Statistical Tests for Data Science (StaTDS). A Python library and Web Client

User Manual
Version 1.0
December 19, 2023

Christian Luna Escudero (i82luesc@uco.es)
Antonio Rafael Moya Martín-Castaño (amoya@uco.es)
José María Luna Ariza (jmluna@uco.es)
Sebastián Ventura Soto (sventura@uco.es)

Contents

List of Tables

List of Figures

1

Introduction

Data Science has experienced exponential growth thanks to the technological advances that have increased the volume of available data. In the Data Science field, there is an ongoing need to develop and compare algorithms to determine the most advantageous one, a pivotal process known as algorithm evaluation [10]. This algorithm evaluation process is not simple, and it includes dozens of techniques and methods to analyze the outcomes of the algorithms. In these analyses, the use of statistical tests is essential. Getting statistically significant differences tells the experts whether one algorithm (or group of them) is substantially different from another (or group of them) by using statistical testing. In this context, it is pertinent to consider two types of tests:

- **Parametric tests** rely on specific assumptions regarding the data distribution, including normality and homogeneity of variance. These tests are most appropriate when the data meet these assumptions, enabling more accurate inferences about population parameters, such as the mean or variance.
- **Non-parametric tests**, in contrast, do not make any assumptions about the underlying data distribution, making them more useful when data do not follow a normal distribution or when parametric assumptions cannot be satisfied. These tests classify, order, or count the data, and they are often less powerful in detecting subtle differences.

In summary, the selection between parametric and non-parametric tests depends on the data characteristics and the assumptions held. Parametric tests are powerful when data satisfy specific assumptions, while non-parametric tests are robust in the presence of non-normal data or data including few samples.

Statistical Tests for Data Science (StaTDS) stands out as a distinctive Python library geared towards the statistical comparison of algorithms, built entirely in pure Python. StaTDS guarantees its perfect functionality and

durability by not depending on external libraries, a common practice in some similar libraries. It supports an extensive range of statistical tests (23 different tests) for diverse use cases, so it outperforms existing libraries for statistical tests. The library includes some tests to determine whether parametric or non-parametric tests the user should use (normality and homoscedasticity test). Not only is it meticulously crafted for Data Scientists, but StaTDS also extends its utility as a web application so a broader audience can use it, ensuring accessibility and ease of use for all users.

The rest of this manual is structured as follows. Chapter 2 reviews the literature and current status of comparative algorithms, as well as the use of statistical tests; Chapter 3 describes how to get and install StaTDS software on your computer; Chapter 4 describes all available tests included in StaTDS and its configurations; Chapter 5 shows the requirements, functionalities and use of the web interface. Finally, Chapter 6 describes what is the best way of reporting bugs.

2 Preliminary

In this chapter, we will address the tasks involved in hypothesis testing. We will detail the required procedures and discuss potential issues, such as Type I and Type II errors. Subsequently, we will present all the statistical tests available in the StaTDS ecosystem (normality, homoscedasticity, parametric and non-parametric). We clarify crucial insights about statistical tests.

2.1 Hypothesis Testing

In the domain of statistics [21], a hypothesis test serves as a method employed to either accept or reject a given hypothesis. In essence, a hypothesis test is a tool used to ascertain whether a hypothesis regarding the value of a statistical parameter within a population is to be supported or discarded. Within the framework of a hypothesis test, an analysis of a data sample is conducted, and based on the resultant findings, a decision is made to either maintain or dismiss a hypothesis concerning a previously established parameter within the population.

In the domain of machine learning and data science, these tests find utility in comparing the outcomes of algorithms or models, to determine whether the observed disparities between them possess statistical significance. Such tests aid in addressing questions such as whether one algorithm genuinely outperforms another in terms of overall performance, or whether the disparities observed may be attributed solely to random chance. When undertaking a hypothesis test, it is imperative to adhere to the prescribed steps, as shown in Figure 2.1.

1. **Formulate the hypothesis**, which will be applied to establish a scientific statement based on experimentation and statistical study. For this purpose, we will set forth the null and alternative hypotheses.

    - Null hypothesis (H0): The researcher formulates this hypothesis, which is temporarily assumed to be true. It can only be disproven

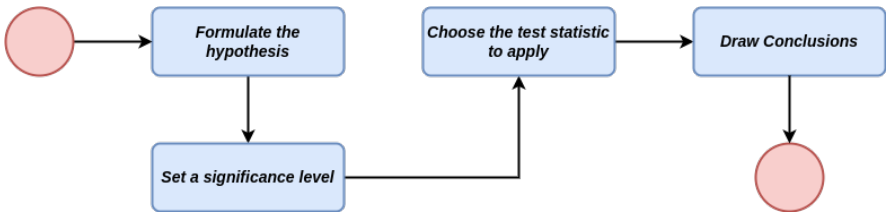


Figure 2.1: Steps to perform a hypothesis test

through a procedure for refutation (application of statistical testing). Generally, it represents the idea that no marked dissimilarities are evident between the compared algorithms.

- Alternative hypothesis (H1): It aspires to supplant or replace the null hypothesis, and it often suggests that there are differences or an effect.

2. **Set a significance level** (α). The researcher establishes this level at the beginning of the process, representing the maximum probability of error they choose to accept when rejecting the null hypothesis. α typically takes small values, such as 0.01 or 0.05. When the significance level is 0.05 (5%), for example, the confidence level is 0.95 (95%), and the two add up to 1 (100%). Once the significance level is defined, two types of errors can occur, such as Type I and Type II errors, which we will discuss as follows:

   - Type I Error (α error) occurs when we reject H0 when it is true. Thus, the probability of making a Type I error is α, which is the significance level we established for our hypothesis test.
   - Type II Error (β error) happens when we accept H0 when it is false. In other words, the probability of making a Type II error in beta (β) relies on the test's power (1-β). In order to reduce the risk of making a Type II error, we can ensure that the test has sufficient power. To do this, we must ensure the sample size is large enough to detect a difference when it truly exists.

3. **Choose the test statistic to apply**. The researcher should select a statistical test that suits the nature of the data and the hypothesis they want to address. In Section 2.2, we will present the types of statistical tests and the various possible scenarios when comparing algorithms.

4. **Draw Conclusions**. The researcher should conduct the statistical test and, based on the p-values, determine whether it meets the significance

level established earlier. This step involves determining whether to reject H0, which implies accepting H1 if H0 is not validated. It is crucial to clarify that when H0 is not validated, it should not be assumed as approved; it should be explicitly stated as "not rejected". The utilization of the word "accept" is avoided in this context due to the potential for a Type II error. Since the probability of making a Type II error can often be relatively high, it is essential to prevent committing it by prematurely accepting H0.

Hypothesis tests can be classified into two types [21]: 1) parametric tests, which assume underlying statistical distributions in data; 2) non-parametric tests, which do not rely on any distribution. Additionally, the application of these tests can vary depending on the scenario and the aspects considered in the results:

- **Pairwise Comparisons.** They allow determining if two algorithms have similar behaviour. There are two main scenarios. The first involves comparing two algorithms within a single dataset, where hypothesis tests such as the McNemar test are employed to assess differences and determine the effectiveness or superiority of one over the other. The second scenario addresses the comparison of algorithms across several data sets, which allows the analysis of whether the effectiveness of the algorithms varies with the specific dataset and if one demonstrates consistently better performance across different contexts.

- **Multiple Comparisons.** It allows determining if various algorithms behave similarly across multiple data sets. All possible pairs of hypotheses will undergo testing. Therefore, $\mu_{\phi^1} = \mu_{\phi^2} = ... = \mu_{\phi^k}$ is the statement under examination.

Once a Multiple Comparisons analysis has been performed, if significant differences arise after concluding, post-hoc tests are necessary. Post-hoc tests determine where our differences come from, and it is possible to consider a comparison among all pairs of algorithms or a comparison between a control algorithm and the rest.

2.2 Statistical Tests

In this section, we will outline the various statistical tests offered by the StaTDS library, explaining the utility of each and the foundational principles in which they are rooted. They enable the evaluation of previously

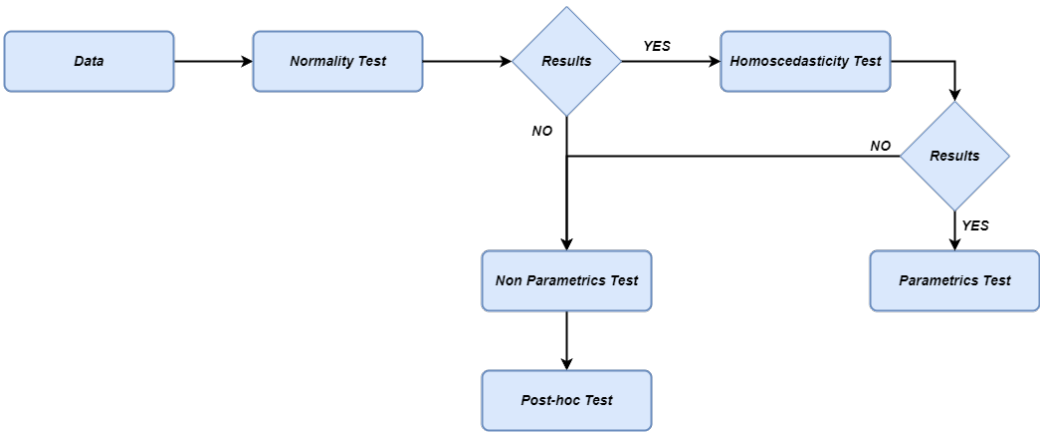


Figure 2.2: Decision on Choosing Parametric vs. Non-Parametric Tests

defined hypotheses. The selection of a specific statistical test is influenced by the nature of the data and the chosen scenario. Figure 2.2 provides a concise guide illustrating when to use each test (normality, homoscedasticity, parametric, and non-parametric tests) developed. In the case of post-hoc tests, they have been primarily developed for non-parametric tests. Finally, Table 2.1 presents a conceptual map of all the tests available for each specified approach.

2.2.1 Normality and Homoscedasticity

Normality and homoscedasticity are fundamental assumptions when determining the applicability of parametric tests in statistical studies. Data normality suggests that it adheres to a Gaussian (normal) distribution, as represented by its probability density function in Equation 2.1.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{2.1}$$

where:

- x denotes the variable for which the PDF is evaluated.
- μ is the mean of the distribution, which locates the center of the peak of the distribution.
- σ is the standard deviation of the distribution, which measures the spread or width of the peak.

Type Test	Name
Normality	Shapiro-Wilk D'Agostino-Pearson Kolmogorov-Smirnov
Homoscedasticity	Levene Bartlett
Parametrics	T Test ANOVA
Non Parametrics	Wilcoxon Binomial Sign Mann-Whitney U Friedman Friedman Aligned Ranks Quade
Post Hoc	Nemenyi Bonferroni Li Holm Holland Finner Hochberg Hommel Rom Schaffer

Table 2.1: Summary Parametric and Non-Parametric Tests

Statisticians use normality tests to evaluate whether a dataset follows a specific distribution. These tests gauge how closely a dataset aligns with a normal distribution and estimate the likelihood that the dataset's underlying random variable follows a normal distribution. We can broadly classify these statistical tests into two categories:

- Graphical Methods: These methods utilize visual representations of data for analysis. Notable examples include:
  - Histogram: It displays the frequency of different values in a dataset. A bell-shaped curve histogram indicates normality.
  - Q-Q Plot (Quantile-Quantile Plot): It matches the data quantiles with those of a standard normal distribution. When data follow a normal distribution, the points roughly align along a straight line.

  - P-P Plot (Percentile-Percentile Plot): Functions like a Q-Q Plot but contrasts percentiles instead of quantiles.
  - Box Plot: Showcases the median, quartiles, and outliers, offering a snapshot of the data's distribution.

- Analytical Methods: These techniques test the data against the presumption that they follow a normal distribution.

These methods offer various approaches to assess normality, and researchers or analysts might select them based on the sample size and the specific context in which they analyze the data. In statistics, a sequence (or a vector) of random variables is homoscedastic if all its random variables have the same finite variance. Statisticians also call this "homogeneity of variance". The complementary notion is called heteroscedasticity, also known as heterogeneity of variance. These tests are essential [18] for validating the assumptions of normality and homogeneity in various statistical analyses such as the Student's t-test and ANOVA. Next, we will delve a bit further into each of these tests.

2.2.1.1 Normality Test

Shapiro-Wilk. Statisticians widely use the Shapiro-Wilk test [23] to assess normality in a dataset. This test examines the null hypothesis that a population that follows a normal distribution produces a sample x_1, x_2, ..., x_n. Researchers introduced the Shapiro-Wilk test in 1965. It calculates a W statistic to determine if a random sample, x_1, x_2, ..., x_n, originates from a normal distribution.

$$W = \frac{(\sum_{i=1}^{n} a_i x_{(i)})^2}{\sum_{i=1}^{n} x_i \overline{x})^2} \tag{2.2}$$

where:

- $x_{(i)}$ represents the ordered sample values ($x_{(1)}$ is the smallest, not to be confused with x_i).
- $\overline{x}$ is the sample mean: $\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$
- a_i are constants generated from the means, variances and covariances of the order statistics of a sample of size n from a normal distribution (see Table 5 in [23]).

Small values of W indicate a departure from normality, and the percentage points for the W statistic, obtained through Monte Carlo simulations, were reproduced by *Shapiro et al.* (see Table 4 in [23]). This test has performed favorably in comparison studies with other goodness-of-fit tests.

D'Agostino-Pearson. The procedure evaluates the null hypothesis, positing that samples derive from a normally distributed population. This test combines both the skewness coefficient—which denotes symmetry and typically has a value of 0 for a normal distribution—and the kurtosis coefficient, which measures peakedness and is usually 0 for a normal distribution. Together, these produce a statistic and p-value. However, the Shapiro-Wilk test is more powerful [12]. We begin by discussing the Skewness and Kurtosis tests. Subsequently, we introduce the D'Agostino-Pearson Test [2], which combines the two aforementioned tests.

The skewness test [3] assesses if the data's skewness significantly deviates from zero. Relying on the principle that data from a normal distribution makes the test statistic $Z(\sqrt{b_1})$ adhere to a standard normal distribution. Here, "skew" is the sample data's skewness, and we determine the standard error using a particular formula:

$$skew = \sqrt{b_1} = \frac{m_3}{m_2^{3/2}} \tag{2.3}$$

$$m_k = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X}^k) \tag{2.4}$$

here, "n" stands for the sample size and $\bar{X}$ is the sample mean:

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{2.5}$$

$$Z(\sqrt{b_1}) = \delta log(Y/\alpha + \sqrt{(Y/\alpha)^2 + 1}) \tag{2.6}$$

where:

$$Y = \sqrt{b_1}\sqrt{\frac{(n+1)(n+3)}{6(n-2)}} \tag{2.7}$$

$$\beta_2(\sqrt{b_1}) = \frac{3(n^2+27n-70)(n+1)(n+3)}{(n-2)(n+5)(n+7)(n+9)} \tag{2.8}$$

$$W^2 = \sqrt{2(\beta_2(\sqrt{b_1})-1)} - 1 \tag{2.9}$$

$$\delta = \frac{1}{\sqrt{log(W)}} \tag{2.10}$$

$$\alpha = \sqrt{\frac{2}{(W^2-1)}} \tag{2.11}$$

The kurtosis test evaluates whether the data's kurtosis significantly varies from zero. This test hinges on the idea that for normally distributed data, the test statistic $Z(b_2)$ will follow a standard normal distribution. In this context, "kurt" represents the sample data's kurtosis, and the standard error comes from a specific formula

$$kurt = b_2 = \frac{m_4}{m_2^2} \tag{2.12}$$

$$m_k = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X}^k) \tag{2.13}$$

here, "n" stands for the sample size and $\bar{X}$ is the sample mean:

$$Z(b_2) = ((1 - \frac{2}{9A}) - \sqrt[3]{\frac{1-2/A}{1+x\sqrt{2/(A-4)}}})/\sqrt{2/(9A)} \tag{2.14}$$

where:

$$A = 6 + \frac{8}{\sqrt{B_1(b_2)}}(\frac{2}{\sqrt{B_1(b_2)}}) + \sqrt{(1 + \frac{4}{B_1(b_2)})} \tag{2.15}$$

- $\sqrt{B_1(b_2)}$ is the third standardized moment of b_2.

$$\sqrt{B_1(b_2)} = \frac{6(n^2-5n+2)}{(n+7)(n+9)}\sqrt{\frac{6(n+3)(n+5)}{n(n-2)(n-3)}} \tag{2.16}$$

- x denotes the standardized version of b_2.

$$x = \frac{b_2 - E(b_2)}{\sqrt{var(b_2)}} \tag{2.17}$$

- $E(b_2)$ is the mean of b_2.

$$E(b_2) = \frac{3(n-1)}{n+1} \tag{2.18}$$

- $var(b_2)$ denotes the variance of b_2.

$$var(b_2) = \frac{24n(n-2)(n-3)}{(n+1)^2(n+3)(n+5)} \tag{2.19}$$

Finally, the D'Agostino-Pearson statistic is defined as [3]:

$$Z(\sqrt{b_1})^2 + Z(b_2)^2 \sim \chi^2(2) \tag{2.20}$$

This test should generally not be used for data sets with less than 20 elements [4]. According to [3], the D'Agostino-Pearson test can be used for all samples where n $\geq$ 9, particularly recommending its use over Shapiro-Wilk for n $>$ 50.

Kolmogorov-Smirnov. The Kolmogorov–Smirnov test [15] serves as a measure of goodness of fit. Specifically, when testing whether samples follow a normal distribution, they are standardized and then measured against a standard normal distribution. This method adjusts the mean and variance of the benchmark distribution to match the sample's estimates. However, using these standards to define the reference distribution changes the fundamental distribution of the test's statistics. Despite these modifications, many studies suggest the test is less precise in identifying normality than the Shapiro–Wilk or D'Agostino-Pearson tests. Yet, these alternative tests have their limitations. For instance, the Shapiro–Wilk test struggles with samples containing many identical values. The Kolmogorov–Smirnov statistic quantifies the distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution.

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{X_i \leq x\}} \tag{2.21}$$

2.2.1.2 Homoscedasticity Test

Levene. Levene's test [13] is a statistical method used for for checking if multiple samples share the same "spread" or variance. To clarify, if one class's scores are all over the place and another's are tightly clustered, your comparisons might be biased. That is where 'homogeneity of variance' comes in, ensuring a level playing field. For instance, the analysis of variance (ANOVA) relies on the fundamental assumption that the variances among the different groups or samples are equal. If this assumption does not hold, the ANOVA results might be misleading or inaccurate. Thus, before using such tests, it's wise to use Levene's test to confirm the homogeneity assumption.

- H0 $\sigma_1^2 = \sigma_2^2 = ... = \sigma_k^2$
- H1 $\sigma_i^2 \neq \sigma_j^2$ for at least one pair (i, j)

The Levene statistic is calculated as:

$$W = \frac{(N-k)}{(k-1)} \frac{\sum_{i=1}^{k} N_i (\overline{Z}_{i.} - \overline{Z}_{i.})^2}{\sum_{i=1}^{k} \sum_{j=1}^{N_i} (Z_{ij} - \overline{Z}_{i.})^2} \tag{2.22}$$

where Z_{ij} can have one of the following three definitions:

1. $Z_{ij} = |Y_{ij} - \overline{Y}_i|$ where $\overline{Y}_i$ is the mean of the $i-th$ subgroup.
2. $Z_{ij} = |Y_{ij} - \tilde{Y}_i|$ where $\tilde{Y}_i$ is the median of the $i-th$ subgroup.
3. $Z_{ij} = |Y_{ij} - \overline{Y}'_i|$ where $\overline{Y}'_i$ is the 10 % trimmed mean of the $i-th$ subgroup.

$\overline{Z}_{i.}$ are the group means of the Z_{ij} and $\overline{Z}_{..}$ is the overall mean of the Z_{ij}. The three choices for defining Z_{ij} determine the robustness and power of Levene's test. Levene's original paper [13] only proposed using the mean. The Levene test rejects the hypothesis with the critical value of F distribution with $k-1$ and $N-k$ degrees of freedom.

Bartlett. Bartlett's test [22] is used to test if k samples have equal variances. It is more sensitive than Levene to depart from normality. If you have strong evidence that your data do come from a normal, or nearly normal, distribution, then Bartlett's test has better performance. The Bartlett test defined the following hypothesis:

- H0 $\sigma_1^2 = \sigma_2^2 = ... = \sigma_k^2$

- H1 $\sigma_i^2 \neq \sigma_j^2$ for at least one pair (i, j)

The Bartlett statistic is defined as:

$$T = \frac{(N-k)log(s_p^2) - \sum_{i=1}^{k}(N_i - 1)log(s_i^2)}{1 + (1/(3(k-1)))((\sum_{i=1}^{k} 1/(N_i - 1)) - 1/(N-k))} \tag{2.23}$$

where:

- N is the total sample size.
- N_i is the sample size of the i-th group.
- k is the number of groups.
- s_p^2 is the pooled variance. The pooled variance is a weighted average of the group variances. Their form is:

$$s_p^2 = \sum_{i=1}^{k}(N_i - 1)s_i^2/(N-k) \tag{2.24}$$

- s_i^2 is the variance of the i-th group.

The Bartlett test rejects the hypothesis with the critical value of χ^2 distribution with $k-1$ degrees of freedom.

2.2.2 Parametrics

Parametric statistical tests make certain assumptions about the population parameters and the distributions from which the data originated. They assume that data is distributed in a specific way, commonly assuming that data follows a normal distribution. Some examples of parametric tests include Student's T-tests and ANOVA tests, which assume data is from a normal distribution.

2.2.2.1 Pairwise Comparisons

T-test. The t-test, also known as the Student's t-test [21], is specifically designed to evaluate the differences between the means of two groups. If your study involves more than two groups or requires multiple pairwise comparisons, consider using an ANOVA or a post-hoc test instead. It is essential to note that the t-test operates under the same assumptions as other parametric tests. Thus, your data must:

1. be independent.
2. follow (approximately) normal distribution.
3. have a similar amount of variance withing each group being compared (homogeneity of variance / homoscedasticity).

If your data do not fit these assumptions, you can try a non-parametric alternative such as Wilconxon Test for data with unequal variances. When choosing a t-test, you will need to consider two things: whether the groups being compared come from a single population or two different populations, and whether you want to test the difference in a specific direction.

- If the groups come from a single population, perform a paired t-test.
- If the groups come from two different populations, perform a unpaired t-test.

The t-test estimates the true difference between two group means using the ratio of the difference in group means over the pooled standard error of both groups.

In the cases of unpaired t-test, t-value follows the next equation:

$$t = \frac{\overline{x}_1 - \overline{x}_2}{s_{\overline{x}_1 - \overline{x}_2}} \tag{2.25}$$

where:

- t is the t value.
- x_1 mean value of the first group.
- x_2 mean value of the second group.
- $s_{\overline{x}_1 - \overline{x}_2}$ is the standard deviation of the mean value difference. To calculate $s_{\overline{x}_1 - \overline{x}_2}$ we follow:

$$s_{\overline{x}_1 - \overline{x}_2} = s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}} \tag{2.26}$$

where:

- n_1 size of the first group.
- n_2 size of the second group.

- s_p is the estimated value for the standard deviation:

$$s_p = \sqrt{\frac{(n_1 - 1)s_i^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}} \tag{2.27}$$

You can compare your calculated t-value against the values in a critical value chart (e.g., Student's t table) to determine whether your t value is greater than what would be expected by chance. The number of degrees of freedom is given by $df = n_1 + n_2 - 2$. If so, you can reject the null hypothesis and conclude that the two groups are in fact different.

In the cases of paired t-test, t-value follows the next equation:

$$t = \frac{\overline{x}_{diff}}{s_{\overline{x}}} \tag{2.28}$$

where $s_{\overline{x}}$ is the standard error of the mean value:

$$s_{\overline{x}} = \frac{s_{diff}}{\sqrt{N}} \tag{2.29}$$

- x_{diff} = Difference between the groups.
- $\overline{x}_{diff}$ = Mean value of the difference between the two groups.
- N = Sample size.
- $_{diff}$ = Standard deviation.
- $s_{\overline{x}}$ = Estimated standard error of mean.

2.2.2.2 Multiple Comparisons

ANOVA. Analysis of Variance Models (ANOVA) is one of the most common analysis activities in multiple comparison. It is used to check whether or not there is a statistically significant difference between the means of three or more independent groups. Two types are the most common: one-way and two-way.

- **One-way ANOVA:** It is used to determine how one factor affects a response variable [16]. It is useful when we want to compare the effect of multiple levels of one factor and we have multiple observations at each level. The factor can be either discrete (different machine, different plants, different shifts, etc.) or continuous (different gas flows, temperatures, etc.). As a matter of clarification, let us consider a

scenario where a professor wants to determine if three different study techniques lead to varying exam scores. To test this, the professor recruits 30 students for a study, randomly assigning each to one of the three study techniques for preparing an exam. After a month, all students take the same test. The goal is to see if the study technique used significantly impacts the exam scores.

In general, the ANOVA table for the one-way case is given by 2.2:

Source	Degrees of Freedom	Sum of Squares	Mean Square	F_0
Factor	$k-1$	$SS_F = \sum_{i=1}^{k} n(\bar{x}_i \bar{x}_{..})^2$	$MS_F = SS_B/(k-1)$	$F = MS_B/MS_W$
Residual	$N-k$	$SS_E = \sum_{i=1}^{k}\sum_{j=1}^{k} n(\bar{x_{ij}} \bar{x}_{i.})^2$	$MS_E = SS_E/(N-k)$	
Corr. Total	$N-1$	$SS_T = SS_F + SS_E$	$F = MS_B/MS_W$	

Table 2.2: One-Way ANOVA Table

where:

- k is the number of groups
- N is the total number of samples
- SS_F (Sum of Squares Between Factor): where n_i is the number of subjects in the i-th group.
- SS_E (Sum of Squares Within Factor): where s_i is the standard deviation of the i-th group.
- SS_T (Total Sum of Squares)
- MS_F (Mean Square Between Factor)
- MS_E (Mean Square Within Factor)

■ **Two-way ANOVA:** Used to determine how two factor affects a response variable, and to determine wheter or not there is an interactive between the two factor on the response variable. As a matter of clarification, let us imagine a botanist interested in understanding whether plant growth is influenced by two factors: sunlight exposure and watering frequency. To investigate this, she plants 40 seeds and allows them to grow for two months. During this period, she varies the conditions for each plant in terms of sunlight exposure and how frequently they are watered. After two months, the botanist measures and records the height of each plant. The purpose of this study is to analyze not only the individual effects of sunlight exposure and watering frequency on plant growth but also to see if there's an interaction effect between these two factors.

2.2.3 Non-parametrics

Non-parametric tests are statistical techniques that do not assume a specific distribution of the data, being particularly useful when the assumptions of parametric tests, such as normality and homoscedasticity, cannot be met. They are often used with ordinal or nominal data and can be applied in situations where sample sizes are small and/or the data have significant outliers.

2.2.3.1 Pairwise Comparisons

Wilcoxon. The Wilcoxon signed-ranks test [5] is a non-parametric alternative to the paired t-test, which ranks the differences in performances of two samples for each data set, ignoring the signs and compares the ranks for the positive and the negative differences. The steps for this test are the following:

1. Rank the module of the performance differences between both algorithms.

2. Calculate the sum of the ranks R^+ and R^- where the first (resp. the second) algorithm outperforms the other.

3. Calculate $\mathrm{T} = \min(R^+, R^-)$.

4. Calculate the statistics z.

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{2.30}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{2.31}$$

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{2.32}$$

Depending on the number of samples in data, the statistic can be compared to a normal distribution (if $N > 25$) or there is a need to use tables with critical values [5].

Binomial Sign. Binomial Sign Test or Sign Test [21] is a statistical method to compare two algorithms in several data sets. It is a non-parametric test that counts the number of losses, ties and wins. Given pairs of observations (such as weight pre- and post-treatment) for each subject, the sign test determines if one member of the pair (such as pre-treatment) tends to be greater than (or less than) the other member of the pair (such as post-treatment). The steps of this test will be outlined below:

- Calculate the differences between the samples:
  - For each pair, subtract one observation from the other, denoting the differences as d.
  - If $d = 0$, that pair is ignored for the rest of the analysis.
- Assign Signs to the differences:
  - If $d_i > 0$, assign a "+" to that pair of observations.
  - If $d_i < 0$, assign a "-" to that pair of observations.
- Count the signs: Count the total number of "+" signs and the total number of "-" signs.
- Determine the test statistic: The Sign Test statistic is min(number of +, number of -).
- Find the p-value using a binomial distribution. The null hypothesis states there's an equal number of signs (i.e. 50/50). Therefore, the test is a simple binomial experiment with:
  - Probability(P) = 0.5.
  - Number of trials = num(d_i) $\neq 0$
  - Number of successes = min(number of +, number of -).

$$P(x) = \binom{n}{x} p^x (1-p)^{n-x} = \frac{n!}{(n-x)!x!} p^x (1-p)^{n-x} \tag{2.33}$$

Mann-Whitney-U. In statistics, the Mann–Whitney U test [21] (also known as the Mann–Whitney–Wilcoxon (MWW/MWU)) is a non-parametric test of the null hypothesis that for randomly selected values X and Y from two populations, the probability of X being greater than Y is equal to the probability of Y being greater than X. It is an alternative to the t-test for independent samples when the data do not meet the requirements of normality and homogeneity of variances required by the t-test. Furthermore, it

is particularly useful when the sample sizes are small. The steps of this test will be outlined below:

- Combine the Data: Begin by combining the two samples into a single data set.
- Assign Ranks: Order the data from smallest to largest and assign a rank to each value. In the case of ties (identical values), assign the average of the ranks that would correspond to those values.
- Sum Ranks: Separate the ranks by the original sample and sum the ranks for each group.
- Calculate the U Statistic: Use the sum of ranks R_1 y R_2 for each sample and the sample sizes n_1 y n_2, to calculate the U statistics for each group:

$$U_1 = R_1 - \frac{n_1(n_1+1)}{2} \tag{2.34}$$

$$U_2 = R_2 - \frac{n_2(n_2+1)}{2} \tag{2.35}$$

$$\text{where } U = \min(U_1, U_2) \tag{2.36}$$

The following equation was proposed in [21] to provide an approximation of the U statistic in a normal distribution (0,1).

$$z = \frac{U - \frac{n_1 n_2}{2}}{\sqrt{\frac{n_1 n_2 (n_1+n_2+1)}{12}}} \tag{2.37}$$

2.2.3.2 Multiple Comparisons

Friedman. The Friedman test is a non-parametric test analogue of the ANOVA. The objective of this test is to determine if we may conclude from a sample of results that there is difference among treatment effects. First step to calculate the test statistic is to convert the original results to ranks. Thus, it ranks the algorithms for each problem separately, the best performing algorithm should have the rank of 1, the second best rank 2, etc. In case of ties, average ranks are computed.

Let r_i^j be the rank of the jth of k algorithms on the ith of n samples. Friedman test requires to compute of the average ranks of algorithms ($R_j = \frac{1}{n}\sum_i = i^n r_j^j$). The Friedman test define the null hypothesis how:

- H0 $R_1 = R_2 = ... = R_k$
- H1 $R_i \neq R_j$ for at least one pair (i, j)

The Friedman statistic (see Equation 2.38) is distributed according to χ^2 with $k-1$ degrees of freedom, when n and k are big enough (as a rule of a thumb, n > 10 and k > 5). For a smaller number of algorithms and data sets, exact critical values have been computed.

$$\chi_F^2 = \frac{12n}{k(k+1)[\sum_j = 1^k - \frac{k(k+1)^2}{4}]} \tag{2.38}$$

Iman and Davenport showed in [11] that Friedmans χ_F^2 presents a conservative behavior and proposed a better statistic (see Equation 2.39) which is distributed according to the F-distribution with $k-1$ and $(k-1)(n-k)$ degrees of freedom. Furthermore, the p-value could be computed through normal approximations.

$$F_F = \frac{(n-1)\chi_F^2}{n(k-1)\chi_F^2} \tag{2.39}$$

Friedman Aligned Ranks. The Friedman Aligned Ranks test [6] is a variation of the Friedman test that seeks to enhance the test's power by aligning the observations by subtracting a measure of central tendency. In this case, we calculate the average performance of each data set as the average performance achieved by all algorithms. Then, it calculates the difference between the performance obtained by an algorithm and the value of the location. We repeat this step for algorithms and data sets. The resulting values, termed aligned observations, which maintain their distinctiveness about the data set and the amalgamation of algorithms with which they affiliate, are subsequently ranked from 1 to kn. Although the general procedure of rank comparison remains the same, the adjustment made before the analysis allows for greater sensitivity in detecting differences between groups, especially when the number of algorithms for comparison is limited.

The Friedman Aligned Ranks test statistic can be written as shown in Equation 2.40:

$$T = \frac{(k-1)\sum_{j=1}^{k} R_j^2 - \left(\frac{k(n^2)}{4(kn+1)^2}\right)}{\{k(n(kn+1)(2kn+1))/6\} - (1/k)\sum_{i=1}^{n} R_i^2} \tag{2.40}$$

- $\hat{R}_i$: is equal to the rank total of the ith data set

- $\hat{R}_j$ is the rank total of the jth algorithm

The test statistic T is compared for significance with a chi-square distribution for $k-1$ degrees of freedom.

Quade. Quade test [6] does not consider all data sets to be equals in terms of importance a difference of Friedman test. In this case, the Quade test conducts a weighted ranking analysis of the sample of results. In this case, the rankings calculated on each data set are scaled depending on the observed differences in the performance of the algorithms.

The process begins by identifying the ranks r_{ji} in a manner similar to the Friedman test. Subsequently, using the original values of each classifier, denoted as xij. Ranks are assigned to the data sets based on the size of the sample range in each set. The sample range in data set i is the difference between the largest and smallest observation within that set.

$$\text{Range in data set:} i = \max_j\{x_{ij}\} - \min_j\{x_{ij}\} \tag{2.41}$$

Obviously, there are n sample ranges, one for each data set. Assign rank 1 to the data set with the smallest range, rank 2 to the second smallest, and so on, to the data set with the largest range, which gets rank n. Use average ranks in case of ties. Let $Q_1, Q_2, ..., Q_n$ be the ranks assigned to data sets $1, 2, ..., n$, respectively.

Finally we adjust the Q_i ranks by multiplying Q_i by a value that measures the difference between the rank in the current dataset (r_i^j) and the average rank within data sets ($k + 1/2$). Thus, we consider S_{ij} as:

$$S_{ij} = Q_i[r_i^j - \frac{k+1}{2}] \tag{2.42}$$

is a statistic that represents the relative size of each observation within the data set, adjusted to reflect the relative significance of the data set in which it appears. In [6] to establish a relationship of Quade Test with the Friedman test, we can use use rankings without average adjusting:

$$W_{ij} = Q_i[r_i^j] \tag{2.43}$$

Let Sj denote the sum for each classifier, $S_j = \sum_{i=1}^{n} S_{ij}$ for $j = 1, 2, ..., k$. The test statistic is:

$$T_3 = \frac{(n-1)B}{A_2 - B} \tag{2.44}$$

where:

- $A_2 = n(n+1)(2n+1)(k)(k+1)(k-1)/72$
- $B = \frac{1}{n}\sum_{j=1}^{k} S_j^2$

T_3 is distributed according to the F-distribution with $k-1$ and $(k-1)(n-1)$ degrees of freedom. In the case that $A_2 = B$, the point should be considered to be within the critical region of the statistical distribution, and the p-value should be calculated as $(1/k!)^{n-1}$.

2.2.4 Post-hoc

After finding a statistically significant result, we use a post-hoc analysis to pinpoint the sources of our differences. Once we reject the null hypothesis, we can compare all pairs or contrast all algorithms with a control.

Authors demonstrated in [6] that when working with the most commonly used non-parametric statistical tests, one can calculate the "z value" which is used for post hoc analysis. The notation used in the computation of the z is as follows:

- k is the number of algorithms being compared.
- n is the number of samples / data sets.
- Friedman test: the expression for computing the test statistic in the Friedman test [6] is as follows:

$$z = (R_i - R_j)/\sqrt{\frac{k(k+1)}{6n}} \tag{2.45}$$

where R_i, R_j are the average rankings by Friedman of the algorithms compared.

- Friedman Aligned Ranks test:

$$z = (\hat{R}_i - \hat{R}_j)/\sqrt{\frac{k(n+1)}{6}} \tag{2.46}$$

where $\hat{R}_i, \hat{R}_j$ are the average rankings by Friedman Aligned Ranks of the algorithms compared.

- Quade test: The test statistic for comparing two algorithms can easily be calculated with Equation 2.47.

$$z = (T_i - T_j)/\sqrt{\frac{k(k+1)(2n+1)(k-1)}{18n(n+1)}} \tag{2.47}$$

where $T_i = \frac{W_i}{n(n+1)/2}$, $T_j = \frac{W_j}{n(n+1)/2}$, and W_i and W_j are the rankings without average adjusting by Quade of the algorithms compared. In fact, T_i and T_j compute the correct average magnitudes.

$$Wj = \sum_{i=1}^{n} W_{ij} \text{for } j = 1, 2, ..., k \tag{2.48}$$

The z value in all cases is used to find the corresponding probability (p-value) from the table of normal distribution N(0,1), which is then compared with an appropriate level of significance α. The post-hoc tests differ in the way they adjust the value of a to compensate for multiple comparisons.

2.2.4.1 Bonferroni-Dunn

The Bonferroni-Dunn test is a one-step method that adjusts the α value based on the number of comparisons, using the formula $\alpha' = \alpha/K$, where K represents the number of comparisons.

- For a comparison against a control: $K = k - 1$
- For an all pairs comparison: $K = k(k-1)/2$

While it remains conservative and effectively controls type I errors, it may have limited statistical power, especially when making many comparisons.

2.2.4.2 Holm

The Holm test operates as a step-down procedure, systematically adjusting the value of α. Begin by ordering the p-values from smallest to largest: $p_1 \leq p_2 \leq ... \leq p_{k-1}$. Correspondingly, assign these to hypotheses $H_1, H_2, ..., H_{k-1}$. In the Holm procedure:

- Initiate with the most significant p-value.
- Reject H_1 if p_1 is less than $\alpha/(k-1)$. After which, evaluate p_2 against $\alpha/(k-2)$.
- Continue this method sequentially. If, for instance, the second hypothesis is rejected based on its p-value, move to the third hypothesis.
- The process stops when we retain a null hypothesis. We also keep all subsequent without further testing.

2.2.4.3 Holland

Holland test [6] also adjust the value of α in a step-down manner, as Holm'm method does. Begin by ordering the p-values from smallest to largest: $p_1 \leq p_2 \leq ... \leq p_{k-1}$. Correspondingly, assign these to hypotheses $H_1, H_2, ..., H_{k-1}$. In the Holland procedure:

- Initiate with the most significant p-value.
- Reject H_1 if p_1 is less than $1 - (1 - \alpha)^{k-1}$. After which, evaluate p_2 against $1 - (1 - \alpha)^{k-2}$.
- Continue this method sequentially. If, for instance, the second hypothesis is rejected based on its p-value, move to the third hypothesis.
- The process stops when we retain a null hypothesis. We also keep all subsequent without further testing.

2.2.4.4 Finner

Finner test is similar to Holm and also adjusts the value of α in a step-down manner. In the Finner procedure:

- Initiate with the most significant p-value.
- Reject H_1 if p_1 is less than $1 - (1 - \alpha)^{(k-1)/i}$. After which, evaluate p_2 against $\alpha/(k - 2)$.
- Continue this method sequentially. If, for instance, the second hypothesis is rejected based on its p-value, move to the third hypothesis.
- The process stops when we retain a null hypothesis. We also keep all subsequent without further testing.

2.2.4.5 Hochberg

The Hochberg test [6] operates ad a step-up procedure, systematically adjusting the value of α. Hochberg is step-up method works in the opposite direction. In the Hochberg procedure:

- Initiate with the largest $p - value$ comparing with $alpha$
- After which, the next largest with $\alpha/2$.
- The next with $\alpha/3$, and so forth until it encounters a hypothesis it can reject.

- We then reject all hypotheses with smaller p-values as well.

This procedure sometimes detects differences that other methods do not [6].

2.2.4.6 Hommel

The Hommel procedure [6] is acknowledged to be more intricate in terms of both computation and comprehension. Essentially, it requires identifying the greatest value of j such that for every k ranging from 1 to j, the condition $p_{n-j+k} > k\alpha/j$. In the case that such a j does not exist, we can reject all hypotheses, otherwise we reject all for which $p_i \leq \alpha/j$. The Hommel procedure to calculate adjusted p-value is described in Algorithm 1.

Algorithm 1 Algorithm for calculating APVs based on Hommel's procedure.

$APV_i \leftarrow p_i$ para todo i.
for $j = k$ **to** 2 **do**
    $B \leftarrow \emptyset$.
    for each i, $i > (k - 1 - j)$ **do**
        Compute value $c_i = (j \cdot p_i)/(j + i - k + 1)$.
        $B \leftarrow B \cup c_i$.
    end for
    Find the smallest c_i value in B, call it $c_{\min}$.
    if $APV_i < c_{\min}$ **then**
        $APV_i \leftarrow c_{\min}$.
    end if
    for each i, $i \leq (k - 1 - j)$ **do**
        $c_i \leftarrow \min(c_{\min}, j \cdot p_i)$.
        if $APV_i < c_i$ **then**
            $APV_i \leftarrow c_i$.
        end if
    end for
end for

2.2.4.7 Rom

Rom test [6] is a modification to Hochberg test to increase its power. This test changes the way alpha is adjusted.

$$\alpha_{k-i} = \left[\sum_{j=1}^{i-1} \alpha^j - \sum_{j=1}^{i-2} \binom{i}{k} \alpha_{k-1-j}^{i-j} \right] / i \tag{2.49}$$

where $\alpha_{k-1} = \alpha$ and $\alpha_{k-2} = \alpha/2$

2.2.4.8 Li

The Li test [14] proposed a two-step rejection procedure:

- Step 1: Reject all H_i if $p_{k-1} \leq \alpha$. Otherwise, accept the hypothesis associated to p_{k-1} and go to Step 2.
- Step 2: Reject any remaining H_i with $p_i \leq (1 - pk - 1)/(1 - \alpha)\alpha$

2.2.4.9 Shaffer

Shaffer test [19] make use the logical relation among the family of hypotheses for adjusting the value of α. Shaffer proposed two procedures:

- Shaffer's static procedure: similar to Holm's step down method at stage j, reject H_i if $p_i \leq \alpha/t_i$, where t_i is the maximum number of hypotheses which can be true given that any (i, ..., 1) hypotheses are false. In this case, all t_i are fully determined for the given hypotheses H_i, independent of the observed p-values.

$$S(k) = \bigcup_{j=1}^{k} \left(\binom{j}{2} + x : x \in S(k-j) \right), \tag{2.50}$$

where $S(k)$ is the set of possible numbers of true hypotheses with k algorithms being compared, $k \geq 2$, and $S(0) = S(1) = \{0\}$

- Shaffer's dynamic procedure: this increases the power of the first by substituting $\alpha = t_i$ at stage i by the value $\alpha = t_i*$ where t_i* is the maximum number of hypotheses that could be true, given that the previous hypotheses are false. It is a dynamic procedure, since t_i* depends not only on the logical structure of the hypotheses, but also on the hypotheses already rejected at step i. Obviously, this procedure has more power than the first one. However, we will not use this second procedure, given that it is included in an advanced procedure which we will describe in the following.

2.2.4.10 Nemenyi

Nemenyi test [5] is similar to the Tukey test for ANOVA and is used when all classifiers are compared to each other. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least

the critical difference where critical values (see Equation 2.51) q_α are based on the Studentized range statistic divided by 2.

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{2.51}$$

3 Getting and installing StaTDS software

3.1 Getting the library

This section describes the steps to download, install, and configure everything you need to use the library with all available tests and post-hoc included and to start analysing your experiments. StaTDS could be downloaded using two different ways: 1) using pip; 2) git as command line or directly from the webpage at `https://github.com/kdis-lab/StaTDS`.

3.2 Using pip

Ensure that Python and pip are correctly installed on your operating system before proceeding. Once you have completed this step, utilize the following commands for library installation according to your preferred configuration:

- If you only want to use the statistical tests:

```
$ pip install statds
```

- If you also want to generate PDFs:

```
$ pip install statds[pdf]
```

- If you want all the features:

```
$ pip install statds[full-app]
```

3.3 Using Git repository

3.3.1 Command line

The installation process for Git is detailed for each supported operating system in [9]. Additionally, a comprehensive guide on downloading StaTDS is provided. Git can be easily installed on widely used operating systems such as Windows, Mac, and Linux. It is worth noting that Git comes pre-installed on the majority of Mac and Linux machines by default.

GNU-Linux. In this operating system Git packages could be installed using the package manager of your distribution. For Debian based systems, it could be installed using:

```
$ apt update && apt install git-core
```

For Fedora based systems, it could be installed using:

```
$ yum install git-core
```

macOS X. If Xcode has been previously installed, Git may already be present. Nevertheless, Git can also be installed using Homebrew by executing the following command:

```
$ brew install git
```

Windows. Windows users can easily install it using a software called Git for Windows [8], which is available on its website (https://gitforwindows.org/). Access the website (see Figure 3.1) and click the "Download" button. This button automatically detects your operating system and downloads the most suitable version. If the detector cannot determine the best build, it will redirect you to the Github releases page for this software, where you can find all available versions. As of the time of writing this document, the latest available version is 2.42.0 (see Figure 3.2). The binary installer can be found under the "Assets" dropdown (see Figure 3.3). We recommend using *.exe* files since they are easier to install.

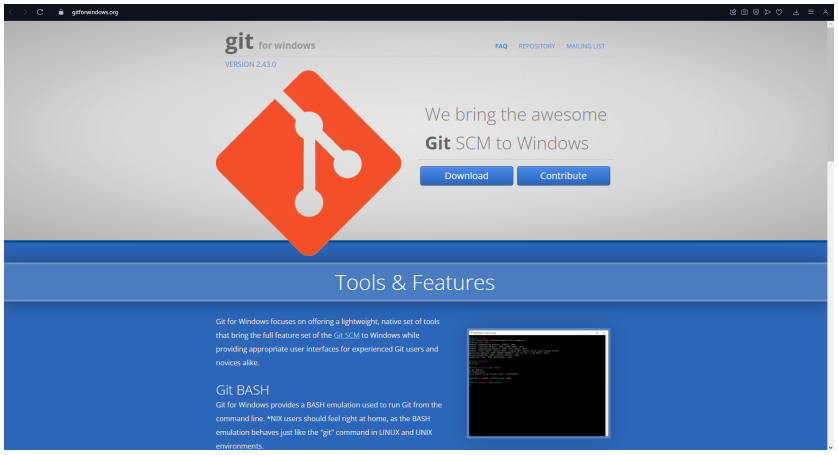


Figure 3.1: Git for Windows website. The download button has to be clicked to obtain the latest version.

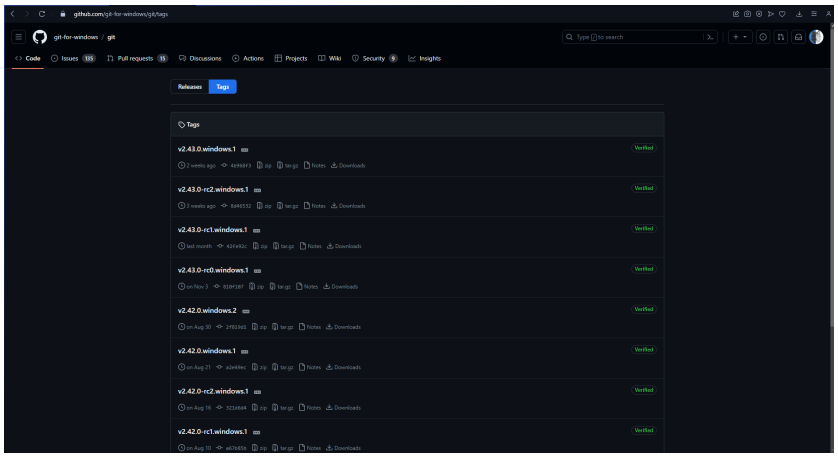


Figure 3.2: Release Git for Windows. All the publicly available versions are listed on this page.

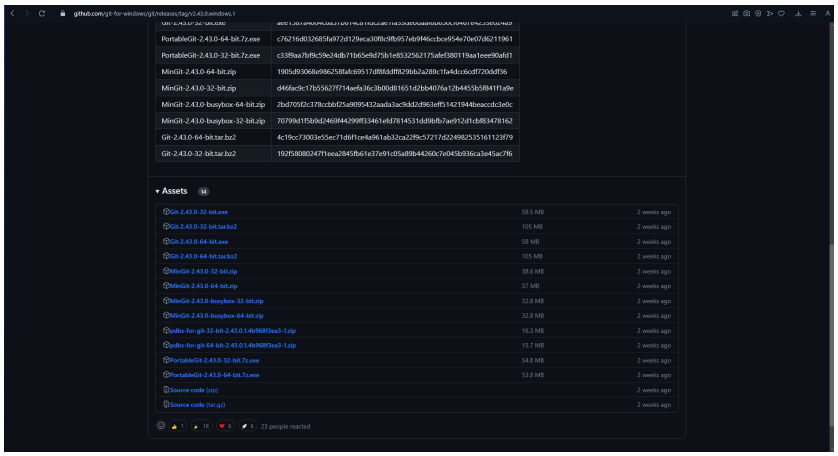


Figure 3.3: List of binary files to install git.

After downloading the installer, it is encouraged to double-click it to start the installation process. The Git Setup wizard screen will appear. Please take a moment to read the license before proceeding, then click the "Next" button. Several steps will be displayed to configure all the options for Git on Windows (see Figure 3.4). Frequently, the default options are sufficient, and no changes are necessary.

Once you have installed Git, you can check if the installation has been successful using the following command. In the case of Windows, you should use the Command Prompt, while for GNU-Linux or macOS, you can validate this in any terminal.

```
$ git --version
```

Finally, StaTDS could be easily downloaded running:

```
$ git clone https//github.com/kdislab/StaTDS
```

The following action will create a new folder with the name StaTDS at the current file location. This directory will contain all the source files and the user manual for the previously specified version. To proceed with the building process, you should access the directory.

3.3.2 Using Github web

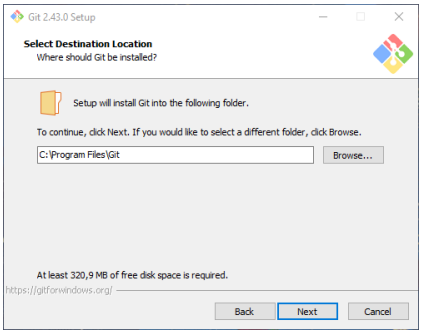
Github provides a direct link to download StaTDS without requiring having git installed. In this way, it is much more easy to download, but it has certain drawbacks. For example, it does not enable to update automatically, so this whole process has to be repeated for each new released version. StaTDS could be downloaded visiting the Github repository publicly available at `https://github.com/kdis-lab/StaTDS`, and clickling on the green button with the text "Code". Then, a small pop-up opens, showing a Download ZIP button (see Figure 3.5). After clickling on that button, a compressed file is downloaded with all the content of the repository. ZIP file only has to be extracted to obtain all the content of the repository.

3.3.3 Building StaTDS Code

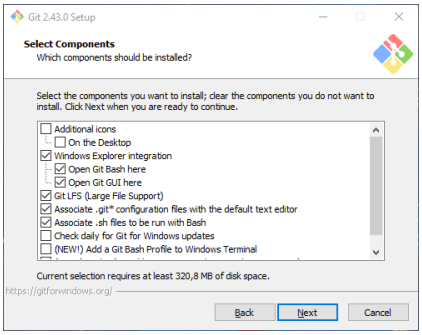
StaTDS has been developed using Python (3.8+) and utilizes the pip package manager to install all its dependencies. Therefore, the first step to "build" StaTDS is to install Python (3.8+).

GNU-Linux. In this operating system, Python could be installed using the package manager of your distribution. For Debian based systems, it

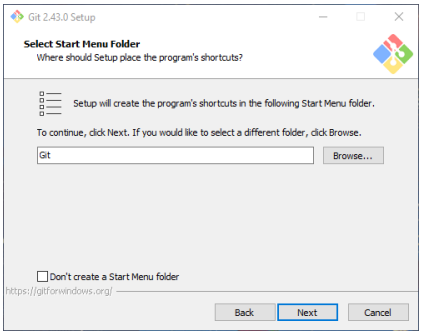
(a) Path of installation.



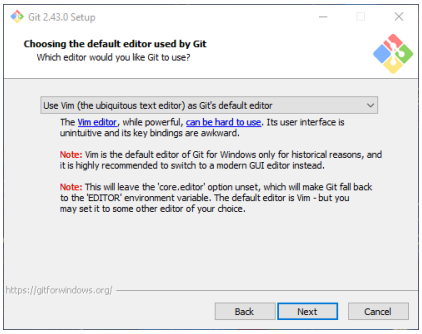
(b) Components of installation.



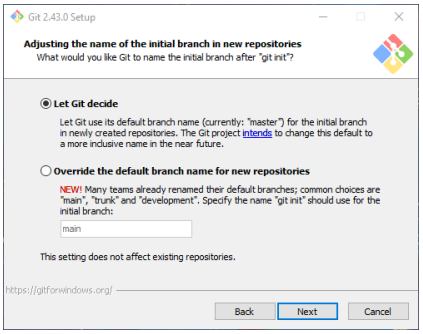
(c) Menu folder.



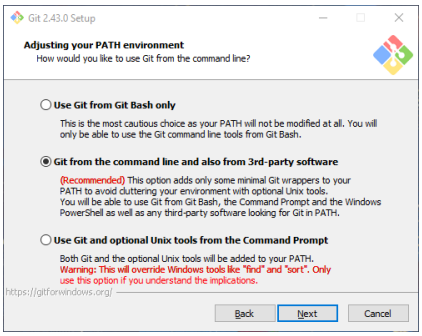
(d) Default editor.



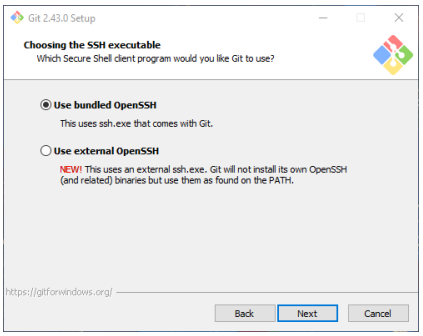
(e) PATH Environment.



(f) HTTPS configuration.



(g) Ending conventions.



(h) Terminal emulator.

Figure 3.4: Different steps of the installation of Git for Windows.

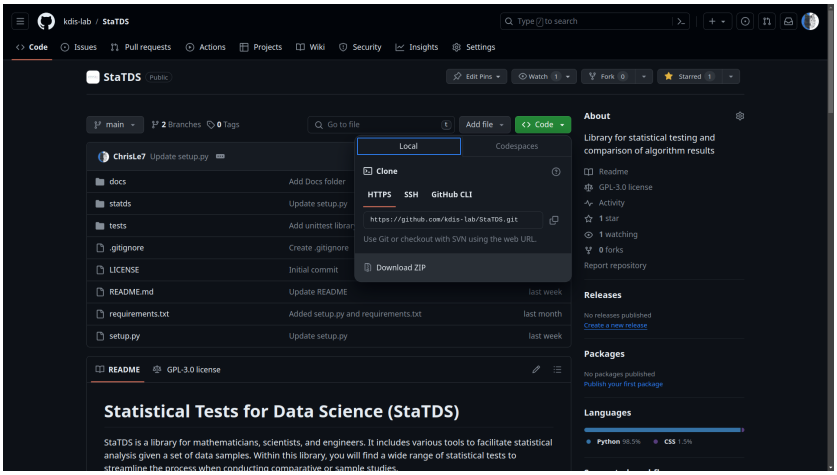


Figure 3.5: Using Github website to download StaTDS, clicking on the "Code" button.

could be installed using:

```
$ apt update && apt install python3.8
```

For Fedora based systems, it could be installed using:

```
$ yum install python3.8
```

macOS X.

```
$ brew install python@3.8
```

Windows. To install Python 3.8 on a Windows system, follow these steps:

1. Go to the official Python website and download the Python 3.8 installer.
2. Once the installer has finished downloading, run the installation file and follow the instructions provided by the installation wizard.
3. In the configuration window, verify that you select the setting "Add Python 3.8 to PATH". This step allows you to use Python from the Windows command line.
4. Proceed with the installation by following the prompts until the process concludes.
5. Python 3.8 is installed and ready for utilization on your Windows system.

Once Python has been successfully installed, we can proceed to install all the dependencies of StaTDS. Use the following command:

```
$ python -m pip install --upgrade pip # To update pip
$ python -m pip install --upgrade build # To update build
$ python -m build
$ pip install dist/statds-1.0-py3-none-any.whl
```

4 StaTDS Library

StaTDS provides a unified and improved interface for comparing algorithms using statistical tests. This chapter aims to highlight the three most important aspects of the library. First, we will present the library's architecture, highlighting its modules and their functions. Secondly, we will introduce examples of application usage, highlighting the possible tests you can perform with StaTDS. Finally, we will discuss report generation by presenting visual examples.

4.1 StaTDS library architecture

The library is a development in open-source Python. Its organization follows a pattern similar to libraries like Scipy or Pandas, built on the idea of being organized by packages. Figure 4.1 shows the architecture of StaTDS; the following will detail each of the modules and their functions:

- stats: Contains functions for calculating p-values and critical values for the various statistical distributions required for statistical tests (both parametric and non-parametric).

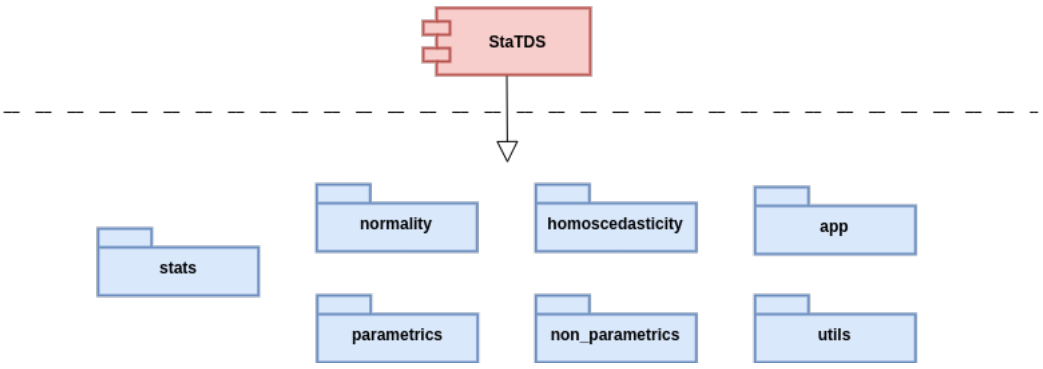


Figure 4.1: Architecture of StaTDS

- normality: Includes all the normality tests and some functions to Q-Q and P-P plots.
- homoscedasticity: Includes all the homoscedasticity tests.
- parametrics: Includes all the parametric statistical tests and functions to determine if data follow a normal distribution and homogeneity.
- no_parametrics: Includes all non-parametric statistical tests and some functions needed to graph results.
- app: Contains all the necessary files for the graphical interface, allowing users to deploy the application if they wish.
- utils: Includes functions to generate analysis and structure.

4.2 Statistical tests

There are two possible ways to use the various statistical tests available from the library. Firstly, you can integrate the library into your scripts and use its functions directly. Alternatively, you can leverage the *utils* module, which enables JSON file analysis for various experiments using the "stats.utils.`analysis_of_experiments`" function. Below, we will present several usage examples and their results. The following will showcase different usage examples along with their corresponding outputs.

4.2.1 Using the library functions

This tutorial covers some basic usage patterns and best practices to help you get started with StaTDS.

```
import statds
import pandas as pd
```

As discussed in Section 2.2.1, you need to check the assumptions of normality and homogeneity before using parametric tests. For this reason, the following will demonstrate how to apply these depending on the chosen normality test.

4.2.1.1 Normality tests

- Shapiro Wilk:

stats.normality.`shapiro_wilk_normality`(*data, alpha*)

Perform the Shapiro-Wilk test for normality. The Shapiro-Wilk test tests the null hypothesis that the data was drawn from a normal distribution.

Parameters

data [numpy.array] Array of sample data.

alpha [float] The significance level for test analysis.

Returns

statistics_w [float] The W statistic for the test, a measure of normality.

p_value [float] The p-value for the hypothesis test. A small p-value (typically ≤ 0.05) rejects the null hypothesis, indicating the data is not normally distributed.

cv_value [float] Critical value for the Shapiro-Wilk test. This is currently set to None.

hypothesis [string] A string stating the conclusion of the test based on the p-value and alpha. It indicates whether the null hypothesis can be rejected or not.

Listing 4.1: Example of Shapiro Wilk Normality

```
from statds.normality import shapiro_wilk_normality
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
results = []

for i in range(1, len(columns)):
    results.append(shapiro_wilk_normality(dataset[columns[i]].to_numpy(), alpha))

statistic_list, p_value_list, cv_value_list, hypothesis_list = zip(*results)

results_test = pd.DataFrame({"Algorithm": columns[1:], "Statistic": statistic_list, "p-value": p_value_list, "Results": hypothesis_list})
print(results_test)
```

- D'Agostino-Pearson:

stats.normality.`d_agostino_pearson`(*data, alpha*)

Perform the D'Agostino and Pearson's omnibus test for normality. This test combines skew and kurtosis to produce an omnibus test of normality, testing the null hypothesis that a sample comes from a normally distributed population.

Parameters

data [numpy.array] Array of sample data. It should be a one-dimensional numpy array.

alpha [float] The significance level for test analysis, default is 0.05.

Returns

statistics_dp [float] The D'Agostino and Pearson's test statistic.

p_value [float] The p-value for the hypothesis test. A small p-value (typically ≤ 0.05) rejects the null hypothesis, indicating the data is not normally distributed.

cv_value [float] Critical value for the test based on the chi-squared distribution with 2 degrees of freedom.

hypothesis [string] A string stating the conclusion of the test based on the p-value and alpha. It indicates whether the null hypothesis can be rejected or not.

Note The test calculates skewness and kurtosis of the data, standardizes these values, and then calculates a test statistic that follows a chi-squared distribution with 2 degrees of freedom under the null hypothesis. The p-value is then derived from this chi-squared distribution.

Listing 4.2: Example of D'Agostino Pearson Normality

```
from statds.normality import d_agostino_pearson
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
results = []

for i in range(1, len(columns)):
    results.append(d_agostino_pearson(dataset[columns[i]].to_numpy(), alpha))

statistic_list, p_value_list, cv_value_list, hypothesis_list = zip(*results)

results_test = pd.DataFrame({"Algorithm": columns[1:], "Statistic": statistic_list, "p-value": p_value_list, "Results": hypothesis_list})
print(results_test)
```

- Kolmogorov-Smirnov:

stats.normality.`kolmogorov_smirnov`(*data, alpha*)

Perform the Kolmogorov-Smirnov test for goodness of fit. This non-parametric test compares a sample with a reference probability distribution (in this case, the normal distribution), assessing whether the sample data follows the same distribution as the reference distribution.

Parameters

data [numpy.array] Array of sample data. It should be a one-dimensional numpy array.

alpha [float] The significance level for test analysis, default is 0.05.

Returns

d_max [float] The maximum difference between the Empirical Cumulative Distribution Function (ECDF) of the data and the Cumulative Distribution Function (CDF) of the reference distribution (normal distribution in this case).

p_value [float] The p-value for the hypothesis test. A small p-value (typically ≤ 0.05) rejects the null hypothesis, indicating the data is not normally distributed.

cv_value [float] Critical value for the Kolmogorov-Smirnov test. This is currently set to None.

hypothesis [string] A string stating the conclusion of the test based on the p-value and alpha. It indicates whether the null hypothesis can be rejected or not.

Note The test calculates the maximum difference (d_max) between the ECDF of the sample data and the CDF of the normal distribution. The KS statistic is then derived from this difference and the sample size. The p-value is estimated from the KS statistic. This test is non-parametric and does not assume a normal distribution of the data.

Listing 4.3: Example of Kolmogorov Smirnov Normality

```
from statds.normality import kolmogorov_smirnov
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
results = []

for i in range(1, len(columns)):
    results.append(kolmogorov_smirnov(dataset[columns[i]].to_numpy(), alpha))

statistic_list, p_value_list, cv_value_list, hypothesis_list = zip(*results)

results_test = pd.DataFrame({"Algorithm": columns[1:], "Statistic": statistic_list, "p-value": p_value_list, "Results": hypothesis_list})
print(results_test)
```

4.2.1.2 Homoscedasticy tests

After reviewing the normality test results, you must conduct another test to see if the data has consistent variances. The Levene test or Bartlett's test are suitable options for this.

- Levene:

stats.homoscedasticity.`levene_test`(*data, alpha, center*)

Perform the Kolmogorov-Smirnov test for goodness of fit. This non-parametric test compares a sample with a reference probability distribution (in this case, the normal distribution), assessing whether the sample data follows the same distribution as the reference distribution.

Parameters

data [numpy.array] A pandas DataFrame where each column represents a different group/sample. The first column is ignored.

alpha [float] The significance level for test analysis, default is 0.05.

center [str] The method for calculating the center of each group - 'mean', 'median', or 'trimmed'. Default is 'mean'.

Returns

statistic_levene [float] The Levene test statistic. A higher value indicates a greater likelihood of differing variances.

p_value [float] The p-value for the hypothesis test. A small p-value (typically ≤ 0.05) rejects the null hypothesis, suggesting that the variances across groups are not equal.

cv_value [float] The critical value for the test at the specified alpha level.

hypothesis [string] A string stating the conclusion of the test based on the test statistic and alpha. It indicates whether the null hypothesis of equal variances can be rejected or not.

Note The Levene test is robust to non-normal distributions, making it preferable to Bartlett's test when data are not normally distributed. The choice of 'center' parameter (mean, median, trimmed) can affect the test's sensitivity to departures from normality. The test statistic is computed based on the absolute deviations from the group centers and then compared against an F-distribution to obtain the p-value.

Listing 4.4: Ejemplo de base de datos

```
from statds.homoscedasticity import levene_test
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
statistic, p_value, rejected_value, hypothesis = levene_test(dataset, alpha, center='mean')
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}")
```

- Bartlett:

stats.homoscedasticity.`bartlett_test`(*data, alpha, center*)

Perform Bartlett's test for homogeneity of variances. This test checks the null hypothesis that all input samples (represented as columns in the dataset) come from populations with equal variances. It is commonly used before conducting ANOVA, as equal variances are an assumption of ANOVA.

Parameters

data [numpy.array] A pandas DataFrame where each column represents a different group/sample. The first column is ignored.

alpha [float] The significance level for test analysis, default is 0.05.

Returns

statistic_bartlett [float] Bartlett's test statistic. A higher value indicates a greater likelihood of differing variances.

p_value [float] The p-value for the hypothesis test. A small p-value (typically ≤ 0.05) rejects the null hypothesis, suggesting that the variances across groups are not equal.

cv_value [float] The critical value for the test at the specified alpha level.

hypothesis [string] A string stating the conclusion of the test based on the test statistic and alpha. It indicates whether the null hypothesis of equal variances can be rejected or not.

Note Bartlett's test is sensitive to departures from normality. Therefore, if the data are not normally distributed, Levene's test is a more appropriate choice. The test statistic is compared against a chi-squared distribution to obtain the p-value. The formula for the test statistic takes into account the number of groups and the total number of samples.

Listing 4.5: Example of Bartlett Homoscedasticity

```
from statds.homoscedasticity import bartlett_test
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
statistic, p_value, rejected_value, hypothesis = bartlett_test(dataset, alpha)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
```

4.2.1.3 Parametrics tests

Once it is known that the data meet the conditions of normality and homogeneity, you can proceed to use parametric tests. To compare two sets of data, use the t-test. To compare several data sets, use the ANOVA test.

- t-test:

stats.parametrics.`t_test_paired`(*data, alpha, verbose*)

Perform a paired t-test. This statistical test is used to compare the means of two related groups of samples, typically before and after a specific treatment or intervention. The test assumes that the differences between pairs are normally distributed.

Parameters

dataset [numpy.array] A pandas DataFrame with exactly two columns, each representing a different group/sample. These groups must be related or paired in some way (e.g., measurements before and after an intervention).

alpha [float] The significance level for the test, default is 0.05.

verbose [boolean] A boolean (True or False). If True, prints the test statistic, rejected value, p-value, and hypothesis.

Returns

statistic_t [float] The t-test statistic. A higher absolute value indicates a greater difference between the paired groups.

rejected_value [float] The critical value for the test at the specified alpha level.

p_value [float] The p-value for the hypothesis test (currently not calculated and set to None).

hypothesis [string] A string stating the conclusion of the test based on the test. It indicates whether the null hypothesis of equal variances can be rejected or not.

Note The paired t-test is appropriate for comparing two means from the same group or individual under two different conditions. The test statistic is calculated by dividing the mean difference between paired observations by the standard error of the mean difference. The test is sensitive to the normality assumption of the differences between pairs.

Listing 4.6: Example of T Test

```
from statds.parametrics import t_test_paired
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
selected_columns = [columns[1], columns[2]]
statistic, rejected_value, p_value, hypothesis = t_test_paired(dataset[selected_columns], alpha )
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p-value {p_value}")
```

- ANOVA:

stats.parametrics.`anova_cases`(*data, alpha*)

Perform an ANOVA (Analysis of Variance) test. This statistical test is used to compare the means of two or more groups to determine if at least one group mean is significantly different from the others. It's commonly used when there are three or more groups.

Parameters

dataset [numpy.array] A pandas DataFrame where each column represents a different group/sample. The first column is ignored.

alpha [float] The significance level for the test, default is 0.05.

Returns

summary_results [pandas.DataFrame] A pandas DataFrame with a summary of each group's mean, standard deviation, and standard error.

anova_results [pandas.DataFrame] A pandas DataFrame with the ANOVA results, including degrees of freedom, sum of squares, mean square, F-statistic, and rejected value for between groups and within groups.

statistical_f_anova [float] The F-statistic for the ANOVA test. A higher value suggests a greater difference between group means.

p_value [float] The p-value for the hypothesis test.

rejected_value [float] The critical value for the test at the specified alpha level.

hypothesis [string] A string stating the conclusion of the test based on the F-statistic and alpha.

Note ANOVA tests the null hypothesis that all group means are equal. The test assumes that the groups are sampled from populations with normal distributions and equal variances. The F-statistic is calculated based on the ratio of variance between the groups to the variance within the groups. A significant F-statistic (p-value less than alpha) indicates that at least one group mean is significantly different.

Listing 4.7: Example of ANOVA Test

```
from statds.parametrics import anova_test
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
statistic, p_value, rejected_value, hypothesis = anova_test(dataset, alpha)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
```

4.2.1.4 Non-parametrics tests

As discussed in Section 2.2.3, non-parametric tests do not assume a specific distribution of the data, being particularly useful when the assumptions of parametric tests, such as normality and homoscedasticity, cannot be met. The following are some ways of applying these tests depending on whether we compare two or multiple groups:

- Two Groups:
  - Wilcoxon:

stats.no_parametrics.`wilconxon`(*data, alpha, verbose*)

Perform the Wilcoxon signed-rank test. This non-parametric test is used to compare two related samples, matched samples, or repeated measurements on a single sample to assess whether their population mean ranks differ. It is an alternative to the paired Student's t-test when the data is not normally distributed.

Parameters

dataset [pandas.DataFrame] A pandas DataFrame with exactly two columns, each representing a different condition or time point for the same subjects.

alpha [float] The significance level for the test, default is 0.05.

verbose [boolean] A boolean (True or False). If True, prints the detailed results table.

Returns

w_wilcoxon [float] The Wilcoxon test statistic, which is the smallest of the sums of the positive and negative ranks.

cv_alpha_selected [float] The critical value for the test at the specified alpha level (only for small sample sizes, otherwise None).

p_value [float] The p-value for the hypothesis test.

hypothesis [string] A string stating the conclusion of the test based on the test statistic, critical value, or p-value and alpha.

Note The Wilcoxon signed-rank test makes fewer assumptions than the t-test and is appropriate when the data are not normally distributed. It ranks the absolute differences between pairs, then compares these ranks. The test is sensitive to ties and has different procedures for small and large sample sizes. For large samples, the test statistic is approximately normally distributed, allowing the use of normal approximation for p-value calculation.

Listing 4.8: Example of Wilcoxon Test

```
from statds.no_parametrics import wilconxon
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
selected_columns = [columns[1], columns[2]]
statistic, p_value, rejected_value, hypothesis = wilconxon(dataset[selected_columns], alpha)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
```

– Binomial Sign:

stats.no_parametrics.`binomial`(*data, alpha, verbose*)

Perform a binomial sign test. This non-parametric test is used to determine if there is a significant difference between the medians

of two dependent samples. It's an alternative to the paired t-test and Wilcoxon signed-rank test, particularly useful when the data does not meet the assumptions of these tests or when the data is on an ordinal scale.

Parameters

dataset [pandas.DataFrame] A pandas DataFrame with exactly two columns, each representing a different condition or time point for the same subjects.

alpha [float] The significance level for the test, default is 0.05.

verbose [boolean] A boolean (True or False). If True, prints the detailed results table.

Returns

statistical_binomial [float] The binomial test statistic, which is the largest of the counts of positive or negative differences.

cv_alpha_selected [float] The critical value for the test at the specified alpha level (not calculated in this function, hence None).

p_value [float] The p-value for the hypothesis test, calculated from the binomial distribution.

hypothesis [string] A string stating the conclusion of the test based on the test statistic and p-value in comparison to alpha.

Note The binomial sign test counts the number of positive and negative differences between paired observations and then tests if the observed proportion of positive (or negative) differences is significantly different from 0.5 (no difference). The test assumes that the distribution of differences is symmetric around the median and ignores pairs with no difference.

Listing 4.9: Example of Binomial Sign Test

```
from statds.no_parametrics import binomial
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
selected_columns = [columns[1], columns[2]]
statistic, p_value, rejected_value, hypothesis = binomial(dataset[selected_columns], alpha)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
```

– Mann-Whitney U:

stats.no_parametrics.`mannwhitneyu`(*data, alpha, criterion, verbose*)

Perform the Mann-Whitney U test, also known as the Wilcoxon rank-sum test. This non-parametric test is used to determine whether there is a significant difference between the distributions of two independent samples. It's an alternative to the independent t-test when the data does not meet the assumptions of the t-test.

Parameters

dataset [pandas.DataFrame] A pandas DataFrame with exactly two columns, each representing a different independent sample.

alpha [float] The significance level for the test, default is 0.05.

criterion [boolean] A boolean that determines the direction for ranking the observations. If False, ranks are in ascending order; if True, in descending order.

verbose [boolean] A boolean (True or False). If True, prints the detailed results table.

Returns

z_value [float] The z-value computed from the U statistic.

cv_alpha_selected [float] The critical value for the test at the specified alpha level (not calculated in this function, hence None).

p_value [float] The p-value for the hypothesis test, calculated from the normal approximation of the U distribution.

hypothesis [string] A string stating the conclusion of the test based on the test statistic and p-value in comparison to alpha.

Note The Mann-Whitney U test ranks all the observations from both groups together and then compares the sum of ranks in each group. It is appropriate for ordinal data and is robust against non-normal distributions. The test assumes that the two groups are independent and that the observations are ordinal or continuous. The normal approximation for the p-value calculation is valid for large sample sizes.

Listing 4.10: Example of Mann-Whitney Test

```
from statds.no_parametrics import mannwhitneyu
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
selected_columns = [columns[1], columns[2]]
statistic, p_value, rejected_value, hypothesis = mannwhitneyu(dataset[selected_columns], ↩
    ↪ alpha)
print(hypothesis)
print(f"Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
```

- Multiple Groups:
  - Friedman:

stats.no_parametrics.`friedman`(*data, alpha, criterion, verbose*)

Perform the Friedman test, a non-parametric statistical test similar to the parametric ANOVA, but for repeated measures. The Friedman test is used to detect differences in treatments across multiple test attempts. It ranks the treatments for each block (or subject), then considers these ranks.

Parameters

dataset [pandas.DataFrame] A pandas DataFrame with the first column as the block or subject identifier and the remaining columns as different treatments or conditions.

alpha [float] The significance level for the test, default is 0.05.

criterion [boolean] A boolean that determines the direction for ranking the observations. If False, ranks are in ascending order; if True, in descending order.

verbose [boolean] A boolean (True or False). If True, prints the detailed results table.

Returns

rankings_with_label [dictionary] A dictionary with the average ranks of each treatment or condition.

stadistic_friedman [float] The Friedman test statistic, which is chi-squared distributed under the null hypothesis.

reject_value [tuple] A tuple containing the critical value for the test and the p-value (if applicable).

hypothesis [string] A string stating the conclusion of the test based on the test statistic and p-value in comparison to alpha.

Note The Friedman test is appropriate when data is ordinal and the assumptions of parametric tests (like ANOVA) are not met. It considers the ranks of the treatments within each block, summing these ranks, and then analyzing the sums' distribution. The test is robust against non-normal distributions and is ideal for small sample sizes.

Listing 4.11: Example of Friedman Test

```
from statds.no_parametrics import friedman
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = friedman(dataset, alpha, criterion=↩
    ↪ False)
print(hypothesis)
print(f'Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
print(rankings)
```

– Friedman Aligned Ranks:

stats.no_parametrics.`friedman_aligned_ranks`(*data, alpha, criterion, verbose*)

Perform the Friedman Aligned Ranks test, an extension of the Friedman test. This test is used when dealing with multiple treatments or conditions over different subjects or blocks, especially in cases where the assumptions of the classical Friedman test may not hold. The test aligns the data by subtracting the mean across treatments for each subject before ranking.

Parameters

dataset [pandas.DataFrame] A pandas DataFrame with the first column as the block or subject identifier and the remaining columns as different treatments or conditions.

alpha [float] The significance level for the test, default is 0.05.

criterion [boolean] A boolean that determines the direction for ranking the observations. If False, ranks are in ascending order; if True, in descending order.

verbose [boolean] A boolean (True or False). If True, prints the detailed results table including aligned ranks.

Returns

rankings_with_label [dictionary] A dictionary with the average ranks of each treatment or condition.

stadistic_friedman [float] The Friedman Aligned Ranks test statistic.

reject_value [tuple] A tuple containing the critical value for the test and the p-value (if applicable).

hypothesis [string] A string stating the conclusion of the test based on the test statistic, critical value, and alpha.

Note The Friedman Aligned Ranks test modifies the standard Friedman test by aligning the data for each subject before ranking. This alignment is achieved by subtracting the average rank across treatments for each subject, making the test more robust to certain types of data irregularities, like outliers. It is appropriate for ordinal data and assumes that the groups are independent and identically distributed within each block.

Listing 4.12: Example of Friedman Aligned Ranks Test

```
from statds.no_parametrics import friedman_aligned_ranks
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = friedman_aligned_ranks(dataset, alpha, criterion=False)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
print(rankings)
```

– Quade:

stats.no_parametrics.`quade`(*data, alpha, criterion, verbose*)

Perform the Quade test, a non-parametric statistical test used to identify significant differences between three or more matched groups. This test is particularly useful for blocked designs where treatments are applied to matched groups or blocks. The Quade test considers the relative differences between treatments within each block and ranks these differences.

Parameters

dataset [pandas.DataFrame] A pandas DataFrame with the first column as the block or subject identifier and the remaining columns as different treatments or conditions.

alpha [float] The significance level for the test, default is 0.05.

criterion [boolean] A boolean that determines the direction for ranking the observations. If False, ranks are in ascending order; if True, in descending order.

verbose [boolean] A boolean (True or False). If True, prints the detailed results table including ranks.

Returns

rankings_with_label [dictionary] A dictionary with the average ranks of each treatment or condition.

stadistic_quade [float] The Quade test statistic.

reject_value [tuple] A tuple containing the critical value for the test and the p-value (if applicable).

hypothesis [string] A string stating the conclusion of the test based on the test statistic, critical value, and alpha.

Note The Quade test adjusts for differences in treatment effects within each block by incorporating the range of each block into the ranking process. This makes it more sensitive to treatment effects in the presence of block-to-block variability. It's appropriate for ordinal data and assumes that the treatments are independent and identically distributed within each block.

Listing 4.13: Example of Quade Test

```
from statds.no_parametrics import quade
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = quade(dataset, alpha, criterion=↩
    ↪ False)
print(hypothesis)
print(f'Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
print(rankings)
```

4.2.1.5 Post hoc

Before using post hoc tests, you must apply a test to identify significant differences.

- Bonferroni:

stats.no_parametrics.`bonferroni`(*ranks, num_cases, alpha, verbose*)

This function performs the Bonferroni correction for multiple comparisons of algorithms or treatments. The Bonferroni correction is a method to adjust significance levels when multiple statistical tests are conducted simultaneously.

Parameters

dataset [dictionary] A dictionary where keys are the names of the algorithms and values are their respective ranks. The ranks must be obtained based on multiple non-parametrics test.

num_cases [int] An integer representing the number of cases, datasets, or instances over which the rankings were calculated.

alpha [float] A float representing the significance level used in the test. It defaults to 0.05, which is a common choice in statistical testing.

control [str] An optional string specifying a control algorithm against which others will be compared. If None, all algorithms are compared against each other.

type_rank [str] A string indicating the type of ranking used (e.g., "Friedman"). Defaults to "Friedman".

verbose [boolean] A boolean indicating whether to print additional information (like the critical distance). Defaults to False.

Returns

ranks_values [pandas.DataFrame] A DataFrame with the results of the comparisons, including adjusted z-values and adjusted p-values.

figure [matplotlib.plot.figure] A figure representing the ranking of the algorithms and the critical distances visually, often as a CD diagram.

Note This function is useful in statistical analysis where multiple algorithms or treatments are compared and there is a need to control the Type I error (false positive) that increases with the number of comparisons.

Listing 4.14: Example of Bonferroni-Dunn Test

```
from statds.no_parametrics import friedman, bonferroni
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = friedman(dataset, alpha, criterion=↩
    ↪ False)
```

```
6    print(hypothesis)
7    print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
8    print(rankings)
9    num_cases = data.shape[0]
10   results, figure = bonferroni(rankings, num_cases, alpha, control = None, type_rank = "↩
         ↪ Friedman")
11   print(results)
12   figure.show()
```

- Holm:

stats.no_parametrics.`holm`(*ranks, num_cases, alpha, verbose*)

This function implements the Holm-Bonferroni method, an adjustment for multiple comparisons. It is used to control the family-wise error rate when comparing multiple algorithms or treatments. This method is more powerful than the simple Bonferroni correction, especially when the number of comparisons is large.

Parameters

dataset [dictionary] A dictionary where keys are the names of the algorithms and values are their respective ranks. The ranks must be obtained based on multiple non-parametrics test.

num_cases [int] An integer representing the number of cases, datasets, or instances over which the rankings were calculated.

alpha [float] A float representing the significance level used in the test. It defaults to 0.05, which is a common choice in statistical testing.

control [str] An optional string specifying a control algorithm against which others will be compared. If None, all algorithms are compared against each other.

type_rank [str] A string indicating the type of ranking used (e.g., "Friedman"). Defaults to "Friedman".

verbose [boolean] A boolean indicating whether to print additional information (like the critical distance). Defaults to False.

Returns

ranks_values [pandas.DataFrame] A DataFrame with the results of the comparisons, including adjusted z-values and adjusted p-values.

figure [matplotlib.plot.figure] A figure representing the ranking of the algorithms and the critical distances visually, often as a CD diagram.

Note This function is useful in statistical analysis where multiple algorithms or treatments are compared and there is a need to control the Type I error (false positive) that increases with the number of comparisons.

Listing 4.15: Example of Holm-Dunn Test

```
from statds.no_parametrics import friedman, holm
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = friedman(dataset, alpha, criterion=↩
    ↪ False)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
print(rankings)
num_cases = data.shape[0]
results, figure = holm(rankings, num_cases, alpha, control = None, type_rank = "Friedman"↩
    ↪ )
print(results)
figure.show()
```

- Holland:

stats.no_parametrics.`holland`(*ranks, num_cases, alpha, verbose*)

This function applies the Holland step-down procedure for controlling the family-wise error rate in multiple comparisons. It's an improvement over the simple Bonferroni method and is particularly useful when dealing with a large number of comparisons.

Parameters

dataset [dictionary] A dictionary where keys are the names of the algorithms and values are their respective ranks. The ranks must be obtained based on multiple non-parametrics test.

num_cases [int] An integer representing the number of cases, datasets, or instances over which the rankings were calculated.

alpha [float] A float representing the significance level used in the test. It defaults to 0.05, which is a common choice in statistical testing.

control [str] An optional string specifying a control algorithm against which others will be compared. If None, all algorithms are compared against each other.

type_rank [str] A string indicating the type of ranking used (e.g., "Friedman"). Defaults to "Friedman".

verbose [boolean] A boolean indicating whether to print additional information (like the critical distance). Defaults to False.

Returns

ranks_values [pandas.DataFrame] A DataFrame with the results of the comparisons, including adjusted z-values and adjusted p-values.

figure [matplotlib.plot.figure] A figure representing the ranking of the algorithms and the critical distances visually, often as a CD diagram.

Note The Holland method adjusts the alpha values for each comparison based on their rank in p-value, offering a more nuanced control over Type I errors compared to the Bonferroni method. It's particularly effective in scenarios with multiple algorithm comparisons, ensuring a more accurate interpretation of statistical results.

Listing 4.16: Example of Holland Test

```
from statds.no_parametrics import friedman, holland
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = friedman(dataset, alpha, criterion=↩
    ↪ False)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
print(rankings)
num_cases = data.shape[0]
results, figure = holland(rankings, num_cases, alpha, control = None, type_rank = "↩
    ↪ Friedman")
print(results)
figure.show()
```

- Finner:

stats.no_parametrics.`finner`(*ranks, num_cases, alpha, verbose*)

This function implements the Finner correction, a statistical method for controlling the family-wise error rate in multiple comparisons. The Finner correction is an alternative to other methods like Bonferroni or Holm, and it is generally more powerful than the Bonferroni correction.

Parameters

dataset [dictionary] A dictionary where keys are the names of the algorithms and values are their respective ranks. The ranks must be obtained based on multiple non-parametrics test.

num_cases [int] An integer representing the number of cases, datasets, or instances over which the rankings were calculated.

alpha [float] A float representing the significance level used in the test. It defaults to 0.05, which is a common choice in statistical testing.

control [str] An optional string specifying a control algorithm against which others will be compared. If None, all algorithms are compared against each other.

type_rank [str] A string indicating the type of ranking used (e.g., "Friedman"). Defaults to "Friedman".

verbose [boolean] A boolean indicating whether to print additional information (like the critical distance). Defaults to False.

Returns

ranks_values [pandas.DataFrame] A DataFrame with the results of the comparisons, including adjusted z-values and adjusted p-values.

figure [matplotlib.plot.figure] A figure representing the ranking of the algorithms and the critical distances visually, often as a CD diagram.

Note The Finner method is particularly useful in scenarios involving multiple comparisons, where it provides a balance between statistical power and control over Type I errors. This makes it a valuable tool in comparative studies of algorithms or treatments across various datasets.

Listing 4.17: Example of Finner Test

```
from statds.no_parametrics import friedman, finner
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = friedman(dataset, alpha, criterion=↩
    ↪ False)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
print(rankings)
num_cases = data.shape[0]
results, figure = finner(rankings, num_cases, alpha, control = None, type_rank = "Friedman↩
    ↪ ")
print(results)
figure.show()
```

- Hochberg:

stats.no_parametrics.`hochberg`(*ranks, num_cases, alpha, verbose*)

This function implements the Hochberg step-up procedure for multiple comparisons correction. It is an alternative to the Bonferroni method and is generally more powerful, especially when there are a large number of comparisons. The Hochberg procedure controls the family-wise error rate more effectively than some other methods.

Parameters

dataset [dictionary] A dictionary where keys are the names of the algorithms and values are their respective ranks. The ranks must be obtained based on multiple non-parametrics test.

num_cases [int] An integer representing the number of cases, datasets, or instances over which the rankings were calculated.

alpha [float] A float representing the significance level used in the test. It defaults to 0.05, which is a common choice in statistical testing.

control [str] An optional string specifying a control algorithm against which others will be compared. If None, all algorithms are compared against each other.

type_rank [str] A string indicating the type of ranking used (e.g., "Friedman"). Defaults to "Friedman".

verbose [boolean] A boolean indicating whether to print additional information (like the critical distance). Defaults to False.

Returns

ranks_values [pandas.DataFrame] A DataFrame with the results of the comparisons, including adjusted z-values and adjusted p-values.

figure [matplotlib.plot.figure] A figure representing the ranking of the algorithms and the critical distances visually, often as a CD diagram.

Note The Hochberg procedure is particularly valuable when conducting multiple comparisons in statistical analysis, as it provides a more refined approach to controlling the Type I error rate compared to more conservative methods.

Listing 4.18: Example of Hochberg Test

```
1 from statds.no_parametrics import friedman, hochberg
2 data = pd.read_csv("dataset.csv")
3 alpha = 0.05
```

```
    columns = list(dataset.columns)
    rankings, statistic, p_value, critical_value, hypothesis = friedman(dataset, alpha, criterion=↩
        ↪ False)
    print(hypothesis)
    print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
    print(rankings)
    num_cases = data.shape[0]
    results, figure = hochberg(rankings, num_cases, alpha, control = None, type_rank = "↩
        ↪ Friedman")
    print(results)
    figure.show()
```

- Hommel:

stats.no_parametrics.**hommel**(*ranks, num_cases, alpha, verbose*)

This function implements the Hommel correction, a statistical method for adjusting p-values when performing multiple comparisons. The Hommel correction is a more powerful alternative to the Bonferroni correction, particularly when the number of comparisons is large.

Parameters

dataset [dictionary] A dictionary where keys are the names of the algorithms and values are their respective ranks. The ranks must be obtained based on multiple non-parametrics test.

num_cases [int] An integer representing the number of cases, datasets, or instances over which the rankings were calculated.

alpha [float] A float representing the significance level used in the test. It defaults to 0.05, which is a common choice in statistical testing.

control [str] An optional string specifying a control algorithm against which others will be compared. If None, all algorithms are compared against each other.

type_rank [str] A string indicating the type of ranking used (e.g., "Friedman"). Defaults to "Friedman".

verbose [boolean] A boolean indicating whether to print additional information (like the critical distance). Defaults to False.

Returns

ranks_values [pandas.DataFrame] A DataFrame with the results of the comparisons, including adjusted z-values and adjusted p-values.

figure [matplotlib.plot.figure] A figure representing the ranking of the algorithms and the critical distances visually, often as a CD diagram.

Note The Hommel correction is particularly useful in scenarios with multiple comparisons, as it provides a balance between controlling the family-wise error rate and maintaining statistical power. This makes it an effective tool in the comparative analysis of algorithms or treatments across various datasets.

Listing 4.19: Example of Hommel Test

```
from statds.no_parametrics import friedman, hommel
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = friedman(dataset, alpha, criterion=↩
    ↪ False)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
print(rankings)
num_cases = data.shape[0]
results, figure = hommel(rankings, num_cases, alpha, control = None, type_rank = "↩
    ↪ Friedman")
print(results)
figure.show()
```

- Rom:

stats.no_parametrics.`rom`(*ranks, num_cases, alpha, verbose*)

This function implements the Rom method, which is a step-down procedure for adjusting p-values in the context of multiple comparisons. The Rom method is an improvement over the Bonferroni correction, providing a more powerful approach, especially when the number of comparisons is large.

Parameters

dataset [dictionary] A dictionary where keys are the names of the algorithms and values are their respective ranks. The ranks must be obtained based on multiple non-parametrics test.

num_cases [int] An integer representing the number of cases, datasets, or instances over which the rankings were calculated.

alpha [float] A float representing the significance level used in the test. It defaults to 0.05, which is a common choice in statistical testing.

control [str] An optional string specifying a control algorithm against which others will be compared. If None, all algorithms are compared against each other.

type_rank [str] A string indicating the type of ranking used (e.g., "Friedman"). Defaults to "Friedman".

verbose [boolean] A boolean indicating whether to print additional information (like the critical distance). Defaults to False.

Returns

ranks_values [pandas.DataFrame] A DataFrame with the results of the comparisons, including adjusted z-values and adjusted p-values.

figure [matplotlib.plot.figure] A figure representing the ranking of the algorithms and the critical distances visually, often as a CD diagram.

Note The Rom method is particularly useful in scenarios involving multiple comparisons, as it offers a more accurate control of the family-wise error rate compared to simpler methods like Bonferroni, especially in cases with a large number of algorithms or treatments being compared.

Listing 4.20: Example of Rom Test

```
from statds.no_parametrics import friedman, rom
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = friedman(dataset, alpha, criterion=↩
    ↪ False)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
print(rankings)
num_cases = data.shape[0]
results, figure = rom(rankings, num_cases, alpha, control = None, type_rank = "Friedman")
print(results)
figure.show()
```

- Li:

stats.no_parametrics.`li`(*ranks, num_cases, alpha, verbose*)

This function implements the Li method for adjusting p-values in multiple comparisons. The Li method is particularly useful when there is a control algorithm to compare against other algorithms. It adjusts p-values based on the performance of the control algorithm relative to others.

Parameters

dataset [dictionary] A dictionary where keys are the names of the algorithms and values are their respective ranks. The ranks must be obtained based on multiple non-parametrics test.

num_cases [int] An integer representing the number of cases, datasets, or instances over which the rankings were calculated.

alpha [float] A float representing the significance level used in the test. It defaults to 0.05, which is a common choice in statistical testing.

control [str] An optional string specifying a control algorithm against which others will be compared. If None, all algorithms are compared against each other.

type_rank [str] A string indicating the type of ranking used (e.g., "Friedman"). Defaults to "Friedman".

verbose [boolean] A boolean indicating whether to print additional information (like the critical distance). Defaults to False.

Returns

ranks_values [pandas.DataFrame] A DataFrame with the results of the comparisons, including adjusted z-values and adjusted p-values.

figure [matplotlib.plot.figure] A figure representing the ranking of the algorithms and the critical distances visually, often as a CD diagram.

Note The Li method is effective in scenarios where a specific algorithm (the control) is of particular interest, and the comparisons are made between this control and other algorithms. It provides a nuanced way to adjust for multiple comparisons by considering the performance of the control algorithm.

Listing 4.21: Example of Li Test

```
from statds.no_parametrics import friedman, li
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = friedman(dataset, alpha, criterion=↩
    ↪ False)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
print(rankings)
num_cases = data.shape[0]
results, figure = li(rankings, num_cases, alpha, control = None, type_rank = "Friedman")
print(results)
figure.show()
```

- Shaffer:

stats.no_parametrics.`shaffer`(*ranks, num_cases, alpha, verbose*)

This function applies the Shaffer's multiple comparison procedure, which is a more refined method for adjusting p-values in the context of multiple comparisons. Shaffer's method is an extension of the Bonferroni correction and is generally more powerful, particularly when the number of comparisons is large.

Parameters

dataset [dictionary] A dictionary where keys are the names of the algorithms and values are their respective ranks. The ranks must be obtained based on multiple non-parametrics test.

num_cases [int] An integer representing the number of cases, datasets, or instances over which the rankings were calculated.

alpha [float] A float representing the significance level used in the test. It defaults to 0.05, which is a common choice in statistical testing.

control [str] An optional string specifying a control algorithm against which others will be compared. If None, all algorithms are compared against each other.

type_rank [str] A string indicating the type of ranking used (e.g., "Friedman"). Defaults to "Friedman".

verbose [boolean] A boolean indicating whether to print additional information (like the critical distance). Defaults to False.

Returns

ranks_values [pandas.DataFrame] A DataFrame with the results of the comparisons, including adjusted z-values and adjusted p-values.

figure [matplotlib.plot.figure] A figure representing the ranking of the algorithms and the critical distances visually, often as a CD diagram.

Note Shaffer's method is particularly effective in scenarios with multiple comparisons, as it provides a more accurate control of the family-wise error rate compared to simpler methods like Bonferroni, especially when the number of algorithms or treatments being compared is large.

Listing 4.22: Example of Shaffer Test

```
from statds.no_parametrics import friedman, shaffer
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = friedman(dataset, alpha, criterion=↩
    ↪ False)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
print(rankings)
num_cases = data.shape[0]
results, figure = shaffer(rankings, num_cases, alpha, type_rank = "Friedman")
print(results)
figure.show()
```

- Nemenyi:

stats.no_parametrics.`nemenyi`(*ranks, num_cases, alpha, verbose*)

The Nemenyi test is a post-hoc analysis method used in statistics to compare multiple algorithms or treatments. It is often used after a Friedman test has indicated significant differences across algorithms.

Parameters

ranks [dictionary] A dictionary where keys are the names of the algorithms and values are their respective ranks. The ranks must be obtained based on multiple non-parametrics test.

num_cases [int] An integer representing the number of cases, datasets, or instances over which the rankings were calculated.

alpha [float] A float representing the significance level used in the test. It defaults to 0.05, which is a common choice in statistical testing.

verbose [boolean] A boolean indicating whether to print additional information (like the critical distance). Defaults to False.

Returns

ranks_values [dictionary] The list of rank values for each algorithm.

critical_distance_nemenyi [float] The critical distance for the Nemenyi test, which is a threshold used to determine if differences between algorithm rankings are statistically significant.

figure [matplotlib.plot.figure] A figure representing the ranking of the algorithms and the critical distances visually, often as a CD diagram.

Note The Nemenyi test is post-hoc non-parametric and is used when the assumptions of parametric tests (like ANOVA) are not met. It's particularly useful in scenarios where multiple algorithms are compared across various datasets.

Listing 4.23: Example of Nemenyi Test

```
from statds.no_parametrics import friedman, nemenyi
data = pd.read_csv("dataset.csv")
alpha = 0.05
columns = list(dataset.columns)
rankings, statistic, p_value, critical_value, hypothesis = friedman(dataset, alpha, criterion=↩
    ↪ False)
print(hypothesis)
print("Statistic {statistic}, Rejected Value {rejected_value}, p−value {p_value}")
print(rankings)
num_cases = data.shape[0]
ranks_values, critical_distance_nemenyi, figure = nemenyi(rankings, num_cases, alpha)
print(ranks_values)
print(critical_distance_nemenyi)
figure.show()
```

4.2.2 Using a JSON file

Another way to use this library is through the *utils* module, which allows for the execution of several statistical tests collectively.

- The *generate_json* function, using two Python lists, facilitates the construction of the data structure required by the *utils.analysis_of_experiments* function.
- The *analysis_of_experiments* function takes three arguments and displays results on the system's standard output or in a PDF report.
  - dataset: A data set for conducting experiments.
  - experiments: A dictionary of experiments and their parameters.
  - generate_pdf: Indicates whether the results of the various experiments should be displayed on the console or generate a PDF report.

Listing 4.24: Example of Analysis of experiments

```
from statds.utils import generate_json, analysis_of_experiments
import pandas as pd
df = pd.read_csv("test_all_vs_all.csv")
# analysis_of_experiments(df, analysis_form, generate_pdf=True)
columns = list(df.columns)

aux = generate_json(["A", "B", "C", "D"],
```

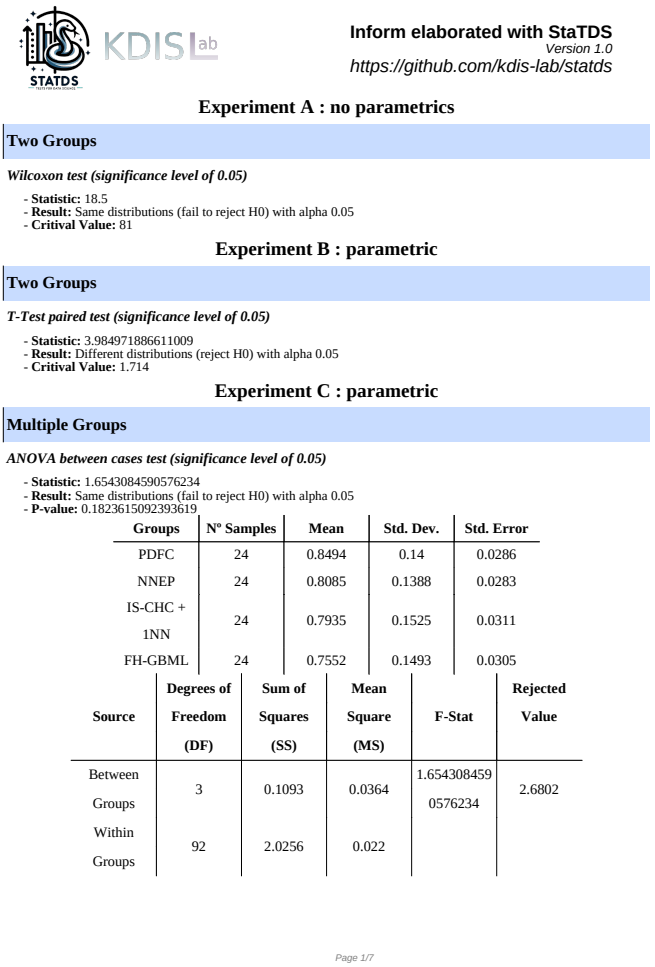
```
                                [{"alpha": 0.05, "test": "Wilcoxon", "first_group": columns[1],
                                  "second_group": columns[-1]},
                                 {"alpha": "0.05", "test": "T-Test paired",
                                  "first_group": columns[1], "second_group": columns[-1]},
                                 {"alpha": 0.05, "test": "ANOVA between cases", "criterion": True},
                                 {"alpha": [0.05, 0.01, 0.1], "test": "Friedman", "criterion": True, "↩
                                     ↪ post_hoc": "Bonferroni"}])
    analysis_of_experiments(df, aux, generate_pdf=False)
    analysis_of_experiments(df, aux, generate_pdf=True, name_pdf="example_inform.pdf")
```

Figure 4.2 shows a sample of the console interface when using the *analysis_of_experiments* function without the pdf generation. All the results are summarized in the console as it is shown.

```
Experiment A : no parametrics
Two Groups
Wilcoxon test (significance level of 0.05)
        - **Statistic:** 18.5
        - **Result:** Same distributions (fail to reject H0) with alpha 0.05
        - **Critival Value:** 81
Experiment B : parametric
Two Groups
T-Test paired test (significance level of 0.05)
        - **Statistic:** 3.984971886611009
        - **Result:** Different distributions (reject H0) with alpha 0.05
        - **Critival Value:** 1.714
Experiment C : parametric
Multiple Groups
ANOVA between cases test (significance level of 0.05)
        - **Statistic:** 1.6543084590576234
        - **Result:** Same distributions (fail to reject H0) with alpha 0.05
        - **P-value:** 0.1823615092393619
        Groups  Nº Samples      Mean  Std. Dev.  Std. Error
0          PDFC          24  0.849375   0.139963    0.028570
1          NNEP          24  0.808542   0.138825    0.028337
2  IS-CHC + 1NN          24  0.793458   0.152517    0.031132
3       FH-GBML          24  0.755167   0.149255    0.030467
          Source  Degrees of Freedom (DF)  Sum of Squares (SS)  Mean Square (MS)    F-Stat Rejected Value
0  Between Groups                        3             0.109271          0.036424  1.654308         2.6802
1   Within Groups                       92             2.025613          0.022018
2           Total                       95             2.134884          0.058441
Experiment D with alpha 0.05 : no parametrics
Multiple Groups
Friedman test (significance level of 0.05)
        - **Statistic:** 16.225000000000023
        - **Result:** Different distributions (reject H0) with alpha 0.05
        - **P-value:** 0.0010197319283888895
     PDFC     NNEP  IS-CHC + 1NN  FH-GBML
0  1.77083  2.47917       2.47917  3.27083
Post hoc: Bonferroni test (significance level of 0.05)
              Comparison  Statistic (Z)   p-value  Adjusted alpha  Adjusted p-value  alpha         Results
0            PDFC vs NNEP       1.900658  0.057347        0.008333          0.344081   0.05  H0 is accepted
1     PDFC vs IS-CHC + 1NN       1.900658  0.057347        0.008333          0.344081   0.05  H0 is accepted
2          PDFC vs FH-GBML       4.024922  0.000057        0.008333          0.000342   0.05  H0 is rejected
3     NNEP vs IS-CHC + 1NN       0.000000  1.000000        0.008333          1.000000   0.05  H0 is accepted
4          NNEP vs FH-GBML       2.124265  0.033648        0.008333          0.201888   0.05  H0 is accepted
5  IS-CHC + 1NN vs FH-GBML       2.124265  0.033648        0.008333          0.201888   0.05  H0 is accepted
Saved Graph with name
```

Figure 4.2: Example of console interface analysis_of_experiments without pdf

Additionally, Figure 4.3 illustrates a sample report in pdf summarizing all the results of the statistical analysis. To obtain it, it is required to indicate so in the *analysis_of_experiments* function by means of *generate_pdf=True* followed by the name of the report (*name_PDF=NAME*).



Inform elaborated with StaTDS
Version 1.0
https://github.com/kdis-lab/statds

Experiment A : no parametrics

Two Groups

Wilcoxon test (significance level of 0.05)

- **Statistic:** 18.5
- **Result:** Same distributions (fail to reject H0) with alpha 0.05
- **Critival Value:** 81

Experiment B : parametric

Two Groups

T-Test paired test (significance level of 0.05)

- **Statistic:** 3.984971886611009
- **Result:** Different distributions (reject H0) with alpha 0.05
- **Critival Value:** 1.714

Experiment C : parametric

Multiple Groups

ANOVA between cases test (significance level of 0.05)

- **Statistic:** 1.6543084590576234
- **Result:** Same distributions (fail to reject H0) with alpha 0.05
- **P-value:** 0.1823615092393619

Groups	Nº Samples	Mean	Std. Dev.	Std. Error
PDFC	24	0.8494	0.14	0.0286
NNEP	24	0.8085	0.1388	0.0283
IS-CHC + 1NN	24	0.7935	0.1525	0.0311
FH-GBML	24	0.7552	0.1493	0.0305

Source	Degrees of Freedom (DF)	Sum of Squares (SS)	Mean Square (MS)	F-Stat	Rejected Value
Between Groups	3	0.1093	0.0364	1.6543084590576234	2.6802
Within Groups	92	2.0256	0.022		

Page 1/7

Figure 4.3: Example of report in pdf

5 StaTDS Web Client

The reason for developing the StaTDS Web Client lies in the immense conveniences it can offer. For instance, users can access it from any location on the internet without extra software installations or knowledge of Python. Another benefit is streamlined test management: the interface allows users to manage multiple tests.

We developed this tool using the Dash framework [17], which facilitates quick data app creation in Python. This integration enables direct use of the library's features without external services. We chose Dash mainly for its simplicity in building interactive dashboards with Python. While developers can use only Python, they also have the option to customize with JavaScript, HTML, or CSS.

In this chapter, we will discuss the developed web app. We will start by outlining the system requirements for installation and access. Then, we will describe the graphical interface you will see when using the tool, focusing on its layout and usability. Next, we will go over the various features the application provides. We will conclude with a Help and Support section.

5.1 System Requirements

5.1.1 Own Installation

If you wish to install it natively, there are two distinct methods: using the Dockerfile available in the GitHub repository or using the application's code and deploying it on your system. The following will detail the two methods to carry out this deployment task.

5.1.1.1 Using Docker

Firstly, to begin with, it is essential to download the repository from GitHub to obtain the Dockerfile. Before this step, ensure that Docker is installed on

your computer [7]. With Docker ready to use, you can build the application's image by executing the following command:

```
docker build -t name-lib ./
```

After the image has been successfully created, the next step is to instantiate a container using that image.

```
docker run -p 8050:8050 --name container name-lib
```

Now, you can access to the interface with your Web navigator through the following url: `http://localhost:8050`

5.1.1.2 Using Dash and Python

In this case, it is essential to install Dash in your environment of Python. The following commands install all depends:

```
pip install statds["full-app"]
```

After you successfully install the app, the next step is to start the server by specifying the port and IP host:

```
from statds import app

app.start_app(port="8050", host="0.0.0.0")
```

Now, you can access to the interface with your Web navigator through the following url: `http://localhost:8050`

5.1.2 Access to the interface

To ensure an optimal and efficient experience when using our tool, we recommend the following minimum requirements for users to access the tool:

- Web Browser: Although we designed our application to be compatible with most modern browsers, we recommend using the following versions or newer
  - Google Chrome: version 85.0.
  - Mozilla Firefox: version 80.0.
  - Microsoft Edge: version 85.0.
  - Safari: version 14.0.
- Operating System: Our application works on any operating system that supports the recommended web browsers, including Windows, MacOS, Linux, and mobile platforms like Android and iOS.

- Internet Connection: As the tool operates online, you'll need a stable internet connection. We've designed the tool to work well with low-speed connections, but for the best experience, we recommend a speed of at least 1 Mbps.
- Screen Resolution: While the application is fully functional on various screen resolutions, for a better visual experience, we recommend a minimum resolution of 1280 x 720 pixels.
- File Upload: While the tool allows .csv file uploads for analysis, it's important to note that the recommended maximum size for these files is 10MB to ensure reasonable processing times.

Once you've checked the listed requirements, you can access the tool via the URL (server.com). The server hosting the application is within the University of Córdoba's network, so it adheres to all their system's access restrictions.

5.2 Dashboard & Interface Design

When you click the link, you'll land directly on the Dashboard's Home Page" (see Figure 5.1). Highlighted features include:

- Navigation Bar: This bar guides your navigation throughout the application and remains consistent in every section. You can find functionalities like Data Analysis", Normality and Homoscedasticity", Import Data", and "Export Results".

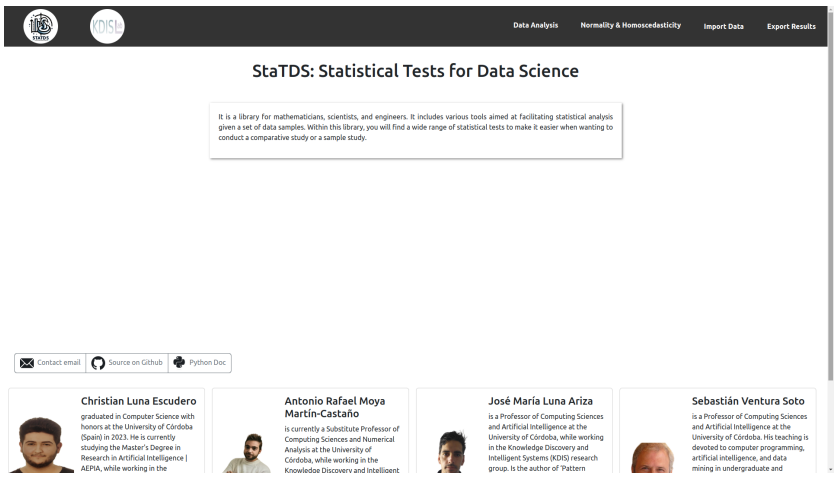


Figure 5.1: Home Page - StaTDS Web Client

- Main Panel: Here, you will see information about the library and the GitHub repository.

Each functionality's interface shares a similar design, splitting the main content into one or two sections based on the user's information needs.

5.3 Navigation

When defining the user navigation, there is only one possible user profile, so there is no need to log in. The default user navigation likely resembles what is shown in Figure 5.2. Below, we detail the typical steps taken by the user:

- The user accesses the main page and navigates to any functionality.
- The user wishes to upload their test results, so they navigate to the "Import Data" page.
- The user provides the data for the study.
- Once supplied, they are redirected to the main page to view the data.
- They can then use "Data Analysis" for statistical tests or "Normality and homoscedasticity" to study the data's normality and homogeneity. From here, users can navigate freely and choose their next steps.

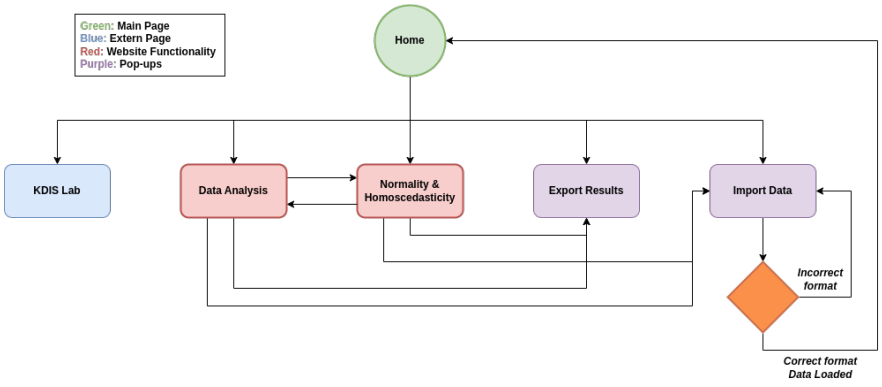


Figure 5.2: Navigation Graph

5.4 System functionalities

We will now explain in detail each of the functionalities available to the user, with various examples and screenshots to help understand the use of the application.

5.4.1 Import Data set

When you click the “Import Data” button on the navigation bar, it shows a dropdown menu (see Figure 5.3). The menu lets you input data: type into a text box (showing a data structure example) or upload a .csv file. Whether you choose the text box or file upload, you can set the delimiter for the data.

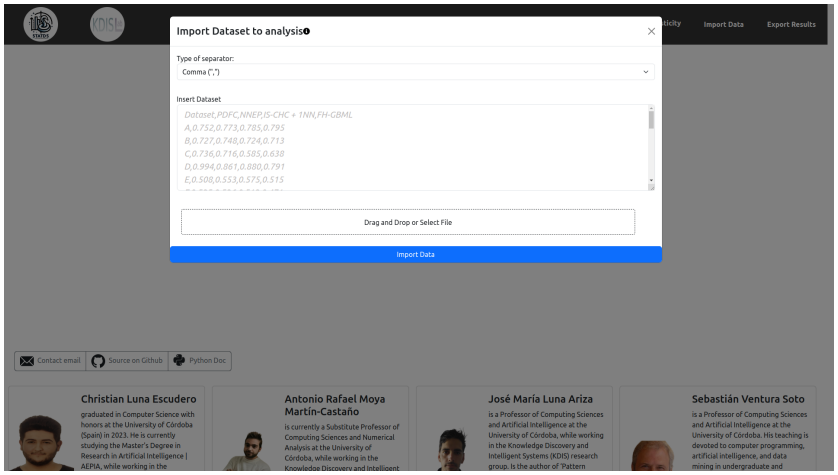


Figure 5.3: Import Data - StaTDS Web Client

After providing data through Import Data”, the application takes you to the Main Panel”. Here, you will see your data alongside the information previously displayed in Figure 5.4, letting you check the accuracy of your upload.

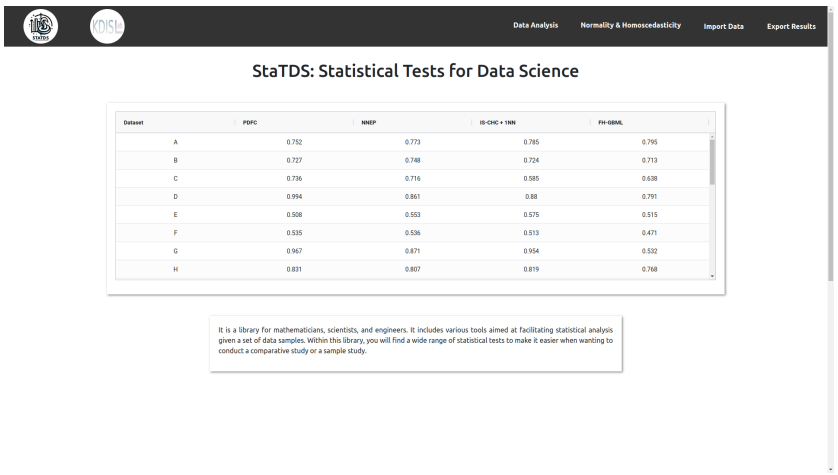


Figure 5.4: Home Page With Data Load - StaTDS Web Client

5.4.2 Data Analysis

Figures 5.5 and 5.6 show this "view/page" panel. In both cases, the left side features a form with different test options users can apply. We can split this form into four parts:

- First three parts:
  - Selection of the significance levels for the tests (α).
  - Selection of the statistical test for comparing two groups.
  - Selection of the statistical test for comparing multiple groups.

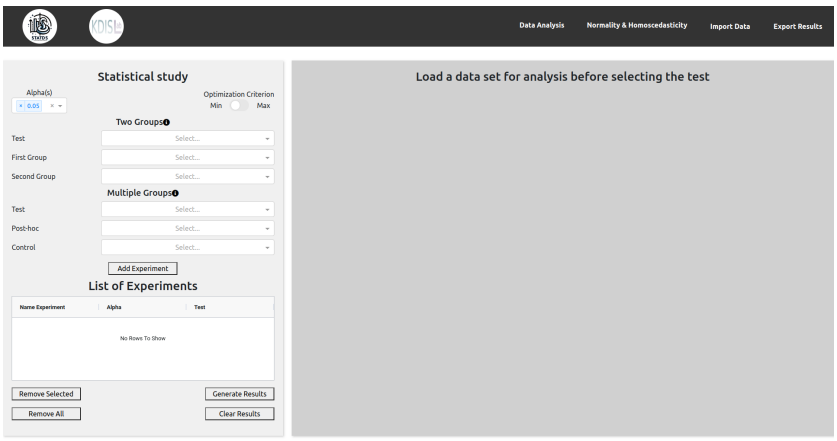


Figure 5.5: Data Analysis Without Data Load - StaTDS Web Client

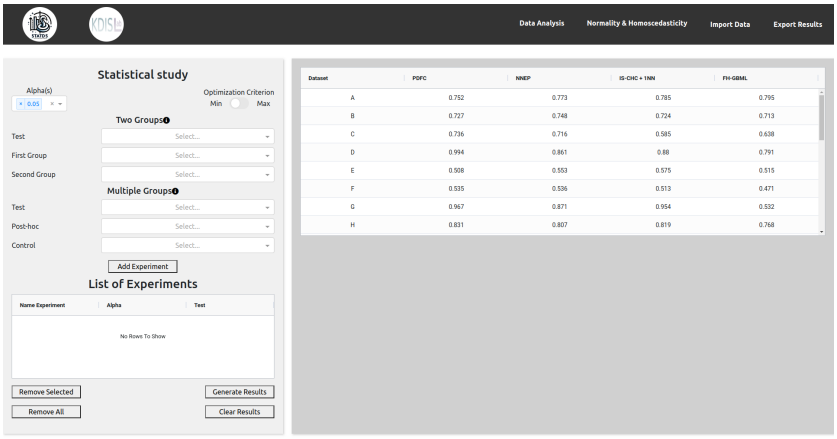


Figure 5.6: Data Analysis With Data Load - StaTDS Web Client

- Fourth part: This section displays a summary table of the upcoming tests.

After selecting experiments, click the "Send" button, then the application runs all experiments and updates the right side to display results (see Figure 5.7).

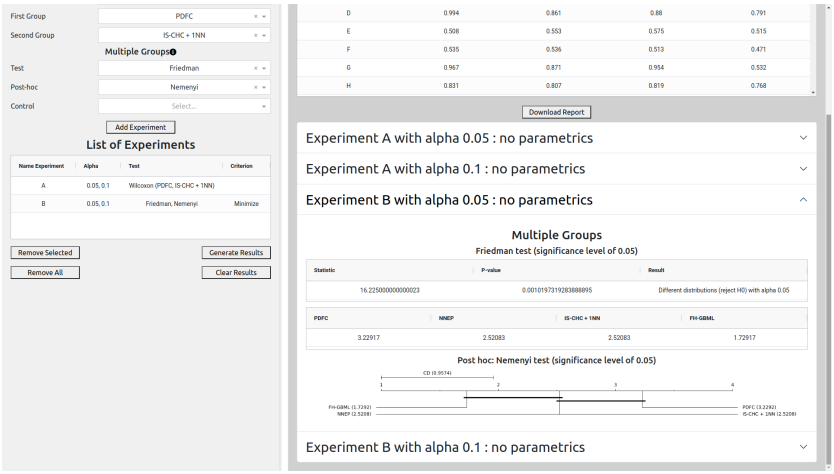


Figure 5.7: Data Analysis With results - StaTDS Web Client

5.4.3 Normality & Homoscedasticity

Before comparing algorithm behavior with parametric tests, ensure the data meets the Normality and Homogeneity assumptions. In the "normality view" (see Figure 5.8), you can select the test for normality and homogeneity. After selecting and clicking the "Send" button, the right panel displays the results, as shown in Figure 5.9

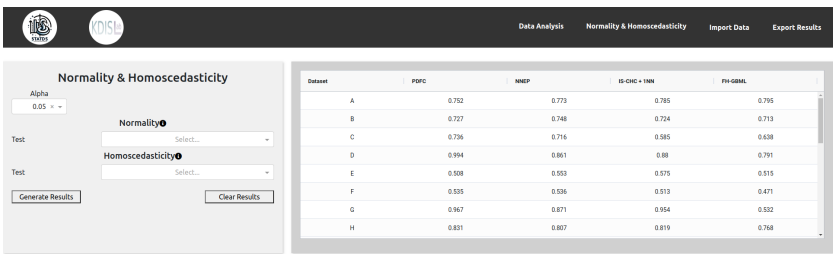
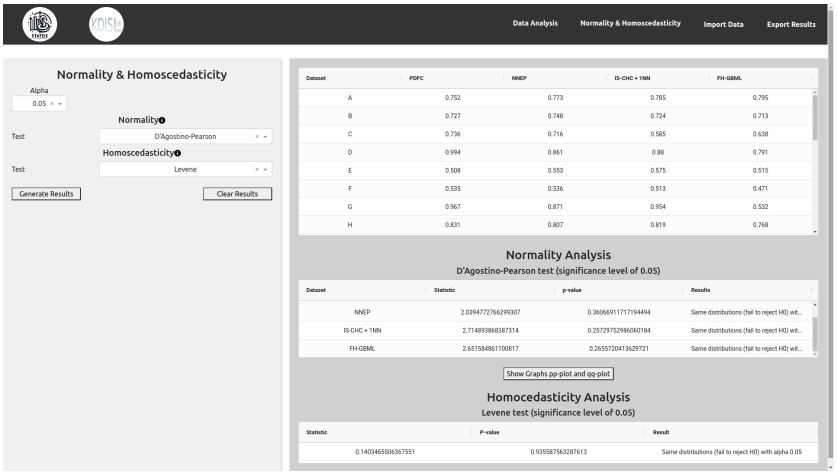
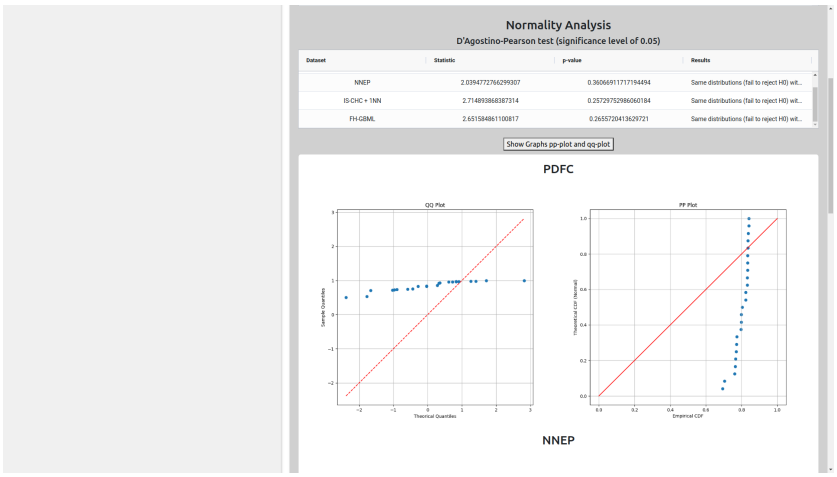


Figure 5.8: Normality & Homoscedasticiy View - StaTDS Web Client



(a) Analytical Methods



(b) QQ-Plot and PP-Plot

Figure 5.9: Normality & Homoscedasticity Results - StaTDS Web Client

5.4.4 Export Results

The web application also allows users to export the results from the different tables. It's possible to generate the LaTeX code to embed these results easily into any document written in LaTeX. To access this feature, one should click on the "Export Results" button in the navigation bar, which will open a "pop-up" (see Figure 5.10) containing the code for each table displayed in the results.

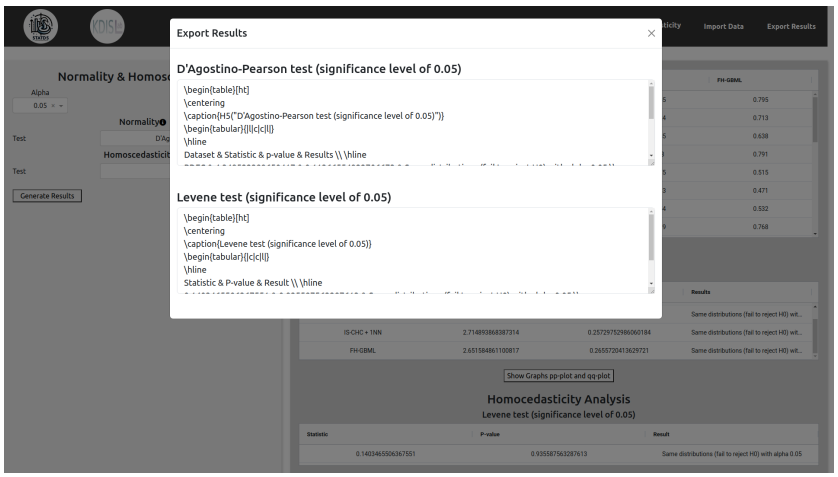


Figure 5.10: Export Results - StaTDS Web Client

5.5 Help & Support

5.5.1 How Can I Determine Which Statistical Test to Use?

To ascertain the correct statistical test, refer to the procedure outlined in Figure 5.11. Begin by evaluating whether your data satisfies the criteria for normality and homoscedasticity, prerequisites for employing parametric tests. Should these conditions be unmet, recourse to non-parametric tests is advisable.

5.5.2 Is There a Way to Conduct a Large Set of Statistical Tests on the Same Dataset?

You can conduct an extensive array of statistical tests on a single dataset using the library functions utils.generate_json and utils.analysis_of_experiments. These functions facilitate the definition of multiple alpha levels and

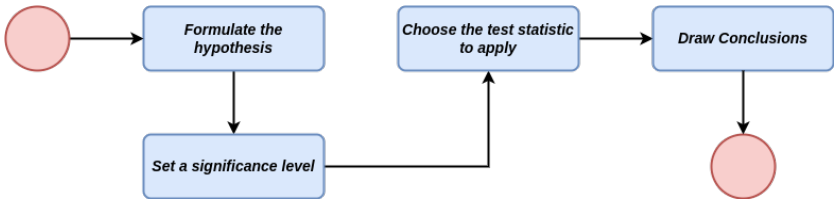


Figure 5.11: Steps to perform a hypothesis test

the concurrent execution of various tests. The web interface offers an enhanced user experience; by utilizing the form presented in Figure 5.7, you can input experiments and initiate the generation of results once you have selected all the requisite tests.

5.5.3 How Can I Copy the Result Tables from the Conducted Statistical Tests?

In the application, result tables are provided as pandas.DataFrame objects, with the method of export being at the discretion of the user. On the web interface, the "Export Results" feature depicted in Figure 5.10 enables the direct replication of the LaTeX code.

5.5.4 What Data File Formats are Supported in the Application?

The supported formats are .csv, .xlss, and .txt. For .csv and .txt files, they should follow the format shown below:

```
Dataset,C1,C2,C3,C4
A,0.752,0.773,0.785,0.795
B,0.727,0.748,0.724,0.713
C,0.736,0.716,0.585,0.638
D,0.994,0.861,0.880,0.791
E,0.508,0.553,0.575,0.515
F,0.535,0.536,0.513,0.471
```

For .xlss files, data should follow the same format but on the first sheet of the document.

5.5.5 Where Can I Find Updated Prototypes of the Functions?

You can find the latest prototypes for the functions in the following Web Documentation-link, for which a reference.

5.5.6 Is There a Way in the Application to Learn About the Tests?

To familiarize yourself with the tests, engage with the help buttons strategically positioned adjacent to the selection tools in the application.

6 Reporting bugs

Feel free to open an issue at Github if anything is not working as expected `https://github.com/kdis-lab/StaTDS`. Merge request are also encouraged, it will be carefully reviewed and merged if everything is all right.

Bibliography

[1] RALPH D'agostino and Egon S Pearson. "Tests for departure from normality. Empirical results for the distributions of b^2 and square $\sqrt{b^1}$". In: *Biometrika* 60.3 (1973), pp. 613–622.

[2] Ralph B d'Agostino. "An omnibus test of normality for moderate and large size samples". In: *Biometrika* 58.2 (1971), pp. 341–348.

[3] Ralph B D'agostino, Albert Belanger, and Ralph B D'Agostino Jr. "A suggestion for using powerful and informative tests of normality". In: *The American Statistician* 44.4 (1990), pp. 316–321.

[4] *D'Agostino-Pearson Test*. Real Statistics Using Excel. 2023. URL: `https://real-statistics.com/tests-normality-and-symmetry/statistical-tests-normality-symmetry/dagostino-pearson-test/`. (accessed: 11-10-2023).

[5] Janez Demšar. "Statistical comparisons of classifiers over multiple data sets". In: *The Journal of Machine learning research* 7 (2006), pp. 1–30.

[6] Salvador García et al. "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power". In: *Information sciences* 180.10 (2010), pp. 2044–2064.

[7] *Get Docker — Docker Docs*. Docker Inc. 2023. URL: `https://docs.docker.com/get-docker`. (accessed: 11-10-2023).

[8] *Git - for Windows*. URL: `https://gitforwindows.org/`. (accessed: 12-10-2023).

[9] *Git - Installing Git — git-scm.com*. URL: `https://git-scm.com/book/en/v2/Getting-Started-Installing-Git`. (accessed: 11-10-2023).

[10] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[11] Ronald L Iman and James M Davenport. "Approximations of the critical region of the fbietkan statistic". In: *Communications in Statistics-Theory and Methods* 9.6 (1980), pp. 571–595.

[12] Siddik Keskin. "Comparison of several univariate normality tests regarding type I error rate and power of the test in simulation based small samples". In: *Journal of Applied Science Research* 2.5 (2006), pp. 296–300.

[13] Howard Levene et al. "Contributions to probability and statistics". In: *Essays in honor of Harold Hotelling* 278 (1960), p. 292.

[14] Jianjun David Li. "A two-step rejection procedure for testing multiple hypotheses". In: *Journal of Statistical Planning and Inference* 138.6 (2008), pp. 1521–1527.

[15] *NIST/SEMATECH e-Handbook of Statistical Methods*. National Institute of Standards and Technology. 2023. URL: `http://www.itl.nist.gov/div898/handbook/`. (accessed: 11-10-2023).

[16] *One-Way ANOVA - NIST/SEMATECH e-Handbook of Statistical Methods*. National Institute of Standards and Technology. 2023. URL: `https://www.itl.nist.gov/div898/handbook/ppc/section2/ppc231.htm`. (accessed: 11-10-2023).

[17] Plotly. *Dash Documentation & User Guide*. URL: `https://dash.plotly.com/`. (accessed: 12-10-2023).

[18] Nornadiah Mohd Razali, Yap Bee Wah, et al. "Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests". In: *Journal of statistical modeling and analytics* 2.1 (2011), pp. 21–33.

[19] Juliet Popper Shaffer. "Modified sequentially rejective multiple test procedures". In: *Journal of the American Statistical Association* 81.395 (1986), pp. 826–831.

[20] David J Sheskin. *Handbook of parametric and nonparametric statistical procedures*. Chapman and hall/CRC, 2003.

[21] David J Sheskin. *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2020.

[22] George W Snedecor and William G Cochran. "Statistical Methods, eight edition". In: *Iowa state University press, Ames, Iowa* 1191.2 (1989).

[23] Shapiro Ss. "An analysis of variance test for normality (complete samples)". In: *Biometrika* 52 (1965), pp. 591–611.

A StaTDS Tutorial using Jupyter Notebook

This tutorial enhances the accessibility of StaTDS (library version) by offering a extensive collection of Jupyter notebooks. You can access these notebooks on GitHub at https://github.com/kdis-lab/statds or Google Colab. Each section below outlines the key topics covered in the respective notebooks.

A.1 Statistical comparison of classification algorithms

This Jupyter notebook includes a comparison of four classification algorithms: Random Forest, Gradient Boosting, Support Vector Machine, and Artificial Neural Network. In this comparison, measures such as accuracy and false positive rates are evaluated. We conduct a series of tests on ten diverse datasets, using 5-fold cross-validation to evaluate the algorithms' performance. You can access this study through the following link. The notebook includes some specific parts implemented by pandas (dataset loading) and sklearn (classification algorithms, training and testing) that do not belong to StaTDS and could be done with other different libraries.

A.2 Statistical comparison of regression algorithms

This Jupyter notebook presents a comparative analysis of four regression algorithms: Random Forest, Support Vector Machine, Lasso Regression, and Ridge Regression. We evaluate them by using mean squared error and R-squared. We examine the algorithms' behavior on ten different datasets, employing a bootstrap evaluation method with a size of 5. You can access this study through the following link. The notebook includes some specific parts

implemented by pandas (dataset loading) and sklearn (regression algorithms) that do not belong to StaTDS and could be done with other different libraries.

A.3 Statistical comparison of clustering algorithms

This Jupyter notebook includes a comparison of three clustering algorithms: K-Means, Agglomerative Clustering, and DBSCAN, with an emphasis on Silhouette Score and Davies-Bouldin Score as evaluation metrics. We implement various tests to assess the performance of these algorithms over ten different datasets. You can access this study through the following link. The notebook includes some specific parts implemented by pandas (dataset loading) and sklearn (clustering algorithms) that do not belong to StaTDS and could be done with other different libraries.

A.4 Statistical comparison of association rule mining algorithms

This Jupyter notebook presents a comparison of two well-known algorithms for mining association rules: Apriori and FP-Growth. The algorithms are compared in terms of runtime (the lower the better) for a varied set of datasets. You can access this study through the following link. The notebook includes some specific parts implemented by pandas (dataset loading) and mlxtend (association rule mining algorithms) that do not belong to StaTDS and could be done with other different libraries.

B

StaTDS App video-tutorials

This tutorial enhances the accessibility of StaTDS (web app version) by offering a extensive collection of videos. You can access these on Youtube at https://www.youtube.com/@StaTDS.

B.1 Import Data

Learn the basics of importing various data formats into StaTDS, including CSV, Excel, and SQL databases. This tutorial demonstrates the step-by-step process to ensure an easy data import experience (see link@YouTube).

B.2 Normality tests

Discover how to perform and interpret different normality tests using StaTDS. This section covers the following tests with practical examples:

- Shapiro-Wilk: link@YouTube
- D'Agostino-pearson: link@YouTube
- Kolmogorov-Smirnov: link@YouTube

B.3 Homoscedasticity tests

Explore techniques for testing homoscedasticity (equal variances) in your data with tutorials on the following tests:

- Levene: link@YouTube
- Batlett: link@YouTube

B.4 Parametric tests

Master parametric testing methods in StaTDS, including paired and unpaired t-tests, and ANOVA for both between and within cases. Some practical examples and tips are included in the following links:

- T-Test paired: link@YouTube
- T-Test unpaired: link@YouTube
- ANOVA between cases: link@YouTube
- ANOVA within cases: link@YouTube

B.5 Non-parametric tests

Learning non-parametric testing with StaTDS, ideal for data that does not meet parametric assumptions, is easy. Some available video-tutorials include Wilcoxon, Binomial Sign, Mann-Whitney U, Friedman, Friedman Aligned Ranks, and Quade tests.

- Wilconxon: link@YouTube
- Binomial sign: link@YouTube
- Mann-Whitney U: link@YouTube
- Friedman: link@YouTube
- Friedman Aligned Ranks: link@YouTube
- Quade: link@YouTube

B.6 Post-hoc tests

Some descriptions of using post-hoc tests are also available on tests such as: Nemenyi, Bonferroni, Li, Holm, Holland, Finner, Hochberg, Hommel, Rom, and Schaffer.

- Friedman + Nemenyi: link@YouTube
- Friedman + Bonferroni: link@YouTube
- Friedman + Li: link@YouTube

- Friedman + Holm: link@YouTube
- Friedman + Holland: link@YouTube
- Friedman + Finner: link@YouTube
- Friedman + Hochberg: link@YouTube
- Friedman + Hommel: link@YouTube
- Friedman + Rom: link@YouTube
- Friedman + Schaffer: link@YouTube