraints, including high commercial acquisition costs and restricted code transparency, factors that impede exact research reproduction [43, 44].

To overcome these licensing cost constraints, the Gnu Regression, Econometrics

and Time-series Library (GRETL) serves as a free, accessible counterpart that combines an intuitive graphical interface with a robust command language for advanced regression analysis [45, 46].

However, GRETL presents distinct constraints. Its narrow specialization restricts its utility to econometrics and time-series analysis, which limits its adoption in broader data science or machine learning tasks. Also, the development community remains small in comparison to expansive ecosystems like R and Python.

Other specialized platforms focus on industrial operations and operations management. The Minitab software represents a prominent standard within manufacturing industries, quality engineering, and applied statistical education. Minitab distinguishes itself through its pre-programmed toolsets for Six Sigma methodology and statistical process control, features that ensure direct execution of control charts and capability analysis without coding requirements. Its main constraint is the high cost of licenses, which remains unaffordable for the general public [47, 48].

Another significant analytical environment is Wolfram Mathematica, which is fundamentally centered on symbolic representation and computation, effectively converting every operation into a unified, computable expression [49]. The platform includes a comprehensive suite of built-in capabilities for statistical inference, encompassing parametric testing, regression analysis, and automated distribution fitting [50]. While its functional design offers a relatively accessible learning curve for initial data exploration, it remains a proprietary language bound to expensive licensing models. Furthermore, because it functions as a self-contained ecosystem, its workflows can be somewhat disconnected from mainstream software development environments, limiting the direct transferability of code to broader open-source production pipelines [51].

Concluding this spectrum of dedicated analytical environments is MATLAB, a high-performance platform developed by MathWorks. Structurally optimized for matrix and vector computations, MATLAB is heavily utilized within engineering, econometrics, and computational signal processing rather than traditional social science statistics [52].

It provides a robust, native programming language alongside specialized toolboxes, such as the Statistics and Machine Learning Toolbox which allow researchers to perform complex inferential tasks, parameter estimations, and stochastic simulations [53]. While MATLAB offers mathematical precision and an integrated development environment (IDE) optimized for numerical analysis, its proprietary nature and cost structure present similar accessibility barriers to those of SAS, frequently limiting its deployment to well-funded academic departments and specialized industrial research facilities [54].

Departing from these high-performance platforms primarily focused on corporate and advanced research environments, GeoGebra presents itself as a visual pedagogical tool. Designed to encourage intuitive learning, it offers an accessible environment that includes dedicated modules for elementary inferential statistics, such as basic hypothesis testing, confidence intervals, and probability distribution modeling [55, 56].

While effective for building initial conceptual intuition, the platform remains limited to basic statistical designs, and its closed interface prevents the transition to code-based reproducibility.

As showed, the current landscape of data analytics comprises a diverse spectrum of platforms, each tailored to specific operational domains and requiring distinct learning curves. Consequently, proficiency in many of these tools has become a standardized requirement across various academic disciplines, including psychology, medicine, and economics.

However, a primary systemic constraint of these conventional platforms lies in their licensing cost models. Because complimentary versions are frequently restricted or non-existent for enterprise deployments, they establish significant financial barriers to entry, impacting both individual skill acquisition and institutional implementation within professional environments.

Furthermore, architectural limitations segment these tools into distinct functional compromises. BI suites remain fundamentally optimized for descriptive dashboards, lacking the mathematical flexibility required for rigorous inferential modeling, while dedicated academic software often operates within proprietary technological silos. This isolation hinders seamless integration with modern data pipelines and enforces a steep, non-transferable learning curve for users. In contrast, open-source programming paradigms, like `R` and `Python`, emerge as a compelling alternative, offering unparalleled algorithmic transparency, absolute workflow reproducibility, and native integration capabilities across disparate data environments.

2.2 The R Statistical Ecosystem

The `R` programming language represents the baseline architecture for academic research and computational statistics. Developed by and for statisticians, it remains the standard environment across institutional research, epidemiology, biostatistics, and the social sciences. Its widespread adoption is driven by the `Comprehensive R Archive Network (CRAN)`, a repository hosting thousands of specialized packages that document and implement advanced mathematical methodologies [57, 58].

Because the language treats data structures as fundamental statistical vectors and data frames by design, it functions as a primary vehicle for peer-reviewed analysis and formal algorithmic validation, cementing its role as the traditional benchmark in scientific inquiry.

At the core of the statistical analysis lies the native `stats` package, which provides the foundational mathematical engine for classical frequentist inference, paired with the `Companion to Applied Regression (car)` package, which extends its diagnostic capabilities [59, 60].

From an inferential standpoint, the `stats` package delivers high mathematical fidelity, implementing the baseline procedures, such as Student's t-tests, chi-square independence tests, and ordinary least squares regressions, required for peer-reviewed

validation.

The `car` package provides critical diagnostic utilities, including variance inflation factors (VIF) and tests for heteroscedasticity, which serve as essential pedagogical tools for teaching assumption verification.

To bridge the gap between programmatic execution and user accessibility, the `R Commander (Rcmdr)` package provides a structured GUI designed to ease the initial syntactic barriers of code-based analysis. Within academic curricula, `Rcmdr` functions as an indispensable pedagogical scaffolding tool by translating menu-driven, point-and-click operations into explicit `R` code in real-time [61].

This dual-window architecture allows graduate students to execute complex inferential procedures while observing the exact programmatic syntax generated by their choices. The platform demystifies command-line operations, guiding researchers from visual reliance toward an understanding of programmatic data workflows and reproducible script execution.

Expanding beyond local script execution, the integration of the `Shiny` framework allows researchers and educators to transition from static computational outputs to dynamic, interactive web-based environments. It provides a robust mechanism for visualizing abstract inferential theory through reactive statistical simulation [62].

Instructors can deploy applications that illustrate complex probabilistic phenomena, such as the Central Limit Theorem, by allowing real-time adjustments to underlying parameters like sample sizes, alpha levels, and effect magnitudes.

This interactive architecture enables students to observe the immediate recalculation of test statistics, confidence intervals, and p-values under varying experimental conditions. By translating mathematical abstractions into perceptible, empirical outcomes, it enriches the conceptual foundation required to comprehend the mechanics of rigorous scientific inference.

Finally, in addition to web frameworks, the `R` ecosystem includes standalone graphical interfaces designed for researchers who prefer visual menus. Recommended as modern, free alternatives, `JASP` and `Jamovi` integrate the statistical precision of the `R` engine within an accessible framework.

The primary advantage of these platforms is the real-time execution of analyses, where any modification to the data spreadsheet triggers an instantaneous update of the results. Also, the output generates publication-ready tables formatted to American Psychological Association (APA) standards. For specialized research, `JASP` stands out due to its comprehensive Bayesian framework, which permits users to perform complex hypothesis testing and parameter estimation without the need to write intricate code loops. In contrast, `Jamovi` prioritizes an intuitive data-analysis integration, offering a syntax mode that reveals the underlying `R` script for each menu action to facilitate a clear transition into programming. Both applications allow direct `R` code execution, providing an integrated environment where users can toggle between graphical inputs and raw script manipulation [63, 64].

Despite the analytical features of the `R` components examined so far, `R` still lacks the

versatility of general-purpose programming paradigms because is a domain-specific language optimized almost exclusively for data analysis. This limitation introduces friction when integrating statistical models into broader production pipelines, such as web application deployment, microservices architectures, or complex system-level automation, requiring development teams to maintain cumbersome bridges between the analytical core and the overarching software infrastructure.

2.3 The Python Statistical Suite

Addressing the structural limitations of both legacy applications and domain-specific open-source tools, Python has emerged as a dominant paradigm within the contemporary enterprise and scientific landscape. Primarily, it stands as the foundational programming language driving machine learning, mathematical modeling, and complex scientific simulations, bridging the gap between theoretical research and scalable industrial application [65, 66].

Also, unlike environments designed exclusively for statistical analysis, Python allows practitioners to build end-to-end data pipelines within a unified ecosystem. This versatility enables the seamless integration of inferential modeling with database systems, cloud computing platforms, and web application interfaces.

Consequently, mathematical frameworks can be effortlessly deployed directly into production environments, eliminating the friction traditionally associated with migrating academic code to enterprise-grade software architectures.

In response to this cross-disciplinary adoption, substantial efforts have been made to lower the barrier to entry for individuals with non-technical profiles. Accordingly, this chapter explores the most relevant tools and initiatives designed to facilitate the use and learning of statistical methods for a broader audience.

2.3.1 Core Libraries in Python

The Python programming language has emerged as a dominant force in data science and statistics due to its modular architecture. At the base of this scientific stack lies NumPy and Pandas.

The NumPy library provides the essential support for large, multidimensional arrays and matrices, utilizing vectorized operations that significantly outperform standard Python lists in computational speed, making it the standard for numerical simulations and linear algebra [67, 68]. Building upon this foundation, Pandas serves as the primary library for data manipulation. The DataFrame, a two-dimensional labeled data structure, enables the seamless manipulation of heterogeneously typed columns, data alignment, and complex reshaping, forming the prerequisite for any modern statistical workflow in Python [69, 70].

For graphical analysis, Matplotlib and Seaborn are the most widely used libraries, providing representations such as scatter plots, boxplots, heatmaps, bar plots, and den-

sity plots, which are fundamental to the exploratory phase of inferential statistics [71, 72].

The mathematical engines for rigorous inference and modeling are provided by `SciPy` and `Statsmodels`. While the `scipy.stats` module contains a vast library of probability distributions and fundamental statistical functions, such as t-tests and Wilcoxon rank-sum tests, `Statsmodels` is more specifically dedicated to the estimation of complex statistical models and data exploration, it provides comprehensive result summaries for linear regression, generalized linear models (GLM), and time-series analysis [73, 74].

While `Python` possesses a robust mathematical foundation that can be leveraged for scientific research, the fragmented nature of its ecosystem often requires users to master multiple tools and manage their complex integration. To address this challenge, several user-friendly wrappers have been developed to condense the analytical workflow into single-line commands.

The `Pingouin` library is a prominent example, designed for users who require exhaustive statistical outputs, including effect sizes and power analysis, without the verbosity of `Statsmodels` [75]. Similarly, `Researchpy` bridges the gap between `Pandas` and specialized statistical tools, offering a simplified method to produce publication-ready frequency tables and summary statistics [76].

The philosophy of these two libraries is to allow users to apply statistical methods directly after loading their data. However, these tools still present hurdles for novice users, the absence of a GUI, or a structured manual focused specifically on inferential statistics, results in a learning curve that remains prohibitive for individuals without a background in informatics.

2.3.2 Interactive Learning Environments

Efforts to democratize `Python` for mathematical applications have led to the development of unified frameworks that simplify technical execution for non-experts.

`Jupyter` notebooks have become a frequent tool for teaching complex computational and mathematical themes in `Python`. A `Jupyter` notebook is an open source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text [77]. By combining executable code with rich markdown explanations, these environments facilitate a literate programming approach that is highly effective for educational purposes.

Utilizing this powerful format, several recognized repositories have been structured as interactive books to teach diverse topics. For example, data analytics and geostatistics are the main focus of `DataScienceInteractivePython`, which includes specific interactive modules on spatial data visualization, probability distributions, hypothesis testing, and spatial continuity [78]. Another prominent example is `Kalman and Bayesian Filters in Python`, which covers introductory stochastic processes, discrete Bayes algorithms, Gaussian distributions, and advanced particle filters [79].

Other examples include Bayesian Methods for Hackers, which, through practical programming, introduces Bayesian inference, Markov Chain Monte Carlo (MCMC) methods, and loss functions [80]. Similarly, the Python Data Science Handbook provides interactive tutorials on foundational libraries and machine learning algorithms [81]. Additionally, repositories like Think Stats utilize Jupyter to teach exploratory data analysis and probability rules using real datasets [82].

A critical component that enhances the accessibility of these educational resources is their integration with Binder. Binder is an open source service that packages a repository into a custom, reproducible computing environment and deploys it in the cloud [83]. This technology allows students to execute the interactive notebooks directly through their web browsers, entirely eliminating the need to install Python or configure local computational environments.

Despite these technological advancements, relying purely on notebook repositories presents significant challenges. The primary disadvantage is that the user must be familiar with the Jupyter environment itself, as there is no clear GUI to guide them step by step through the analytical process.

To mitigate the navigational and syntactic difficulties inherent to raw Jupyter, hybrid solutions such as Visual Python have been developed [84]. Visual Python operates as a graphical code generator that integrates directly into Jupyter Notebooks, JupyterLab, and Google Colab. By providing a point and click GUI, it assists users in automatically generating the necessary Python syntax to execute complex operations. The platform is structured around an extensive suite of tools categorized by functionality, including modules for data manipulation, advanced data visualization, machine learning model implementation, and fundamental statistical analysis. This approach significantly lowers the technical barrier to entry, allowing users to interact with their datasets visually while maintaining a transparent, reproducible record of the generated Python code.

The primary limitation of Visual Python, however, lies in its foundational design as a general purpose analytical suite rather than a dedicated educational resource. Because its interface exposes a vast array of advanced data engineering and machine learning options, it can easily overwhelm a student whose sole objective is to master introductory inferential statistics. Moreover, while the tool excels at automating code generation, it lacks the pedagogical guidance and structured pathways necessary to help users understand underlying mathematical assumptions or interpret the resulting statistical outputs.

To transition from template-driven graphical generators to conversation-based systems, recent advancements introduce generative artificial intelligence into interactive learning environments. Julius AI represents this paradigm shift, functioning as an intelligent statistical assistant that utilizes large language models to run Python-based analyses without manual script generation. The platform operates through a chatbot format where users upload datasets and reference specific files within a digital workspace. By interpreting plain-text intent, the assistant constructs and executes the

required syntax in background environments, utilizing core libraries such as `Pandas` and `matplotlib`. This interface permits immediate data exploration, including statistical summaries, quick data visualizations, and outlier identification. This approach lowers technical barriers for novice researchers, allowing prompt-driven interactions while exposing the generated code to preserve an underlying record of the data manipulation [85, 86].

However, the primary limitation of `Julius AI` arises from its stochastic foundation as a generative language model rather than a deterministic statistical engine. The underlying algorithm operates on probabilistic word prediction, a factor that introduces inherent risks of code hallucination and non-reproducible outputs under identical prompts. This instability creates friction for students who require consistent, verifiable feedback to master introductory inferential statistics. Moreover, while the tool provides rapid analytical execution, the conversation-based automation can bypass the deep conceptual evaluation necessary to comprehend mathematical assumptions, building a false sense of competence without structured pedagogical guidance.

As a result, a significant gap remains in the current technological landscape. While current tools offer either rigorous code based ecosystems, comprehensive but unstructured interactive textbooks, or enterprise level graphical generators, there is a distinct absence of a focused middle ground. The academic community requires a streamlined, user friendly graphical interface that retains the educational transparency of `Python` code generation, yet is pedagogically structured specifically to guide non informatics students through the rigorous steps of inferential statistics without cognitive overload.

3 Software Architecture and Design

The project's architecture is founded on the principle of Separation of Concerns (SoC). As defined in software engineering, this principle states that each element of a program should do exactly one thing and one thing only [87].

SoC provides a powerful way to structure software systems, making them easier to maintain and reuse. A modular program implements this principle through information hiding, where each module exposes only a well-defined interface while keeping its internal details hidden. Consequently, individual sections of the program can be modified or reused independently, without needing to understand or alter other sections.

This design ensures that the mathematical rigor of the statistical tests is never compromised by the requirements of the user interface, and vice versa. By decoupling the computational engine from the visual representation, the system remains modular, scalable, and easy to audit.

3.1 Architectural Overview

Each component inside the project has a specific responsibility. This organization is reflected in the project's directory structure, ensuring a clean development workflow and high maintainability:

- **The Orchestrator (`app.py`):** Acts as the main entry point of the application. It initializes the execution environment and routes the user's navigation between the various statistical modules.
- **The Logic Layer (`/logic`):** This directory constitutes the computational core of the system. It contains pure `Python` modules that handle data processing and statistical testing. These functions are GUI-agnostic, receiving raw data, typically as `pandas.DataFrame` objects, and returning structured results such as p-values,

effect sizes, and test statistics without any dependency on the interface.

- **The Presentation Layer (`/gui`):** This layer is responsible for the user experience and interaction. It handles file uploads, input validation, and the rendering of interactive charts. Its primary role is to act as a bridge, taking raw outputs from the logic layer and formatting them into human-readable interpretations and executable `Python` code snippets.
- **Configuration (`.streamlit/` and `requirements.txt`):** These files manage the global environment, ensuring consistent styling, theme settings, and dependency versioning for core libraries such as `Pingouin`, `SciPy`, and `Statsmodels`.

3.1.1 Frontend Framework

The `Streamlit` library was selected as the primary framework for the implementation of the GUI due to its ability to transform `Python` scripts into interactive web applications without requiring extensive knowledge of frontend technologies like `HTML`, `CSS`, or `JavaScript`.

According to the official documentation [88], the framework is designed to handle the complexities of data-heavy applications by abstracting the client-server communication, allowing the developer to focus exclusively on the `Python`-based logic. Unlike traditional desktop-based libraries such as `Tkinter` or `PyQt`, `Streamlit` offers a modern, reactive web-based environment that facilitates rapid deployment and accessibility through a browser.

A defining characteristic of this framework is its unique execution model, which reruns the source script from top to bottom whenever a user interacts with a widget. This reactive behavior ensures that the internal state of the application remains synchronized with the statistical parameters selected by the user, such as significance levels or dataset filters.

Furthermore, `Streamlit` provides native integration with the `PyData` ecosystem, allowing for the direct rendering of `pandas.DataFrame` objects and high-fidelity visualizations generated by `Matplotlib`.

3.2 Statistical Concepts Integrated into the Tool

Consistent with the previously discussed architectural framework, the decoupling of the computational logic from the graphical interface facilitates a modular evaluation of the statistical methodologies embedded within the system.

This section details the specific techniques implemented within the `/logic` module, providing an overview of their mathematical foundations and the specialized `Python` libraries that serve as the primary engine for these analytical calculations.

3.2.1 Normality Tests

Testing for normality is a fundamental step in inferential statistics, as many parametric tests, such as the t-test and ANOVA, rely on the assumption that the data follows a normal distribution. Therefore, conducting a normality test is practically the mandatory first step in most analytical situations [89]. In this software, the engine for these calculations is the `scipy.stats` library.

Shapiro-Wilk Test

The Shapiro-Wilk test is widely regarded as one of the most powerful tests for normality and it is most recommended for sample sizes in the approximate range of 30 to 50, where it provides a reasonable balance between power and practicality [90, 91].

It calculates a W statistic that tests whether a random sample comes from a normally distributed population. The statistic is defined as:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{3.1}$$

where $x_{(i)}$ are the ordered sample values (i.e., $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$) and a_i are constants derived from the covariances of the order statistics.

The test is implemented using the `scipy.stats.shapiro` function, which relies on Royston's algorithm [92]. This algorithmic approach eliminates the need for traditional lookup tables of critical values, which are historically limited to specific sample sizes. Instead, Royston's method uses an analytical approximation to directly compute a continuous p-value for the W statistic.

D'Agostino's K^2 Test (Pearson)

This test combines skewness (b_1) and kurtosis (b_2) to produce an overall test of normality [93].

The K^2 statistic is defined as:

$$K^2 = Z_1^2(b_1) + Z_2^2(b_2), \tag{3.2}$$

where Z_1 and Z_2 are the standard normal deviates (Z-scores) obtained from the transformations of the sample skewness and kurtosis, respectively. Under the null hypothesis of normality, both Z_1 and Z_2 approximately follow a standard normal distribution $\mathcal{N}(0, 1)$, which implies that the combined K^2 statistic follows a chi-square distribution with 2 degrees of freedom.

The test's primary advantage lies in its ability to decompose the omnibus statistic into separate components for skewness and kurtosis, allowing practitioners to identify the specific nature of a departure from normality.

The test is most appropriate when the sample size is moderate to large; it is considered particularly suitable for $n > 50$ [94].

While it performs well across various conditions, recent research indicates that it is especially powerful in detecting moderate deviations from normality, making it a well-suited choice for many real-world datasets [95].

Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (K-S) test is a non-parametric test that compares the empirical cumulative distribution function (CDF) of the sample, $F_n(x)$, against the CDF of the theoretical normal distribution, $F(x)$. The test statistic D is the maximum absolute difference between them:

$$D_n = \sup_x |F_n(x) - F(x)|. \tag{3.3}$$

It is primarily suited for moderate to moderately large sample sizes [96]. In practice, the standard K-S test is overly conservative when parameters are estimated from the data, a common scenario in real-world analysis, and suffers from low statistical power compared to alternatives like Shapiro-Wilk or Anderson-Darling [97]. Consequently, the K-S test is best employed in pedagogical settings to introduce concepts of empirical CDFs and maximum absolute deviations, or in situations where the theoretical parameters are genuinely known [98].

Anderson-Darling Test

As an improvement over the K-S test, the Anderson-Darling test gives more weight to the tails of the distribution [99]. The statistic A^2 measures the distance between the CDF of the theoretical distribution, $F(x)$, and the empirical distribution function of the sample, $F_n(x)$.

Unlike the K-S test, it introduces a weighting function $w(x) = [F(x)(1 - F(x))]^{-1}$, which places more importance on the tails of the distribution.

For a sample of size n with ordered observations $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$, the computational formula is:

$$A^2 = -n - \frac{1}{n} \sum_{i=1}^{n} (2i - 1) \left[\ln F(x_{(i)}) + \ln \left(1 - F(x_{(n-i+1)}) \right) \right], \tag{3.4}$$

where $F(x_{(i)})$ is the CDF of the normal distribution for the i-th sorted observation.

The Anderson-Darling test traditionally relies on comparing the A^2 statistic against fixed critical values [100]. This limitation often forces students to manually interpret tables, hindering the decision-making process.

To address this, the software utilizes the `interpolate` method provided by the library `scipy.stats` [101]. This method performs a mathematical interpolation between the critical points tabulated yielding a continuous p-value. This ensures a consistent pedagogical experience, allowing the student to apply the same decision rule ($p < \alpha$, where α denotes the significance level) across all statistical tests within the platform.

3.2.2 Inference for Single Populations

For single-population analysis, the tool provides parametric approaches, focused on the population mean (μ) and non-parametric approaches, focused on the population median (M).

Parametric Approach: One-Sample t-test and Confidence Interval

When the data meets the normality assumption (verified in Section 3.2.1) or the sample size is large ($n \geq 30$) to invoke the Central Limit Theorem, the software utilizes the Student's t-distribution [102, 103]. This operates under the assumption that when the population mean (μ) is unknown, the population standard deviation (σ) is also unknown. Therefore, the standard normal distribution is incorrect for p-value calculations in this context.

To test the null hypothesis $H_0 : \mu = \mu_0$ against an alternative hypothesis ($H_1 : \mu < \mu_0$, $\mu > \mu_0$, or $\mu \neq \mu_0$), the one-sample t-statistic is computed as:

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}, \tag{3.5}$$

where $\bar{x}$ is the sample mean, s is the sample standard deviation, and n is the sample size. This statistic follows a t-distribution with $n - 1$ degrees of freedom.

Given the confidence level $1 - \alpha$, the software calculates the $100(1 - \alpha)\%$ confidence interval (CI) for the mean using the standard error of the mean (SEM):

$$\text{CI} = \left[\bar{x} - t_{1-\alpha/2,n-1} \left(\frac{s}{\sqrt{n}} \right), \bar{x} + t_{1-\alpha/2,n-1} \left(\frac{s}{\sqrt{n}} \right) \right], \tag{3.6}$$

where $t_{1-\alpha/2,n-1}$ denotes the $1 - \alpha/2$ quantile of the Student's t-distribution with $n - 1$ degrees of freedom. These calculations are executed using `scipy.stats.ttest_1samp` for the p-value and `scipy.stats.t.interval` for the exact confidence bounds, providing a unified parametric module.

Non-Parametric Approach: Wilcoxon Signed-Rank Test

For small sample sizes that exhibit significant deviations from normality, the mean ceases to be a reliable measure of central tendency. In these cases, the software implements the Wilcoxon Signed-Rank test to test the null hypothesis $H_0 : M = M_0$ against the alternative hypothesis ($H_1 : M < M_0$, $M > M_0$, or $M \neq M_0$). [104].

The test procedure involves calculating the absolute differences $|d_i|$, where $d_i = x_i - M_0$, excluding any zero differences, and ranking them from smallest to largest. The test statistic W is the sum of the signed ranks:

$$W = \sum_{i=1}^{n_r} \left[\text{sgn}(d_i) \cdot R_i \right], \tag{3.7}$$

where n_r is the reduced sample size, $\text{sgn}(d_i)$ is the sign of the difference, and R_i is the rank of $|d_i|$.

To achieve this, the software uses the function `stats.wilcoxon`, which evaluates if the median of these differences is symmetrically distributed around zero.

Confidence Intervals for the Median

Constructing a confidence interval for a median is mathematically more complex than for a mean. To address this educational challenge, the software offers two methodologies:

Distribution-Free Interval via Order Statistics
This methodology calculates a non-parametric confidence interval for the population median without relying on a priori distributional assumptions. Assuming continuity for the variable under analysis, the number of sample observations falling below the true median follows a Binomial distribution $B(n, 0.5)$. A large-sample normal approximation can then be applied to determine the ranking bounds within the sample [105].

Under this approach, the central limit theorem is leveraged to approximate the binomial variance, yielding a mean $\mu = n/2$ and a standard deviation $\sigma = \sqrt{n}/2$. Using the critical value $z_{1-\alpha/2}$ (quantile $1 - \alpha/2$ of the standard normal distribution) for the desired confidence level, the exact index positions within the sorted sample array are defined as:

$$\text{lower\_idx} = \max\left(0, \left\lfloor \frac{n}{2} - z_{1-\alpha/2}\frac{\sqrt{n}}{2} \right\rfloor\right), \tag{3.8}$$

$$\text{upper\_idx} = \min\left(n-1, \left\lceil \frac{n}{2} + z_{1-\alpha/2}\frac{\sqrt{n}}{2} \right\rceil\right). \tag{3.9}$$

By sorting the dataset in ascending order, the values occupying these calculated ranks establish the distribution-free boundaries.

Bootstrap Confidence Interval
To introduce students to modern computational statistics, the software includes a Bootstrap approach. Bootstrapping creates an empirical sampling distribution of the median by resampling the original dataset with replacement thousands of times [106].

The final confidence interval is extracted via the percentile method, taking the $(\alpha/2)$ and $(1-\alpha/2)$ quantiles directly from the empirical bootstrap distribution, which eliminates any reliance on underlying population symmetry or closed-form variance calculations.

$$\text{CI}_B = \left[\widehat{M}^*_{(\alpha/2)}, \widehat{M}^*_{(1-\alpha/2)}\right], \tag{3.10}$$

where $\hat{M}^*_{(p)}$ represents the p-th percentile of the bootstrapped medians.

3.2.3 Inference for Two Independent Populations

This module relies on `NumPy` for data manipulation and `scipy.stats` for the execution of statistical tests. All analytical procedures implemented within this framework focus on the comparison of independent populations, assuming that data groups possess no statistical correlation or common subjects.

Alternative methodologies exist for paired or dependent experimental designs, such as the dependent t-test and the Pitman-Morgan test for correlated variances. Although the software does not natively feature these dependent variants, paired datasets can still be analyzed within this framework by transforming them into a single difference variable and utilizing the previously described one-sample tests. Nonetheless, the automated pipeline remains optimized for separate, non-overlapping cohorts.

F-test for Equality of Variances

To test the null hypothesis that two normal populations have equal variances, $H_0 : \sigma_1^2 = \sigma_2^2$, against the alternative hypothesis ($H_1 : \sigma_1^2 < \sigma_2^2$, $\sigma_1^2 > \sigma_2^2$, or $\sigma_1^2 \neq \sigma_2^2$), the software computes the F-test.

The F-statistic is the ratio of the two sample variances:

$$F = \frac{s_1^2}{s_2^2}, \tag{3.11}$$

where s_1^2 and s_2^2 are the unbiased sample variances calculated via `numpy.var` with `ddof=1`. The software evaluates the p-value using the CDF (`f.cdf`) from `scipy.stats.f` for both one-sided and two-sided hypotheses. Additionally, it computes the confidence interval for the ratio of variances as:

$$\text{CI} = \left[\frac{s_1^2/s_2^2}{F_{1-\alpha/2,\nu_1,\nu_2}}, \quad \frac{s_1^2/s_2^2}{F_{\alpha/2,\nu_1,\nu_2}} \right], \tag{3.12}$$

where $F_{1-\alpha/2,v_1,v_2}$ and $F_{\alpha/2,v_1,v_2}$ denote respectively the quantiles $1-\alpha/2$ and $\alpha/2$ of the F distribution with ν_1 and ν_2 degrees of freedom.

The primary limitation of the F-test lies in its extreme vulnerability to minor deviations from normality. When the underlying data exhibit non-normal characteristics, such as asymmetry or heavy tails, the test severely inflates Type I error rates. This leads to false-positive conclusions, mistakenly rejecting the null hypothesis of homoscedasticity when the variances might actually be equal [107, 108].

Levene's Test

It is a robust alternative to the F-test. The software uses the library `stats.levene` with the parameter `center='median'`. This configuration executes the Brown-Forsythe variant, which is less sensitive to non-normal distributions because it transforms the raw data into absolute deviations from the group median [109]. The test statistic W is math-

ematically structured as an F-ratio from a one-way analysis of variance performed on these transformed distances, defined generally for k groups as:

$$W = \frac{(N-k)}{(k-1)} \frac{\sum_{i=1}^{k} n_i (\overline{Z}_{i\cdot} - \overline{Z}_{\cdot\cdot})^2}{\sum_{i=1}^{k} \sum_{j=1}^{n_i} (Z_{ij} - \overline{Z}_{i\cdot})^2}, \tag{3.13}$$

where k represents the number of distinct groups (with $k = 2$ in this specific context), n_i is the sample size of the i-th group, and N is the total number of observations across all groups. The transformed variable is defined as $Z_{ij} = |Y_{ij} - \widetilde{Y}_i|$, where Y_{ij} is the j-th observation in the i-th group and $\widetilde{Y}_i$ is the median of that specific group. Furthermore, $\overline{Z}_{i\cdot}$ denotes the mean of the transformed variables within group i, and $\overline{Z}_{\cdot\cdot}$ signifies the grand mean of all transformed observations.

Two-Sample t-test

To compare two population means, this tool utilizes `scipy.stats.ttest_ind` to test the null hypothesis $H_0 : \mu_1 = \mu_2$ against the alternative hypothesis ($H_1 : \mu_1 < \mu_2$, $\mu_1 > \mu_2$, or $\mu_1 \neq \mu_2$).

When equal variances are assumed, the t statistic is calculated as:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}, \tag{3.14}$$

where s_p is the pooled standard deviation, defined by:

$$s_p = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}. \tag{3.15}$$

Based on these parameters, the software determines the $100(1-\alpha)\%$ confidence interval for the difference between the population means $(\mu_1 - \mu_2)$ through the formulation:

$$\text{CI} = \left[(\bar{x}_1 - \bar{x}_2) - t_{1-\alpha/2,\nu} \cdot s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}, (\bar{x}_1 - \bar{x}_2) + t_{1-\alpha/2,\nu} \cdot s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}} \right], \tag{3.16}$$

where $t_{1-\alpha/2,\nu}$ represents the critical value from the Student t-distribution with $\nu = n_1 + n_2 - 2$ degrees of freedom.

Conversely, if variances are unequal, the software automatically applies Welch's t-test, where the t statistic is defined as:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}. \tag{3.17}$$

In this case, the degrees of freedom are adjusted using the Satterthwaite-Welch

approximation to maintain the validity of the p-value under heteroscedasticity.

It serves as an optimal option when both groups are normally distributed, or when the sample sizes are large enough ($n_1, n_2 \geq 30$) to rely on the central limit theorem to mitigate minor distributional asymmetries [110, 111].

For this condition, the corresponding confidence interval for the mean difference is computed as:

$$\text{CI} = \left[(\bar{x}_1 - \bar{x}_2) - t_{1-\alpha/2,\nu} \cdot \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}},\ (\bar{x}_1 - \bar{x}_2) + t_{1-\alpha/2,\nu} \cdot \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}} \right], \tag{3.18}$$

where the adjusted degrees of freedom ν are derived from the Satterthwaite-Welch equation.

Mann-Whitney U Test

When the data does not meet the requirements for parametric tests, the software implements the Mann-Whitney U test (`stats.mannwhitneyu`). This procedure detects differences in the distribution of values between two independent samples, testing the null hypothesis $H_0 : P(X > Y) = 0.5$ against the alternative hypothesis ($H_1 : P(X > Y) < 0.5$, $P(X > Y) > 0.5$, or $P(X > Y) \neq 0.5$), where X and Y represent randomly selected observations from the respective populations.

The statistic is calculated by ranking all observations R_i:

$$U_i = R_i - \frac{n_i(n_i + 1)}{2}. \tag{3.19}$$

In the presence of heavy skewness, non-linear data scaling, or prominent outliers, this non-parametric methodology remains robust. This resilience comes from its operation on the relative ordering of data rather than raw magnitudes, which protects the nominal Type I error rate and maintains high statistical efficiency without distortion from anomalous extreme values [112, 113].

Bootstrap for difference of medians

Bootstraping addresses the challenge of calculating confidence intervals for the difference of medians when no simple analytical formula exists.

The software utilizes a two-sample bootstrap procedure where the statistic of interest is defined as $\theta = \tilde{x}_1 - \tilde{x}_2$.

The process follows the percentile method, where the original datasets X_1 and X_2 are resampled with replacement B times. In each iteration b, the software computes the difference between the medians of the resampled groups:

$$\widehat{\theta}_b^* = \text{median}(X_{1,b}^*) - \text{median}(X_{2,b}^*). \tag{3.20}$$

By aggregating these iterations, an empirical sampling distribution of the difference is constructed. The $100(1-\alpha)\%$ CI is then derived by extracting the $(\alpha/2)$ and $(1-\alpha/2)$ percentiles from this distribution:

$$\text{CI} = \left[\widehat{\theta}^*_{(\alpha/2)}, \quad \widehat{\theta}^*_{(1-\alpha/2)}\right]. \tag{3.21}$$

One-vs-Rest

To extend the utility of the two-population inferential tests to multi-class categorical variables without requiring manual dataset manipulation, the software implements a One-vs-Rest (OvR) framework.

This module aligns with standard practices in multi-class classification, where inferential statistics frequently serve as a preliminary step for feature selection. In this context, two-sample hypothesis tests operate as formal filter methods to evaluate the relationship between independent variables and the target groups. By testing each feature against the OvR split, the system identifies variables that exhibit significant differences, a process that reduces dimensionality and isolates the most relevant predictors [114–116].

This feature dynamically re-encodes a categorical variable with $k > 2$ levels into a dichotomous factor. The user selects a specific category of interest to constitute the "One" group (G_{One}), while the software automatically pools all remaining categories to form the "Rest" group:

$$G_{\text{Rest}} = \bigcup_{i \neq \text{One}} G_i. \tag{3.22}$$

This architecture operates as a pragmatic heuristic for target-centric research, benchmarking, and exploratory data analysis, offering a highly efficient alternative to exhaustive pairwise comparisons by directly isolating the distinctiveness of a single population against a global baseline.

Although highly practical for operational workflows, this heuristic pooling introduces specific statistical trade-offs that must be considered. Mathematically, the "Rest" group ceases to be homogeneous and instead forms a mixture distribution, which frequently exhibits multimodality or severe skewness that violates standard parametric normality assumptions.

Furthermore, the baseline variance incorporates the internal differences between the pooled sub-populations, systematically inducing heteroscedasticity. Consequently, while this heuristic approach optimizes practical utility and simplification, users must account for these distributional distortions, along with potential masking effects that can affect overall statistical power.

3.2.4 Inference for k Populations

This module integrates `scipy.stats` for core tests, `Statsmodels` for standard post-hoc analysis, and `Pingouin` for complementary alternatives.

Tests for Homoscedasticity

The software evaluates the equality of variances across k groups, testing the null hypothesis $H_0 : \sigma_1^2 = \sigma_2^2 = \cdots = \sigma_k^2$ against the alternative hypothesis $H_1 : \sigma_i^2 \neq \sigma_j^2$ for at least one pair (i, j). This evaluation is conducted via two implemented methods: Bartlett's Test and Levene's Test.

Bartlett's Test

Implemented in `stats.bartlett` is optimal for situations where the data across all k groups are verified to follow a normal distribution [108].

The test statistic B is calculated as:

$$B = \frac{(N-k)\ln(s_p^2) - \sum_{i=1}^{k}(n_i - 1)\ln(s_i^2)}{1 + \frac{1}{3(k-1)}\left(\sum_{i=1}^{k}\frac{1}{n_i - 1} - \frac{1}{N-k}\right)}, \tag{3.23}$$

where s_i^2 is the variance of the i-th group and s_p^2 is the pooled variance.

In non-normal scenarios, Bartlett's test inflates the Type I error rate, generating a high frequency of false-positive results that mistakenly reject the null hypothesis of homoscedasticity [107].

Levene's Test

As a robust alternative to Bartlett's test for multiple groups, the software has the Levene's test (`stats.levene`). The underlying methodology, the mathematical formulation of the W statistic, and the specific configuration utilizing the Brown-Forsythe variant are completely identical to the procedure detailed in the two-population analysis.

One-Way Analysis of Variance (ANOVA)

To test the null hypothesis that k populations have equal means $H_0 : \mu_1 = \mu_2 = \cdots = \mu_k$, against the alternative hypothesis $H_1 : \mu_i \neq \mu_j$ for at least one pair (i, j), the software implements the F-test via `stats.f_oneway`:

$$F = \frac{MSB}{MSW} = \frac{\sum_{i=1}^{k} n_i(\bar{x}_i - \bar{x})^2/(k-1)}{\sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \bar{x}_i)^2/(N-k)}, \tag{3.24}$$

where MSB represents the mean square between groups, MSW is the mean square within groups, and N is the total sample size. Under the null hypothesis, this test statistic follows an F-distribution with $k - 1$ numerator degrees of freedom and $N - k$ denominator degrees of freedom

ANOVA represents a powerful and efficient tool when the data within each group follow a normal distribution or when group sample sizes are large enough to safely invoke the central limit theorem [117, 118].

The `stats.f_oneway` funcion includes an `equal_var` parameter to handle variables that violate the assumption of homoscedasticity. When set to `True` (the default operational state), the system performs the standard one-way ANOVA test described above. Conversely, when set to `False`, the function dynamically executes Welch's ANOVA test.

Welch's ANOVA is an alternative formulation designed to maintain statistical validity under variance heterogeneity and unbalanced sample sizes. Instead of utilizing a pooled within-group variance estimate (MSW), Welch's method weights each group based on its specific sample variance and size using the term $w_i = n_i/s_i^2$, where s_i^2 represents the sample variance of group i.

The test statistic is adjusted around these weights by correcting the denominator degrees of freedom (df_{Welch}) via the following formulation:

$$df_{\text{Welch}} = \frac{k^2 - 1}{3\sum_{i=1}^{k} \frac{1}{n_i - 1}\left(1 - \frac{w_i}{\sum_{j=1}^{k} w_j}\right)^2}. \tag{3.25}$$

Post-hoc Pairwise Comparisons

The software provides three distinct pathways for pairwise comparisons of means:

Tukey's HSD
Used when variances are equal. It utilizes the Studentized Range distribution to control the Family-Wise Error Rate (FWER) [119]. The software leverages `pairwise_tukeyhsd` from `statsmodels.stats.multicomp`.

Games-Howell Test
Implemented via the `Pingouin` library for cases where the assumption of homoscedasticity is violated. It calculates the confidence intervals using Welch's degrees of freedom (ν):

$$\text{CI} = \left[\left(\bar{x}_i - \bar{x}_j\right) - q_{\alpha,k,\nu}\sqrt{\frac{1}{2}\left(\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}\right)},\ \left(\bar{x}_i - \bar{x}_j\right) + q_{\alpha,k,\nu}\sqrt{\frac{1}{2}\left(\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}\right)} \right], \tag{3.26}$$

where $q_{\alpha,k,\nu}$ denotes the critical value from the Studentized range distribution at a significance level α for k groups and ν degrees of freedom, s_i^2 and s_j^2 represent the sample variances, and n_i and n_j are the sample sizes for groups i and j, respectively.

Kruskal-Wallis H-test

As a non-parametric alternative to ANOVA, the software implements the Kruskal-Wallis test. This test evaluates the null hypothesis that k population medians are equal ($H_0 : M_1 = M_2 = \cdots = M_k$) against the alternative hypothesis ($H_1 : M_i \neq M_j$ for at least one pair (i, j)).

Because its mathematical framework converts raw numerical distances into a unified pooled ranking system, it successfully eliminates the biasing impact of extreme outliers and severe skewness, protecting the nominal Type I error rate without requiring complex data transformations [120, 121].

The test statistic H is mathematically derived as follows:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1), \tag{3.27}$$

where R_i is the sum of ranks for the i-th group, n_i represents the sample size of that specific group, and N signifies the total number of combined observations across all groups.

This is executed via `stats.kruskal`, providing a p-value that indicates whether at least one sample stochastically dominates another.

Pairwise Bootstrap for Median Differences

Similarly to the two-population case, the bootstrap methodology is utilized to obtain the confidence intervals for the difference of medians. To accommodate multiple groups, this implementation adds an outer loop that systematically evaluates all unique pairwise combinations.

3.2.5 Proportion Tests

This module is designed to evaluate categorical variables that represent binary outcomes, such as success or failure rates.

One-Sample Proportion Tests

To evaluate a single population proportion against a hypothesized baseline value p_0, the software provides two inferential methods based on sample size constraints.

The tool has the exact binomial test (`scipy.stats.binomtest`) for small or exact sample sizes. Under the null hypothesis $H_0 : p = p_0$, the probability of observing exactly x successes in n independent trials is derived directly from the binomial probability mass function:

$$P(X = x) = \binom{n}{x} p_0^x (1 - p_0)^{n-x}. \tag{3.28}$$

Conversely, when the sample size satisfies large-sample approximations, the software has the asymptotic one-sample Z-test (`statsmodels.stats.proportions_ztest`), which evaluates the standard normal test statistic:

$$Z = \frac{\hat{p} - p_0}{\sqrt{\frac{p_0(1-p_0)}{n}}}, \tag{3.29}$$

where $\hat{p} = x/n$ represents the empirical sample proportion.

To complement these hypothesis tests, the software estimates single-proportion confidence intervals using the `statsmodels.stats.proportion.proportion_confint` function, specifically implementing the Wilson score and Clopper-Pearson methodologies. While the Clopper-Pearson interval represents an exact method derived mathematically from the cumulative Beta distribution, which guarantees strict nominal coverage, the Wilson score interval operates as a highly efficient asymptotic alternative based on inverting the score test [122, 123]. The Wilson method provides precise coverage probabilities without the excessive over-conservatism typical of exact intervals, remaining reliable when dealing with small sample sizes or proportions close to the absolute boundaries of zero and one.

Two-Sample Proportion Tests

For comparative analysis between two independent populations, the software evaluates the null hypothesis that the two underlying success rates are equal ($H_0 : p_1 = p_2$) against the alternative hypothesis ($H_1 : p_1 < p_2$, $p_1 > p_2$, or $p_1 \neq p_2$).

This comparison is executed through the two-sample Z-test for independent proportions via the `statsmodels.stats.proportion.proportions_ztest` function. The test statistic utilizes a pooled proportion estimate, $\hat{p}_0 = (x_1 + x_2)/(n_1 + n_2)$, to calculate the standard error under the assumption of the null hypothesis framework:

$$Z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}_0(1 - \hat{p}_0)\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}, \tag{3.30}$$

where $\hat{p}_1$ and $\hat{p}_2$ denote the independent sample proportions.

Also, the `statsmodels.stats.proportion.confint_proportions_2indep` function constructs confidence intervals for the difference between these proportions ($\theta = p_1 - p_2$), supporting the Wald and Newcombe score methodologies.

The Wald interval relies on a standard asymptotic normal approximation to estimate the variance of the difference, offering a computationally direct approach that performs optimally with large, well-balanced sample sizes. Conversely, to address scenarios where sample sizes are limited, proportions approach the boundaries of zero or one, or zero success counts are observed within the groups, the system implements Newcombe's score method. This technique combines the individual Wilson score intervals of both populations, reliably capturing variance asymmetries and preventing the

systematic undercoverage typically exhibited by the Wald approximation under these extreme boundary conditions and small-sample constraints [124, 125].

Finally, although a dedicated one-versus-rest page for applying these methods in multi-categorical variables is not implemented in the software, users can replicate this analysis by manually binarizing the data. This consolidation can be applied either to the grouping variable to merge multiple independent populations, allowing a specific demographic or region to be contrasted against a broader aggregated baseline, or to the target variable to collapse multiple response categories, thereby isolating a single primary outcome of interest by treating all other alternatives as a unified non-event.

3.2.6 Contingency Tables

The analysis of categorical data is centered on the construction and interpretation of contingency tables, which display the frequency distribution of variables. The software utilizes `pandas.crosstab` to aggregate raw data into these matrices, providing the basis for testing independence and measuring the strength of associations.

Chi-square Test of Independence and Homogeneity

This method consists of evaluating whether two categorical variables are independent or, equivalently, whether the proportions of a categorical outcome are homogeneous across k different populations. While the research question may differ (association vs. comparison of proportions), the underlying mathematical procedure is identical [126, 127].

The test statistic is calculated using `scipy.stats.chi2_contingency` and follows the formula:

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}, \tag{3.31}$$

where O_{ij} represents the observed frequency in the cell of the i-th row and j-th column. The expected frequencies E_{ij} are calculated under the assumption of independence as:

$$E_{ij} = \frac{(\text{Row Total}_i \times \text{Column Total}_j)}{n}. \tag{3.32}$$

The resulting statistic is compared against a chi-square distribution with $(r-1)(c-1)$ degrees of freedom to determine the p-value, where r and c are the total number of rows and columns in the contingency table, respectively, and n denotes the total sample size.

The validity of the chi-square test relies on three core assumptions: (i) the data must be obtained through random sampling from the population, (ii) the observations must be independent, meaning each subject or case contributes to exactly one cell in the contingency table, and (iii) the sample size must be sufficiently large. This large-sample approximation holds true if all expected frequencies E_{ij} are greater than or

equal to 1, and at least 80% of the cells have expected frequencies of 5 or more.

Yates' Continuity Correction

This method consists of an adjustment specifically for 2×2 contingency tables to improve the accuracy of the p-value when the continuous chi-square distribution is used to approximate a discrete binomial distribution [127]. It involves subtracting 0.5 from the absolute difference between observed and expected frequencies before squaring:

$$\chi^2_{\text{Yates}} = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(|O_{ij} - E_{ij}| - 0.5)^2}{E_{ij}}. \tag{3.33}$$

Fisher's Exact Test of Independence

This method consists of calculating the exact probability of the observed configuration of a contingency table under the null hypothesis of independence. It is particularly relevant when the assumptions of the chi-square test are not met, specifically when the total sample size is small or any expected frequency is less than five [128, 129].

Exact Method for 2×2 Tables

For a 2×2 contingency table, the probability of observing a specific configuration of frequencies follows a hypergeometric distribution, assuming fixed row and column marginal totals. The frequencies are organized in a matrix structure where a, b, c, and d correspond to positions $(1,1)$, $(1,2)$, $(2,1)$, and $(2,2)$, as illustrated below:

	Column 1	Column 2	Total
Row 1	a	b	$a+b$
Row 2	c	d	$c+d$
Total	$a+c$	$b+d$	n

The probability P of obtaining the observed set of values is given by:

$$P = \frac{\binom{a+b}{a}\binom{c+d}{c}}{\binom{n}{a+c}} = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{n!a!b!c!d!}, \tag{3.34}$$

where n denotes the grand total, such that $n = a + b + c + d$.

The software utilizes the `scipy.stats.fisher_exact` function to compute the corresponding p-value and returns the sample odds ratio (OR) as the effect size statistic, defined as:

$$\text{OR} = \frac{a \cdot d}{b \cdot c}. \tag{3.35}$$

Monte Carlo Simulation for $r \times c$ Tables

When the contingency table exceeds a 2×2 dimension, calculating the exact distribution by enumerating all possible configurations becomes computationally intractable due to

combinatorial explosion. To address this limitation, `scipy.stats.fisher_exact` provides an approximation of the exact p-value via the `MonteCarloMethod`.

This approach first calculates the probability mass of the observed table (P_{obs}) under the null hypothesis of independence with fixed marginal totals. For an $r \times c$ table, this probability is defined as:

$$P_{\text{obs}} = \frac{\prod_{i=1}^{r} R_i! \prod_{j=1}^{c} C_j!}{n! \prod_{i=1}^{r} \prod_{j=1}^{c} O_{ij}!}, \tag{3.36}$$

where O_{ij} is the observed frequency in row i and column j, R_i represents the marginal sum of row i, C_j represents the marginal sum of column j, and n is the grand total. Subsequently, the algorithm generates a large number of random simulated tables that strictly preserve these exact marginal sums. For each generated table k, its corresponding probability mass (P_k) under the null distribution is evaluated. Finally, the empirical p-value is estimated as the proportion of simulated tables that are at least as extreme as the observed data, which corresponds to the fraction of tables where $P_k \leq P_{\text{obs}}$ [130].

Measures of Association

While significance tests determine the existence of a relationship, the measures of association quantify the strength of the effect. Derived from the chi-square statistic or the direct proportions of the contingency table, these metrics (Cramér's V, Pearson's Contingency Coefficient C, the Phi Coefficient ϕ, and the Odds Ratio OR) are strictly appropriate for nominal data, as their mathematical frameworks do not incorporate any ordinal ranking. They are implemented utilizing `NumPy`, `scipy.stats`, and `Statsmodels`.

Cramér's V

This method consists of a rescaling of the chi-square statistic to a range between 0 and 1, providing a comparable measure for tables of any dimension ($r \times c$) [131]. The implementation utilizes the raw chi-square statistic and follows the formula:

$$V = \sqrt{\frac{\chi^2}{n(k-1)}}, \tag{3.37}$$

where n is the total number of observations and $k = \min(\text{rows}, \text{columns})$. The software handles the edge case where $k = 1$ by returning `np.nan` to prevent division by zero, ensuring mathematical consistency.

Pearson's Contingency Coefficient (C)

This method consists of another approach to quantifying association based on the chi-square statistic [132]. It is calculated as:

$$C = \sqrt{\frac{\chi^2}{\chi^2 + n}}. \tag{3.38}$$

Unlike Cramér's V, the upper limit of C depends on the dimensions of the table and never reaches 1.

Phi Coefficient (ϕ)
This method consists of a measure of association specifically designed for 2×2 contingency tables and it is mathematically equivalent to the Pearson correlation coefficient for two binary variables (see Section 3.2.7) [133]. It is defined as:

$$\phi = \sqrt{\frac{\chi^2}{n}}. \tag{3.39}$$

Odds Ratio (OR)
This method consists of comparing the relative odds of an outcome between two groups in a 2×2 table.

The software utilizes `statsmodels.stats.contingency_tables.Table2x2` to compute the ratio:

$$\text{OR} = \frac{n_{11}/n_{12}}{n_{21}/n_{22}} = \frac{n_{11}n_{22}}{n_{12}n_{21}}, \tag{3.40}$$

where n_{ij} denotes the observed frequency in the i-th row and j-th column of the 2×2 contingency table, with $i, j \in \{1, 2\}$. Additionally, the implementation calculates the $100(1-\alpha)\%$ CI for the OR using the `oddsratio_confint` method.

This is performed in the logarithmic scale to handle the inherent skewness of the ratio distribution:

$$\begin{aligned} \ln(\text{CI}) = \Bigg[& \ln(\text{OR}) - z_{\alpha/2}\sqrt{\frac{1}{n_{11}} + \frac{1}{n_{12}} + \frac{1}{n_{21}} + \frac{1}{n_{22}}}, \\ & \ln(\text{OR}) + z_{\alpha/2}\sqrt{\frac{1}{n_{11}} + \frac{1}{n_{12}} + \frac{1}{n_{21}} + \frac{1}{n_{22}}} \Bigg], \end{aligned} \tag{3.41}$$

The results are then exponentiated back to the original scale to provide interpretable bounds for the association.

3.2.7 Correlation Analysis

The analysis of correlation identifies the strength and direction of the relationship between two continuous variables. The software implements three primary coefficients through the `scipy.stats` library.

Pearson Product-Moment Correlation

Strictly appropriate for continuous data, this method consists of measuring the linear relationship between two variables [134, 135]. The correlation coefficient r is defined as the ratio between the covariance of the variables and the product of their standard deviations:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}\,. \tag{3.42}$$

The software utilizes `scipy.stats.pearsonr` to calculate both the coefficient and the p-value to test whether the population correlation is significantly different from zero. The significance is determined using a t-distribution with $n - 2$ degrees of freedom, where the test statistic is:

$$t = r\sqrt{\frac{n-2}{1-r^2}}. \tag{3.43}$$

The confidence interval is calculated analytically using the Fisher transformation, which converts the correlation coefficient into a normally distributed variable z:

$$z = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right). \tag{3.44}$$

The bounds in the z-space are calculated as $[z - z_{1-\alpha/2}SE_z,\ z + z_{1-\alpha/2}SE_z]$, where $SE_z = 1/\sqrt{n-3}$, and then transformed back to the original scale.

This coefficient is optimal when evaluating continuous variables that exhibit a normal bivariate distribution and share a homoscedastic pattern of variance within a single homogeneous population [135, 136]. The metric possesses acute sensitivity to extreme values, meaning isolated outliers can attenuate or inflate the calculated value, which distorts the true nature of the association [134]. When applied to non-linear structures, the coefficient fails to capture the true underlying dependence, which can result in a value close to zero despite a strong functional connection [137].

Spearman Rank Correlation

Appropriate for ordinal data or continuous variables that deviate from parametric assumptions, this method consists of a non-parametric measure of rank correlation that assesses how well the relationship between two variables can be described using a monotonic function. Rank transformation provides resistance to extreme values, ensuring outliers do not distort the estimation of association strength. However, the metric fails to detect complex non-monotonic dependencies, outputting a coefficient near zero when strong parabolic or cyclical patterns characterize the distribution. Also, small sample sizes increase Type II error rates during null hypothesis significance testing [135, 137].

The coefficient ρ is calculated by applying the Pearson formula to the ranks of the data. For samples without tied ranks, it simplifies to:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}, \tag{3.45}$$

where d_i is the difference between the ranks of each observation.

The p-value is calculated via `scipy.stats.spearmanr` using a t-distribution approx-

imation. Because the sampling distribution of ρ is not easily parameterized for small or non-normal samples, the software computes the 95% confidence interval using a bootstrap procedure with resamples, ensuring a valid estimation.

Kendall's Tau (τ)

Specifically designed for ordinal data, this method consists of a non-parametric measure of association based on the concordance and discordance of paired observations. It is particularly effective for small datasets or data with many tied ranks [135].

The coefficient τ is defined as:

$$\tau = \frac{N_c - N_d}{\frac{1}{2}n(n-1)}, \tag{3.46}$$

where N_c is the number of concordant pairs and N_d is the number of discordant pairs.

The software employs `scipy.stats.kendalltau` to obtain the coefficient and a p-value based on a normal approximation for the null hypothesis of independence. Similar to the Spearman implementation, the confidence interval is derived through a bootstrap approach to circumvent the complexities of the analytical standard error in the presence of ties.

3.3 Complementary Tools for Data Management and Descriptive Exploration

To provide a complete analytical workflow, the platform incorporates dedicated modules for data ingestion, data preparation, and univariate descriptive exploration. These utilities ensure that users can execute the entire data pipeline within a single environment prior to running advanced inferential tests.

The data management foundation relies on the `Pandas` library to handle file parsing, manage diverse character encodings, and segment variables into numeric or categorical subsets. This framework includes data transformation capabilities, such as the discretization of continuous vectors into explicit categorical bins.

Summary statistics and exploratory analysis are powered by vectorized operations within `Pandas`, which deliver rapid and precise summaries of the dataset. The system evaluates the structural properties of the data by calculating sample sizes, dataset dimensions, and complete frequency tables. For numeric distributions, the platform extracts metrics of central tendency, including the mean, median, and mode, alongside dispersion parameters such as variance, standard deviation, range, and the interquartile range. Shape characteristics are further assessed through skewness and kurtosis calculations. To complement these quantitative summaries, the system integrates `Matplotlib` to generate visual distributions. This graphic component renders clean histograms, boxplots, and bar charts to aid in the direct identification of outliers, asymmetry, and data trends.

A core objective of these complementary utilities is the preservation of scientific replicability. By uniting comprehensive data manipulation, descriptive synthesis, and visual diagnostics with reproducible code generation, these tools conclude the technical framework of the software.

3.4 Application Distribution and Deployment

The open-source codebase remains public on `GitHub`,[1] where the repository includes a comprehensive `README` file for manual environment setup.

To streamline the user experience and bypass local dependency configuration, the software is hosted via `GitHub Releases`, which serves as a standard hub for stable software builds and compiled binary assets. This approach ensures that users can deploy the software without prior experience in package management or version control.

The package arrives as a compressed `ZIP` folder that contains the primary file, named `python-stats.exe`. Launching this executable opens a background command-line window to host the local server. As soon as this server becomes active, the default web browser opens to display the application interface.

As a result, this distribution mechanism minimizes the operational barrier for the end user. A single click on the file grants full access to the analytical workspace, ensuring a smooth experience free from software conflicts or technical complications.

[1] `https://github.com/danielTeniente/python-statistical-inference`

4

System Demonstration

This chapter illustrates the software's functionality through three practical scenarios, exploring its various integrated methods. First, a brief overview is first provided on how to obtain the software from `GitHub` and the basic steps required for its execution. Then, the three scenarios use well-known datasets: `penguins`, `toothgrowth`, and `heart`.

The `penguins` dataset contains size measurements for three penguin species observed in the Palmer Archipelago, Antarctica. These data points include bill length, bill depth, flipper length, body mass, and sex across the Adelie, Chinstrap, and Gentoo species, serving as a standard resource for data exploration [138].

For the second scenario, the `toothgrowth` dataset records the effect of Vitamin C on tooth growth in guinea pigs. It monitors the length of odontoblasts in sixty animals subjected to three distinct dose levels of Vitamin C administered through two delivery methods, which are orange juice and ascorbic acid [139].

In the final scenario, the `heart` dataset contains medical attributes from patients evaluated for cardiovascular health. Key features include age, resting blood pressure, cholesterol levels, and maximum heart rate, culminating in a target variable that indicates the presence or absence of heart disease [140].

Finally, this demonstration employs a standard significance level of $\alpha = 0.05$ throughout all the examples.

4.1 Software Acquisition and Execution

First, it is necessary to access the project's official `GitHub`.[1] Within the repository page, the *Releases* section contains the downloadable compressed files (Figure 4.1). From there, the user should download the file titled `python-stats.zip` (Figure 4.2).

Once downloaded and extracted, the main folder contains three essential components: a dedicated directory housing all files and dependencies required to run the program (which the user does not need to modify), the `data` folder containing the CSV

[1] `https://github.com/danielTeniente/python-statistical-inference`

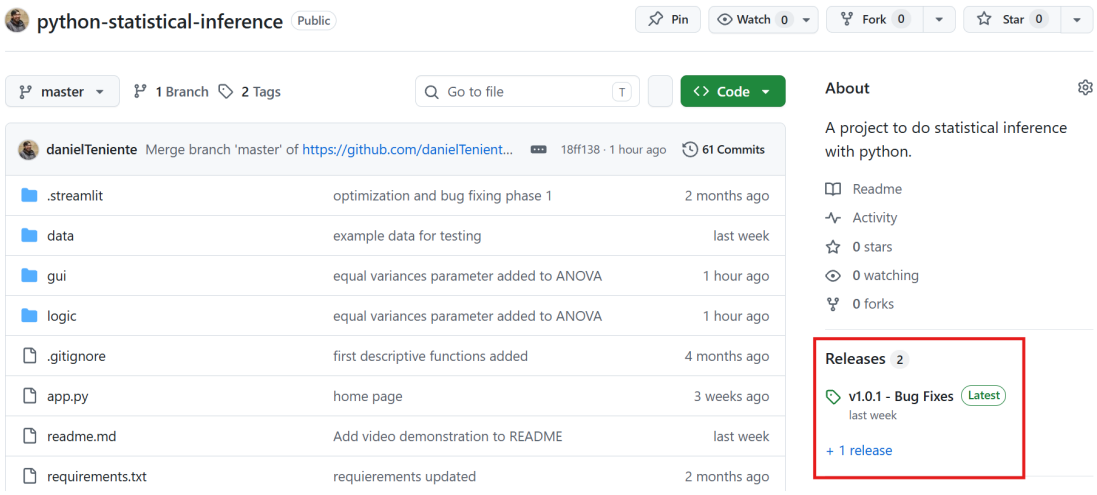


Figure 4.1: *GitHub repository page of the project showing the releases section.*

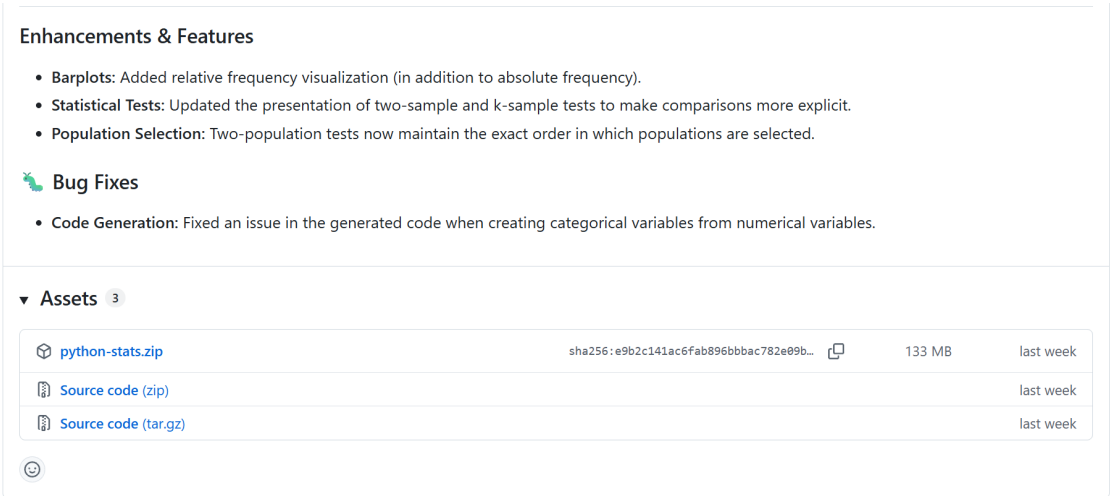


Figure 4.2: *Releases page containing the project ZIP file.*

test datasets utilized in the subsequent demonstrations, and the `.exe` file provided to launch the application (Figure 4.3).

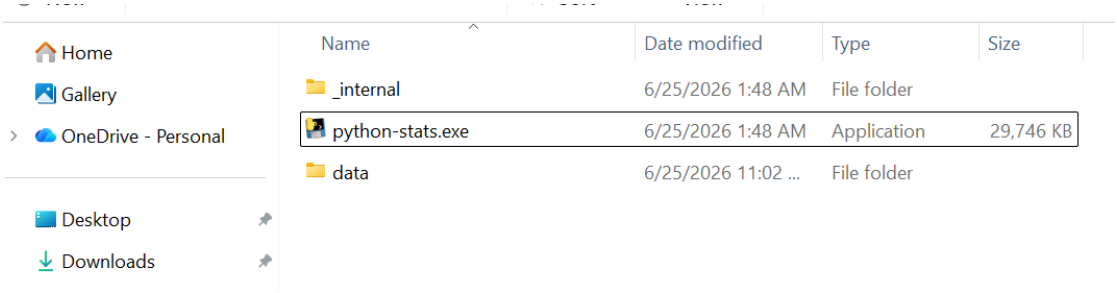


Figure 4.3: *Contents of the project ZIP file.*

To start the software, the user must double-click the executable file. This action automatically opens a Windows command prompt console, indicating that the program is running and a local server has been successfully hosted (Figure 4.4). Following this, the default web browser opens automatically to display the program's graphical user interface (Figure 4.5). It is crucial to note that the application will remain operational only as long as the console window stays open. If the console is closed, the local server will terminate, and the interface in the browser will cease to function.

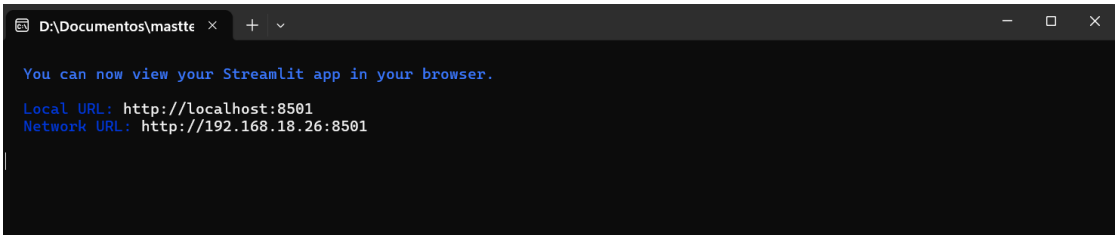


Figure 4.4: *Local server console used to deploy the application.*

Figure 4.5: *Application running on the local server, opened via the default web browser.*

As demonstrated, no installation procedures are required, as the software is distributed as a standalone executable file that allows for direct deployment without requiring a pre-existing Python installation on the host system. With the environment

fully established, it is now possible to proceed with the practical demonstration utilizing the test datasets.

4.2 Scenario #1

This scenario focuses on two objectives: evaluating variations in mean flipper length across species and analyzing the correlation between flipper length and body mass.

The `Home` page serves as the entry point for the application (Figure 4.6). It presents the project title, institutional details, and its educational purpose: providing executable statistical tools with visible Python code for independent replication. All available data analysis options are hosted in the left navigation menu.

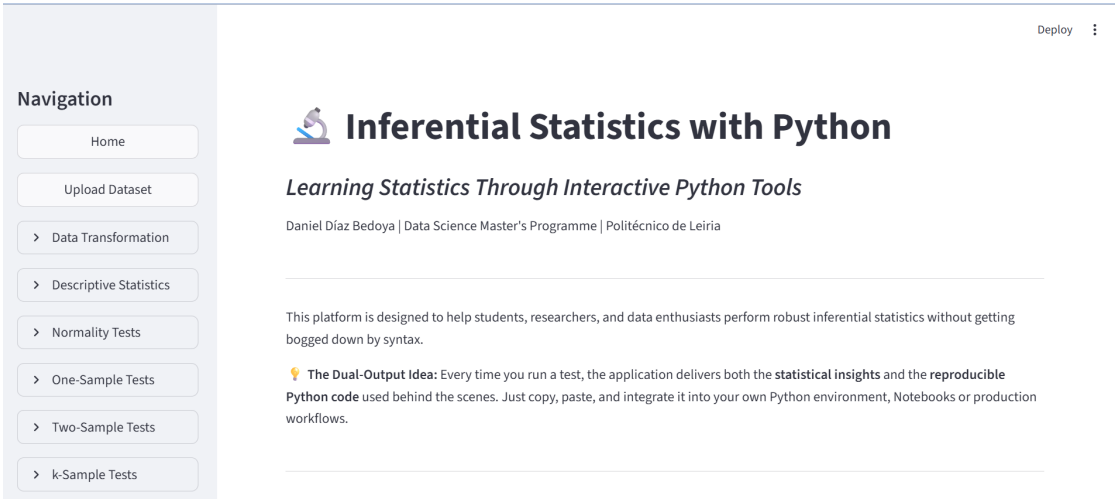


Figure 4.6: *Home page of the application.*

The software requires a CSV dataset as a prerequisite for any data analysis. As illustrated in Figure 4.7, if no dataset has been uploaded, the application pages restrict access to the analytical features, prompting the user to import a file.

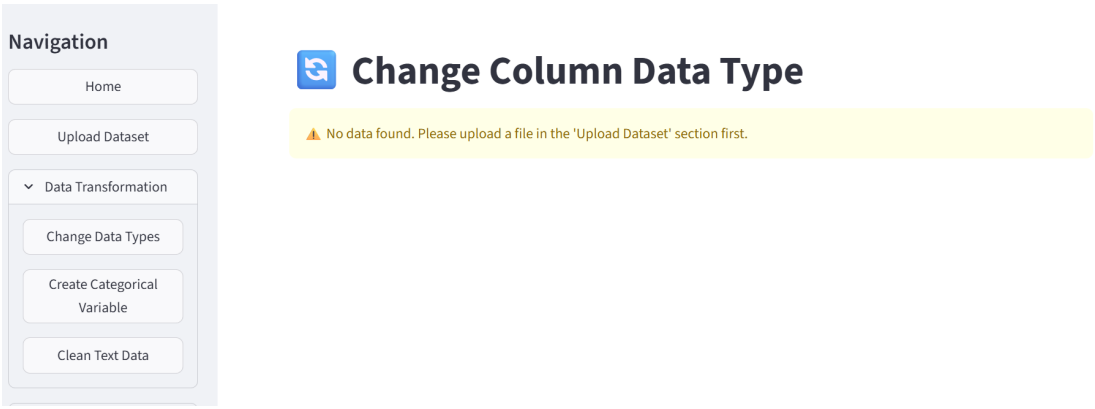


Figure 4.7: *Application interface state when no dataset has been loaded.*

As illustrated in Figure 4.8, the `Upload Dataset` page provides a user-friendly interface to load files via drag-and-drop or a standard file browser. The application in-

cludes settings for data formatting, allowing users to adjust the field separator and choose from widely used character encodings. By default, the automatic detection feature makes an assumption about the file's formatting, sparing the user from manual adjustment.

Consistent with the application's educational objective, loading a dataset immediately triggers the display of the relevant `Python` code at the bottom of the screen. In this instance, the interface outputs the precise `Pandas` code needed to import the file, ensuring users can understand and replicate the data ingestion process locally.

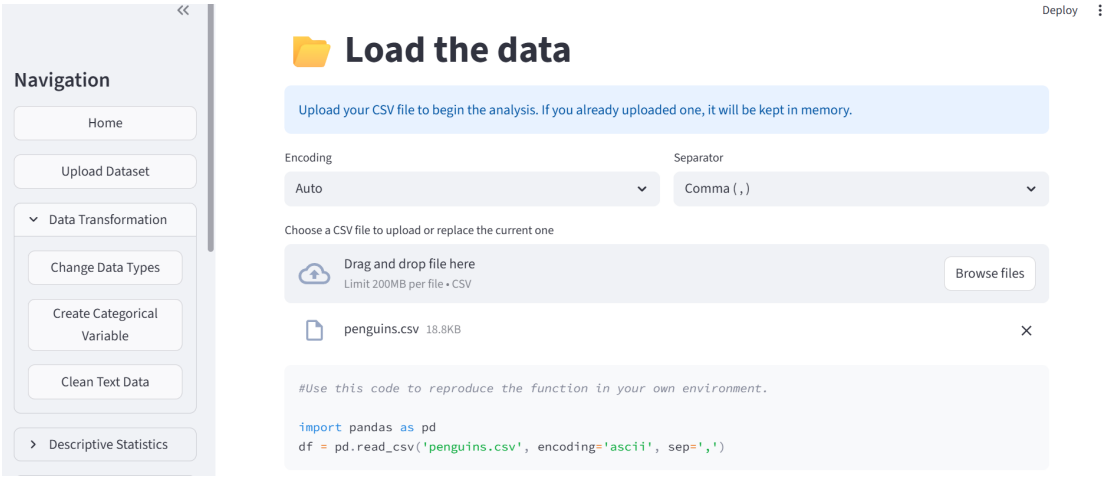


Figure 4.8: *Loading a dataset via file browsing or drag-and-drop.*

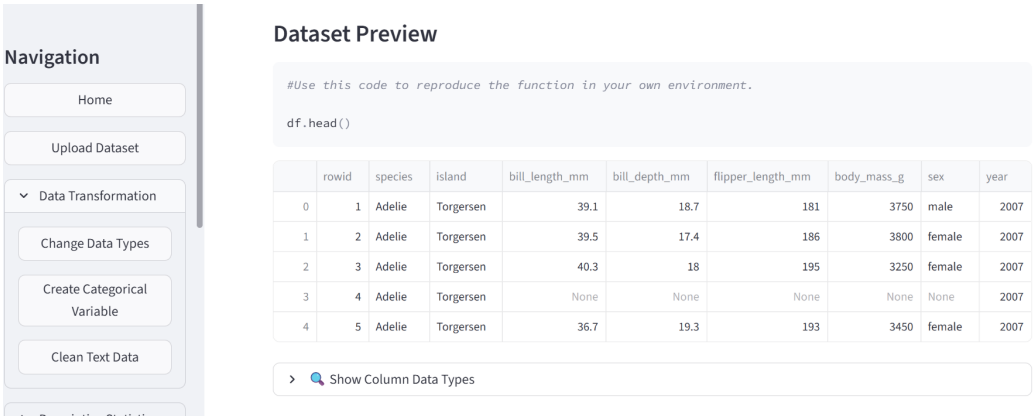
To facilitate data verification, the `Upload Dataset` page includes two components below the ingestion area. The first is a data preview that displays the dataset's head through a `Streamlit` table, enabling a quick check of the initial values (Figure 4.9a). And this is followed by a second table showcasing the default `Pandas` data types (Figure 4.9b), mapping out the data structure automatically recognized by the application.

Within the `Descriptive Statistics` section, the `Numerical Variables` page allows users to evaluate the dataset's key characteristics.

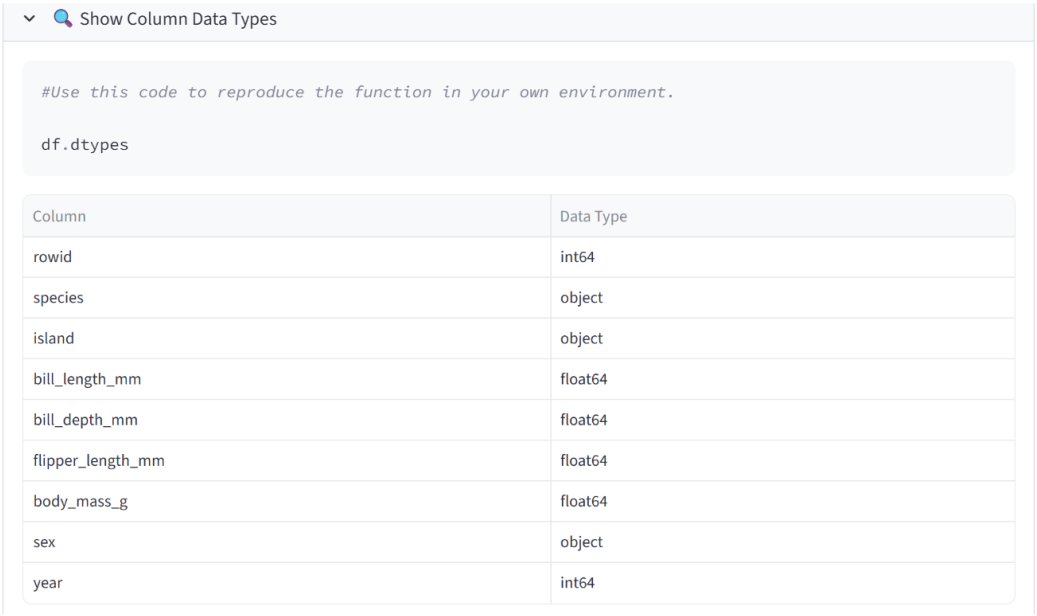
A statistical overview of the dataset is rendered as a native `Streamlit` table (Figure 4.10) once the user clicks the `Generate General Summary` button. This process utilizes the `Pandas` `df.describe` method and also computes the dataset's dimensions via `shape`. As displayed in this scenario, the sample dataset comprises 344 records and 9 variables.

Following the general summary, the interface provides a selection tool to isolate and analyze individual features in greater detail. In this scenario, selecting the flipper length variable `flipper_length_mm` yields an expanded breakdown of its key central tendency and dispersion metrics. Although Figure 4.11 captures only the initial portion of the summary, which includes the mean, median, mode, variance, and standard deviation, the interface additionally provides the minimum, maximum, range, quartiles, interquartile range, skewness, and kurtosis.

Finally, the `Data Visualization` subsection includes a dropdown menu to select a numerical column, accompanied by a slider to define the number of bins for the his-

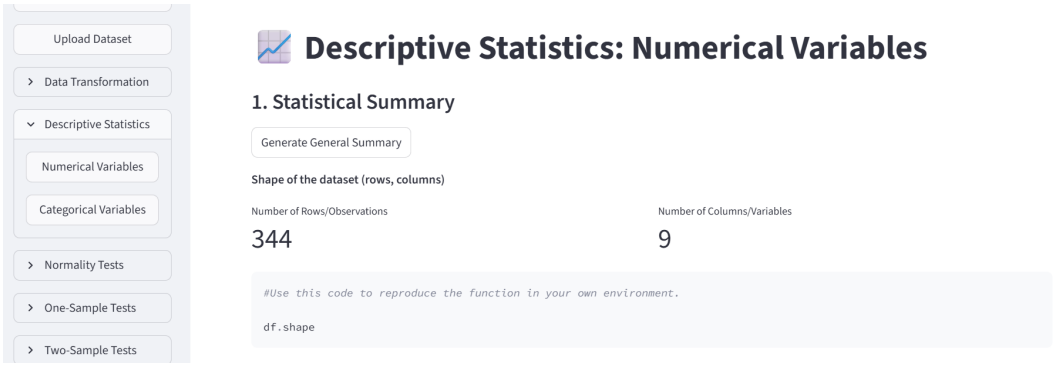


(**a**) *Table preview of the rows and columns in the dataset.*

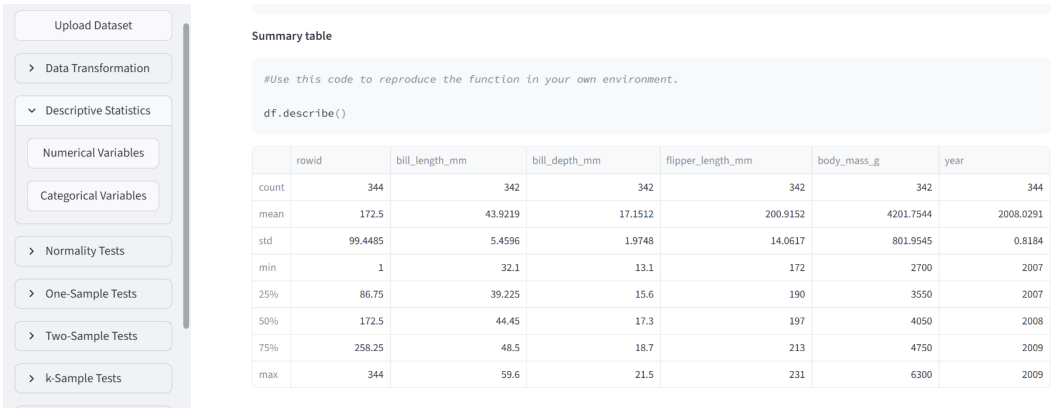


(**b**) *Default data types assigned to the loaded variables.*

Figure 4.9: *Preview of the loaded dataset*



(**a**) *Shape of the dataset.*



	rowid	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	year
count	344	342	342	342	342	344
mean	172.5	43.9219	17.1512	200.9152	4201.7544	2008.0291
std	99.4485	5.4596	1.9748	14.0617	801.9545	0.8184
min	1	32.1	13.1	172	2700	2007
25%	86.75	39.225	15.6	190	3550	2007
50%	172.5	44.45	17.3	197	4050	2008
75%	258.25	48.5	18.7	213	4750	2009
max	344	59.6	21.5	231	6300	2009

(**b**) *Descriptive statistics for the numerical variables.*

Figure 4.10: *Summary of descriptive statistics.*

togram. In this scenario, Sturges' rule is applied to determine the optimal bin count. Given that the dataset contains $n = 344$ observations, the rule dictates that the number of classes k is calculated as $k = 1 + \log_2(344) \approx 9.42$. Following the standard practice in common Python and R implementations, which utilize a `ceiling` function when performing this calculation, the final number of bins is rounded up to 10 [141, 142].

After configuring these parameters, users must click the `Generate Visualizations` button to render both the histogram and the boxplot to evaluate the variable's distribution (Figure 4.12a and Figure 4.12b).

Within the `Data Visualization` subsection, users can also analyze data distributions by category by enabling a checkbox. Activating this element dynamically reveals a dropdown menu to select the grouping variable (Figure 4.13a).

For instance, selecting the species as the grouping criteria and clicking the `Generate Visualizations` button renders a boxplot that illustrates the flipper length distribution across groups (Figure 4.13b).

A visual analysis of these boxplots reveals a distinct variation in length, supporting the hypothesis that the mean flipper length varies across species. Also, the distribution of Adelie exhibits the presence of outliers. In `Matplotlib`, these outliers are identified by extending the whiskers to a maximum of 1.5 times the interquartile range beyond the first and third quartiles. Consequently, any data point falling outside these boundaries is considered an outlier.

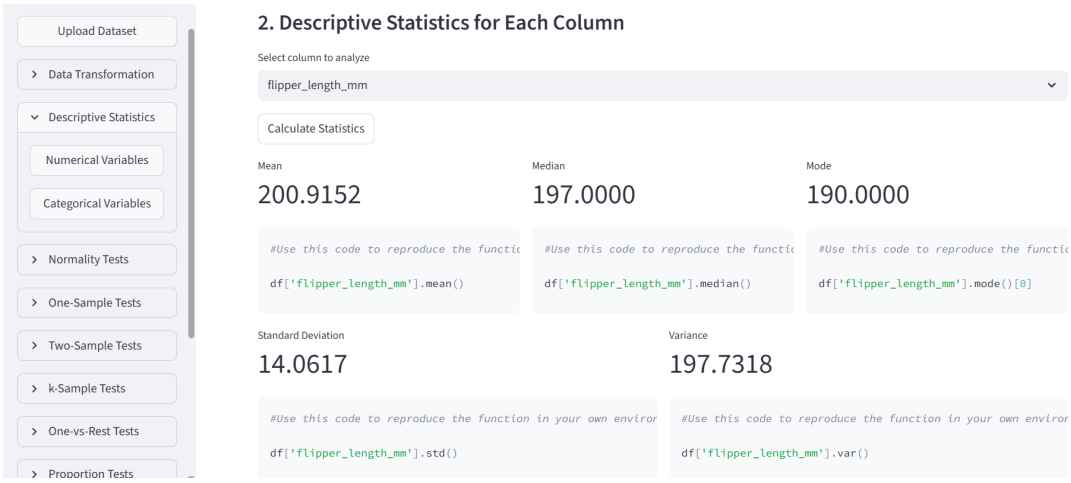
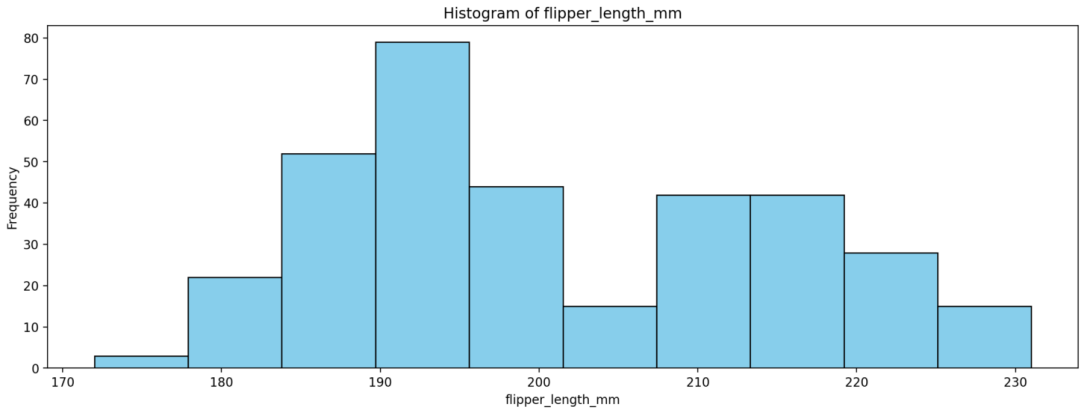
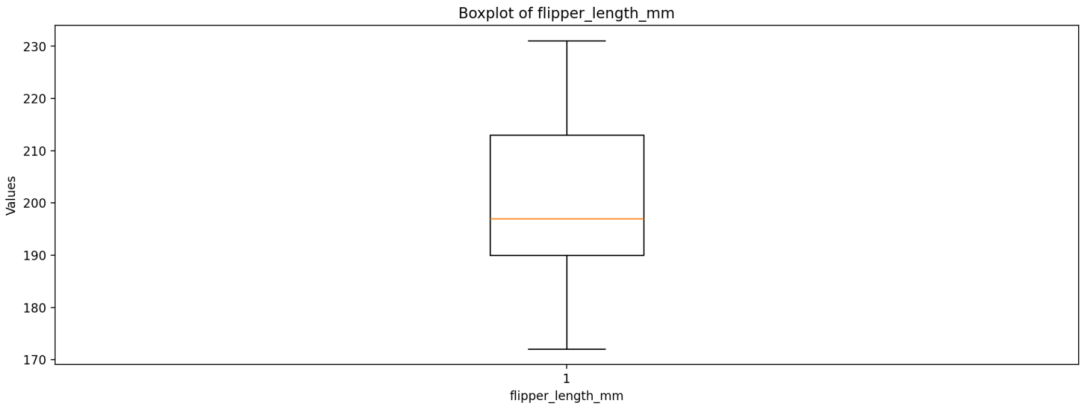


Figure 4.11: *Detailed descriptive statistics for the flipper length variable.*

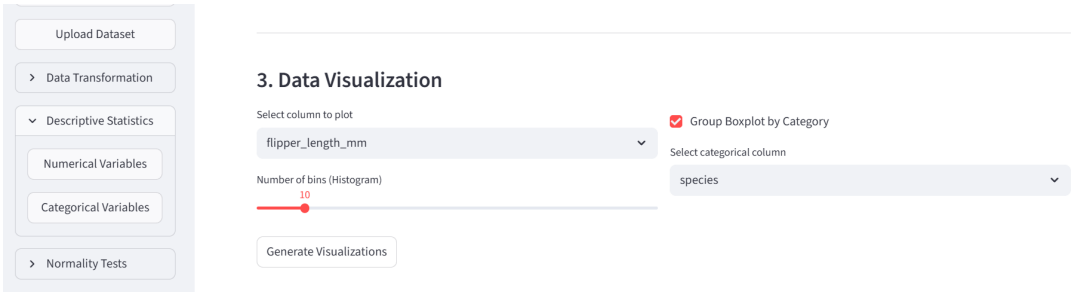


(**a**) *Histogram of the selected variable.*

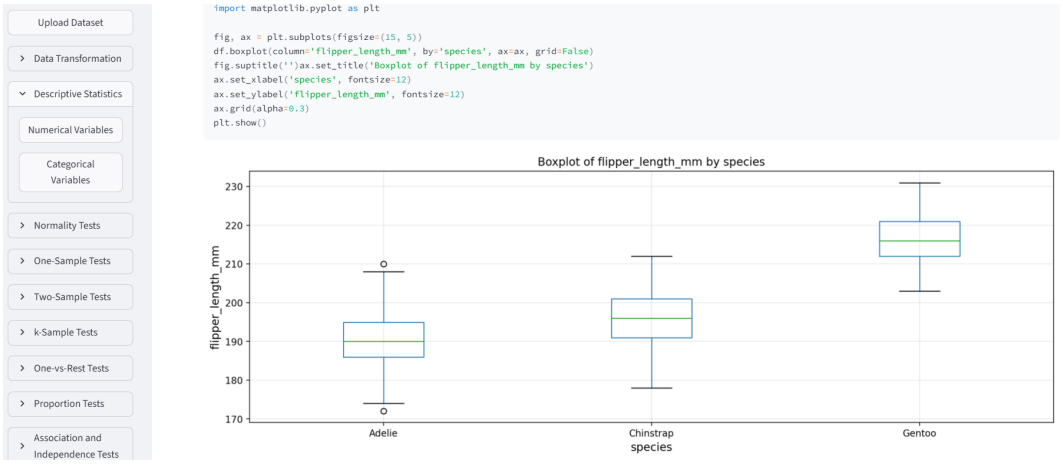


(**b**) *Boxplot of the selected variable.*

Figure 4.12: *Descriptibe plots generated by the software*



(**a**) *Plot configuration settings for categorical analysis.*



(**b**) *Boxplot displaying flipper length variations across species.*

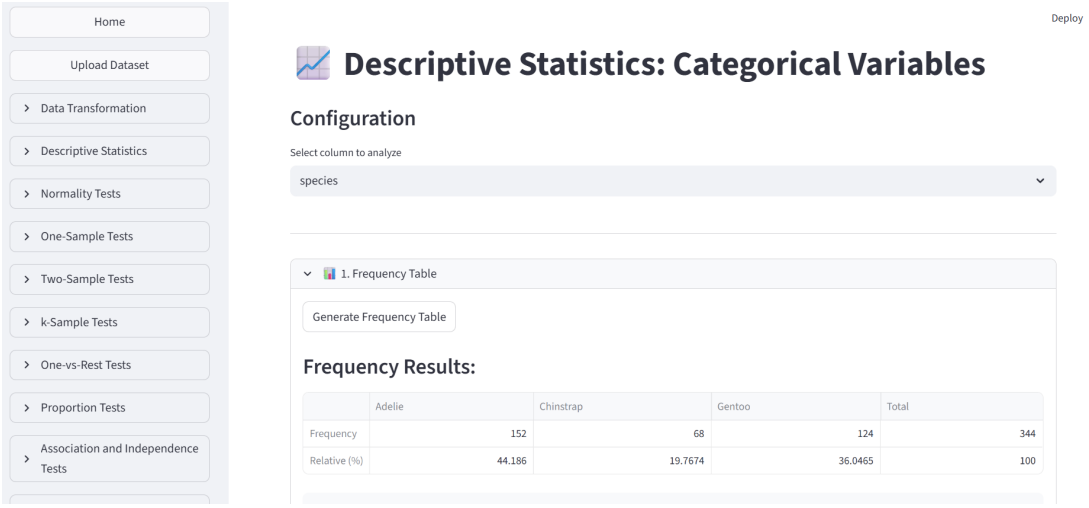
Figure 4.13: *Distribution of flipper length categorized by penguin species.*

Users can isolate and analyze qualitative variables via a dropdown selector by visiting the `Categorical Variables` page, located under the `Descriptive Statistics` section. For the calculation of relative frequencies, any missing values (`NaN`) are automatically omitted, ensuring percentages are computed exclusively over valid observations. Choosing the `species` variable generates a detailed frequency table (Figure 4.14a). This overview makes it evident that all categories have a sample size greater than 30, which satisfies a standard prerequisite for performing robust statistical testing. Furthermore, the application renders a corresponding bar plot directly beneath the tabular data, providing a visual representation of the distribution among the categories (Figure 4.14b).

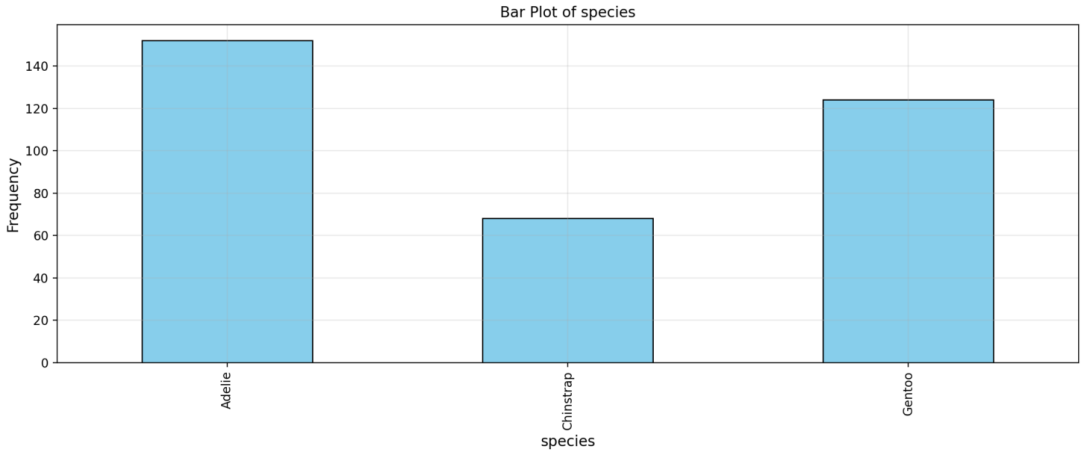
To complement this view, the interface includes a checkbox that allows users to dynamically toggle the chart to display relative frequencies instead of absolute values (Figure 4.15a and Figure 4.15b).

To advance with the analysis, the `Normality tests` section in the left menu provides two distinct analytical options: `Whole Sample Normality` and `Normality Tests by Group`.

Although the primary objective of this scenario is the categorical evaluation of flipper length by species, both modules were utilized for demonstration purposes. The `Whole Sample Normality` interface features selectors for both the target numerical variable and the statistical test, subsequently rendering the outputs across two subsections:

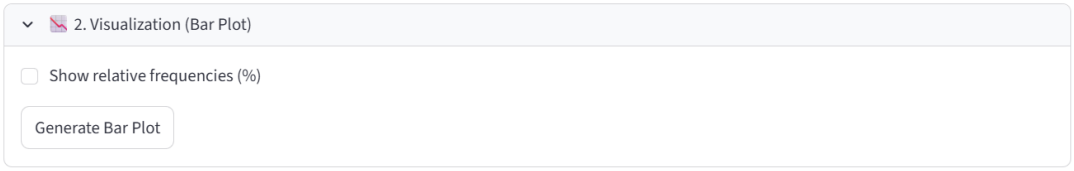


(**a**) *Frequency table for the categorical variable.*

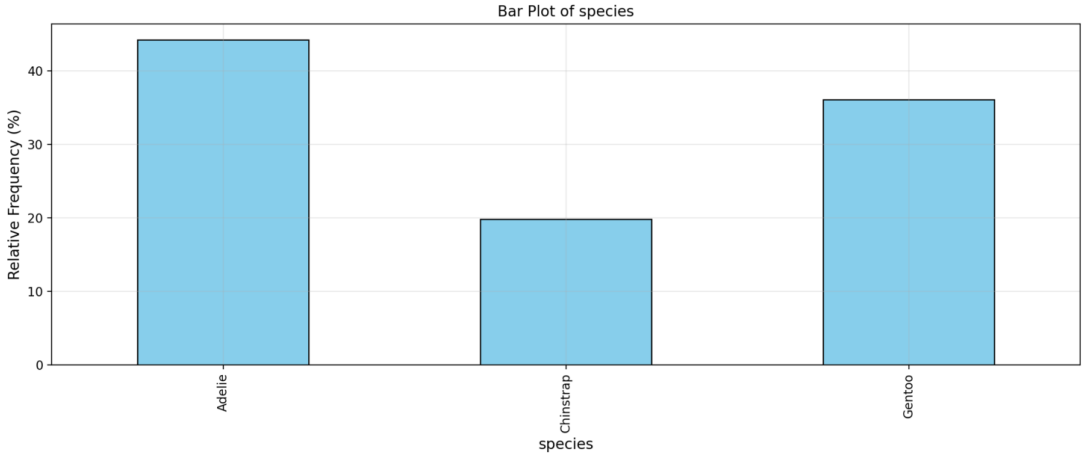


(**b**) *Barplot of the categorical variable.*

Figure 4.14: *Distribution of flipper length categorized by penguin species.*



(**a**) *Checkbox interface for relative frequency selection.*



(**b**) *Relative frequency barplot.*

Figure 4.15: *Relative frequency visualization settings and corresponding barplot output.*

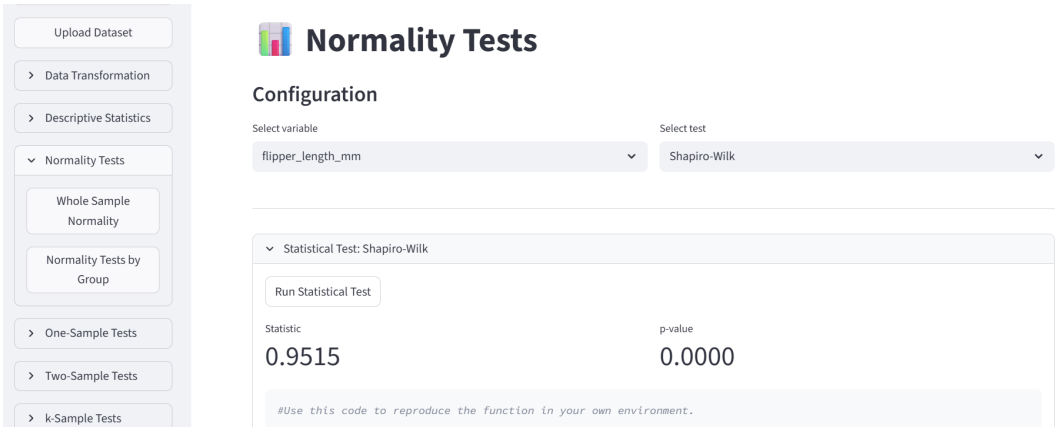
one containing the test metrics and another displaying a QQ-plot (Figure 4.16b). Executing this test for the entire population yields a p-value of zero (Figure 4.16a).

Alternatively, the `Normality Tests by Group` subsection handles categorical distributions. Its interface requires the user to define the target numerical attribute, the grouping variable, and the test method (Figure 4.17a). A key feature of this page is a multi-selection input, which grants the flexibility to include or exclude specific classes from the computation.

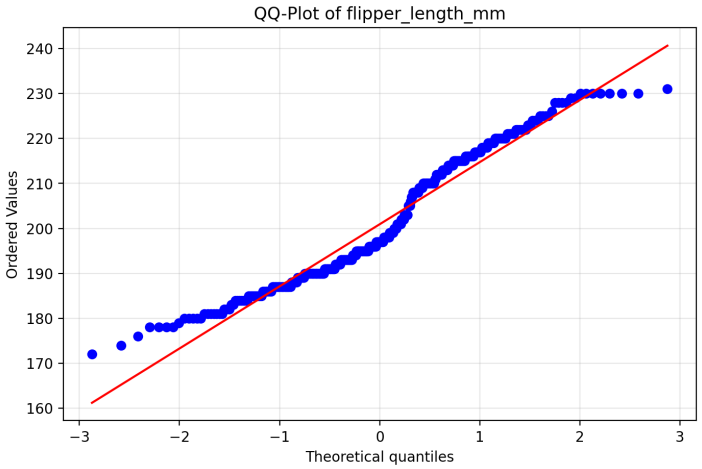
For this demonstration, all three available species categories were kept active. Partitioning the analysis under these parameters reveals that the Gentoo category violates the normality assumption with a p-value below the standard significance level (Figure 4.17b). However, since this category maintains a sample size of 124 observations, the Central Limit Theorem applies, allowing the workflow to proceed safely.

The next stage of the analysis is hosted in the `k-Sample Tests` module, specifically within the `k-Sample Variance Tests` page.

The configuration panel features interactive selectors for the numerical variable, the grouping criteria, and a multi-selection component to filter the active classes under study (Figure 4.18a). Directly beneath these settings, a method selector allows users to choose the variance testing algorithm. Lower down on the page, a selector enables the choice of the statistical test, which in this scenario is set to Levene's test. The execution populates a subsection where the output displays a p-value of 0.7188 (Figure 4.18b). Since this value exceeds the conventional significance threshold, the application fails

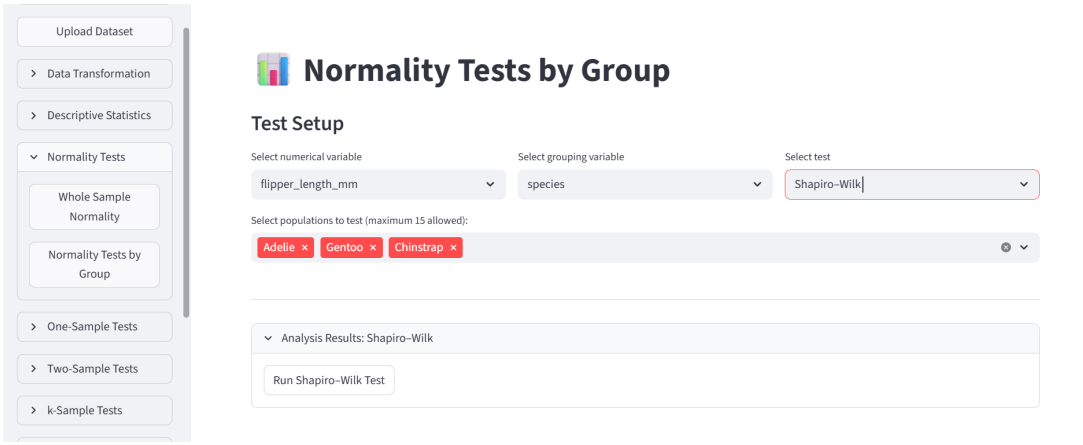


(**a**) *Normality test results calculated for the entire population.*

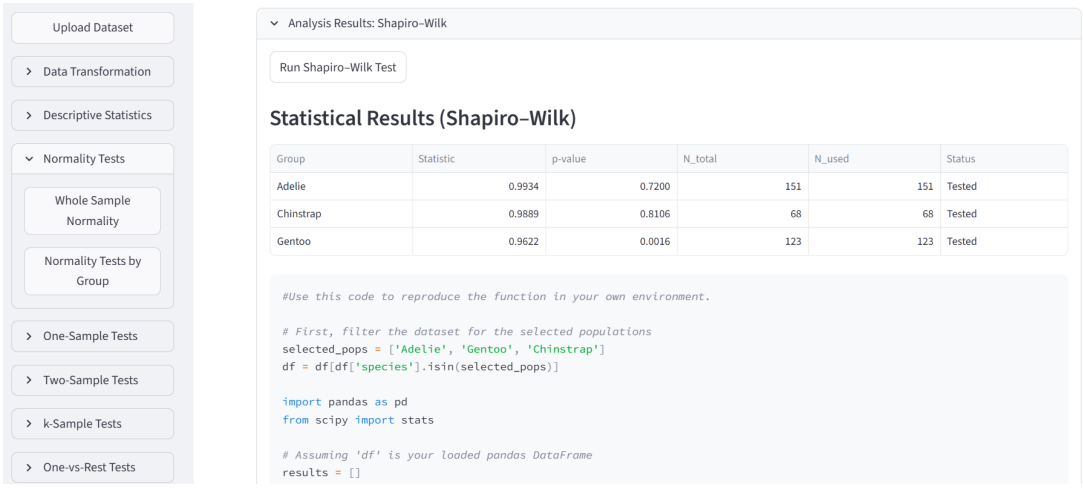


(**b**) *QQ-Plot generated by the software.*

Figure 4.16: *Whole Sample Normality options available in the software*



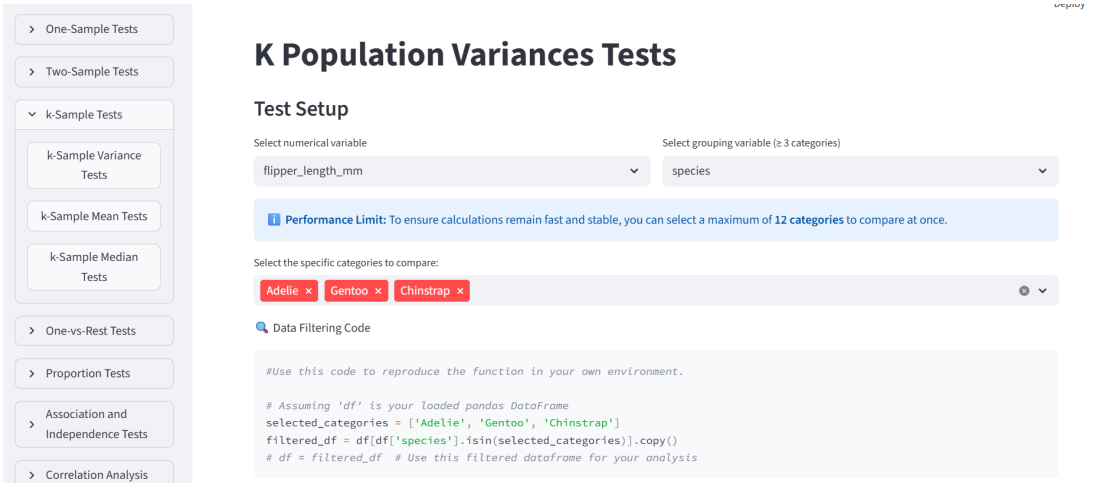
(**a**) *Configuration parameters for the categorical normality test.*



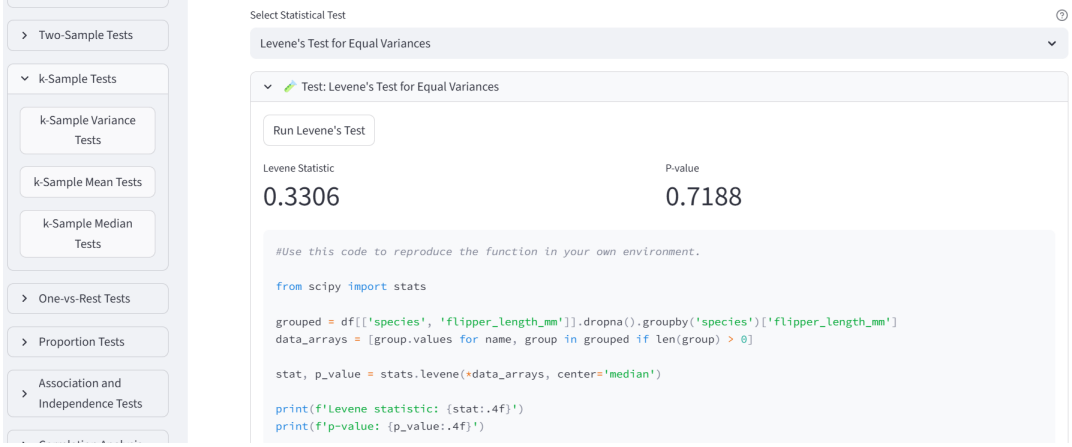
(**b**) *Normality test outcomes across the different species groups.*

Figure 4.17: *Normality analysis partitioned by categories.*

to reject the null hypothesis of equal variances (homoscedasticity). Consequently, the analysis can safely proceed to a test to determine whether the population means differ.



(**a**) *Configuration interface for Levene's test of equal variances.*



(**b**) *Levene's test output showing the resulting p-value for homogeneity of variances.*

Figure 4.18: *k-Samplev Variance Tests page.*

Located within the same `k-Sample Tests` section, the `k-Sample Mean Tests` page shares an identical configuration layout to the variance module regarding variable and category selection. However, it introduces a checkbox to toggle whether equal variances are assumed, a parameter that directly regulates the availability of subsequent post-hoc procedures. Additionally, the interface incorporates a slider at the bottom of the settings panel to adjust the confidence level for the intervals, which is set to 0.95 by default (Figure 4.19). Below the configuration area, the page's first subsection displays the test outputs. In this scenario, the executed ANOVA test yields a p-value of zero (Figure 4.20), firmly rejecting the null hypothesis that the mean flipper lengths are identical across all penguin species.

In the second subsection, the software renders the post-hoc pairwise comparisons. Because the workflow operates under the equal variances assumption, the application utilizes Tukey's method instead of the Games-Howell alternative. The statistical outputs are organized into a data table showing the confidence intervals for the differences

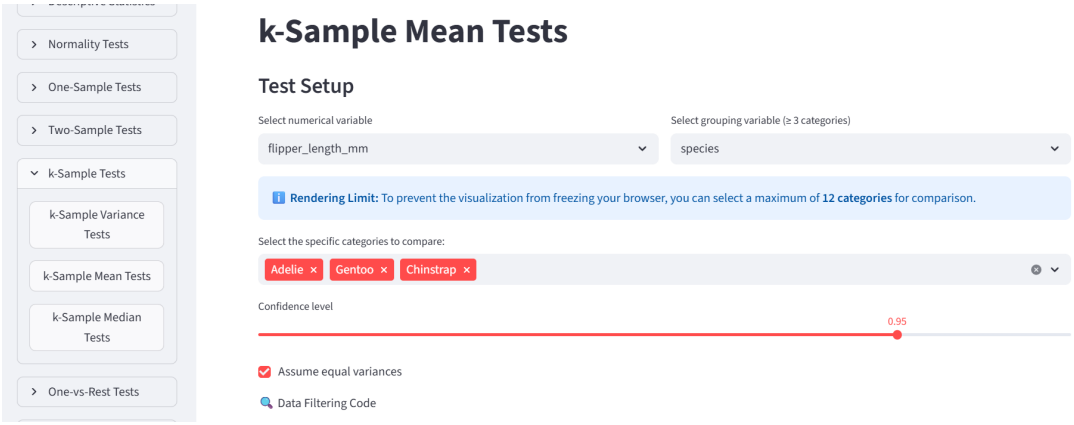


Figure 4.19: *ANOVA test configurations assuming equal variances for k populations.*

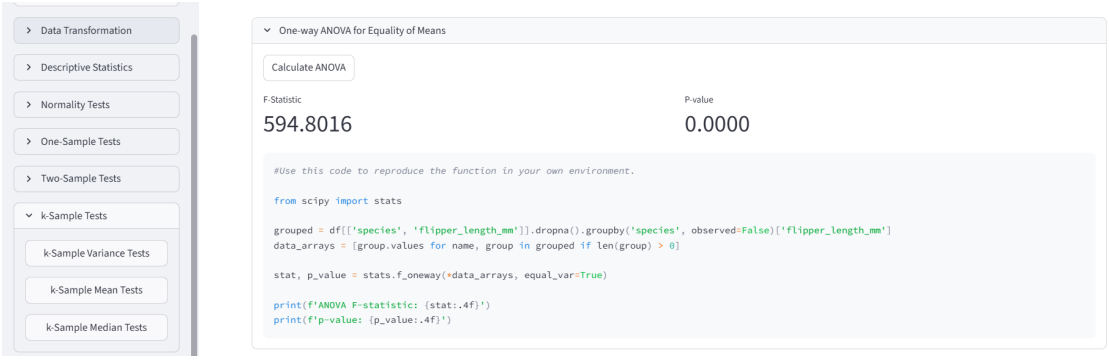


Figure 4.20: *ANOVA test results for evaluating differences in mean flipper lengths.*

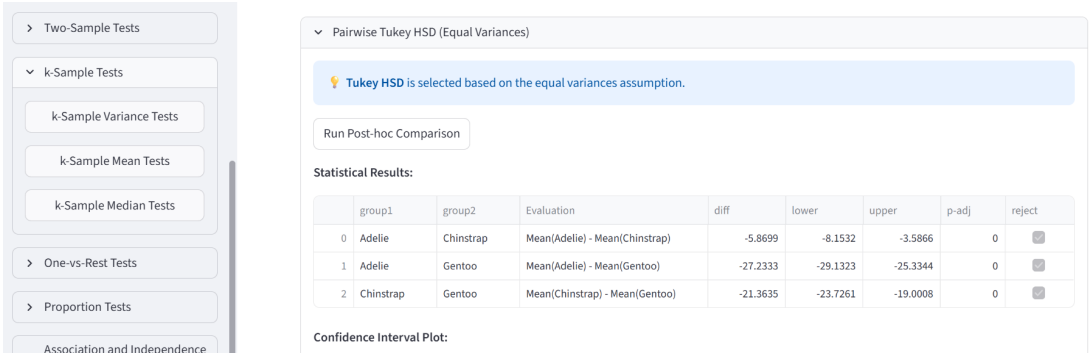
(Figure 4.21a). And to enhance data readability and simplify the visual analysis for the user, the interface generates an interval plot (Figure 4.21b). The graphical representation demonstrates that the intervals do not cross the zero threshold, visually reinforcing the rejection of the global null hypothesis.

To continue with this scenario, navigating to the `Correlation Analysis` section in the menu and selecting the `Correlation Analysis` page allows the evaluation of the correlation between (`flipper_length_mm`) and (`body_mass_g`). The configuration area features dropdown selectors for both numerical attributes, a method selector, and a slider to adjust the confidence level, which defaults to 0.95 (Figure 4.22). In this instance, the Spearman method is selected due to its mathematical robustness against non-normality.

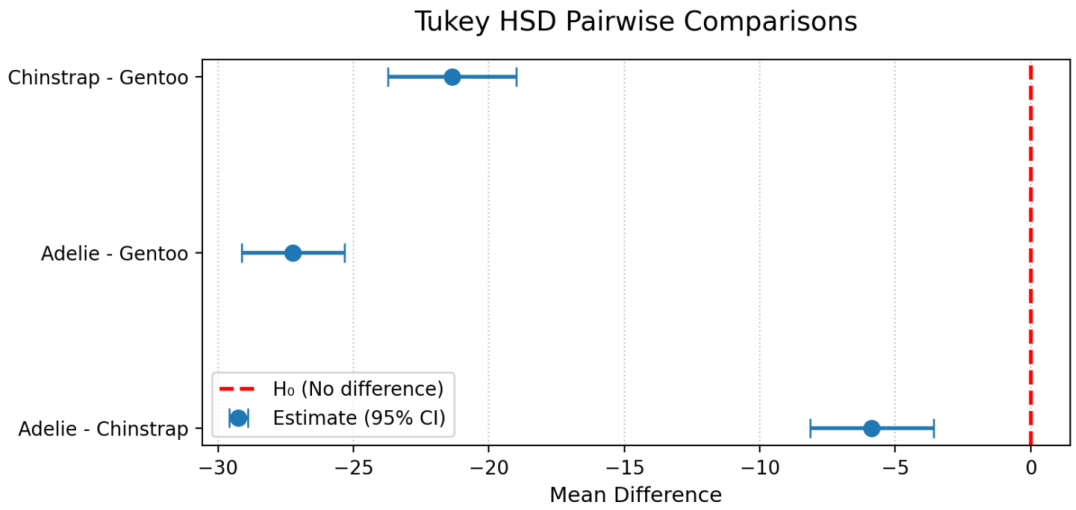
The first subsection displays the test metrics (Figure 4.23), yielding a correlation coefficient of 0.7746 and a p-value of 0.0000, confirming a statistically significant strong positive relationship. Additionally, the 95% Bootstrap confidence interval places the true correlation value between 0.7387 and 0.8005.

To complement the numerical summary, the second subsection handles the graphical deployment. Activating the regression line checkbox modifier (Figure 4.24a) dynamically updates the scatter plot, visualizing the strong linear relationship mapping flipper length to body mass (Figure 4.24b).

To conclude this exercise, the OvR module within the `One-vs-Rest` section of the



(**a**) *Tukey's simultaneous confidence intervals results table.*



(**b**) *Graphical representation of the confidence intervals for mean differences.*

Figure 4.21: *Post-hoc multiple pairwise comparisons using Tukey's method.*

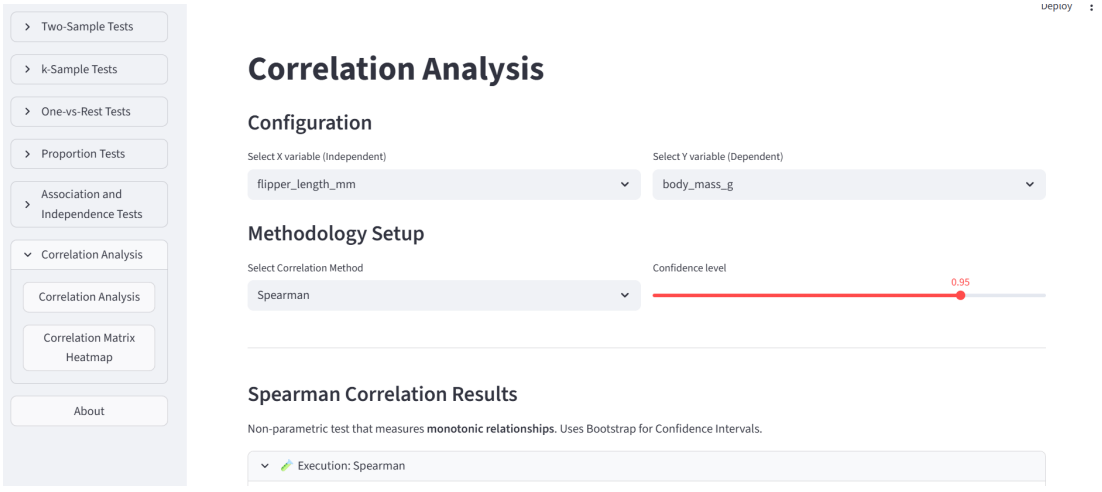


Figure 4.22: *Correlation test configurations.*

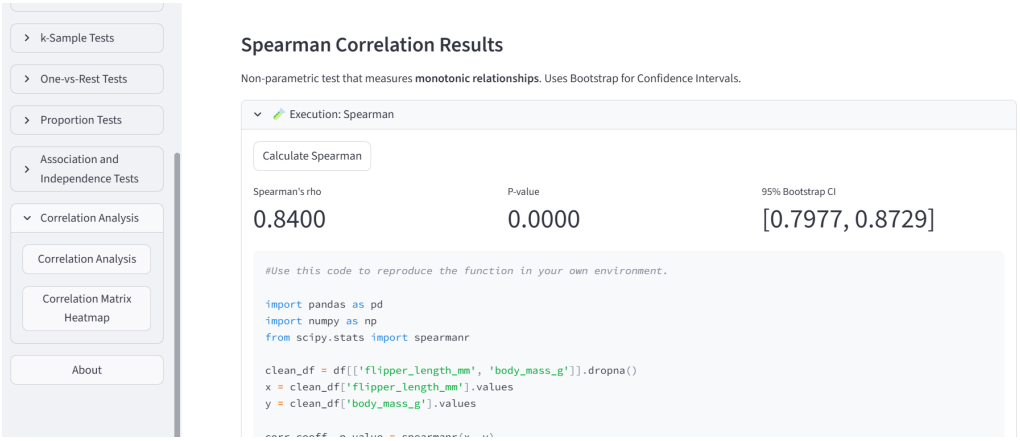
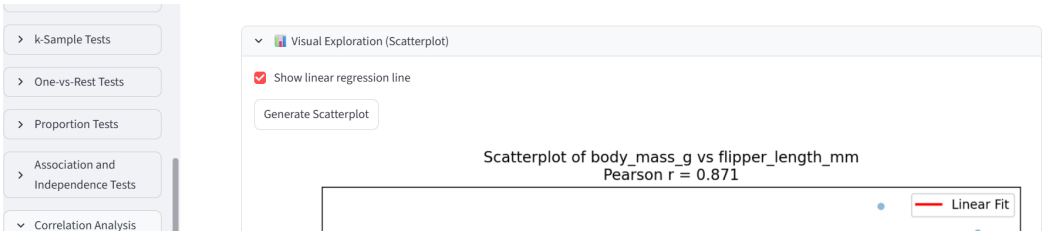
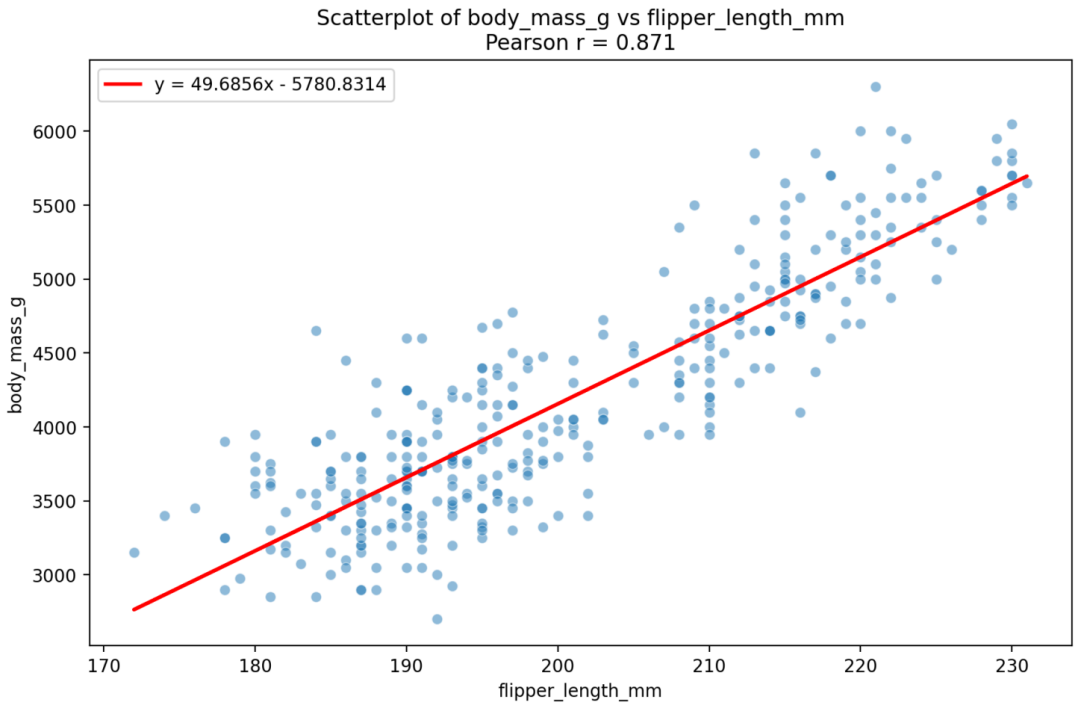


Figure 4.23: *Correlation estimation, confidence interval and test results.*



(**a**) *Enabling the regression line on the scatter plot.*



(**b**) *Scatter plot of the variables* `flipper_length_mm` *and* `body_mass_g`*, including the regression line.*

Figure 4.24: *Scatterplot of the selected variables.*

menu is utilized. The objective is to compare the body mass of penguins from Biscoe island against the other two.

First, the `One-vs-Rest Normality Tests` page is selected to evaluate the normality of both groups. The interface features four dropdown selectors to define the numerical variable, the categorical variable, the target category for comparison, and the hypothesis (which remains at the default `two-sided` setting). Lastly, a slider for the confidence interval is set to 95% (Figure 4.25).

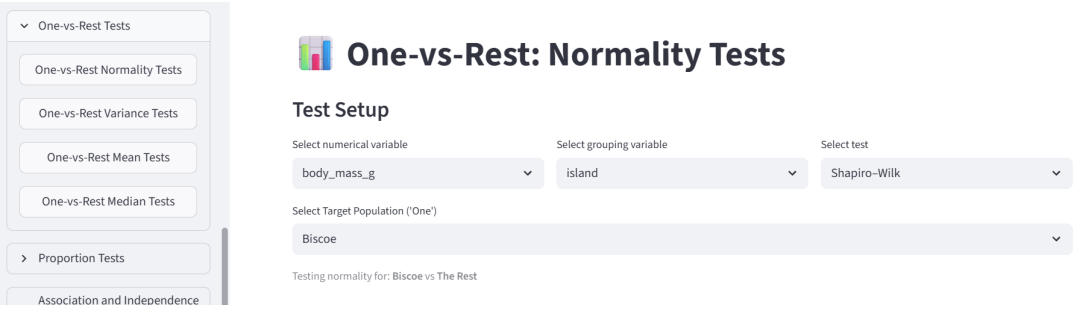


Figure 4.25: *Configuration for the OvR normality test.*

The page subsection provides an execution button for the test. After execution, a table displays the group under evaluation, its associated p-value, the sample size n within the group, and the test status to report errors if necessary (Figure 4.26).

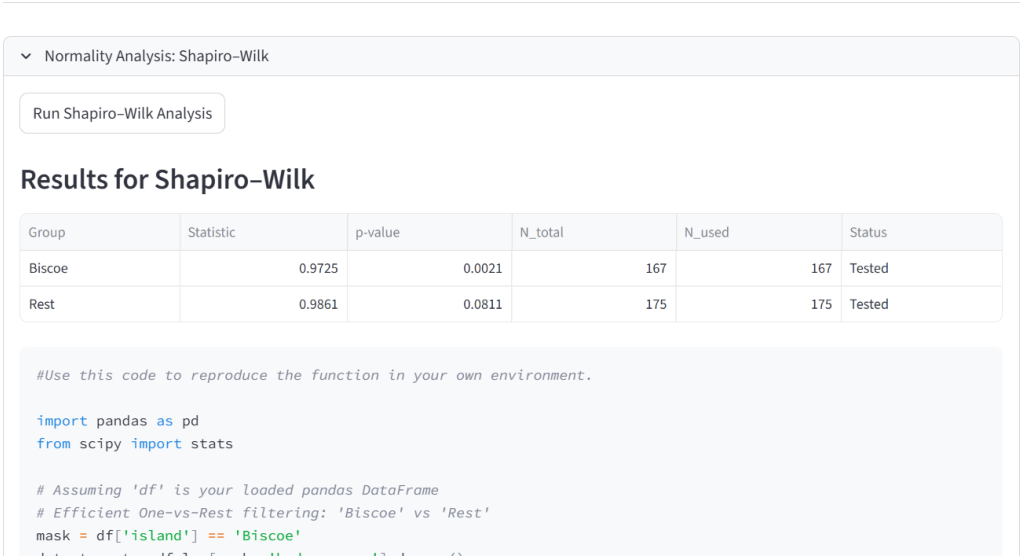


Figure 4.26: *Results of the normality test applied to the OvR approach.*

In this case, the p-value for Biscoe Island is lower than the standard significance level. As a result, the normality assumption is rejected, and the non-parametric path is selected for subsequent testing.

Next, the `One-vs-Rest Median Tests` page is used to compare the median body mass of Biscoe island against the others. Mirroring the normality interface, this page features four dropdown selectors for the numerical variable, categorical variable, selected category, and hypothesis, alongside a slider to adjust the confidence level (Fig-

ure 4.27).

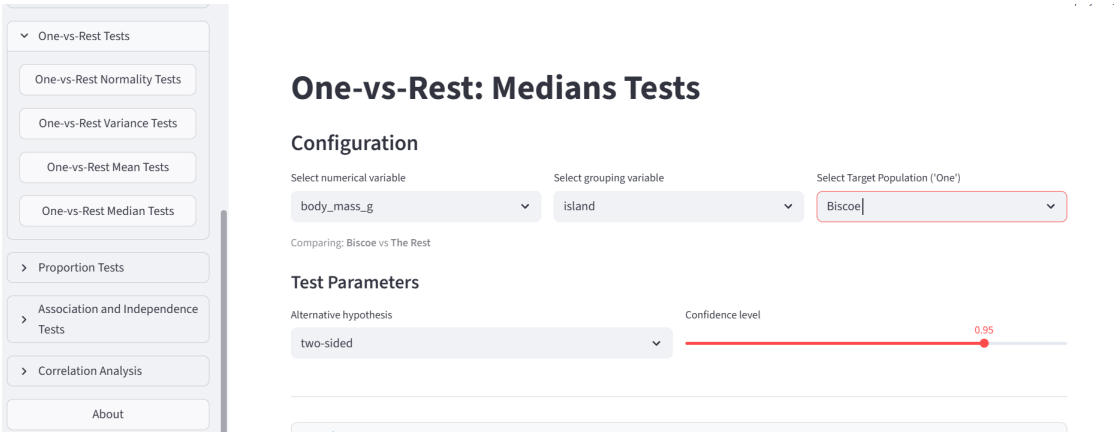


Figure 4.27: *Configuration for the OvR median test.*

Then, the first subsection displays the statistical results and the corresponding confidence intervals. In this analysis, the p-value is minimal, providing evidence to reject the null hypothesis that the median mass of Biscoe island equals the median of the remaining islands (Figure 4.28a). The second subsection generates a plot of the confidence interval. And this plot reveals that the confidence interval excludes 0, which represents the baseline of identical medians (Figure 4.28b).

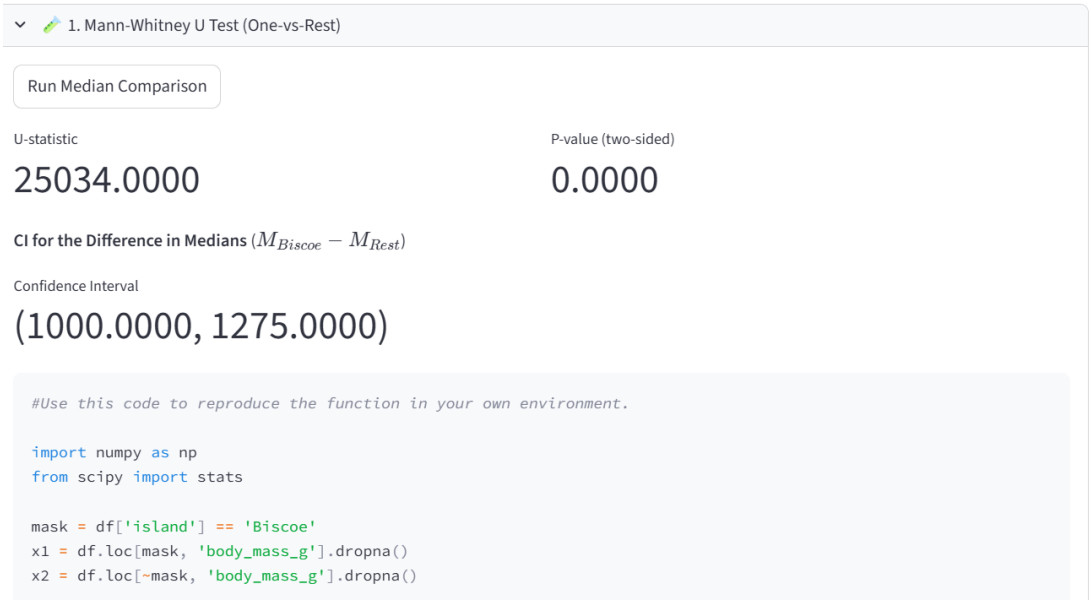
4.3 Scenario #2

Focusing on the `ToothGrowth` dataset, this analysis aims to determine whether tooth growth varies significantly across different Vitamin C doses. To achieve this, the evaluation is designed around comparing the medians of the treatment groups. Given that each treatment combination features a small sample size, the data serves as an ideal case study to justify the transition from parametric models toward non-parametric testing frameworks.

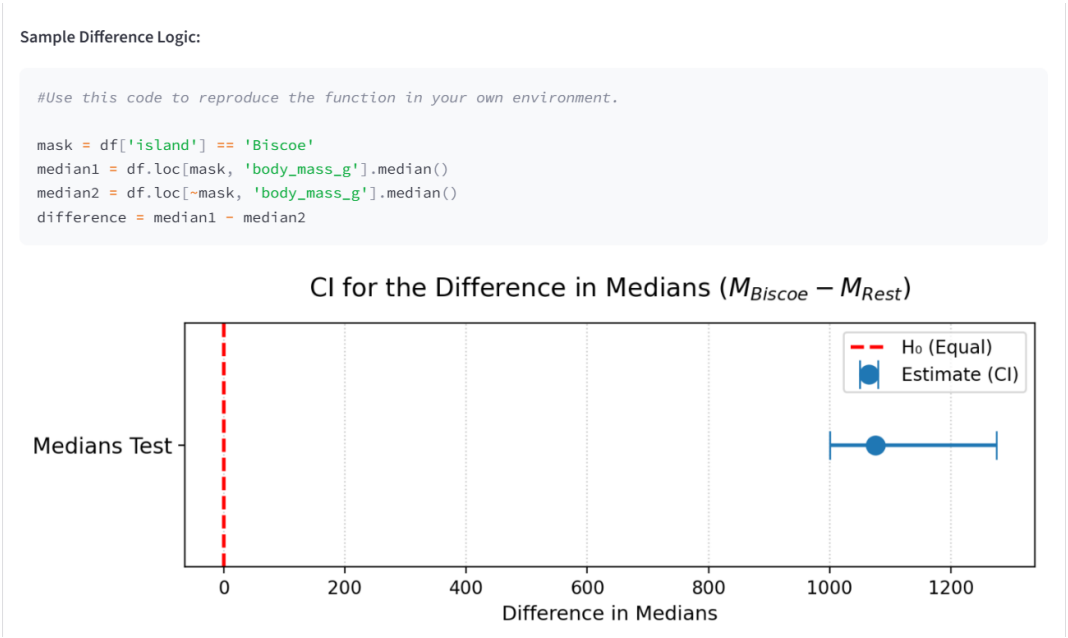
Following the initial data ingestion workflow, the dataset is successfully imported. After this, within the `Data Transformation` section, an additional pre-processing step is introduced. Navigating to the `Change Data Types` page provides an interface featuring a column selector, an indicator displaying the current data type of the chosen variable, and a dropdown menu to designate the target format.

For this analysis, the module is utilized to transform the `dose` variable from `float64` to an ordered `category`, as illustrated in Figure 4.29. When selecting the categorical data type, the interface displays a checkbox to specify whether the category is ordinal. Activating this option reveals a selector that allows the user to arrange the categorical levels in their proper hierarchy. This feature is employed to establish the sequential order as 0.5, 1.0, and 2.0.

The process begins within the `Descriptive Statistics` section by entering the `Categorical Variables` page. Choosing the `dose` variable from the application's selection menu outputs a summary table showing a uniform distribution of 20 samples



(**a**) *Test statistic, confidence interval for the difference, and p-value results.*



(**b**) *Plot of the confidence intervals for the difference of medians.*

Figure 4.28: *Overall results of the median comparison test.*

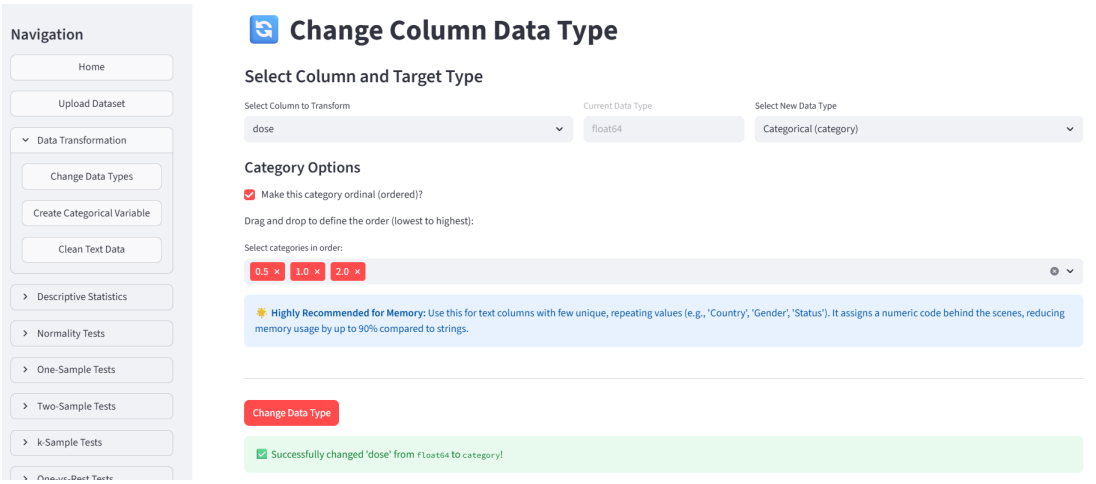


Figure 4.29: *Data type modification page*

per category (Figure 4.30). While a sample size of 20 does not entirely preclude a parametric approach if normality and homoscedasticity criteria are met, this scenario is designed strictly for instructional purposes. Therefore, the workflow intentionally bypasses parametric routing to demonstrate the execution of non-parametric alternatives directly.

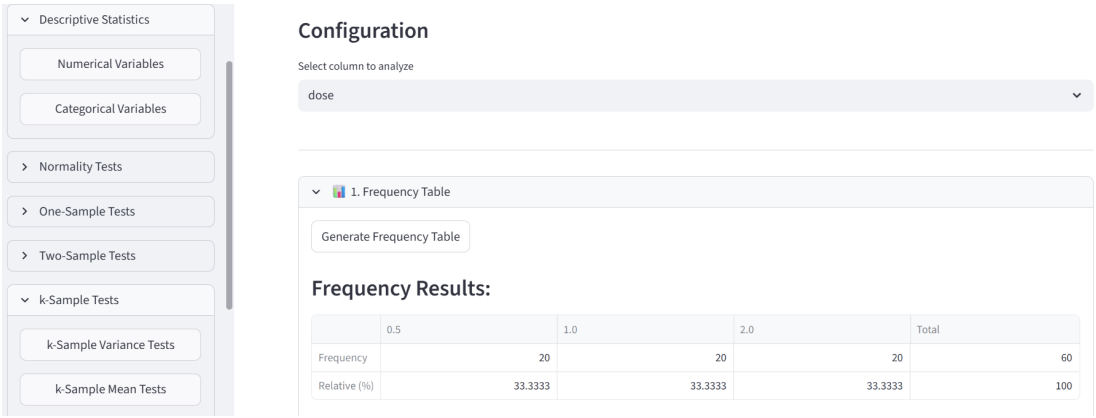


Figure 4.30: *Frequency table for categories.*

The next stage of the workflow is hosted on the `k-Sample Median Tests` page, located within the `k-Sample Tests` section of the left sidebar menu. The upper parameters panel features selectors for the numerical target, the categorical grouping attribute, and a multi-selection menu to filter the active classes (Figure 4.31).

The primary subsection displays the metrics for the Kruskal-Wallis H-test algorithm. In this case, the test delivers a p-value of zero (Figure 4.32), confirming a statistically significant difference in tooth length distributions across the dosage groups.

To supplement this global test, the second subsection handles the estimation of bootstrap confidence intervals for the pairwise differences between medians. This component incorporates an interactive slider allowing users to adjust the number of bootstrap resamples. To optimize data interpretation, the application renders these estimations through a summary table (Figure 4.33a) and a companion plot (Figure 4.33b).

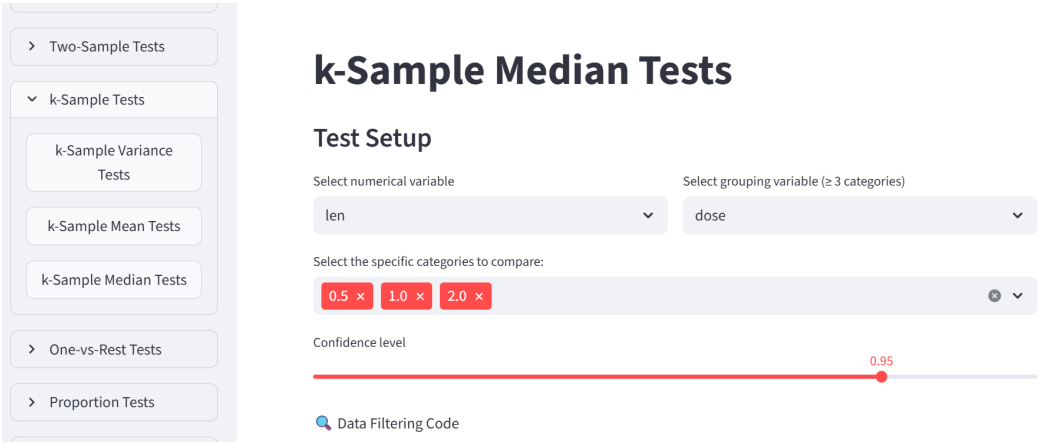


Figure 4.31: *Setup configurations for the non-parametric Kruskal-Wallis H-test.*

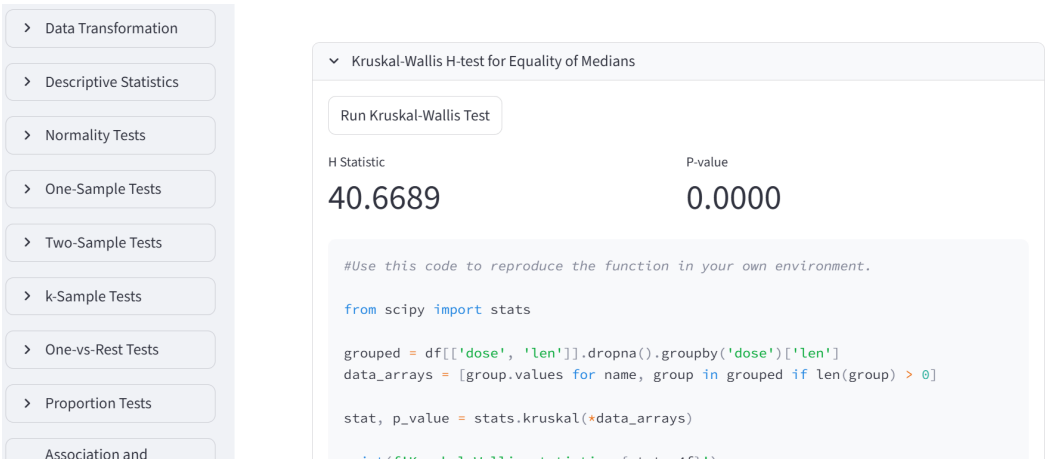
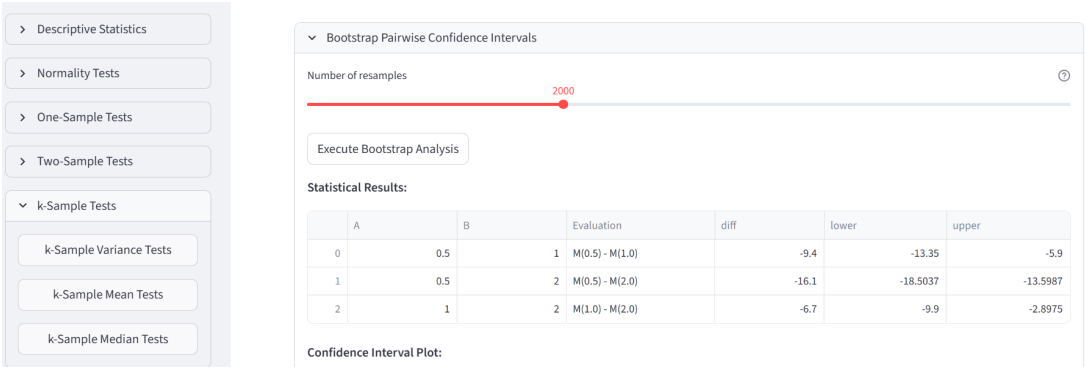
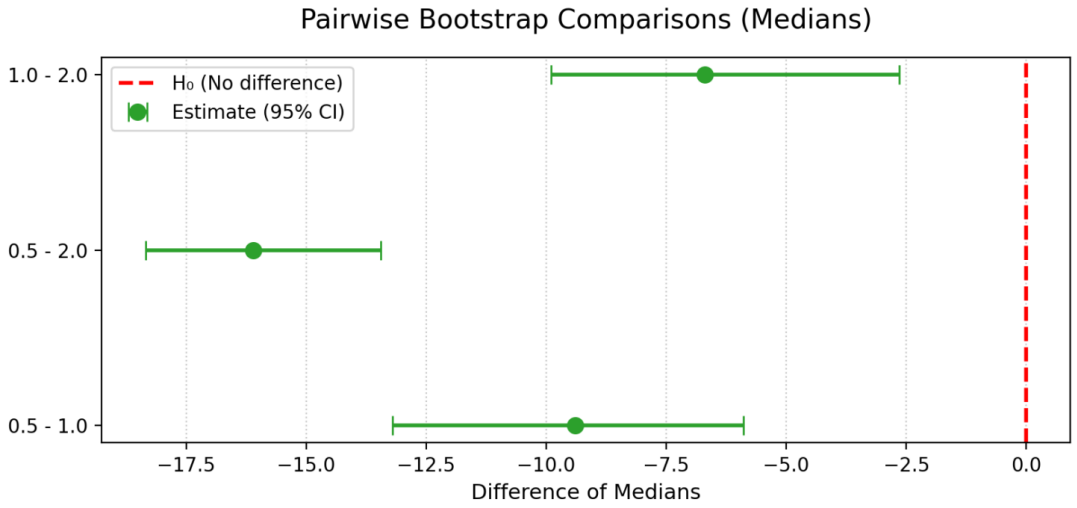


Figure 4.32: *Kruskal-Wallis test output.*



(**a**) *Table of bootstrapped confidence intervals for the medians.*



(**b**) *Plot of the distributions obtained via bootstrapping.*

Figure 4.33: *Bootstrapping estimation for confidence intervals of median differences.*

In summary, the evaluations conducted across these scenarios demonstrate the application's versatility in processing continuous numerical features through both parametric and non-parametric frameworks. While the initial workflow leveraged parametric modeling validated by the Central Limit Theorem and homoscedasticity verifications, the subsequent analysis successfully deployed non-parametric alternatives, specifically the Kruskal-Wallis H-test and bootstrap resampling, to robustly handle restricted sample sizes and violations of normality by evaluating group medians.

Having established the software's performance under varying numerical conditions, the workflow transitions from continuous metrics to discrete structures. The next section details the platform's specialized modules for testing proportions, shifting the analytical focus entirely toward categorical interactions.

4.4 Scenario #3

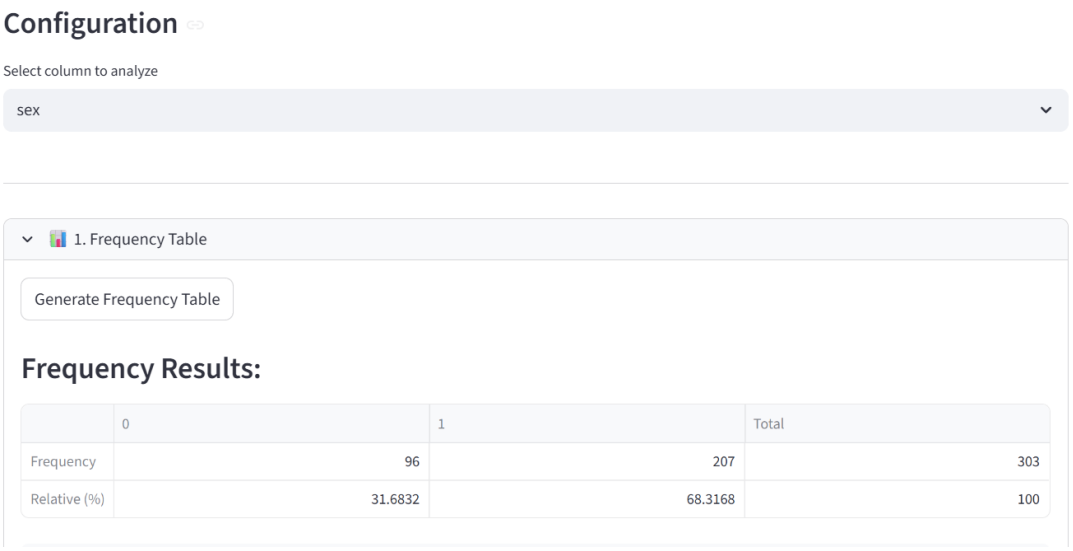
Focusing on the `heart` dataset, this analysis aims to investigate the distribution of cardiovascular disease across distinct demographic groups. To achieve this, the evaluation is designed around comparing the proportions of the clinical condition, represented by the `target` variable, between male and female subjects. As demonstrated in previous scenarios, since proper variable classification is required, the `target` attribute is converted from integer to categorical data type using the `Change Data Type` page. Ultimately, the data serves as an ideal case study to discretize the continuous age variable into specific cohorts, allowing for a subsequent analysis of independence relative to the `target` outcome.

Within the `Descriptive Statistics` section in the left menu, the `Categorical Variables` page is used to display the frequency tables for both the `sex` and `target` variables (Figure 4.34). These outputs reveal a higher percentage of male subjects compared to female subjects within the sample (Figure 4.34a), whereas the proportions concerning the presence or absence of the cardiovascular condition remain balanced (Figure 4.34b).

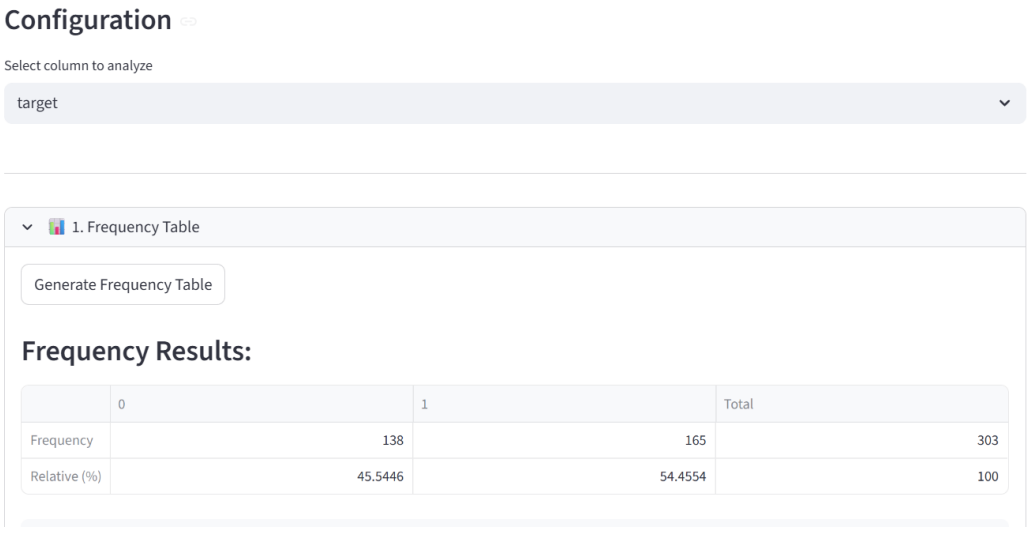
To determine whether the `target` variable represents 50% of the population, the `One-Proportion Test` page within the `Proportions Test` section of the left menu is utilized.

The configuration process requires selecting the categorical variable along with the specific category that defines a successful outcome, which is set to `1` for this analysis. An input box allows for the manual definition of the null hypothesis value between 0 and 1, established here at 0.5, while a selection menu defines the alternative hypothesis, which is configured as `two-sided` by default. Additionally, a slider controls the confidence level for the intervals, while two selection menus allow the user to choose the statistical test and interval estimation method from the available options. The `Binomial Test (Exact)` and the `Wilson Score` method were selected for this case (Figure 4.35).

Clicking the `Run Analysis` button within the execution subsection triggers the statistical evaluation. The test yielded a p-value greater than the standard significance



(**a**) *Frequency table of sex variable.*



(**b**) *Frequency distribution of target variable.*

Figure 4.34: *Frequency tables for gender demographics and target health status.*

level, indicating there was insufficient evidence to reject the null hypothesis that the true proportion was 50%, a conclusion further supported by the fact that the estimated confidence interval encompassed this hypothesized value (Figure 4.36).

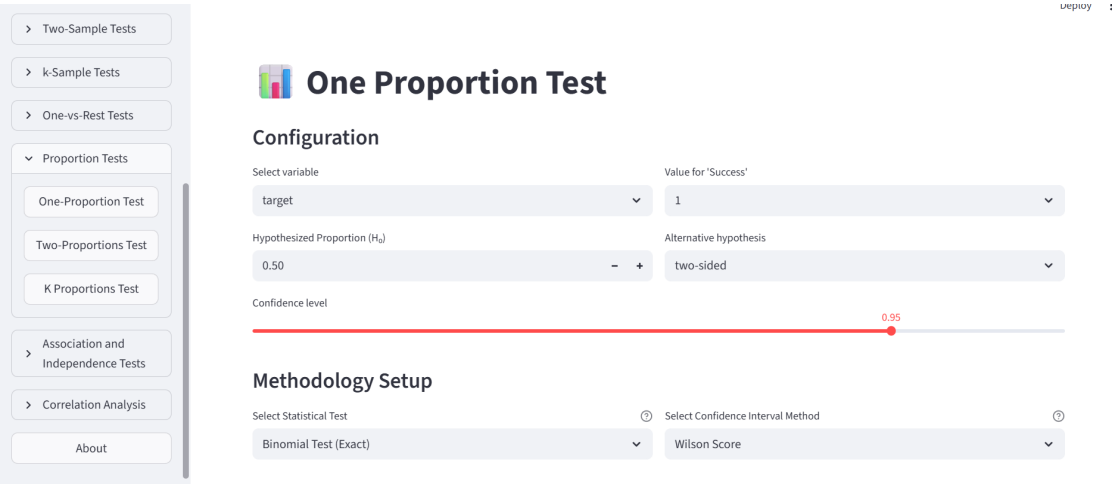


Figure 4.35: *Settings interface for the single-proportion binomial test via Wilson score method.*

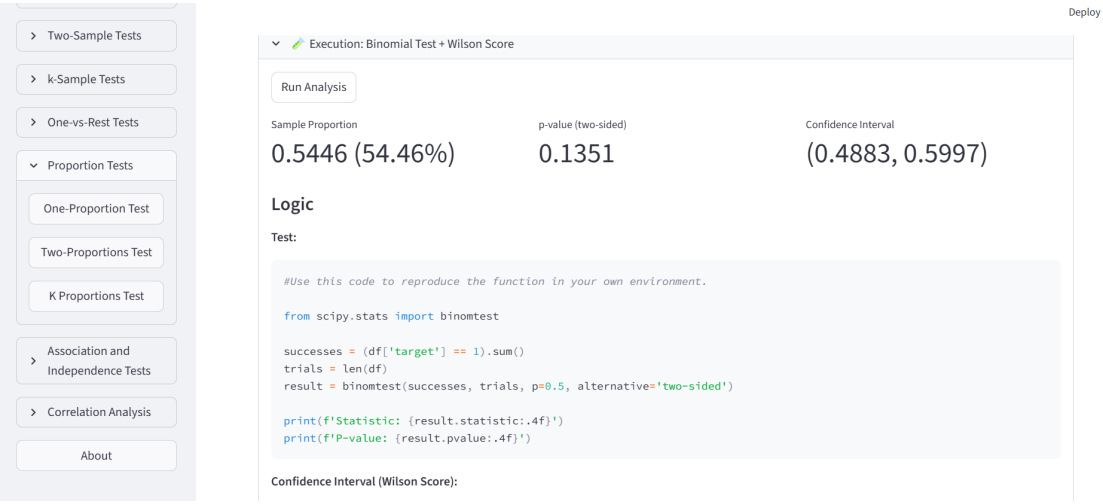


Figure 4.36: *One proportion test outcome and population proportion confidence intervals.*

To evaluate whether the proportions of individuals with the cardiovascular condition are equal across genders, the `Two-Proportions Test` page within the `Proportions Test` section of the left menu is utilized.

This page requires the selection of both a grouping variable and an outcome variable, which are mapped to the `sex` and `target` attributes, respectively. A selector defines the category representing a successful outcome, configured as `1` for this analysis, while the alternative hypothesis menu remains at its default `two-sided` option. Consistent with the previous page, a slider adjusts the confidence level, and two distinct dropdown menus allow for the selection of the statistical test and the confidence interval estimation method. For this case, `Fisher's Exact Test` was employed alongside the `Newcombe` method for confidence interval estimation (Figure 4.37).

Within the first subsection of the page, the option to generate the contingency table is provided, yielding results that suggested that the proportions of the disease are not equal between male and female subjects (Figure 4.38).

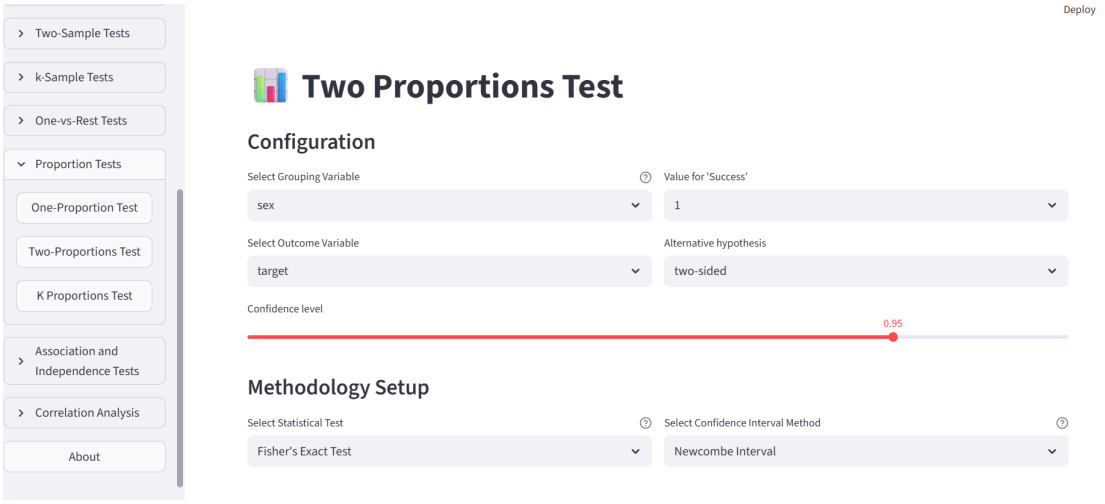


Figure 4.37: *Parameter configurations for Fisher's exact test on two proportions.*

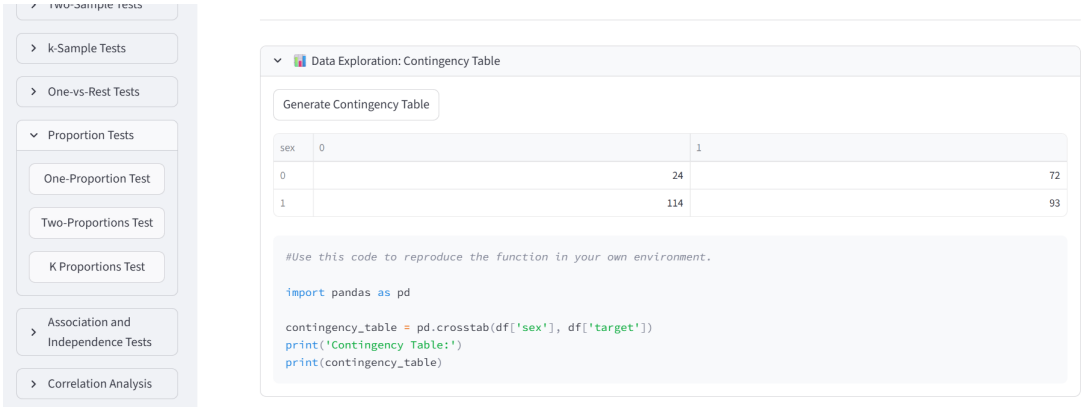


Figure 4.38: *Contingency table comparing cardiovascular disease prevalence across genders.*

In the second subsection, the final test results yielded a p-value of 0, providing sufficient evidence to reject the null hypothesis of equal proportions. Furthermore, the estimated confidence interval excluded 0, further substantiating the statistical divergence between the analyzed groups (Figure 4.39).

A key capability of the software is the feature allowing the discretization of continuous numerical variables into categorical intervals, which is managed via the `Create Categorical Variable` page within the `Data Transformation` section of the left menu (Figure 4.40).

The user interface provides a selection menu for the numerical variable along with an input box to specify the name of the new categorical column. Below these fields, the desired number of categories is defined through either direct numeric input or adjacent adjustment buttons that constrain the value between 2 and 10. Prior to using this feature, the user must determine the number of classes based on their own analytical

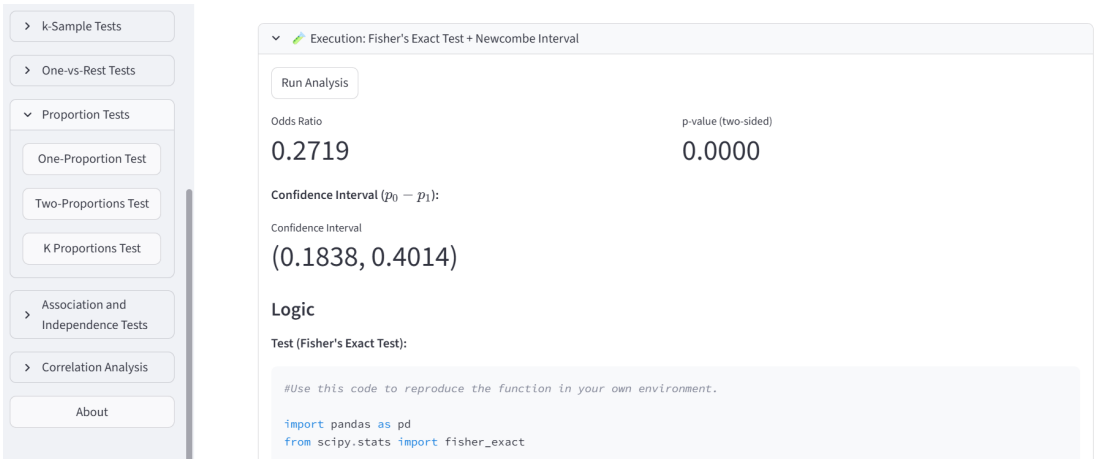


Figure 4.39: *Fisher's exact test results and Newcombe confidence interval for gender proportions.*

reasoning, ensuring it is appropriate for the specific objectives of their study.

Additionally, a checkbox dictates whether the right boundary is included within each interval, a setting that is enabled by default but deactivated for this specific configuration.

The interface then allows for the manual entry of the upper limits to structure the groups. In this instance, the continuous age variable is partitioned into three distinct cohorts to determine whether the presence of the cardiovascular condition is independent of age. This setup maps the data into "young adults" for individuals under 40, "middle-aged adults" for those under 60, and "older adults" for the oldest demographic group (Figure 4.41).

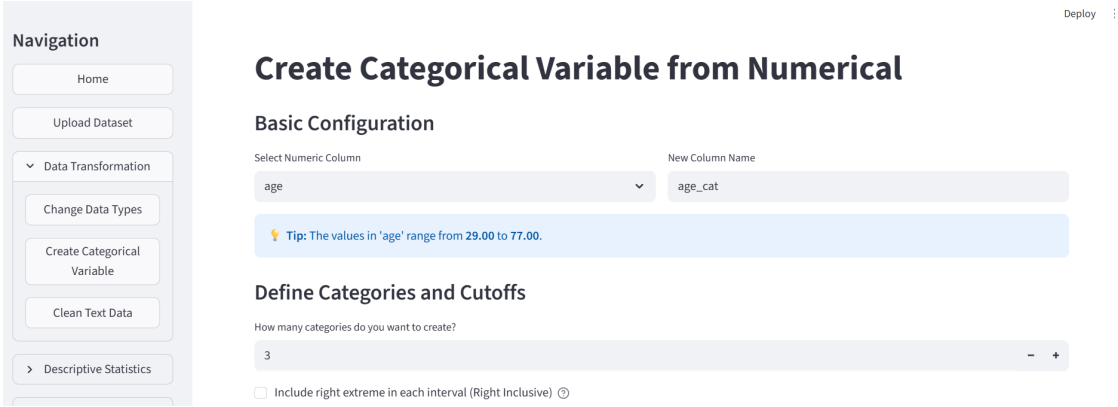


Figure 4.40: *Configuration page to transform numerical ranges into categorical data.*

Within the menu's `Association and Independence Tests` section, the `Tests of Independence` page is used to evaluate the relationship between categorical variables.

This interface provides two distinct selection dropdowns to define each of the categorical variables, alongside a dropdown menu to choose the desired statistical test (Figure 4.42). When the chi-square test is selected, the interface displays a checkbox to apply the Yates' continuity correction, a feature recommended primarily for 2×2 tables. As can be seen in Figure 4.43, the current analysis does not constitute a 2x2 scenario,

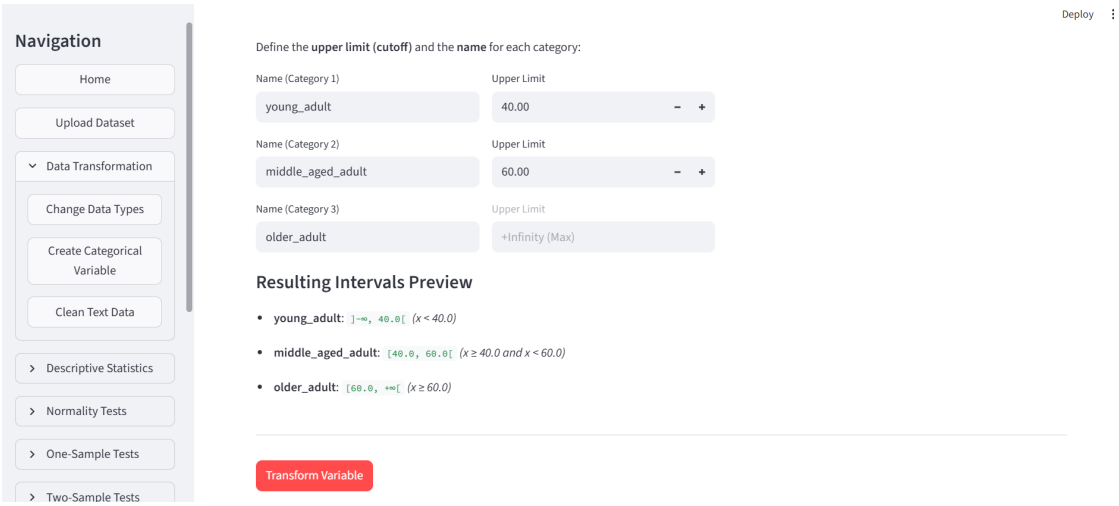


Figure 4.41: *Defined ranges mapping patient age into distinct adult life stages.*

therefore, it is not necessary to apply Yates' continuity correction. The table of expected values is shown below the contingency table (Figure 4.44), revealing that no expected frequency falls below 5 and thus confirming that it is perfectly appropriate to proceed with the standard chi-square test.

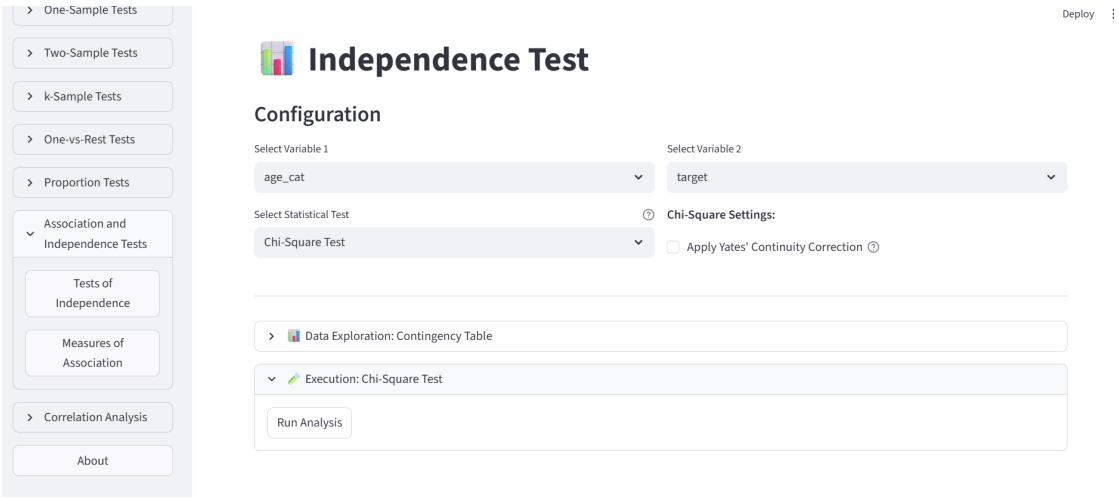


Figure 4.42: *Parameters for the chi-square independence test.*

Within the second subsection of the page, the test results yielded a p-value below the standard significance level, indicating that statistical independence between the analyzed variables is rejected (Figure 4.45).

For demonstrative purposes, `Fisher's Exact Test` is also evaluated. Due to the computationally intensive nature of exact combinatorial calculations for contingency tables larger than 2×2, the interface dynamically adapts, replacing the `Apply Yates` checkbox with a slider dedicated to configuring the number of resamples for a Monte Carlo simulation (Figure 4.46)

When executed on this larger table structure, the test do not return a standard test statistic but provides a simulated p-value (Figure 4.47). This value fell below the stan-

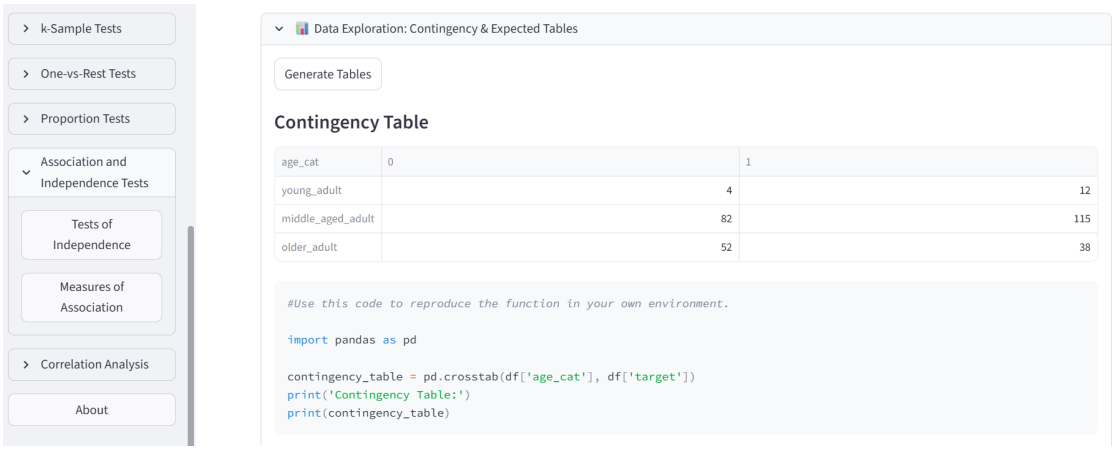


Figure 4.43: *Contingency table for age and target variables.*

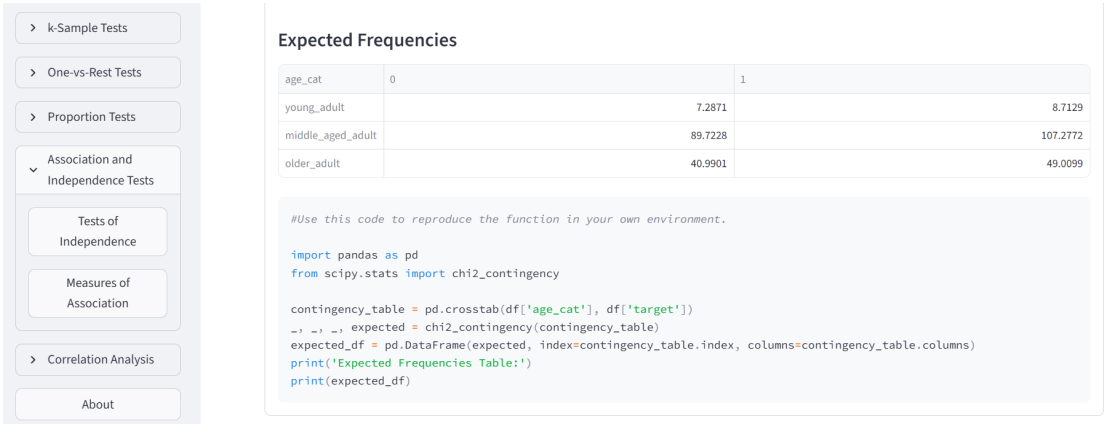


Figure 4.44: *Table of expected values for age and target variables assuming independence.*

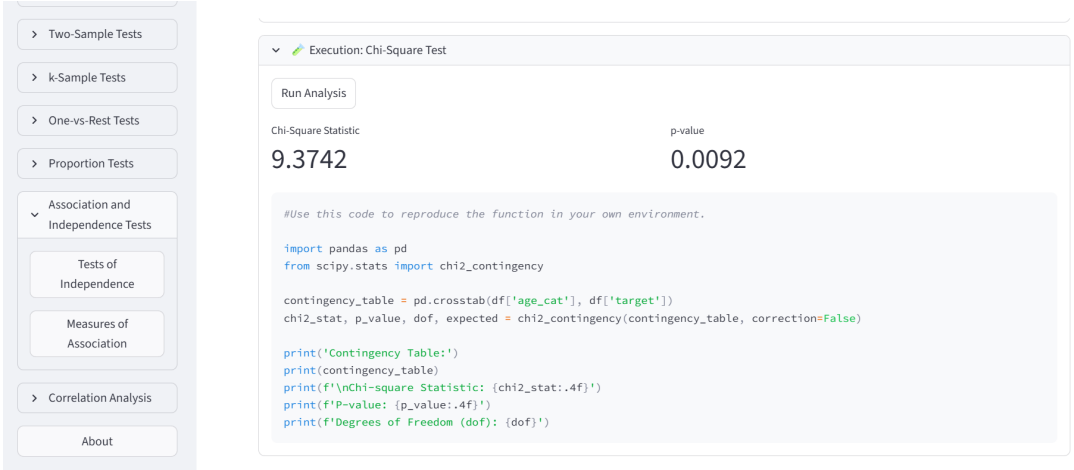


Figure 4.45: *Chi-square test results evaluating age and disease independence.*

dard significance level, mirroring the chi-square result and supporting the rejection of the null hypothesis of independence.

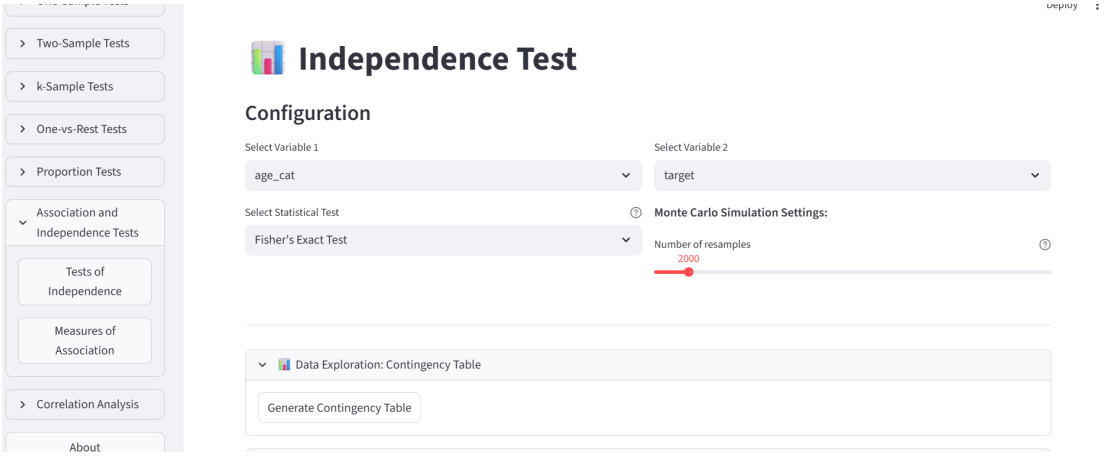


Figure 4.46: *Setup for Fisher's exact test with Monte Carlo simulation for larger tables.*

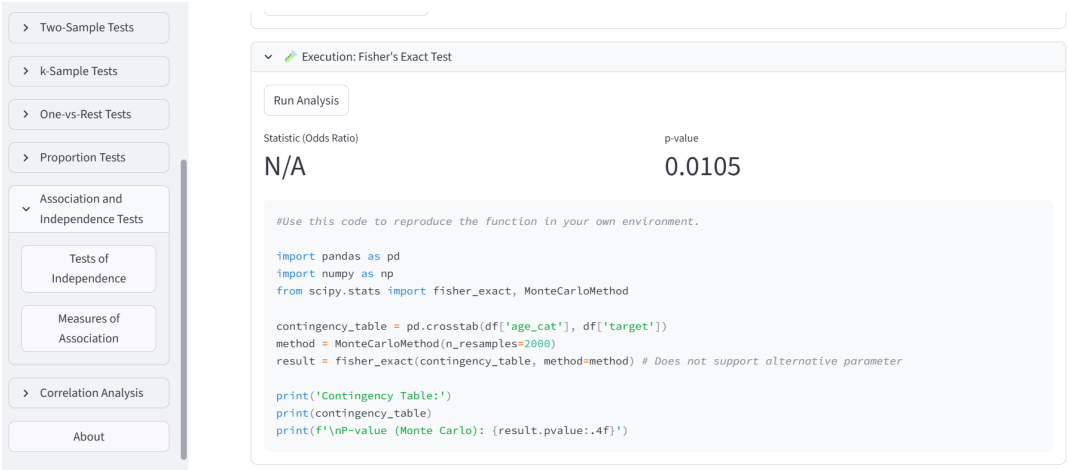


Figure 4.47: *Simulated p-value output from Fisher's exact test on the age-disease matrix.*

Since the prior test of independence successfully rejected the null hypothesis, the analysis shifts toward measuring the intensity of the relationship via the `Measures of Association` page, located within the `Association and Independence Tests` section of the left menu.

The configuration interface requires defining both categorical variables through dropdown menus and selecting the desired metric from the available options (Figure 4.48). While the first subsection reproduces the previously discussed contingency table, the second subsection displays the statistical outputs. Given that `age_cat` is an ordinal variable and `target` is a binary attribute (coded as 0 for healthy and 1 for sick), Kendall's τ was selected as the methodologically appropriate coefficient to evaluate a potential directional trend across the age tiers.

For this configuration, the system computed a Kendall's τ coefficient of -0.082857 with a corresponding p-value of 0.1413. This statistical output indicates an extremely weak negative relationship that lacks statistical significance at the standard significance

level (Figure 4.49). Consequently, no reliable monotonic or directional trend can be confirmed between the progression of age categories and the condition's presence.

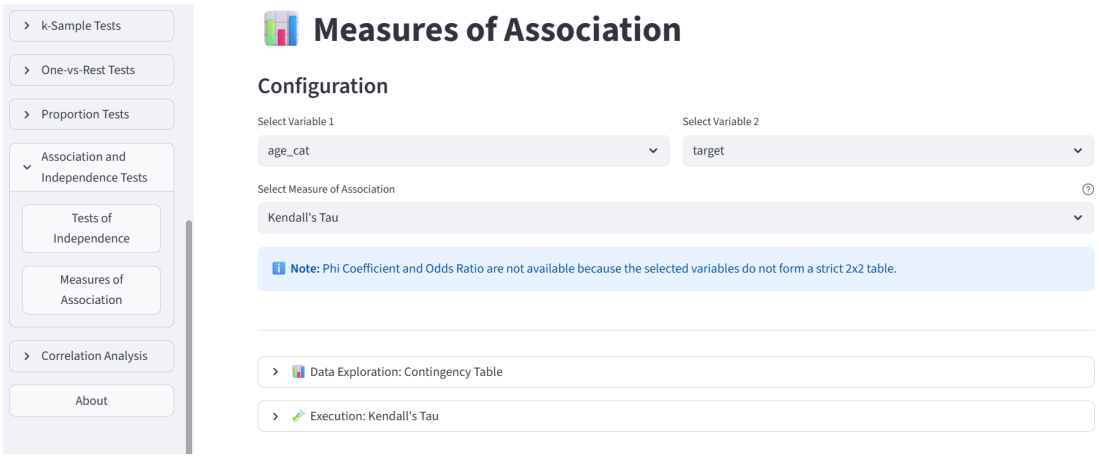


Figure 4.48: *Association page.*

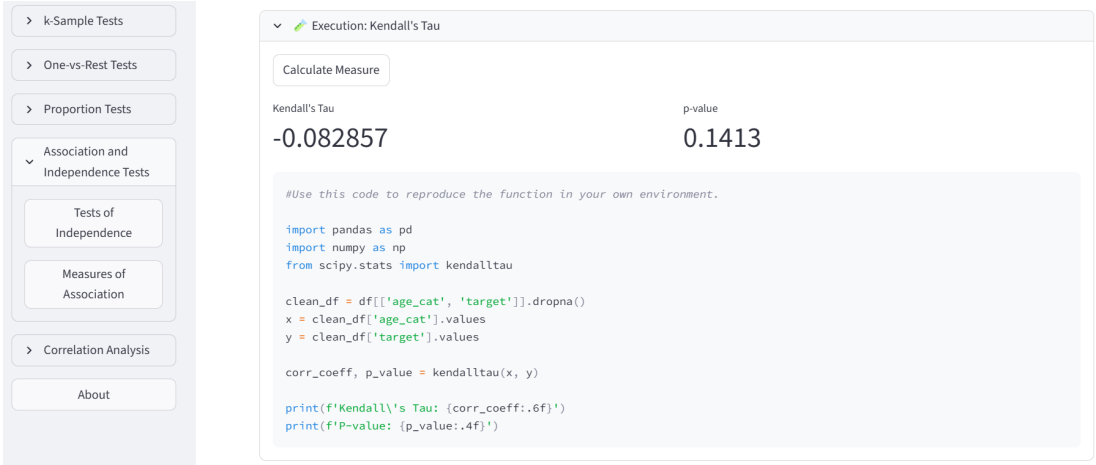


Figure 4.49: *Cramer's V output displaying the calculated effect size coefficient.*

These three practical scenarios presented in this chapter successfully validate the execution capabilities of the application by operationalizing the comprehensive statistical methodologies defined in the previous chapter. By systematically transitioning from parametric and non-parametric frameworks for continuous variables to proportion, independence, and association analyses for categorical attributes, these case studies confirm the practical utility of the software.

5
Conclusions and Future Work

The successful execution of the objectives outlined in the introduction provides a final system that bridges the gap between statistical theory and hands-on application.

By delivering an intuitive visual interface, the software grants access to the core inferential statistics tools required in a standard university curriculum. Beyond simple execution, the application exposes the underlying `Python` code for each operation, a feature that ensures a reproducible and independent workflow.

To maximize academic impact, the repository remains public as an open-source tool on `GitHub`. This open-source approach addresses a persistent challenge in education, where standard software options demand expensive licenses and proprietary languages.

The primary advantage of this project over existing software lies in its deliberate focus on inferential statistics. The literature review in Chapter 2 demonstrates that contemporary educational `Python` platforms often build overambitious suites with an overwhelming number of features that cause student confusion. Furthermore, those applications require a `Jupyter` notebook environment to function at all, whereas this software operates as an independent module. Students can execute statistical tests and extract clean `Python` code for deployment in any preferred environment, including free online platforms such as `Google Colab`.

These design choices ensure immediate applicability in classroom settings. The architecture permits rapid interaction with datasets to implement statistical concepts under faculty guidance, as students select the appropriate options within the interface without a requirement to master complex programming syntax. Because the software removes the coding barrier, professors can incorporate the tool into statistics courses with minimal friction.

Although the outcomes of this project establish a robust foundational framework for inferential statistics, multiple analytical domains remain unexplored that could benefit from a similar educational approach. A crucial future expansion involves the inclusion of modules dedicated to calculating and visualizing probabilities across key theoretical frameworks, specifically focusing on the normal, t, binomial, and Poisson

distributions.

This enhancement would allow users to compute probability density functions for exact values alongside cumulative distribution functions to determine threshold areas. Furthermore, implementing inverse probability calculations would enable students to derive critical values directly, thereby bridging the conceptual gap between probability theory and the determination of rejection regions in hypothesis testing.

Expanding the statistical scope into predictive analytics, a complementary project can be developed to focus exclusively on regression modeling. Regression analysis represents a vast statistical landscape that encompasses simple linear models, multiple linear models, and logistic frameworks for categorical outcomes. This endeavor can incorporate parameter estimation alongside the generation of confidence intervals and fundamental hypothesis tests for model coefficients, such as t-tests and F-tests.

The development of this regression suite can utilize the existing codebase as a structural blueprint, a strategy that capitalizes on an intuitive GUI and real-time `Python` code visibility to increase the overall value for statistical education.

Bibliography

[1] D. Lazer, R. Kennedy, G. King, and A. Vespignani, "The parable of google flu: Traps in big data analysis," *Science*, vol. 343, no. 6176, pp. 1203–1205, 2014. DOI: 10.1126/science.1248506.

[2] D. Butler, "When google got flu wrong," *Nature*, vol. 494, pp. 155–156, 2013. DOI: 10.1038/494155a.

[3] J. Dastin, *Amazon scraps secret ai recruiting tool that showed bias against women*, 2018. [Online]. Available: https://www.reuters.com/article/world/insight-amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK0AG/.

[4] M. Raghavan, S. Barocas, J. Kleinberg, and K. Levy, "Mitigating bias in algorithmic hiring: Evaluating claims and practices," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, ser. FAT* '20, Barcelona, Spain: Association for Computing Machinery, 2020, pp. 469–481. DOI: 10.1145/3351095.3372828.

[5] Stanford Graduate School of Business, *The flip-flop: Why Zillow's algorithmic home-buying venture imploded*, Nov. 2021. [Online]. Available: https://www.gsb.stanford.edu/insights/flip-flop-why-zillows-algorithmic-home-buying-venture-imploded.

[6] J. Cook, *When iBuying algorithms failed Zillow: What it says about the business world's love affair with AI*, 2021. [Online]. Available: https://www.geekwire.com/2021/ibuying-algorithms-failed-zillow-says-business-worlds-love-affair-ai/.

[7] C. Franklin, A. E. Bargagliotti, G. A. Kader, R. L. Schaeffer, and D. A. Spangler, "Guidelines for assessment and instruction in statistics education (GAISE II) report: A Pre-K–12 step in sharing the future of data," American Statistical Association, Tech. Rep., 2020. [Online]. Available: https://www.amstat.org/asa/files/pdfs/GAISE/GAISEIIPreK-12_Full.pdf.

[8] S. Raschka, J. Patterson, and C. Nolet, "Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence," *Information*, vol. 11, no. 4, 2020. DOI: 10.3390/info11040193.

[9] Data Active, *Global BI implementation benchmark study 2025*, 2025. [Online]. Available: https://dataactive.co.za/global-bi-implementation-benchmark-study-2025/.

[10] Microsoft, *Microsoft named a leader in the 2024 gartner magic quadrant for analytics and bi platforms*, 2024. [Online]. Available: https://powerbi.microsoft.com/blog/microsoft-named-a-leader-in-the-2024-gartner-magic-quadrant-for-analytics-and-bi-platforms/.

[11] Microsoft, *Feature availability for free users in power bi service*, 2025. [Online]. Available: https://learn.microsoft.com/power-bi/consumer/end-user-features.

[12] Microsoft, *Power bi data connection documentation*, 2025. [Online]. Available: https://learn.microsoft.com/power-bi/connect-data/.

[13] Microsoft, *Dax overview*, 2023. [Online]. Available: https://learn.microsoft.com/dax/dax-overview.

[14] Microsoft, *Run python scripts in power bi desktop*, 2025. [Online]. Available: https://learn.microsoft.com/power-bi/connect-data/desktop-python-scripts.

[15] Future Market Insights, *Self-service bi market - global market analysis report 2025-2035*, 2025. [Online]. Available: https://www.futuremarketinsights.com/reports/self-service-bi-market.

[16] Tableau, *Tableau desktop and tableau desktop public edition feature comparison*, 2025. [Online]. Available: https://help.tableau.com/current/pro/desktop/en-us/desktop_comparison.htm.

[17] Falvey Library, Villanova University, *Tableau: Data visualization software guide*, 2025. [Online]. Available: https://library.villanova.edu/research/data-services/visualization-best-practices/tableau.

[18] N. Shrestha, *Linear trend line in tableau: Evaluating model fit with r-squared and p-value*, 2024. [Online]. Available: https://www.thedataschool.com/nitesh-shrestha/client-project-expectation-vs-reality/.

[19] J. Hullman and A. Gelman, "Interactive analysis needs theories of inference," *Statistical Modeling, Causal Inference, and Social Science* (*Columbia University*), 2020. [Online]. Available: https://statmodeling.stat.columbia.edu/2020/10/23/interactive-analysis-needs-theories-of-inference/.

[20] Mistral Business Solutions, *Análisis del cuadrante mágico de gartner para analytics 2025*, Jul. 2025. [Online]. Available: https://www.mistralbs.com/blog/analisis-del-cuadrante-magico-de-gartner-para-analytics-2025.

[21] Google Cloud, *Looker studio is data studio | google cloud*. [Online]. Available: https://cloud.google.com/blog/products/data-analytics/looker-studio-is-data-studio.

[22] Valiotti Data, *What is looker studio? 2026 review (free google bi tool, pricing, limits)*, Jan. 2024. [Online]. Available: https://valiotti.com/technologies/looker-studio/.

[23] Improvado, *Looker vs. looker studio: The definitive 2026 comparison*, Oct. 2026. [Online]. Available: https://improvado.io/blog/looker-vs-looker-studio-comparison.

[24] S. Buzrul, "On the pros and cons of using excel for regression analysis," *Turkish Journal of Agriculture - Food Science and Technology*, vol. 12, no. s2, pp. 2234–2241, 2024. DOI: 10.24925/turjaf.v12is2.2234-2241.6931.

[25] M. I. Hertz and A. S. McNeill, "Eleven quick tips for properly handling tabular data," *PLOS Computational Biology*, vol. 20, no. 11, e1012604, Nov. 2024. DOI: 10.1371/journal.pcbi.1012604.

[26] Y.-S. Su, "It's easy to produce chartjunk using Microsoft® Excel 2007 but hard to make good graphs," *Computational Statistics Data Analysis*, vol. 52, no. 10, pp. 4594–4601, 2008. DOI: 10.1016/j.csda.2008.03.007.

[27] Diligent Corporation, *Why spreadsheets create risk for organizational compliance*, Diligent Insights, Mar. 2020. [Online]. Available: https://nl.diligent.com/resources/blog/why-spreadsheets-create-risk-for-organizational-compliance.

[28] Analyse-it, *Standard edition statistical analysis software for excel*, Analyse-it Product Documentation. [Online]. Available: https://analyse-it.com/products/standard.

[29] RegressIt, *What's wrong with excel's own data analysis add-in (analysis toolpak) for regression*. [Online]. Available: https://regressit.com/analysis-toolpak-problems.html.

[30] A. Rahman and M. G. Muktadir, "SPSS: An imperative quantitative data analysis tool for social science research," *International Journal of Research and Innovation in Social Science (IJRISS)*, vol. 5, no. 10, pp. 696–702, Oct. 2021, ISSN: 2454-6186. DOI: 10.47772/IJRISS.2021.51012.

[31] IBM, *NPAR TESTS command - overview*, 2026. [Online]. Available: https://www.ibm.com/docs/bg/spss-statistics/cd?topic=tests-overview-npar-command.

[32] Research & Report, *Why spss falls short in complex analysis*, 2025. [Online]. Available: https://researchandreport.org/why-spss-falls-short-in-complex-analysis/.

[33] J. Collier, "Introduction," in *Using SPSS Syntax: A Beginner's Guide*. London: SAGE Publications, 2010, pp. 1–12. DOI: 10.4135/9781446249369.n1.

[34] E. Upton, X. Cai, P. Jakiela, O. Ozier, and S. Raman, "The software behind the Stats: A student exploration of software trends across disciplines," 2025. DOI: 10.48550/arXiv.2504.06507.

[35] StataCorp, *Biostatistics: Data analysis and statistical software for biostatistics*, 2026. [Online]. Available: https://www.stata.com/disciplines/biostatistics/.

[36] StataCorp, *Stata License Options*, 2026. [Online]. Available: https://www.stata.com/order/license-options/.

[37] S. Correia and M. P. Seay, "Require: Package dependencies for reproducible research," 2024. [Online]. Available: 10.48550/arXiv.2309.11058.

[38] Pharmaceutical Technology, *Exploring the skills gap in statistical programming organizations*, 2023. [Online]. Available: https://www.pharmtech.com/view/exploring-the-skills-gap-in-statistical-programming-organizations.

[39] C. Matthews and B. Shilling, *Validating Clinical Trial Data Reporting with SAS*. Cary, NC: SAS Institute, 2005, ISBN: 9781590475751.

[40] SAS Institute Inc., *Sas clinical standards toolkit: Validation framework overview*, 2016. [Online]. Available: https://support.sas.com/documentation/cdl/en/clinstdtktug/69404/HTML/default/p1tpgi2mr0qsnrn1cqdw4pkd1tkd.htm.

[41] R Consortium, *Johnson & johnson's success story: A hybrid R/SAS strategy for FDA submission*, 2025. [Online]. Available: https://r-consortium.org/posts/johnson-and-johnsons-success-story-a-hybrid-r-sas-strategy-for-fda-submission.

[42] Appsilon Team, *Pharma is not leaving SAS because it stopped working. it's leaving because open source moves faster.* 2024. [Online]. Available: https://www.appsilon.com/post/pharma-is-not-leaving-sas-because-it-stopped-working-its-leaving-because-open-source-moves-faster.

[43] IHS EViews, *EViews Statistical Software*, 2026. [Online]. Available: https://www.eviews.com/.

[44] IHS EViews, *Learning EViews: Basics*, 2026. [Online]. Available: https://www.eviews.com/Learning/basics.html.

[45] A. T. Yalta, A. Cottrell, and P. C. Rodrigues, "Advanced data analysis with gretl," *Computational Statistics*, vol. 41, p. 76, 2026. DOI: 10.1007/s00180-026-01753-3.

[46] A. Cottrell and R. Lucchetti, *gretl: Gnu Regression, Econometrics and Time-series Library* [*Computer Software*], 2026. [Online]. Available: https://gretl.sourceforge.net/.

[47] T. D. Blackburn, *Six Sigma: A Case Study Approach Using Minitab®*. Springer Cham, 2022. DOI: 10.1007/978-3-030-96213-5.

[48] Minitab, LLC, *Minitab Statistical Software*, 2026. [Online]. Available: https://www.minitab.com/en-us/.

[49] Wolfram Research, *Evaluation of expressions*, Wolfram Language & System Documentation, 2026. [Online]. Available: https://reference.wolfram.com/language/tutorial/EvaluationOfExpressions.html.

[50] Wolfram Research, *Wolfram|Alpha examples: Statistical inference*, https://www.wolframalpha.com/examples/mathematics/statistics/statistical-inference/, 2026.

[51] Wolfram Research, *Mathematica license pricing options*, 2026. [Online]. Available: https://www.wolfram.com/mathematica/pricing/.

[52] R. Tomaszewski, "Analyzing citations to software by research area: A case study of matlab," *Science & Technology Libraries*, vol. 42, no. 2, pp. 231–246, 2023. DOI: 10.1080/0194262X.2022.2070329.

[53] MathWorks, *Statistics and Machine Learning Toolbox*, 2026. [Online]. Available: https://www.mathworks.com/products/statistics.html.

[54] MathWorks, *MATLAB and Simulink pricing and licensing options*, 2026. [Online]. Available: https://www.mathworks.com/pricing-licensing.html.

[55] T. Siregar, "Enhancing statistical learning and problem solving through the use of geogebra," *Preprints*, Oct. 2025. DOI: 10.20944/preprints202510.1179.v1.

[56] GeoGebra, *GeoGebra math calculators and resources: Probability and statistics modules*, 2026. [Online]. Available: https://www.geogebra.org/math/statistics.

[57] R Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2026. [Online]. Available: https://www.R-project.org/.

[58] CRAN Team, *The Comprehensive R Archive Network*, 2026. [Online]. Available: https://cran.r-project.org.

[59] R Core Team and contributors worldwide, *The r stats package*, 2025. [Online]. Available: https://search.r-project.org/R/refmans/stats/html/00Index.html.

[60] J. Fox and S. Weisberg, *An R Companion to Applied Regression*, Third. Thousand Oaks CA: Sage, 2019. [Online]. Available: https://www.john-fox.ca/Companion/.

[61] J. Fox, *Using the R Commander: A Point-and-Click Interface for R*. Boca Raton, FL: Chapman and Hall/CRC, 2016. DOI: 10.1201/9781315380537.

[62] W. Chang, J. Cheng, J. Allaire, C. Sievert, B. Schloerke, G. Aden-Buie, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges, *Shiny: Web application framework for R*, 2026. DOI: 10.32614/CRAN.package.shiny.

[63] JASP Team, *JASP (Version 0.97.1)[Computer software]*, 2026. [Online]. Available: https://jasp-stats.org/.

[64] T. jamovi project, *Jamovi (version 2.6) [computer software]*, 2025. [Online]. Available: https://www.jamovi.org.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[66] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[67] NumPy Community, *Numpy documentation*, 2026. [Online]. Available: https://numpy.org/doc/stable/.

[68] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2.

[69] The pandas development team, *Pandas documentation*, 2026. [Online]. Available: https://pandas.pydata.org/docs/.

[70] W. McKinney, "Data structures for statistical computing in python," *SciPy 2010*, 2010. DOI: 10.25080/Majora-92bf1922-00a.

[71] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[72] M. L. Waskom, "Seaborn: Statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. DOI: 10.21105/joss.03021.

[73] SciPy Community, *Statistical functions (scipy.stats)*, 2024. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/stats.html.

[74] S. Seabold and J. Perktold, *Statsmodels: Statistical models, hypothesis tests, and data exploration*, 2024. [Online]. Available: https://www.statsmodels.org/stable/index.html.

[75] R. Vallat, *Pingouin: Statistics in python*, version 0.6.2, 2026. [Online]. Available: https://pingouin-stats.org/.

[76] C. J. Bryant, *Researchpy: Produce pandas dataframes that contain relevant statistical information*, 2026. [Online]. Available: https://github.com/Corey-Bryant/researchpy_documentation.

[77] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and Jupyter Development Team, "Jupyter Notebooks—a publishing format for reproducible computational workflows," in *IOS Press*, 2016, pp. 87–90. DOI: 10.3233/978-1-61499-649-1-87.

[78] M. Pyrcz, *Data science interactive python: Interactive illustrations for geostatistics and data analytics*, version 2.5.0, 2026. [Online]. Available: https://github.com/GeostatsGuy/DataScienceInteractivePython.

[79] R. Labbe, *Kalman and Bayesian filters in Python*, 2024. [Online]. Available: https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python.

[80] C. Davidson-Pilon, *Probabilistic Programming and Bayesian Methods for Hackers*. Addison-Wesley Professional, 2015. [Online]. Available: https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers.

[81] J. VanderPlas, *Python Data Science Handbook: Essential Tools for Working with Data*. "O'Reilly Media, Inc.", 2016. [Online]. Available: https://github.com/jakevdp/PythonDataScienceHandbook.

[82] A. B. Downey, *Think Stats: Exploratory Data Analysis*, 3rd ed. O'Reilly Media, Inc., 2025. [Online]. Available: https://allendowney.github.io/ThinkStats/.

[83] Project Jupyter, M. Bussonnier, J. Forde, J. Freeman, B. Granger, T. Head, C. Holdgraf, K. Kelley, G. Nalvarte, A. Osheroff, M. Pacer, Y. Panda, F. Perez, B. Ragan-Kelley, and C. Willing, "Binder 2.0 - reproducible, interactive, sharable environments for science at scale," *SciPy 2018*, 2018. DOI: 10.25080/Majora-4af1f417-011.

[84] Visual Python Team, *Visual python: A gui-based python code generator*, 2026. [Online]. Available: https://www.visualpython.ai/.

[85] Z. Perkel, *Python Statistical Analysis: Fundamentals, Libraries, and How-to*, Nov. 2025. [Online]. Available: https://julius.ai/articles/using-python-statistical-analysis.

[86] S. Vilakati, "Harnessing generative artificial intelligence for teaching statistics in medical research: Strategies for accurate hypothesis testing," *Teaching Statistics*, vol. 48, no. 1, pp. 64–75, DOI: 10.1111/test.70019.

[87] G. O'Regan, in *Concise Guide to Software Engineering: From Fundamentals to Application Methods*, ser. Undergraduate Topics in Computer Science, Cham, Switzerland: Springer International Publishing, 2017, ch. 19.7.1 Aspect-Oriented Software Engineering, pp. 313–314, ISBN: 978-3-319-57749-4 (print), 978-3-319-57750-0 (ebook). DOI: 10.1007/978-3-319-57750-0.

[88] Streamlit, *Streamlit documentation*, Online documentation, 2025. [Online]. Available: https://docs.streamlit.io/ (visited on 05/13/2026).

[89] L. S. Meyers, G. Gamst, and A. J. Guarino, "Assessing normality," in *Data Analysis Using SAS Enterprise Guide*, Cambridge University Press, 2012, ch. 13.

[90] K. Le Boedec, "Sensitivity and specificity of normality tests and consequences on reference interval accuracy at small sample size: A computer-simulation study," *Veterinary Clinical Pathology*, vol. 45, no. 4, pp. 648–656, DOI: 10.1111/vcp.12390.

[91] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[92] P. Royston, "Remark AS R94: A remark on algorithm AS 181: The w-test for normality," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 44, no. 4, pp. 547–551, 1995. DOI: 10.2307/2986146. (visited on 06/14/2026).

[93] R. B. D'Agostino, "An omnibus test of normality for moderate and large size samples," *Biometrika*, vol. 58, no. 2, pp. 341–348, 1971. DOI: 10.2307/2334522.

[94] C. Avram and M. Mărușteri, "Normality assessment, few paradigms and use cases," *Revista Romana de Medicina de Laborator*, vol. 30, no. 3, pp. 251–260, 2022. DOI: 10.2478/rrlm-2022-0030.

[95] C. Correa-Ávarez, J. Rojas-Mora, A. Zumaqué, and O. Bru-Cordero, "Simulation study on the power and sensitivity of sixteen normality tests under different non-normality scenarios," *TecnoLógicas*, vol. 28, no. 62, pp. 1–24, 2025. DOI: 10.22430/22565337.3293.

[96] N. M. Razali and Y. B. Wah, "Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests," *Journal of Statistical Modeling and Analytics*, vol. 2, no. 1, pp. 21–33, 2011. DOI: 10.1515/bile-2015-0008.

[97] M. A. Stephens, "EDF statistics for goodness of fit and some comparisons," *Journal of the American Statistical Association*, vol. 69, no. 347, pp. 730–737, 1974. DOI: 10.1080/01621459.1974.10480196.

[98] H. W. Lilliefors, "On the Kolmogorov-Smirnov test for normality with mean and variance unknown," *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967. DOI: 10.1080/01621459.1967.10482916.

[99] M. A. Stephens, "Edf statistics for goodness of fit and some comparisons," *Journal of the American statistical Association*, vol. 69, no. 347, pp. 730–737, 1974.

[100] M. A. Stephens, "Tests based on EDF statistics," in *Goodness-of-Fit Techniques*, R. B. D'Agostino and M. A. Stephens, Eds., New York: Marcel Dekker, 1986, pp. 97–193, ISBN: 978-0824774873.

[101] SciPy Developers, *scipy.stats.anderson documentation*, SciPy v1.17.0 Reference Guide, 2024. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.anderson.html.

[102] S. G. Kwak and J. H. Kim, "Central limit theorem: The cornerstone of modern statistics," *Korean Journal of Anesthesiology*, vol. 70, no. 2, pp. 144–156, 2017. DOI: 10.4097/kjae.2017.70.2.144.

[103] A. Ghasemi and S. Zahediasl, "Normality tests for statistical analysis: A guide for non-statisticians," *International Journal of Endocrinology and Metabolism*, vol. 10, no. 2, pp. 486–489, 2012. DOI: 10.5812/ijem.3505.

[104] S. Manikandan, "Measures of central tendency: Median and mode," *Journal of Pharmacology and Pharmacotherapeutics*, vol. 2, no. 3, pp. 214–215, 2011. DOI: 10.4103/0976-500X.83300.

[105] M. J. Campbell and M. J. Gardner, "Calculating confidence intervals for some non-parametric methods," *BMJ*, vol. 296, no. 6634, pp. 1454–1456, 1988. DOI: 10.1136/bmj.296.6634.1454.

[106] T. Hesterberg, "What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum," 2014. DOI: 10.48550/arXiv.1411.5279. arXiv: 1411.5279 [stat.OT].

[107] Y. Zhou, Y. Zhu, and W. K. Wong, "Statistical tests for homogeneity of variance for clinical trials and recommendations," *Contemporary Clinical Trials Communications*, vol. 33, p. 101 119, 2023, ISSN: 2451-8654. DOI: 10.1016/j.conctc.2023.101119.

[108] Y. Wang, P. Rodríguez de Gil, Y.-H. Chen, J. D. Kromrey, E. S. Kim, T. Pham, D. Nguyen, and J. L. Romano, "Comparing the performance of approaches for testing the homogeneity of variance assumption in one-factor ANOVA models," *Educational and Psychological Measurement*, vol. 77, no. 2, pp. 305–329, 2016. DOI: 10.1177/0013164416645162.

[109] M. B. Brown and A. B. Forsythe, "Robust tests for the equality of variances," *Journal of the American Statistical Association*, vol. 69, no. 346, pp. 364–367, 1974. DOI: 10.2307/2285659.

[110] T. K. Kim, "T-test as a parametric statistic," *Korean Journal of Anesthesiology*, vol. 68, no. 6, pp. 540–546, 2015. DOI: 10.4097/kjae.2015.68.6.540.

[111] M. Delacre, D. Lakens, and C. Leys, "Why psychologists should by default use welch's t-test instead of student's t-test," *International Review of Social Psychology*, Apr. 2017. DOI: 10.5334/irsp.82.

[112] N. Nachar, "The Mann-Whitney u: A test for assessing whether two independent samples come from the same distribution," *Tutorials in Quantitative Methods for Psychology*, vol. 4, no. 1, pp. 13–20, 2008. DOI: 10.20982/tqmp.04.1.p013.

[113] A. J. Vickers, "Parametric versus non-parametric statistics on the analysis of randomized trials," *BMC Medical Research Methodology*, vol. 5, no. 1, p. 35, 2005. DOI: 10.1186/1471-2288-5-35.

[114] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, no. null, pp. 1157–1182, Mar. 2003.

[115] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *J. Mach. Learn. Res.*, vol. 5, pp. 101–141, Dec. 2004.

[116] D. Wang, H. Zhang, R. Liu, W. Lv, and D. Wang, "T-test feature selection approach based on term frequency for text categorization," *Pattern Recognition Letters*, vol. 45, pp. 1–10, 2014. DOI: 10.1016/j.patrec.2014.02.013.

[117] M. J. Blanca, R. Alarcón, J. Arnau, R. Bono, and R. Bendayan, "Non-normal data: Is ANOVA still a robust option?" *Psicothema*, vol. 29, no. 4, pp. 552–557, 2017. DOI: 10.7334/psicothema2016.383.

[118] T. K. Kim, "Understanding one-way ANOVA using conceptual figures," *Korean Journal of Anesthesiology*, vol. 70, no. 1, pp. 22–26, 2017. DOI: 10.4097/kjae.2017.70.1.22.

[119] S. Lee and D. K. Lee, "What is the proper way to apply the multiple comparison test?" *Korean Journal of Anesthesiology*, vol. 71, no. 5, pp. 353–360, 2018. DOI: 10.4097/kja.d.18.00242.

[120] V. Bewick, L. Cheek, and J. Ball, "Statistics review 10: Further nonparametric methods," *Critical Care*, vol. 8, no. 3, pp. 196–199, 2004. DOI: 10.1186/cc2857.

[121] E. Ostertagová, O. Ostertag, and J. Ková, "Methodology and application of the Kruskal-Wallis test," *Applied Mechanics and Materials*, vol. 611, pp. 115–120, 2014. DOI: 10.4028/www.scientific.net/AMM.611.115.

[122] C. J. Clopper and E. S. Pearson, "The use of confidence or fiducial limits illustrated in the case of the binomial," *Biometrika*, vol. 26, no. 4, pp. 404–413, 1934. [Online]. Available: http://www.jstor.org/stable/2331986 (visited on 06/25/2026).

[123] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *Journal of the American Statistical Association*, vol. 22, no. 158, pp. 209–212, 1927. [Online]. Available: http://www.jstor.org/stable/2276774 (visited on 06/25/2026).

[124] A. Agresti and B. Caffo, "Simple and effective confidence intervals for proportions and differences of proportions result from adding two successes and two failures," *The American Statistician*, vol. 54, no. 4, pp. 280–288, 2000. [Online]. Available: http://www.jstor.org/stable/2685779 (visited on 06/25/2026).

[125] R. G. Newcombe, "Interval estimation for the difference between independent proportions: Comparison of eleven methods," *Statistics in Medicine*, vol. 17, no. 8, pp. 873–890, 1998. DOI: 10.1002/(SICI)1097-0258(19980430)17:8<873::AID-SIM779>3.0.CO;2-I.

[126] S. D. Bolboacă, L. Jäntschi, A. F. Sestraş, R. E. Sestraş, and D. C. Pamfil, "Pearson-fisher chi-square statistic revisited," *Information*, vol. 2, no. 3, pp. 528–545, 2011. DOI: 10.3390/info2030528.

[127] K. Kishore and V. Jaswal, "Statistics corner: Chi-squared test," *Journal of Postgraduate Medicine, Education and Research*, vol. 57, pp. 40–44, 2023. DOI: 10.5005/jp-journals-10028-1618.

[128] M. L. McHugh, "The chi-square test of independence," *Biochemia Medica*, vol. 23, no. 2, pp. 143–149, 2013. DOI: 10.11613/bm.2013.018.

[129] H.-Y. Kim, "Statistical notes for clinical researchers: Chi-squared test and fisher's exact test," *Restorative Dentistry & Endodontics*, vol. 42, no. 2, pp. 152–155, 2017. DOI: 10.5395/rde.2017.42.2.152.

[130] The SciPy Community, *Scipy.stats.montecarlomethod: Monte carlo method for hypothesis tests*, SciPy Documentation, 2026. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.MonteCarloMethod.html.

[131] W. Bergsma, "A bias-correction for cramér's V and tschuprow's T," *Journal of the Korean Statistical Society*, vol. 42, no. 3, pp. 323–328, 2013. DOI: 10.1016/j.jkss.2012.10.002.

[132] A. Stuart, "The estimation and comparison of strengths of association in contingency tables," *Biometrika*, vol. 40, no. 1/2, pp. 105–110, 1953. DOI: 10.2307/2333101.

[133] J. Ekström, "The phi-coefficient, the tetrachoric correlation coefficient, and the Pearson-Yule debate," Tech. Rep., 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:41800911.

[134] M. M. Mukaka, "A guide to appropriate use of correlation coefficient in medical research," *Malawi Medical Journal*, vol. 24, no. 3, pp. 69–71, 2012.

[135] H. Akoglu, "User's guide to correlation coefficients," *Turkish Journal of Emergency Medicine*, vol. 18, no. 3, pp. 91–93, 2018. DOI: 10.1016/j.tjem.2018.08.001.

[136] E. Saccenti, "What can go wrong when observations are not independently and identically distributed: A cautionary note on calculating correlations on combined data sets from different experiments or conditions," *Frontiers in Systems Biology*, vol. 3, p. 1 042 156, 2023. DOI: 10.3389/fsysb.2023.1042156.

[137] R. J. Janse, T. Hoekstra, K. J. Jager, C. Zoccali, G. Tripepi, F. W. Dekker, and M. van Diepen, "Conducting correlation analysis: Important limitations and pitfalls," *Clinical Kidney Journal*, vol. 14, no. 11, pp. 2332–2337, 2021. DOI: 10.1093/ckj/sfab085.

[138] J. Ju, *Penguins*, Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/jaeyunju/penguins (visited on 06/03/2026).

[139] A. Maszanski, *Toothgrowth*, Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/alexmaszanski/toothgrowth (visited on 06/03/2026).

[140] A. Rezaei, *Heartcsv*, Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/arezaei81/heartcsv (visited on 06/03/2026).

[141] R Core Team, *R source code*, The R Project Subversion Repository. [Online]. Available: https://svn.r-project.org/R/trunk/src/library/grDevices/R/calc.R.

[142] NumPy Developers, *Numpy reference: `numpy.histogram_bin_edges`*. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.histogram_bin_edges.html.