Use of a reproducible R Shiny web app to promote students' interest in coding

Xuemao Zhang*

Abstract

Programming or coding plays a crucial role in ensuring reproducibility in data analysis. Among the various programming languages in the field of data science, R stands out as a powerful tool capable of implementing a diverse range of statistical and graphical techniques. Unfortunately, in general education, many non-computing majors do not take a typical programming course, perhaps due to their fear of the command line. A web app Rstats (`http://esumath.shinyapps.io/rstats`) was developed to promote students' interest in R programming while learning introductory statistics, irrespective of their coding background. The web app provides a user-friendly point-and-click interface for data analysis, allowing students to explore statistical concepts without writing code directly, while also generating and displaying reproducible R code for the analysis. Thus, students can learn the programming language R while performing data analysis with the web app. By removing the intimidation associated with coding features, the web app aims to engage a broader audience in the realm of data and information. Source code of the app is provided for instructors who wish to deploy the web app to their own server. The features of the web app are described in detail and how the web app was applied to a GE statistical data analysis is discussed.

Keywords: General Education, introductory statistics, R, Shiny, web app

1. INTRODUCTION

General Education (GE) courses serve as a means to broaden the perspectives of undergraduate students beyond their major disciplines, preparing them to navigate and contribute effectively to a dynamic and evolving society. As Rushkoff (2012) claims, learning to code familiarizes people with the values of a digital society: how people collaborate and share information. Traditionally, programming classes, such as Introduction to Programming, were not commonly integrated into general education curricula at universities before 2015 (Ferguson 2015). Furthermore, non-computing majors often avoid traditional programming courses, possibly due to their fear of using the command line. This reluctance places them at a technical disadvantage in an era dominated by data and information.

To alleviate the apprehension associated with the command line among non-computing students, integrating computer programming into a broader range of General Education (GE) courses is one way to bridge this gap. An ideal candidate for incorporating programming content is the introductory statistics course, as the practical application of various statistical

*xzhang2@esu.edu, East Stroudsburg University, 200 Prospect Street, East Stroudsburg, PA 18301

concepts and formulas involves numerical computations. Traditional point-and-click statistical software packages like Minitab, SPSS, and Stata, while popular, fall short in cultivating students' programming skills. These tools operate as black boxes, revealing only input and output, hindering students from gaining a deeper understanding of statistical concepts. In contrast, programming languages such as R (R Core Team 2024), Python, and SAS stand out as top choices for statistical data analysis in the field of data science. Both R and Python are open-source, with Python serving as a versatile general-purpose programming language, while R is specifically tailored for data analysis. Considering these factors, introducing programming through R in an introductory statistics class is a compelling option for non-computing majors. This approach not only demystifies the command line but also empowers students with practical programming skills crucial for data analysis in diverse academic and professional contexts.

The R language stands as an embodiment of the S programming language, rooted in the realm of data analysis rather than originating from a conventional programming language background. Upholding the same philosophy as its precursor S, R prioritizes simplicity to facilitate seamless data analysis. As the S language developer John Chambers wrote Peng (2016): "We wanted users to be able to begin in an interactive environment, where they did not consciously think of themselves as programming. Then as their needs became clearer and their sophistication increased, they should be able to slide gradually into programming, when the language and system aspects would become more important." In comparison to Python, R boasts a more user-friendly learning curve, making it accessible to a broader audience. Its widespread adoption is propelled by its open-source nature and its robust capability to implement an extensive array of statistical and graphical techniques. The popularity of R is further amplified by its expansive ecosystem, featuring over 18,000 add-on packages—code contributions aimed at enhancing the core language or addressing specific problem domains—thanks to a dynamic and vibrant developer community. These packages extend the capabilities of R Wickham (2015-b). Notably, the R Shiny package deserves special mention for empowering users to construct web applications directly from R without requiring knowledge in CSS and Javascript (Beeley 2013).

Instructors have been developing Shiny web apps to enhance the learning experience for statistics students, as highlighted by Doi (2016). One notable advantage lies in the accessibility of Shiny web app development, requiring no specialized web development skills, as emphasized by Beeley (2013). For example, Kandlikar (2018) crafted the web app ranacapa for exploratory biodiversity analyses and visualizations of eDNA results. Specifically tailored for statistical analyses, the R Shiny web app ABCMETAapp was innovatively designed for simulation-based estimation of mean and standard deviation in meta-analysis by Kwon (2021). Beyond advanced statistical applications, these Shiny web apps have found utility in undergraduate introductory statistics education. For example, Freire (2019) developed an R Shiny app to facilitate the understanding of probability density functions, while Arnholt (2019) employed a Shiny web app to elucidate the power concept in z-tests and t-tests for introductory statistics students. Williams (2017) created an R Shiny app to assist the learning process of confidence intervals using graphics and data from the US National Basketball Association. Another example is that Stratton (2021) employed a Shiny web app to visualize power curves for multiple estimators and population distributions, showcasing the adaptability of these tools across diverse statistical domains.

In this paper, an R Shiny web app named Rstats is presented, currently accessible on shinyapps.io (http://esumath.shinyapps.io/rstats). This web app serves as a comprehensive tool for conducting various data analysis tasks within the context of introductory statistics courses at the GE level. Rstats employs a user-friendly point-and-click interface, enabling users to effortlessly perform data analyses without the need for programming skills. A distinctive feature of Rstats lies in its emphasis on reproducibility. Upon executing a data analysis task using the web app, corresponding R code is automatically generated and displayed. This generated R code can be easily copied and pasted into the R workspace of a local PC or Mac, allowing students to reproduce the results of their data analyses. This approach facilitates a seamless learning experience, enabling students to familiarize themselves with the R programming language directly within a web browser, without requiring any coding during their interaction with the web app. The point-and-click functionality of Rstats ensures that even students with a limited interest in coding are not deterred by the features.

This article provides an in-depth exploration of the features of the web application, Rstats, which was initially deployed in two undergraduate General Education (GE) statistics courses during the Fall 2021 semester. Section 2 examines of the layout and interface design of the Rstats web application. Explanation of how the web app facilitates probability and quantile calculations is covered in Section 3. Section 4 explores the application's capabilities in statistical inference, encompassing topics such as confidence intervals and hypothesis testing. Graphical illustrations are incorporated to enhance comprehension. Section 5 is a comprehensive coverage of the broader spectrum of data analysis tasks supported by the web app. This includes univariate numerical and categorical data analysis, simple and multiple linear regression analysis, logistic regression analysis, and analysis of variance for 1-factor and 2-factor experimental designs. Finally, reflection on the perceived value of the Rstats web app and potential enhancements are discussed.

2. APPLICATION LAYOUT

The R Shiny app Rstats is designed to fulfill two primary objectives. Firstly, it provides students with an intuitive point-and-click interface for conducting probabilistic calculations and statistical data analyses. This facilitates a more profound understanding of probability distributions and statistical inference, as students can visually inspect the graphical output generated by the software. For instance, in Figure 1, the density curve of the standard normal distribution is depicted, with the orange-shaded area representing the lower tail, illustrating the probability $P(Z \leq 2)$. This visual representation aids students in grasping complex concepts through direct observation.

The second goal of Rstats is to cater to students keen on coding. For this audience, the app enables the extraction of the underlying code for producing the graphical output. Students can then copy this code and execute it on a local R console or Posit Cloud (formerly RStudio Cloud, available at https://posit.cloud/), thereby reproducing the output. As illustrated in Figure 1, students are encouraged to try running the code 'pnorm(2L, mean = 0L, sd = 1L, lower.tail = TRUE)' to replicate the displayed output. The 'L' suffix in the code indicates that the numbers are stored as integers; however, removing the 'L' does not affect the result, as R treats numeric values as floating-point by default. This can help introduce students to

R's numeric data types.

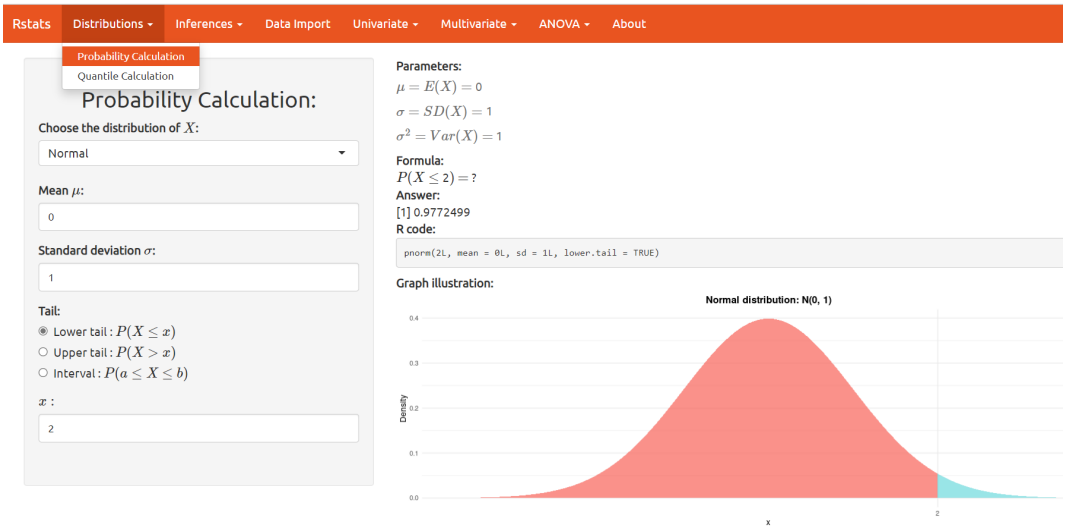

Figure 1: Interface of the web app Rstats.

The web app harnesses the power of R Shiny's reactive programming paradigm, as described by Wickham (2021). Reactive is a style of programming specifying values that evolve over time, with calculations and actions contingent on these dynamic values Bainomugisha (2013). This approach imbues R Shiny apps with interactivity: users manipulate input controls - such as sliders, typing in text boxes, and checkboxes - triggering server-side logic to execute tasks like data reading, subsetting, and model fitting etc. ultimately resulting in outputs updating such as plots redrawing and tables updating. The implementation of reactive programming techniques for constructing an R Shiny app is comprehensively expounded in Chapters 13 to 16 of Wickham (2021). These chapters serve as a practical guide, offering insights into leveraging the capabilities of reactive programming to create engaging and responsive web applications. Notably, a pivotal attribute of this web app lies in its emphasis on reproducibility. All R code employed for data analysis within the app is captured, affording users the ability to seamlessly run the generated R code on an R console outside the web app environment.

The Rstats web app encompasses six distinct menus, as illustrated in Figure 1. Upon initiation, the sub-menu 'Probability Calculation' under the 'Distributions' menu is automatically displayed. Each menu serves a specific purpose, enhancing the app's functionality:

1. Distributions. Users can calculate probabilities and quantiles for various discrete and continuous distributions, aiding in a comprehensive understanding of statistical distributions.

2. Inferences. This menu empowers users to perform statistical inference related to population means, proportions, and variances, given either data or specific statistics.

3. Data Import. Users can upload or manually input data (up to 5 variables), transform variables within the dataset, and download the modified dataset. The app exclusively accepts .csv (comma-separated values) format files for data uploads.

4. Univariate. Univariate data analysis capabilities are provided for the imported dataset, facilitating insights into individual variables.

5. Multivariate. This menu enables users to delve into statistical models such as simple and multiple linear regression, logistic regression, and contingency analysis for multivariate datasets.

6. ANOVA. Users can conduct analysis of variance (ANOVA) for experimental designs involving one or two factors, offering a powerful tool for exploring data from an experimental design.

The web app's reproducibility feature is realized through the utilization of two key techniques: the function *interpolate* Wickham (2015-a) and the *shinymeta* package Cheng (2021). These mechanisms work in tandem to ensure transparency and reproducibility of the executed R code. The function *interpolate* serves a dual purpose by not only executing the provided R code on the server but also documenting the script in a text file visible to users. This approach not only ensures that users can observe the code generated during their interaction with the web app but also facilitates the reproduction of the entire analysis. Complementing this, the *shinymeta* package further enhances the reproducibility feature. By capturing the underlying logic embedded within the Shiny app, *shinymeta* exposes this logic as code visible to users. Essentially *shinymeta* just reverts the Shiny's reactive building blocks.

3. PROBABILITY AND QUANTILE CALCULATIONS WITH THE APP

In this section, the application of the web app for probability and quantile calculations will be described.

3.1. Probability calculations

In this subsection, we will delve into the specifics of probability calculations for Normal distributions, Binomial distributions, and Finite distributions only using the web app. It's important to note that similar procedures can be followed for probability calculations involving other distributions available in the app.

Users can initiate probability calculations by clicking on the sub-menu labeled 'Probability Calculation' under the 'Distributions' menu within the web app. This sub-menu is the default display when users start the app, and it primarily facilitates probability calculations for various probability distributions, both discrete and continuous.

By default, the displayed distribution is the Normal distribution, as indicated in Figure 1. However, users can choose from a range of distributions through the drop-down list beneath the title 'Choose the distribution of X:'. Selections include Finite, Binomial, Poisson, Normal, t, Chi-squared, and F distributions. Once a distribution is chosen, corresponding parameters must be entered. For instance, for a Normal distribution, users would input mean and standard deviation values, and for a t-distribution, degrees of freedom. Notably, finite distributions necessitate users to input distinct values along with corresponding probabilities

(or relative frequencies) or frequencies. Users also have the flexibility to specify the tail probability to be calculated - whether it's the left tail, right tail, or an interval.

Consider a random variable X following a normal distribution with a mean of 1 and a standard deviation of 1.5, denoted as $X \sim N(\mu = 1, \sigma = 1.5)$. We calculate the following three probabilities in this context.

- $P(X < 1.8)$ or $P(X \leq 1.8)$
- $P(X > 1.8)$ or $P(X \geq 1.8)$
- $P(0.5 < X < 1.8)$, or $P(0.5 < X \leq 1.8)$, or $P(0.5 \leq X \leq 1.8)$ or $P(0.5 \leq X < 1.8)$

To perform calculations, users need to modify the default parameter values. The necessary adjustments can be made by entering new values into the two input boxes positioned under the titles '**Mean** μ:' and '**Standard deviation** σ:' respectively. For example, to compute the lower tail probability $P(X < 1.8)$ for the specified normal distribution, users should follow these steps:

1. Input the desired mean and standard deviation values into the respective input boxes.
2. Choose the lower tail probability calculation by selecting the first radio-button under '**Tail:**'.
3. Enter the value for 'x:' as 1.8, indicating the threshold for the probability calculation as shown in Figure 2.

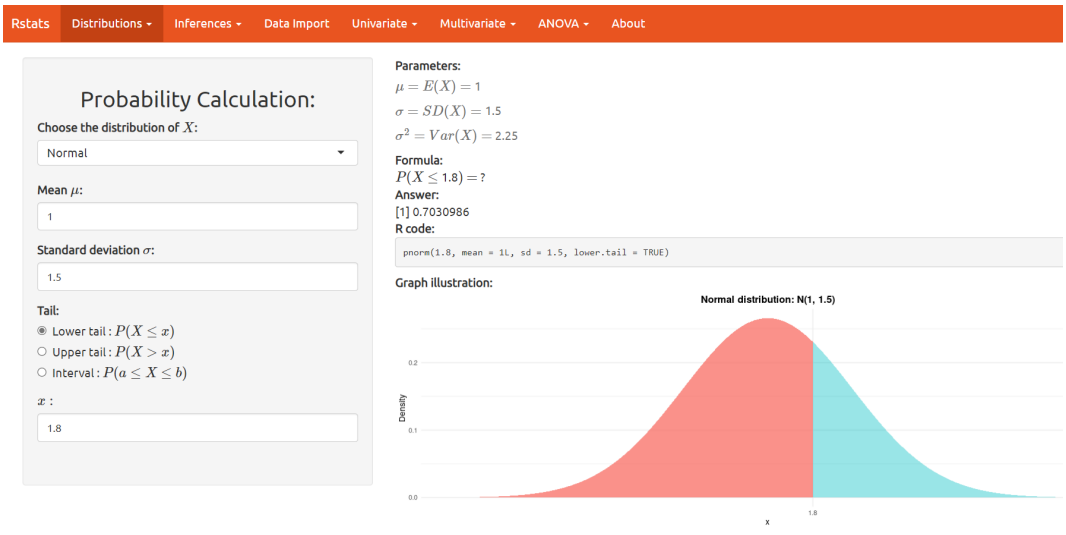

Figure 2: Calculation of $P(X < 1.8)$, where $X \sim N(\mu = 1, \sigma = 1.5)$.

The right panel in Figure 2, depicting the output of the calculation, comprises five distinct components: *Parameters* of the distribution, *Formula* of the calculation, *Answer* of the calculation, *R code*, and *Graph illustration* of the result. *Parameters* displays the parameters

of the normal distribution, providing users with an opportunity to double-check the accuracy of the entered parameter values. Similarly, *Formula* allows users review the formulation of the probability calculation problem, ensuring the correct setup of the query. The segment of *Answer* presents the numerical result of the probability calculation. Users can copy the generated R code for the problem, which is *pnorm(1.8, mean = 1L, sd = 1.5, lower.tail = TRUE)*. This code can be pasted and executed on an R console for further examination or use.

The title of the *Graph illustration* specifies the distribution type and its parameters, and the orange-shaded area on the graph corresponds to the lower-tail area, representing the probability $P(X < 1.8)$. The light-blue shaded area denotes the complement, $P(X \geq 1.8)$. Consistently, the required area is always shaded in orange, while the complementary area is shaded in light-blue for all distributions in this sub-menu.

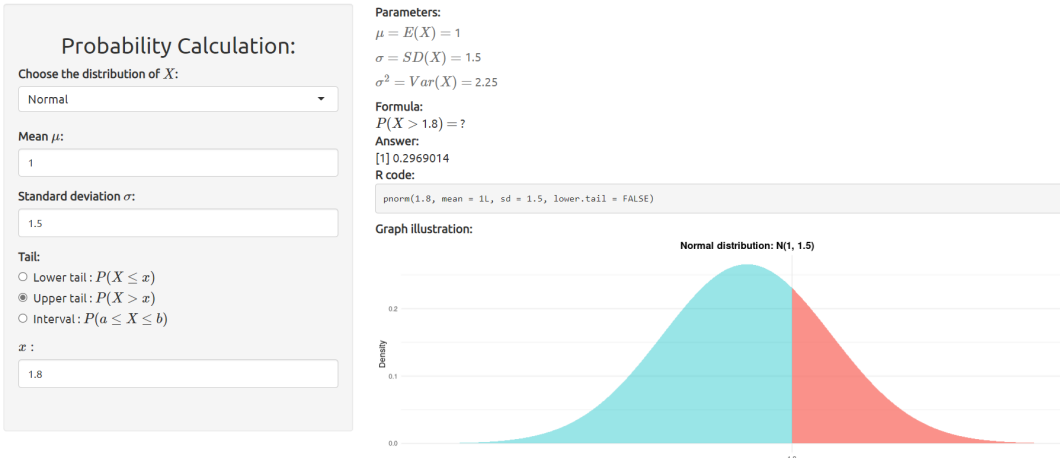

Figure 3: Calculation of $P(X > 1.8)$, where $X \sim N(\mu = 1, \sigma = 1.5)$.

To compute $P(X > 1.8)$, users can simply select the second radio-button under '**Tail:**' and update the value under 'x:' to 1.8, following the illustration in Figure 3. For the calculation of $P(0.5 < X < 1.8)$, users should choose the third radio-button, labeled 'Interval: $P(a \leq X \leq b)$' under '**Tail:**'. Subsequently, input a as 0.5 and b as 1.8, ensuring $a \leq b$, as depicted in Figure 4.

These steps allow users to specify the desired probability calculation scenario, tail, and interval, ensuring the web app performs the intended calculations accurately.

The Binomial distribution, a typical discrete probability distribution in introductory statistics, can be effectively analyzed using the web app. Suppose a random variable X follows a Binomial distribution with the number of trials $n = 20$ and probability of success $p = 0.8$. To calculate point probabilities and sums of several point probabilities, including cumulative probabilities, the following steps can be taken.

1. Select the distribution 'Binomial' from the drop-down list beneath the title 'Choose the distribution of X:' as shown in Figure 5.

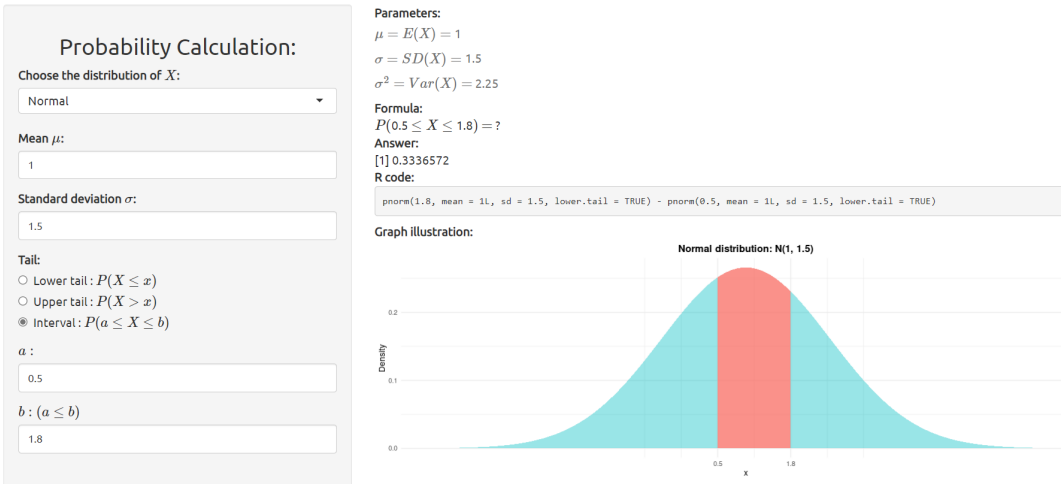

Figure 4: Calculation of $P(0.5 < X < 1.8)$, where $X \sim N(\mu = 1, \sigma = 1.5)$.

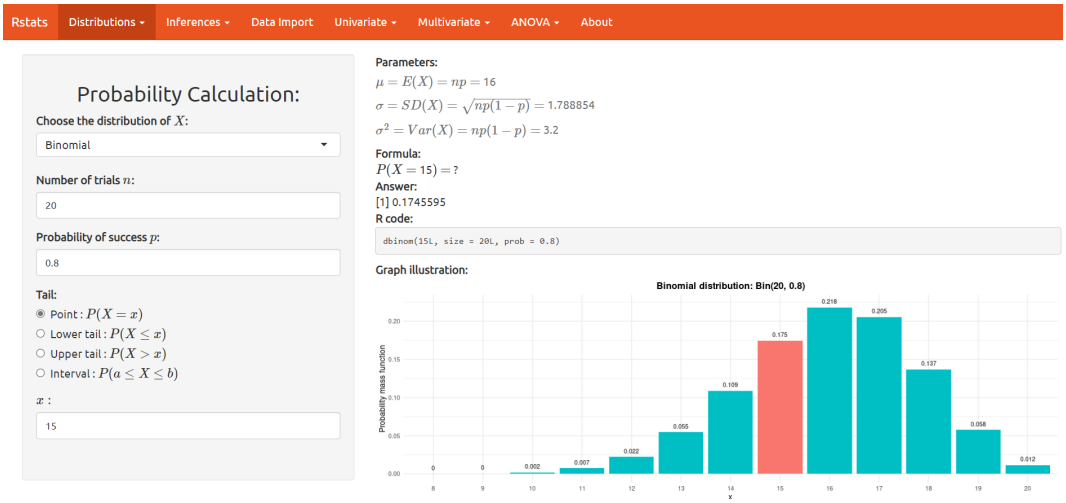

Figure 5: Calculation of $P(X = 15)$, where $X \sim binomial(n = 20, p = 0.8)$.

2. Enter the parameters, specifically '**Number of trials** n:' and '**Probability of success** p:'.

3. To calculate $P(X = 15)$, opt for the first radio-button labeled 'Point: $P(X = x)$' under '**Tail:**' and input the value 15 under 'x:'.

In the right panel, the output of the calculation is displayed, encompassing *Parameters* of the Binomial distribution, *Formula* of the calculation, *Answer* of the calculation, *R code*, and *Graph illustration*. The graph is presented as a bar chart, where the orange bar corresponds to the probability $P(X = 15)$, while other bars are colored light-blue. Additionally, each point probability is labeled above the respective bar in the illustration. This comprehensive representation in the right panel aids users in understanding and validating the calculation results, offering both numerical and visual insights.

Likewise, users have the flexibility to choose other radio-buttons under '**Tail:**' to calculate

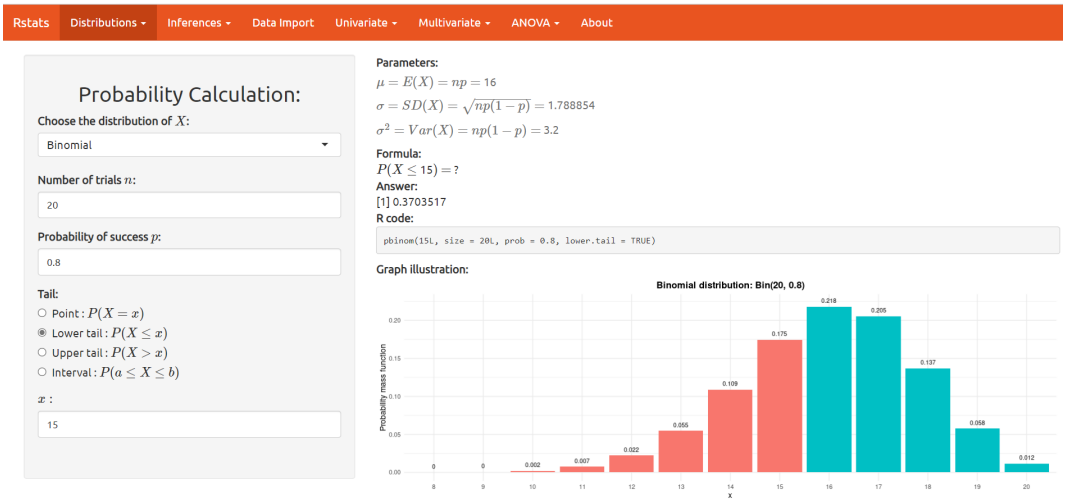

Figure 6: Calculation of $P(X \leq 15)$, where $X \sim binomial(n = 20, p = 0.8)$.

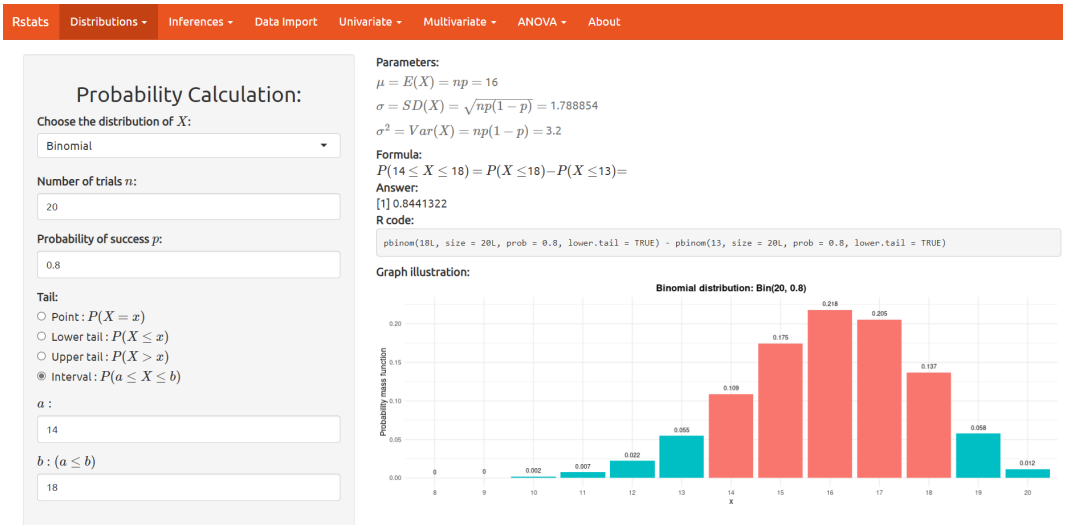

Figure 7: Calculation of $P(14 \leq X \leq 18)$, where $X \sim binomial(n = 20, p = 0.8)$.

various probabilities such as:

- 'Lower tail: $P(X \leq x)$' : The probability that X is less than or equal to a given value x.
- 'Upper tail: $P(X > x)$': The probability that X is greater than a given value x.
- 'Interval: $P(a \leq X \leq b)$': The probability that X falls within a specified interval $[a, b]$, where a is less than or equal to b.

For instance, Figure 6 demonstrates the calculation of the lower tail probability $P(X \leq 15)$, and Figure 7 illustrates the calculation of the interval probability $P(14 \leq X \leq 18)$ for the

binomial distribution $X \sim \text{binomial}(n = 20, p = 0.8)$. By choosing the appropriate radio-button and providing the relevant values for x, a, and b, users can explore a diverse range of probability scenarios for the Binomial distribution.

Table 1: An example of a discrete distribution with finite support

Values	3	5	4	1	2
Probabilities	0.1	0.2	0.4	0.2	0.1
Frequencies	10	20	40	20	10

Discrete distributions with finite support, referred to as finite distributions in the web app, play a crucial role in introductory statistics courses, contributing to both the understanding of probability distribution concepts and measures of center and spread. Using the example distribution from Table 1, the process of entering a finite distribution into the app involves specifying distinct values of X and their corresponding weights w, which could be probabilities, relative frequencies, or frequencies. These values must be separated by commas.

Users can choose any of the three radio-buttons under '**Tail:**' in the app to calculate probabilities, including 'Lower tail: $P(X \leq x)$', 'Upper tail: $P(X > x)$', and interval probabilities 'Interval: $P(a \leq X \leq b)$' for any values a and b where $a \leq b$, as exemplified in Figure 8.

Figure 8: Calculation of $P(X \leq 2)$ for the finite distribution in Table 1.

In the output, the parameters include the mean μ, variance σ^2, and standard deviation σ. The R code for calculating these parameters is provided, enabling students to gain a deeper understanding of the underlying calculations. The R code for calculating the probabilities are not given since only algebraic calculations are involved for a discrete probability distribution with finite support. Students can grasp of the calculation formula for population mean, variance, and standard deviation when they try to figure how the following R code produces the parameters.

```
x = c(3, 5, 4, 1, 2)
w = c(0.1, 0.2, 0.4, 0.2, 0.1)
```

```
mu <- sum(x * w) / sum(w)
sigma2 <- sum((x - mu) ^ 2 * w) / sum(w)
sigma <- sqrt(sigma2)
```

When a finite distribution takes the form of a frequency table, the associated data can be treated as sample data. In such cases, both the sample variance and sample standard deviation will be computed. For instance, if we input the weights w as frequencies, following the structure of Table 1 without altering the original probabilities, both the parameters (considering the data as a population) and the sample statistics (considering the data as a sample) can be calculated, as depicted in Figure 9.

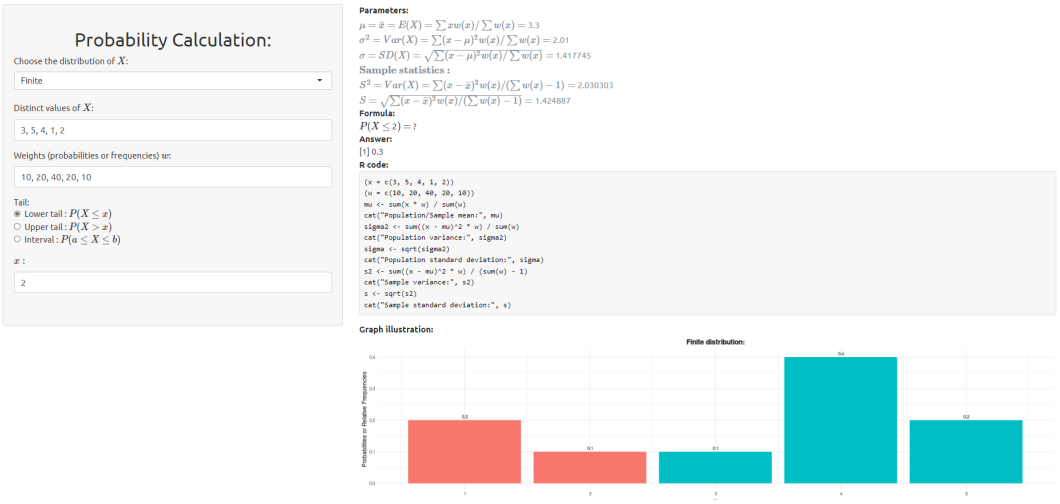

Figure 9: Calculation of $P(X \leq 2)$ for the finite distribution when the distribution is given as a frequency table.

In this scenario, students can gain a deeper understanding of the formulas for mean, variance, and standard deviation for both parameters and statistics by examining the following R code. This code not only produces the calculations for parameters and statistics but also provides an educational opportunity for students to comprehend the underlying principles and distinctions between population and sample calculations. Figure 9 illustrates this process, fostering a more comprehensive understanding of statistical concepts.

```
 x = c(3, 5, 4, 1, 2)
w = c(10, 20, 40, 20, 10)
mu <- sum(x * w) / sum(w)
sigma2 <- sum((x - mu)^2 * w) / sum(w)
sigma <- sqrt(sigma2)
s2 <- sum((x - mu)^2 * w) / (sum(w) - 1)
s <- sqrt(s2)
```

3.2. Quantile calculations

A quantile serves to partition a distribution into two distinct subgroups. Quantile calcula-

tion involves solving a reverse problem compared to probability calculation; it determines a quantile when either the lower-tail or upper-tail probability is specified. For instance, consider finding the quantile x_0 associated with a given lower-tail probability p_0 such that $P(X \leq x_0) = p_0$ for a continuous random variable X. As the solution to this problem involves solving a non-linear equation which is beyond General Education, programming is essential for accurate quantile determination.

The complexity intensifies when dealing with discrete random variables, as their cumulative distribution functions manifest as step functions instead of continuous ones. Consequently, the quantile calculation process for discrete random variables differs significantly. Therefore, the web app Rstats exclusively handles quantile calculations for continuous random variables, specifically those following normal, t, Chi-square, and F distributions. In this subsection, we illustrate quantile calculations using a normal distribution as an exemplar.

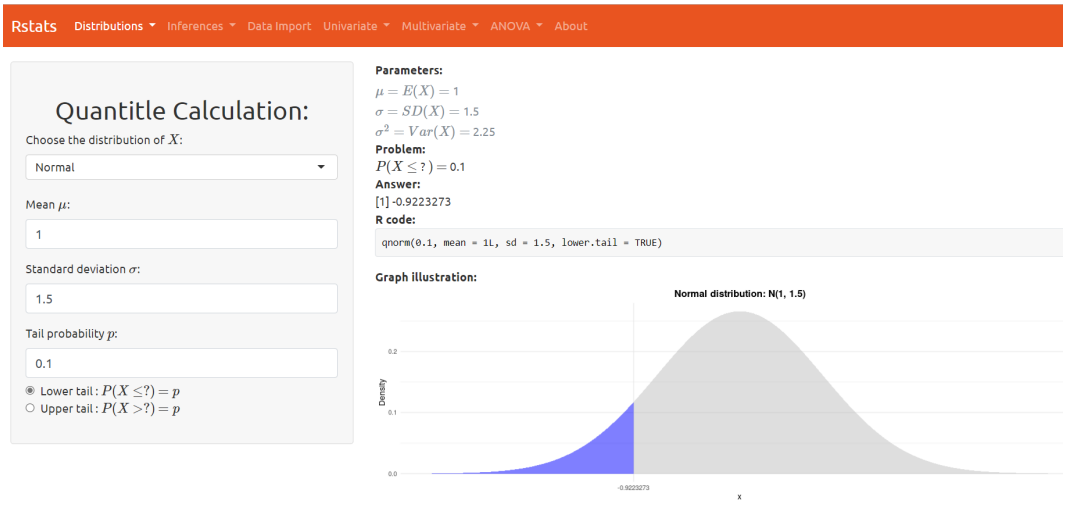

Figure 10: Quantile calculation of $P(X \leq ?) = 0.1$, where $X \sim N(\mu = 1, \sigma = 1.5)$.

Consider a normal random variable X with a mean of 1 and a standard deviation of 1.5, denoted as $X \sim N(\mu = 1, \sigma = 1.5)$. We aim to calculate two quantiles, x_1 and x_2, satisfying the following probabilities.

- $P(X < x_1)$=0.1 and
- $P(X > x_2) = 0.1$.

The sub-menu titled 'Quantile Calculation' is located within the 'Distributions' menu. Similar to the 'Probability Calculation' sub-menu, users must initiate the process by choosing the type of distribution from the drop-down list situated below the heading 'Choose the distribution of X:' as illustrated in Figure 10. Subsequently, the distribution's parameters should be entered.

Following this, users are prompted to input the tail probability p, corresponding to either the lower-tail or upper-tail. The first radio-button, labeled 'Lower tail: $P(X \leq ?) = p$' is preselected by default. However, users may choose the second radio-button, labeled 'Upper

tail: $P(X >?) = p$,' especially if an upper-tail probability is given. It is advisable to make this selection first when dealing with an upper-tail probability since the value of the input tail probability could be changed when users switch between the two radio buttons.

Figure 10 provides a visual representation of how the application addresses the equation $P(X \leq ?) = 0.1$. Once again, the right panel of the interface encompasses five key sections: *Parameters* outlining the distribution characteristics, *Problem* detailing the quantile calculation challenge, *Answer* displaying the calculated result, *R code* showcasing the underlying code, and *Graph illustration* visually representing the outcome.

Within the graph illustration, the shaded blue area signifies the lower-tail probability of 0.1, and the corresponding quantile of -0.922327 is explicitly labeled on the x-axis. Users can easily adapt this process for upper-tail probabilities by selecting the second radio-button labeled 'Upper tail: $P(X >?) = p$' and entering the desired tail probability, such as 0.1. This adjustment is exemplified in Figure 11.

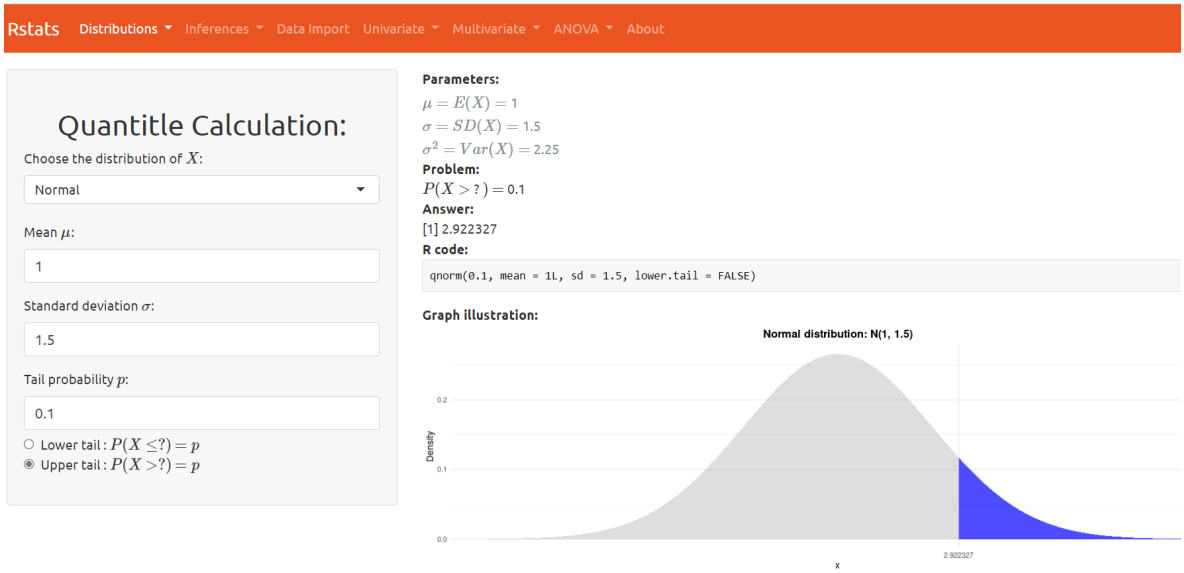

Figure 11: Quantile calculation of $P(X >?) = 0.1$, where $X \sim N(\mu = 1, \sigma = 1.5)$.

4. STATISTICAL INFERENCE WITH THE APP

Statistical inference involves leveraging information derived from a sample to make broader conclusions about the population from which the sample is selected. Essential topics in introductory statistics encompass confidence interval estimation and hypothesis tests, which focus on parameters like population means, proportions, and variances. A fundamental principle guiding statistical inference is the concept of sampling distributions, elucidating the inherent uncertainty in the process.

To facilitate a comprehensive understanding of statistical inference, the app incorporates an 'Inference' menu, featuring two sub-menus: 'Statistics' and 'Data.' Users opt for the 'Statistics' sub-menu when dealing with summary statistics instead of detailed dataset information.

Conversely, the 'Data' sub-menu is suitable for cases where intricate data must be entered as vectors. The app accommodates both one-sample and two-sample statistical inference.

In this section, we demonstrate the functionality of the app's 'Inference' menu using the example of conducting statistical inference about population means with one-sample data entered as a vector. Upon selecting the 'Data' sub-menu, the page titled 'Inference about population parameters when small sample data is manually entered:' appears. The default option in the drop-down list under 'Inference about:' is 'One Population Mean - T Test/Interval'. It's noteworthy that the app seamlessly integrates interval estimation and hypothesis testing. This integration stems from the recognition that confidence intervals can serve both purposes, and the width of a confidence interval serves as an additional measure of statistical significance compared to the p-value of the hypothesis test.

146, 140, 160, 151, 134, 189, 157, 144, 175, 127, 164

Consider a simple random sample of 15-year old boys selected from one city and their weights (in pounds) are listed above. Moreover, suppose the sample is selected from a normal population so conditions for t tests/intervals are met.

Suppose the initial task involves testing the assertion that the sample weights originate from a population with a mean equal to 147 lb at a significance level of $\alpha = 0.05$. Users are guided through the process as follows.

1. Data Entry: Users must input the sample data into the designated box under 'Sample data' as a vector, with values separated by commas.
2. Null Hypothesis (H_0): The mean value of 147 lb under the null hypothesis is specified by entering it into the numerical input field under '$H_0 : \mu = \mu_0 =$'.
3. Alternative Hypothesis (H_1): Users choose the alternative hypothesis from three radio-buttons: $\mu \neq \mu_0$, $\mu < \mu_0$, and $\mu > \mu_0$. The default selection is the two-sided alternative $\mu \neq \mu_0$.
4. Significance Level: The significance level α is set using a slider input, offering values between 0.001 and 0.2 for user selection.

The output, showcased in the right panel of the page as illustrated in Figure 12, comprises five integral components: *Confidence interval*, *Hypothesis test*, *Interpretation* with graph illustration, *R output*, and *R code for the test*. Each section serves a distinct purpose.

- **Confidence Interval** provides comprehensive information about the calculation of the confidence interval, including the confidence level, t-critical value, sample mean, sample standard deviation, and the complete formula of the confidence interval, alongside the calculated result.
- **Hypothesis test** is divided into four components: (1) Statements of the null hypothesis H_0 and the alternative hypothesis H_1; (2) The formula and the value of the test statistic;

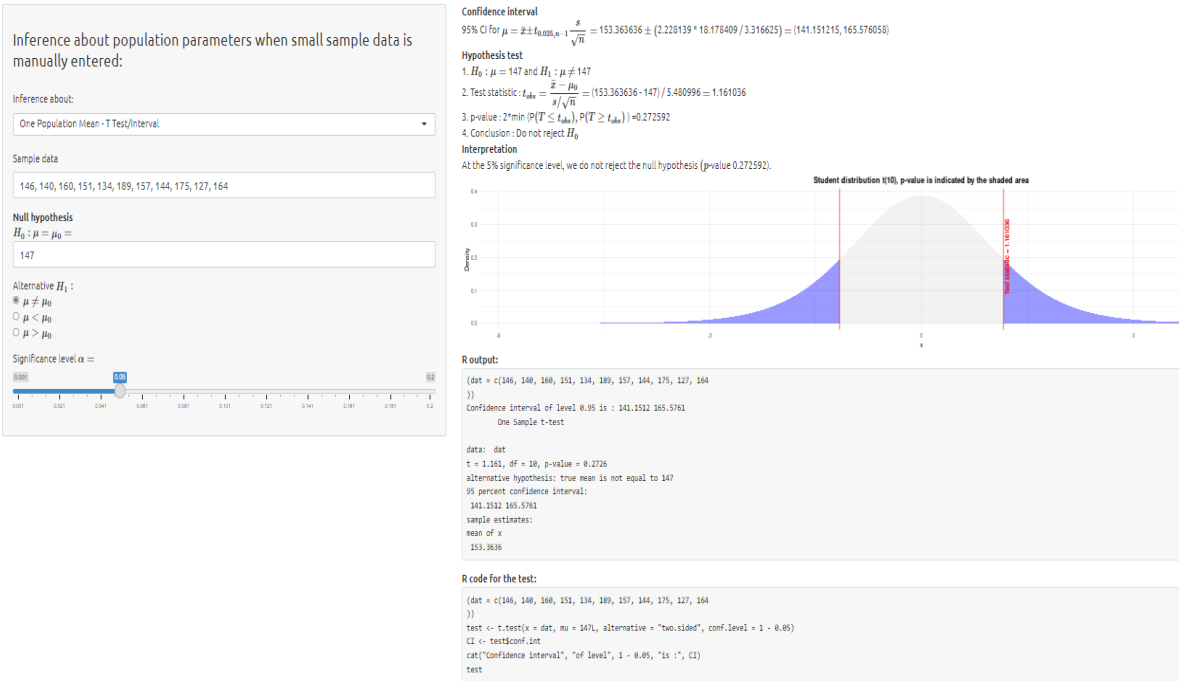

Figure 12: Two-tailed confidence interval estimation of μ and hypothesis testing for $H_0 : \mu = 147$ versus $H_1 : \mu \neq 147$.

(3) The p-value of the test; and (4) The conclusion regarding the rejection or non-rejection of H_0.

- **Interpretation** displays a graph illustration and a succinct explanation. The title of the graph elucidates the distribution of the test statistic under the null hypothesis H_0, and the the region corresponding to the p-value of the test is shaded. The density curve labels the test statistic derived from the sample, offering visual clarity. In the example provided, students can easily discern that the p-value for the two-sided alternative is represented by the sum of two shaded areas.

- **R output** shows the output by R when the R code below is run.

- **R code for the test** provides R code related to confidence interval estimation and hypothesis testing, omitting the code on graph plotting.

While keeping the data and null hypothesis unchanged, users can explore different hypothesis testing scenarios by selecting a different radio-button under 'Alternative H_1:'. For instance, if the new problem is to 'Test the claim that these sample weights come from a population with a mean greater than or equal to 147 lb,' the alternative hypothesis becomes $H_1 : \mu < 147$. In this case, users simply need to select the radio-button $\mu < \mu_0$ to initiate the analysis, as illustrated in Figure 13.

If the problem becomes 'Test the claim that these sample weights come from a population with a mean greater than 147 lb'. Then the alternative becomes $H_1 : \mu > 147$ and users just need

Figure 13: Left-tailed confidence interval estimation of μ and hypothesis testing for $H_0 : \mu = 147$ versus $H_1 : \mu < 147$.

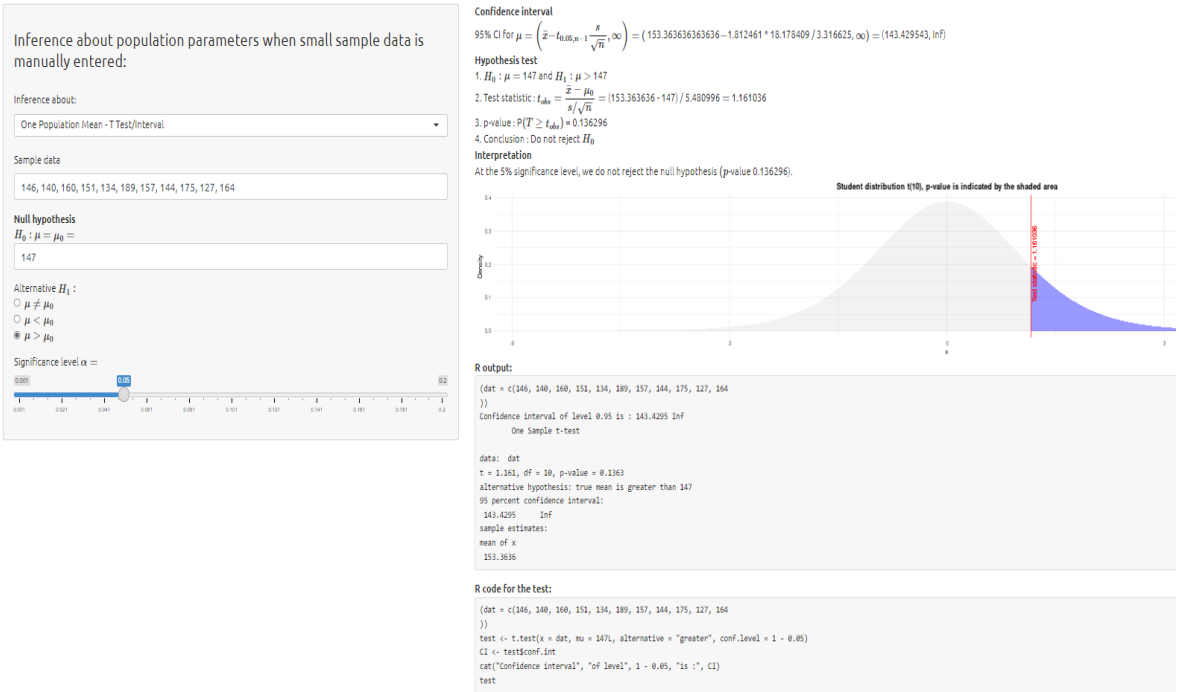

Figure 14: Right-tailed confidence interval estimation of μ and hypothesis testing for $H_0 : \mu = 147$ versus $H_1 : \mu > 147$.

to select the radio-button $\mu > \mu_0$ to conduct the analysis as shown in Figure 14. This user-friendly approach enables seamless exploration of various hypothesis testing problems within the same dataset and null hypothesis, fostering a dynamic and interactive user experience.

The comprehensive presentation of the hypothesis testing procedure, R code, R output and graph illustration ensures that users receive detailed insights into the statistical analysis performed, enhancing their understanding and facilitating effective interpretation of the results.

5. UNIVARIATE DATA ANALYSIS

5.1. Data Import and Data Entry

Data input in the initial two menus is in the form of vectors. However, in R, data tables are typically stored using data frames. A data frame is essentially a compilation of vectors, all of which share the same length, resembling a numerical matrix, except that different columns can different data types. To facilitate the process of data input or import, the application's 'Data Import' menu is designed for this purpose. It serves the dual function of importing data file on a local disk and enabling users to manually enter data. It is imperative, as per the principles outlined by Wickham (Wickham 2017), that the data adheres to the concept of tidiness: each variable must have its own column, each observation must have its own row, and each value must have its own cell.

Data can be seamlessly imported into the application by selecting the 'Upload dataset' checkbox, as illustrated in Figure 15. Once activated, users have the option to upload a CSV (comma-separated-values) file to the application. It is important to note that if the uploaded dataset lacks a header or a row of column names, the checkbox 'The first row is header' should be deselected. Currently, the application exclusively supports CSV files, a sufficiently versatile format for introductory statistics courses. It should be noted that the application does not automatically generate the local path of the uploaded file. For instance, the R code generated by the app for uploading the 'mtcars' file from a local disk is `Data <- read.csv("mtcars.csv", TRUE)`. However, when employing an R console for file upload, it is imperative to specify the complete file path. For example (for windows users), if the path of the file is C:\Users\Downloads\mtcars.csv, the corresponding R code to be entered on an R console should be `Data=read.csv("C:/Users/Downloads/mtcars.csv", header=TRUE)`. It is important to note that R uses forward slashes in file paths, even for Windows users. This careful consideration ensures accurate and error-free file uploading through the R console.

Users have the option to manually input data by selecting the 'Enter data manually' button, as illustrated in Figure 15. Upon selection, the 'DataEntry' tab, depicted in Figure 16, will be presented.

Manual data entry may be time-consuming for large datasets, hence the application supports a maximum of five columns or variables. Initially, the interface provides ten rows, with the flexibility for users to add or delete rows akin to Excel functionality.

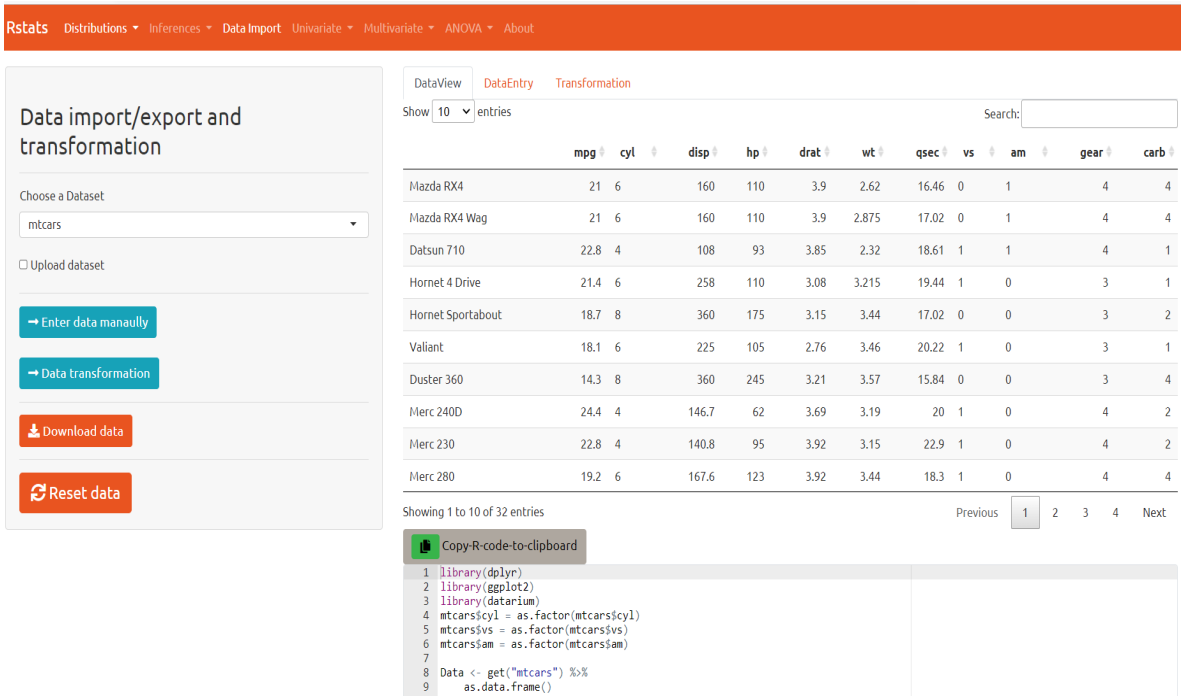

Figure 15: Menu *Data Import* of the web app Rstats.

To begin, users must specify the number of variables in the designated input box labeled 'Number of variables (columns), up to 5 only' restricted to a maximum of five variables. If only one variable is entered, a checkbox labeled 'Frequency Table (frequencies must be integers)' becomes visible. This option allows users to input data as a frequency table, with the first column representing distinct values and the second column representing frequencies, as demonstrated in Figure 16. Upon configuring the desired number of columns, users must confirm their selection by clicking the 'Set columns' button. Once set, the number of variables cannot be altered until the 'Reset columns' button is clicked. This feature prevents accidental modifications to the spreadsheet, located in the top-right panel. Furthermore, to preserve entered data, users must click the 'Submit' button. This action ensures that the data is saved and available to use for the remaining menus within the application .

The web app offers fundamental data transformation capabilities. Users can access these features by selecting the 'Data Transformation' button within the 'Data Import' menu or by clicking on the 'Transformation' tab. Key transformations include power and logarithm operations on numerical variables, as well as the conversion between numerical variables and factors (categorical variables). When converting a numerical variable into a factor, users have two options. They can employ the direct method, which converts all numerical values into factor levels, or opt for the binning method. The binning method utilizes the *cut* function in R to segment the numerical variable's range into intervals. Subsequently, the values in the dataset are categorized based on the interval to which they belong.

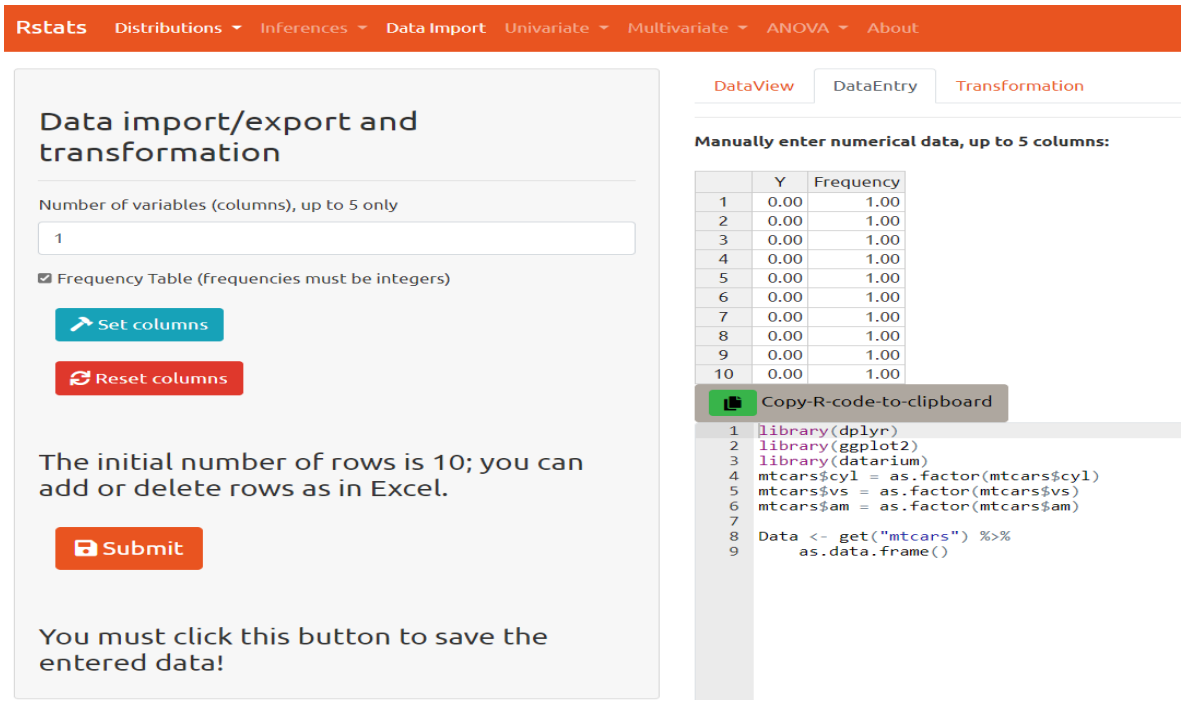

Figure 16: Manually data entry with the web app Rstats.

5.2. Univariate Numerical Data Analysis

Clicking on the 'Univariate' menu reveals two sub-menus: 'Numerical Data Analysis' and 'Categorical Data Analysis'. These options facilitate the analysis of univariate data by providing summary statistics and statistical inference. To illustrate the functionality of univariate data analysis, let's consider the built-in R dataset *mtcars* Henderson (1981).

Figure 17 depicts the interface for univariate numerical data analysis, accessible upon selecting 'Numerical Data Analysis' from the 'Univariate' menu. In this interface, users must first choose a numerical variable from the dataset. Subsequently, they can perform various descriptive statistics and statistical inference, including:

1. Numerical summary
2. Plots and Normality test
3. Inference about population mean(s)
4. Inference about population variance(s)

By default, the radio button 'Numerical Summary' is selected. Users can then see different summary statistics available in 'Summary' from the dropdown list titled 'Summary Statistics' obtained using the *summary* function in R. Further more, users can choose 'Mean', 'Variance', and 'Standard Deviation' from the dropdown list to get the corresponding statistics. Moreover, users can calculate these summary statistics for multiple groups by selecting a grouping

(categorical) variable from the last dropdown list shown in Figure 17. This feature enables comparative analysis across different categories within the dataset.

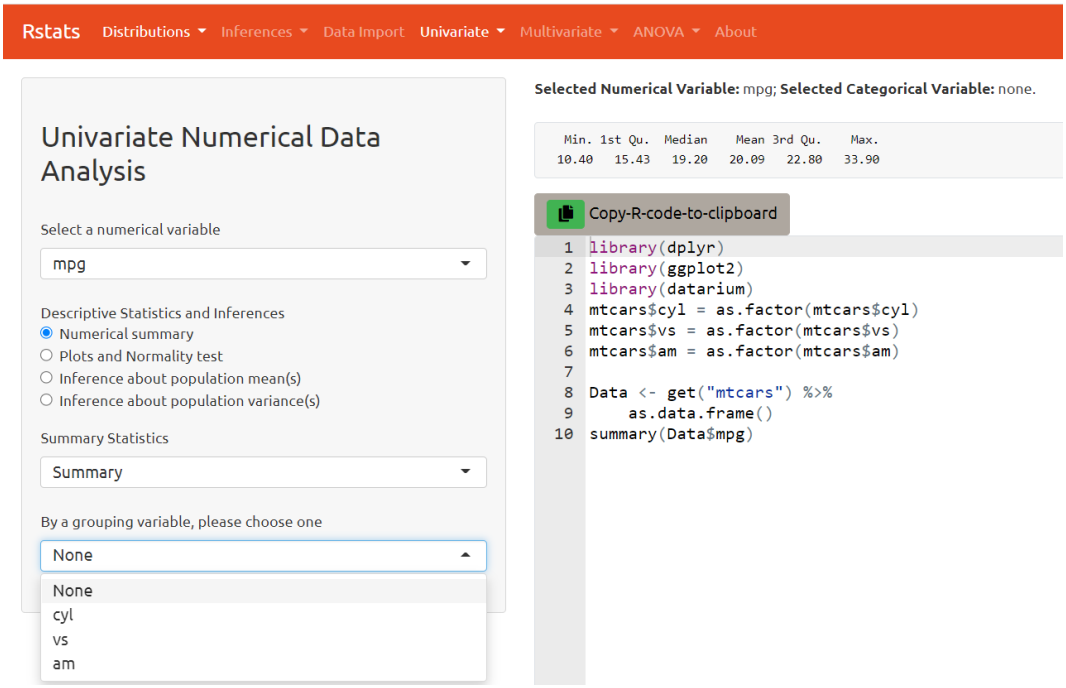

Figure 17: Univariate numerical data analysis interface of the web app Rstats.

When users select the radio button 'Plots and Normality test', they gain access to various visualization and normality assessment tools. From the dropdown list titled 'Plot Type', users can opt for any of the following options:

- Histogram
- Boxplot
- Normal Quantile Plot
- Shapiro-Wilk Test of Normality

For example, in Figure 18, the interface displays a histogram of the variable *mpg*, with adjustable sliders allowing users to modify the number of bins for a different visualization. Again these analyses can be conducted for multiple groups of data based on a chosen grouping variable.

Statistical inference regarding population means and variances can be performed for both one-sample and two-sample analyses. By default, the analysis 'Type' is set to 'One Sample', as depicted in Figure 19. In this setup, users begin by entering the mean value under the null hypothesis μ_0, with the default value being 0. They also select an alternative hypothesis from $\mu \neq \mu_0$, $\mu < \mu_0$ and $\mu > \mu_0$. Additionally, users can adjust the significance level α using a slider to achieve the desired confidence level $1 - \alpha$ for the confidence interval of μ.

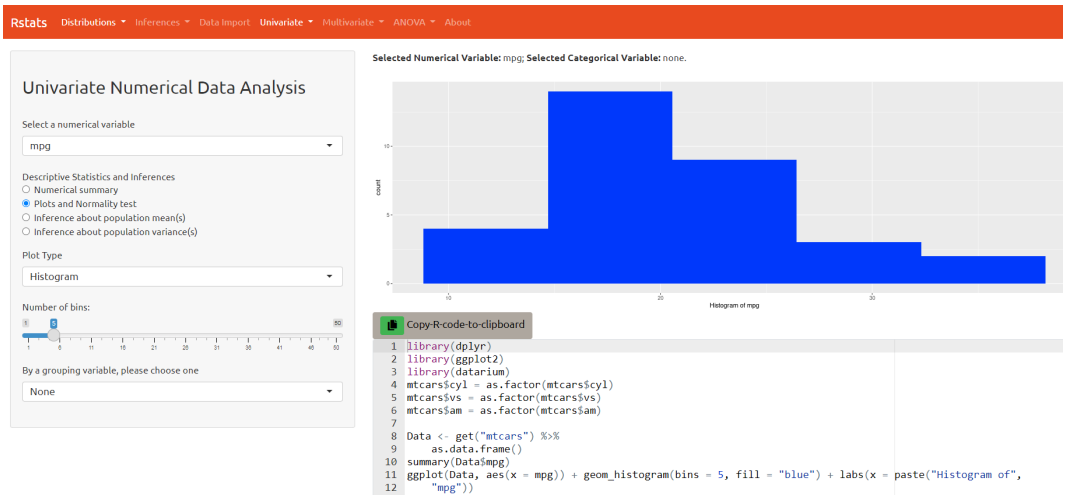

Figure 18: Plots and normality test of a numerical variable by the web app Rstats.

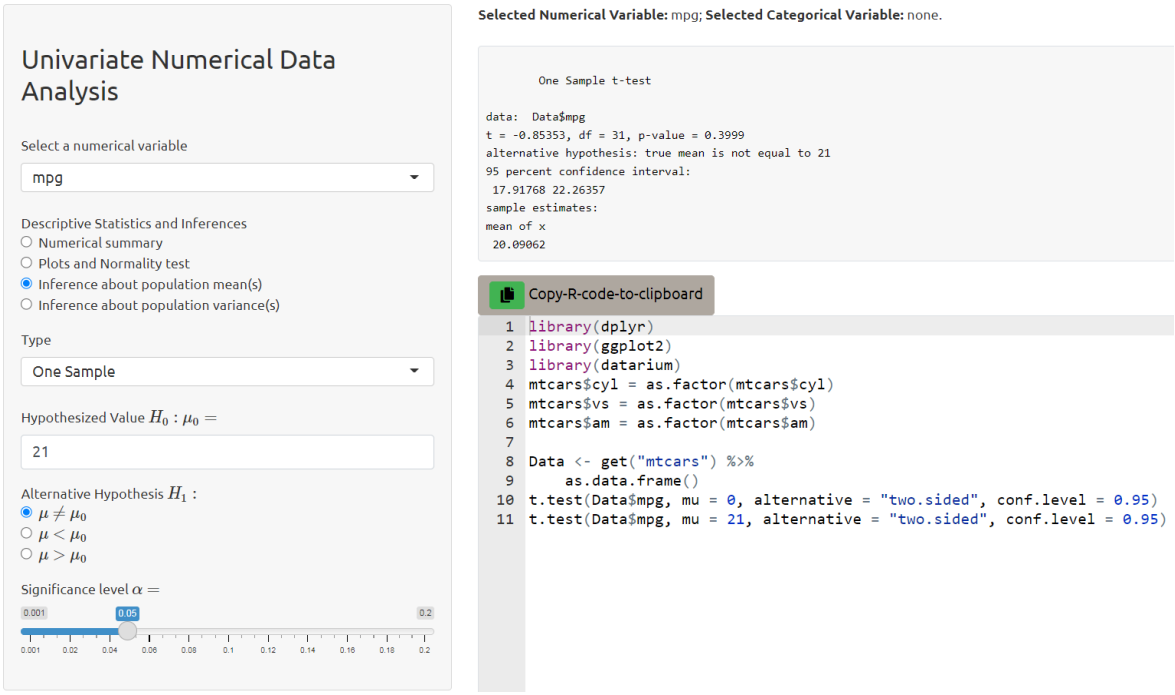

Figure 19: One-sample inference about population means by the web app Rstats.

In the case of a two-sample statistical inference, users must select a grouping (categorical) variable, and then choose two levels of this variable as the two samples. For instance, Figure 20 compares the means of *mpg* for 4-cylinder cars and 6-cylinder cars. By default, the value $\mu_1 - \mu_2$ under the null hypothesis is set to 0, indicating no difference between the two population means. Users then select an alternative hypothesis from $\mu_1 - \mu_2 \neq \mu_0$, $\mu_1 - \mu_2 < \mu_0$ and $\mu_1 - \mu_2 > \mu_0$. Additionally, users can specify whether the equal variance assumption is valid and whether the two-sample t-test is paired, as illustrated in Figure 20. This flexibility allows for comprehensive analysis and comparison of population means and variances across different

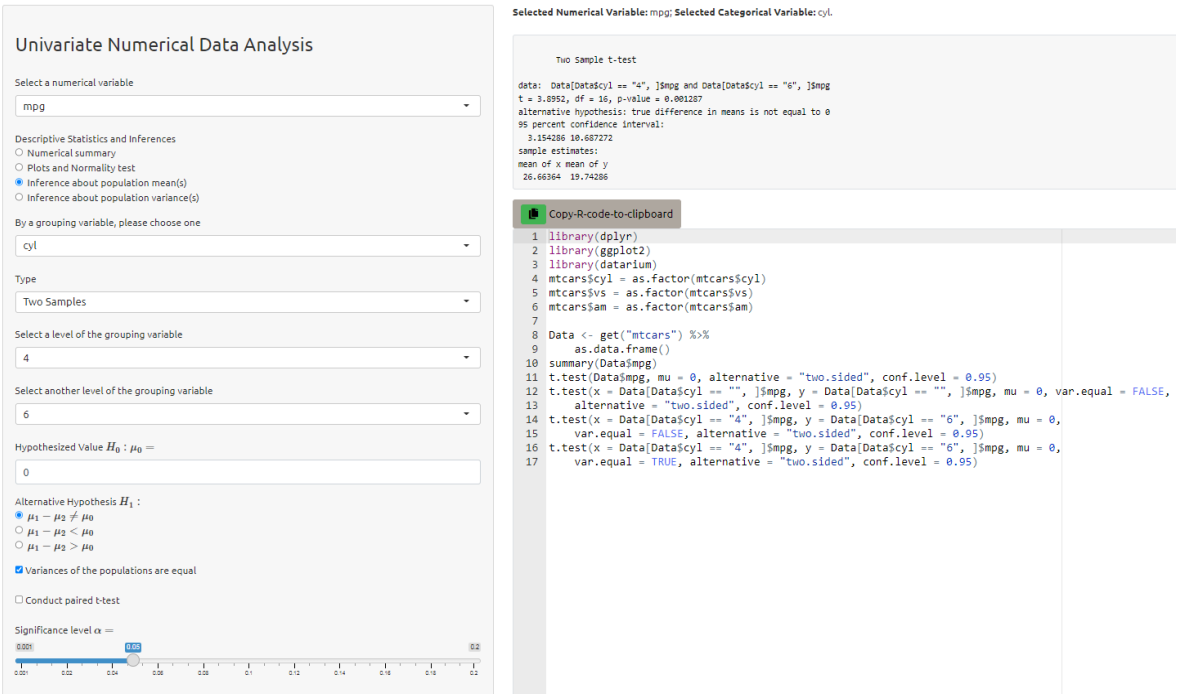

Figure 20: Two-sample inference about population means by the web app Rstats.

groups within the dataset.

Statistical inference concerning population variances are not explicitly demonstrated in this paper. However, it's worth noting that the web app's user-friendly interface makes conducting such analyses straightforward for users. With its intuitive design and accessible features, users can effortlessly perform both one-sample and two-sample statistical inference about population variances.

5.3. Univariate Categorical Data Analysis

When analyzing a dataset with categorical variables, users can access the 'Categorical Data Analysis' sub-menu to examine descriptive statistics, plots, and statistical inference about population proportions. Figure 21 displays the frequency table of the categorical variable *cyl* from the dataset *mtcars* after selecting the 'Summary statistics and plots' radio button. Additionally, users can generate bar charts and pie charts by choosing 'Bar chart' and 'Pie chart', respectively, from the drop-down list under 'Summary statistics and plots'.

Once users have reviewed the summary statistics and plots of the data, they can proceed to examine statistical inference about population proportions. This can be done by selecting the 'Inference about population proportion(s)' radio button. By default, the inference problem is set to 'One population proportion' in the drop-down list under 'Inference about'. Other available inference problems include 'Two population proportions', 'Three or more population proportions' and 'Goodness-of-fit' test.

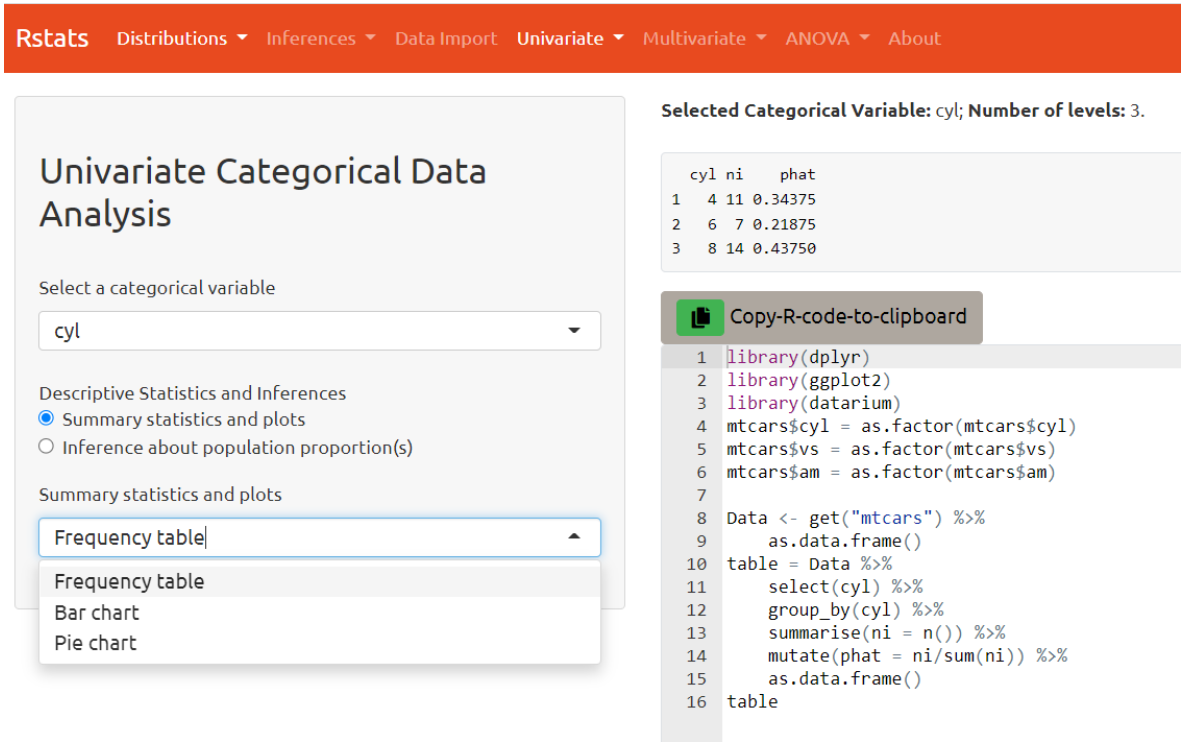

Figure 21: Descriptive statistics and plots of categorical data by the web app Rstats.

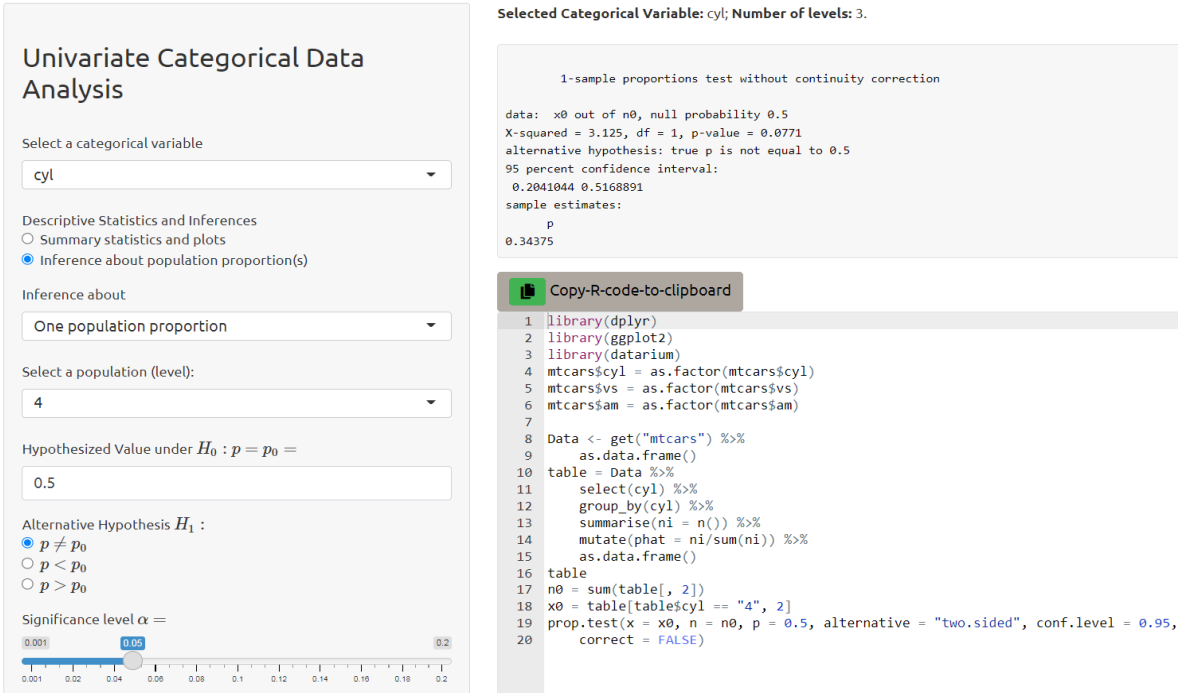

Figure 22: Inference about one population proportions by the web app Rstats.

In Figure 22, the process of hypothesis testing and confidence interval estimation for the proportion p of 4-cylinder cars is depicted. Users are required to input the value of p under the null hypothesis and select an alternative, similar to the inference about population means discussed in the preceding subsection. Additionally, the significance level slider is utilized to adjust the confidence level $1 - \alpha$.

To perform inference regarding the equality between two population proportions using the 'Two population proportions' option in the 'Inference about' drop-down list, users must select two levels of a chosen categorical variable. Similarly, for testing the equality of three or more population proportions, users need to specify the desired levels of the categorical variable. In the case of the goodness-of-fit test, users are required to input probabilities for all categories under the null hypothesis. While no screenshots are included for these three inference problems in the paper, the web app interface provides a user-friendly guide for conducting these types of data analyses.

6. MULTIVARIATE DATA ANALYSIS

Under the 'Multivariate' menu, users have access to a range of multivariate data analysis tools. These include: Simple Linear Regression analysis (SLR Model), Multiple Linear Regression analysis (MLR Model) incorporating logistic regression models, and Contingency Analysis designed for two categorical variables. Within this section, we consistently employ the *mtcars* dataset to illustrate the practical application of these analyses within the web app.

6.1. Simple Linear Regression Analysis

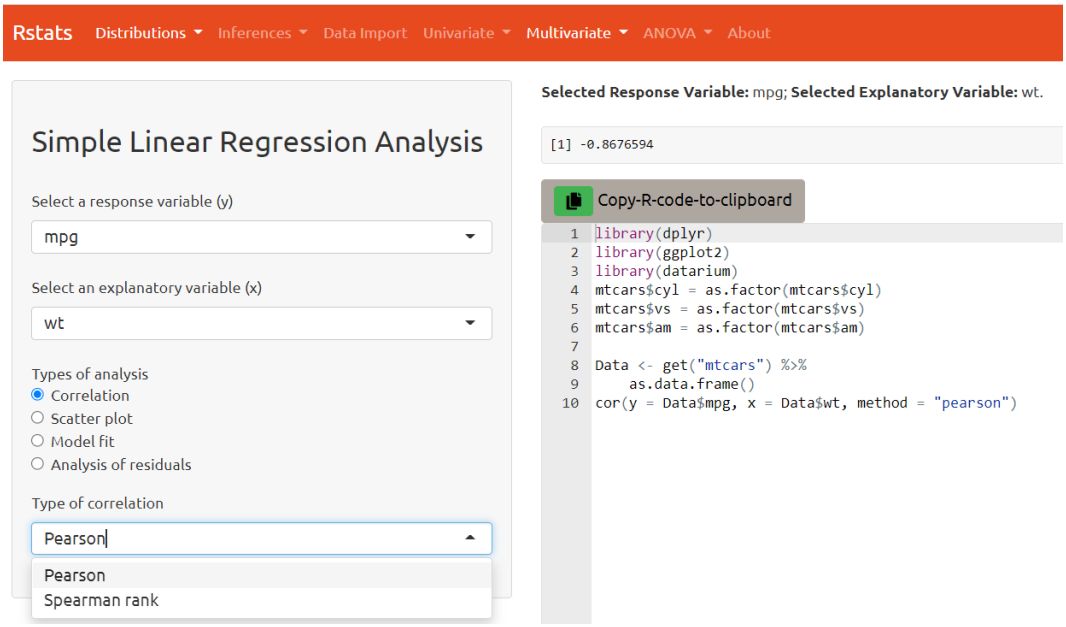

Figure 23: Pearson linear correlation between two numerical variables.

To initiate the analysis, users must first select two numerical variables: a response variable y and an explanatory variable x. In Figure 23, for instance, the response variable *mpg* and the explanatory variable *wt* are chosen. Following variable selection, users can engage in four types of analysis: 'Correlation', 'Scatter plot', 'Model fit', and 'Analysis of residuals'. Figure 23 illustrates the Pearson linear correlation between *mpg* and *wt*. Moreover, users have the option to compute the Spearman rank correlation by selecting the second item from the 'Type of correlation' drop-down list.

Figure 24 depicts the scatter plot of the two selected variables. If the checkbox labeled 'Scatter plot by a grouping variable' is checked, a categorical variable must be chosen from a drop-down list located under the checkbox (not depicted in this document). Upon selection, the points in the scatter plot will be colored according to the categorical variable, while the fitted regression line remains unchanged. In other words, the simple linear regression model fit applies to the entire dataset rather than distinct subsets based on the chosen categorical variable.

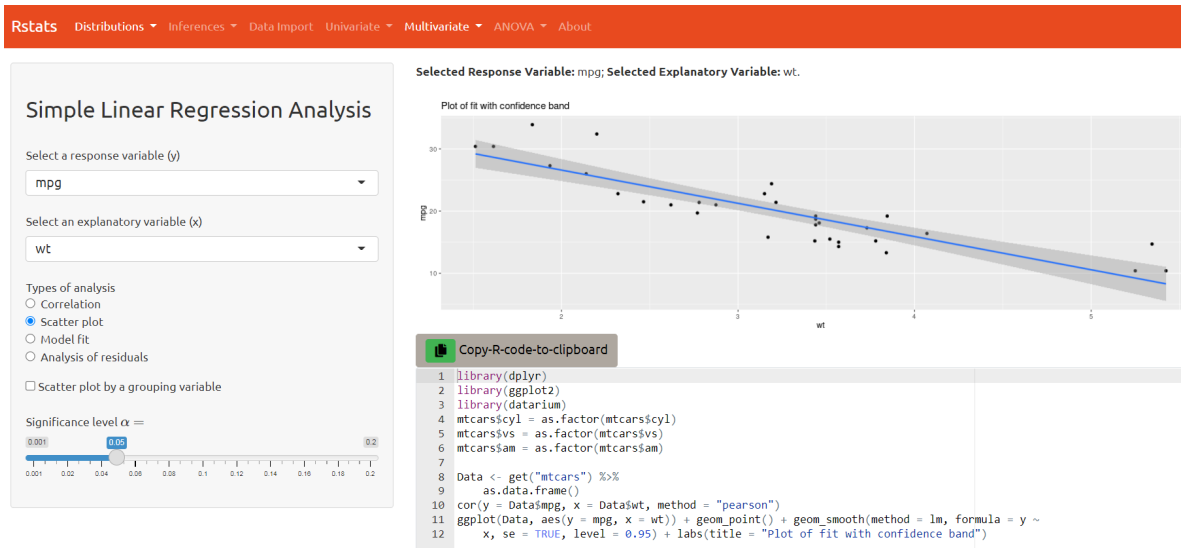

Figure 24: Scatter plot with fitted regression line between two numerical variables.

Figure 25 presents the summary of the simple linear regression model fit of the data generated by the R function *lm*. Within the drop-down list labeled 'SLR analysis results', users can opt for several analyses:

- Selecting 'CI of the parameters' yields confidence intervals of the regression parameters, with the confidence level controlled by the significance level.
- Choosing 'ANOVA table' provides the ANOVA table of the model fit.
- Opting for 'Prediction of response for future observations' generates confidence intervals and prediction confidence intervals for specified future values of the explanatory variable.

Figure 26 illustrates the process of analyzing residuals following the fitting of a simple linear

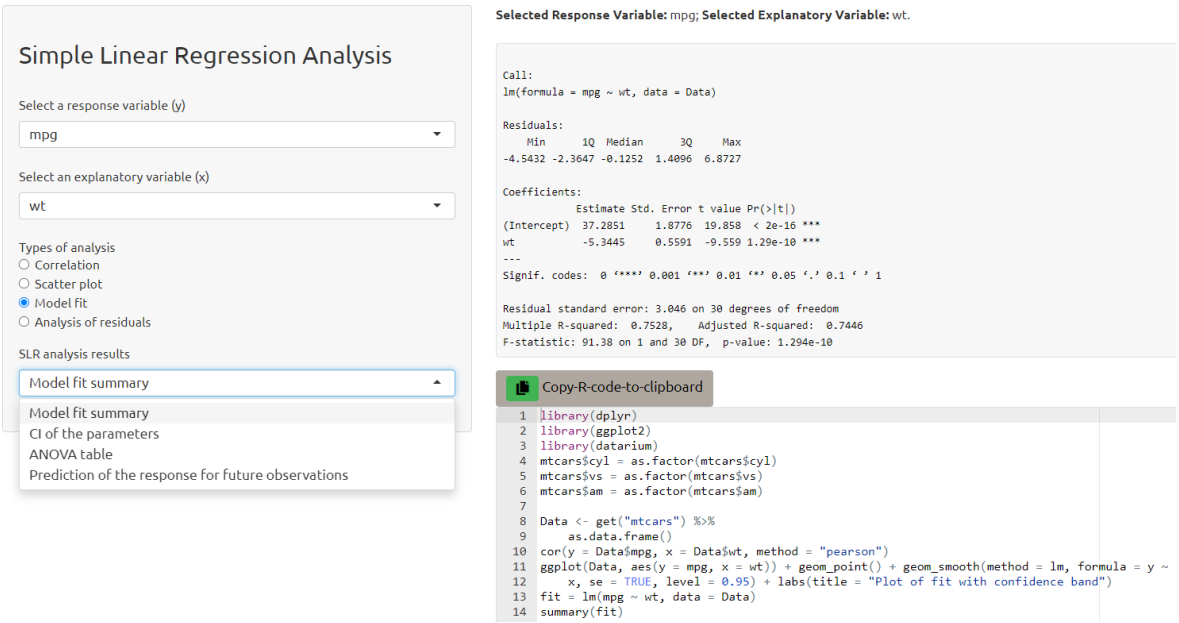

Figure 25: Simple linear regression model fit for two numerical variables.

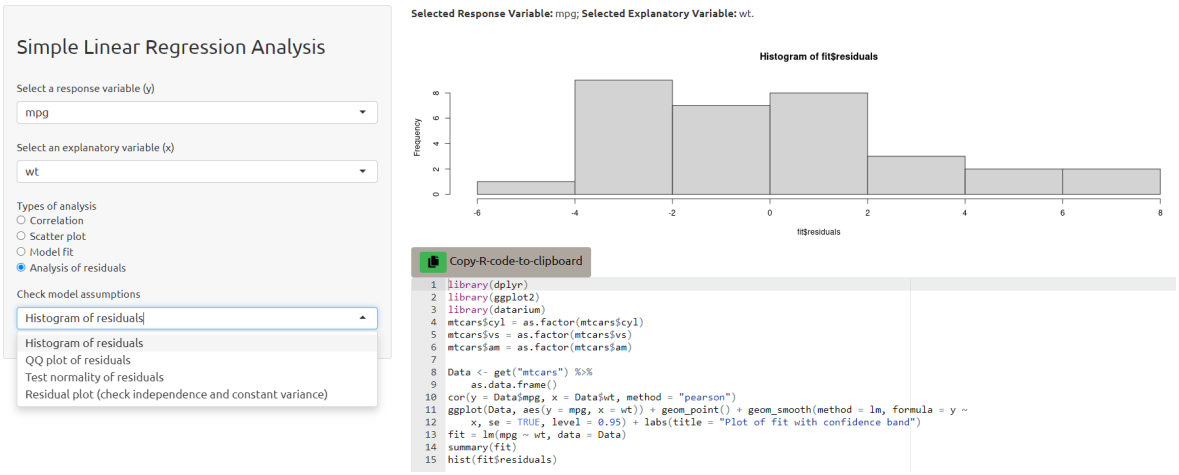

Figure 26: Analysis of residuals obtained by a simple linear regression model fit.

regression model using the web application. Within the application, users are presented with a dropdown menu titled 'Check model assumptions,' which contains four options:

- 'Histogram of residuals': This option generates a histogram that visualizes the distribution of residuals, aiding in assessing their normality.
- 'QQ plot of residuals': This option produces a QQ plot (Quantile-Quantile plot) of residuals against a theoretical normal distribution, facilitating the evaluation of normality assumptions.
- 'Test normality of residuals': This option performs statistical tests to assess the normal-

ity of residuals, providing quantitative measures of deviation from normality assumptions.

- 'Residual plot (check independence and constant variance)': This option generates a residual plot, enabling users to evaluate whether the assumptions of independence and constant variance hold for the residuals.

6.2. Multiple Linear Regression Analysis

The sub-menu 'MLR' provides access to various types of regression models, including polynomial regression models, general multiple linear regression models, and logistic regression models. However, it's worth noting that the nonlinear logistic regression models, which deviate from the linear regression analysis framework, are not separately listed in the menu for the sake of user convenience during analysis.

For polynomial regression models, users have access to four types of analysis: Correlations, Scatter plot, Model fit, and Analysis of residuals. In Figure 27, the scatter plot depicts the relationship between *mpg* and *hp* with a degree 2 polynomial regression fit. Users can adjust the significance level slider to control the confidence level of the confidence band.

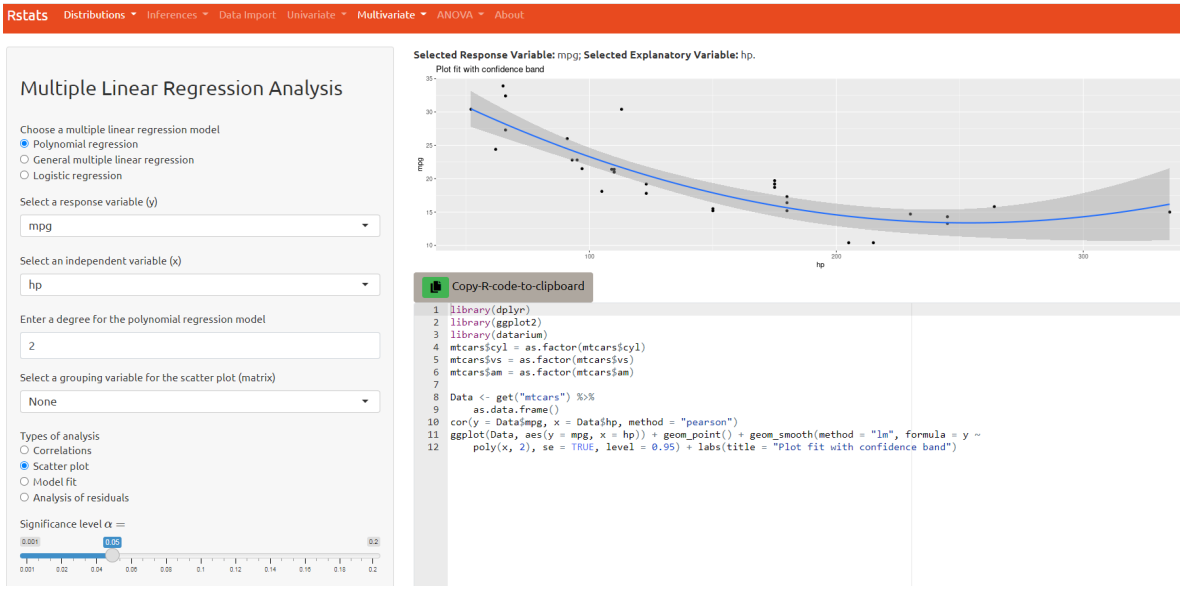

Figure 27: Scatter plot with a degree 2 polynomial regression model fit between two numerical variables.

Within the 'Model fit' section, users can choose from the following options:

- 'Model fit': Provides an overview of the fitted model.
- 'CI of the parameters': Displays confidence intervals for the regression parameters.
- 'ANOVA table': Presents the ANOVA table for assessing the significance of the model.

- 'Prediction of response for future observations': Enables users to obtain confidence intervals and prediction confidence intervals for specified future values of the explanatory variable, akin to the menu options available in the 'SLR' model.

Once again, the options for Analysis of residuals mirror those available for fitting a simple linear regression model.

Figure 28 presents a model fit for general multiple linear regression models, while Figure 29 illustrates a scatter plot depicting the relationship between a binary response and a numerical exploratory variable with the logistic regression fit. Each of these analyses comprises correlation (matrix), scatter plot (matrix), model fit, and analysis of residuals, with the exception of logistic regression analysis, which lacks the functionality of analyzing residuals.

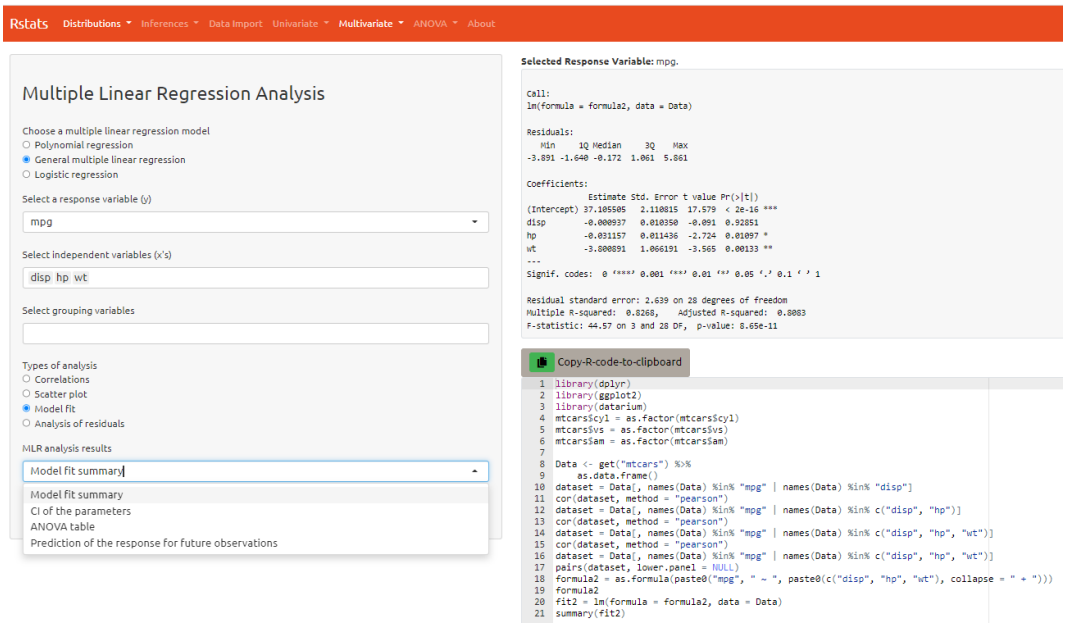

Figure 28: A model fit for general multiple linear regression models.

6.3. Contingency Analysis

When the 'Contingency Analysis' sub-menu, found under the 'Multivariate' menu, is selected, users gain the ability to construct a contingency table and perform tests of independence on two categorical variables. By default, upon selecting two categorical variables, a cross-table is automatically generated. Figure 30 illustrates the Chi-square test of independence conducted on the two categorical variables *cyl* and *vs* using the generated contingency table after the 'Statistical inference' radio button is selected.

7. Analysis of Variance

The web application empowers users to perform both one-way (factor) ANOVA and two-way

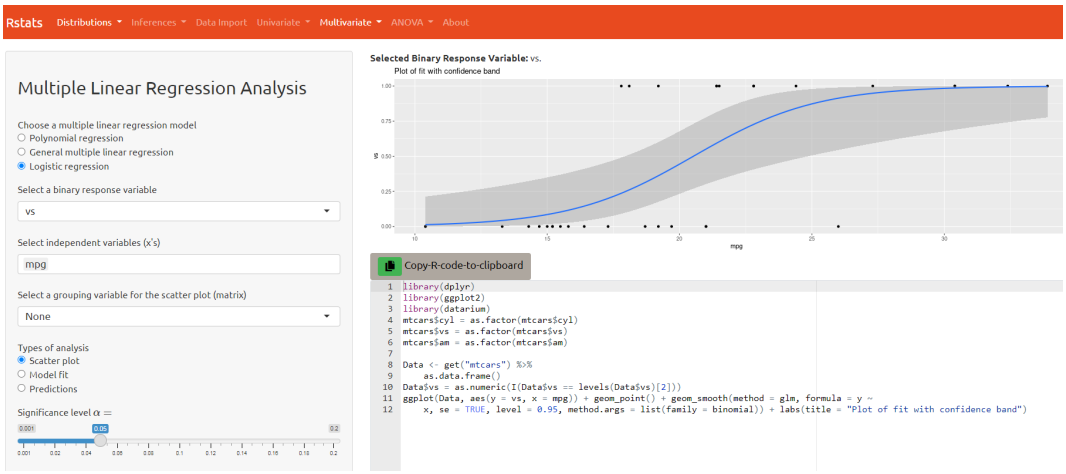

Figure 29: Scatter plot of a numerical explanatory variable and a binary response variable with a logistic regression model fit.

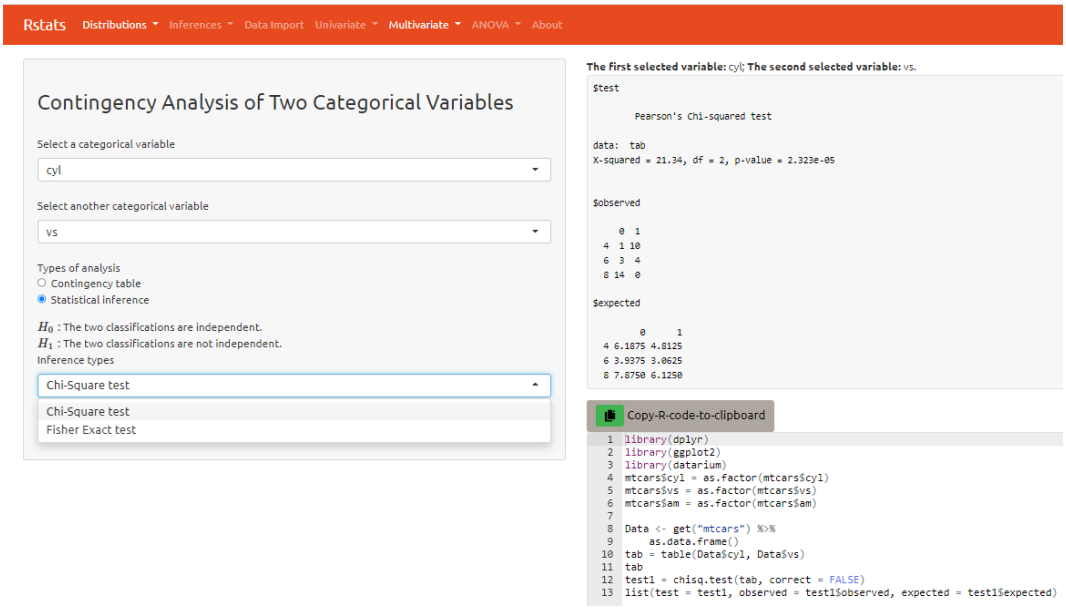

Figure 30: Chi-square test of independence of two categorical variables.

(factor) ANOVA analyses. To begin, users select a numerical response variable and then proceed to choose one categorical variable for one-way ANOVA or two categorical variables for two-way ANOVA. Each analysis type offers a range of comprehensive features that can be accessed by toggling the following radio buttons.

- Descriptive statistics and plots
- Model fit
- Analysis of residuals

- Post hoc multiple comparisons.

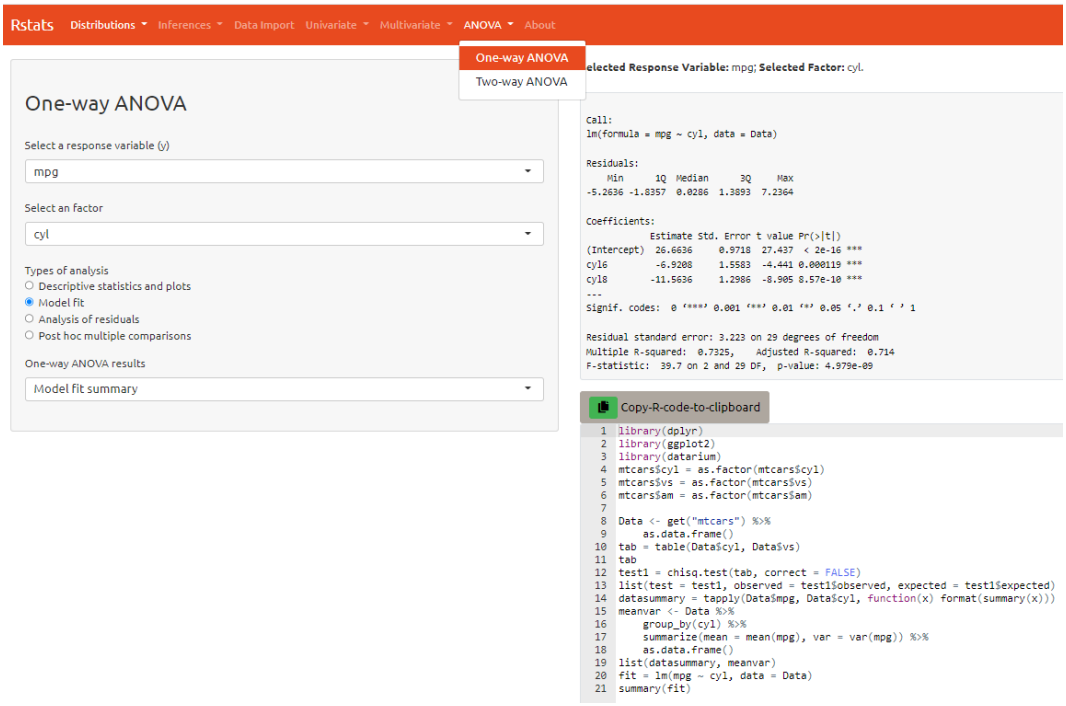

Figure 31: Model fit for one-way Analysis of Variance.

In Figure 31, the model fit of a one-way ANOVA analysis is demonstrated, utilizing the numerical response variable `mpg` and the categorical variable `cyl`. Additionally, users can access the ANOVA table by selecting either ‘One-way ANOVA results’ for one-way ANOVA or ‘Two-way ANOVA results‘ for two-way ANOVA from the provided drop-down list.

Descriptive statistics encompass group means and medians based on the factor(s), while plots include histograms and box-plots categorized by the factor(s), along with an interaction plot specifically designed for the two-way ANOVA. Analysis of residuals involves examining histograms and QQ-plots of the residuals, conducting hypothesis tests on the normality of residuals, generating residual plots, and assessing the assumption of constant variance within the statistical model.

For post hoc multiple comparisons, various methods are available to determine which treatments exhibit significant differences from others. These methods include the Bonferroni test, the Holm procedure, and Tukey’s test.

8. Summary

The web application Rstats, as described in this paper, is tailored specifically for introductory statistics courses aimed at non-computing majors. Therefore, the functionality of the app is intentionally limited. For instance, while it lacks advanced data cleaning capabilities like sub-setting, it does support several basic data transformations such as power and log

transformations for numerical variables, as well as conversions between numerical and categorical variables. However, missing values are automatically removed during the calculation of descriptive statistics.

The primary objective of the web app is to cultivate students' interest in programming through its reproducibility feature. Based on an Rstats web app user experience survey of 38 students in Spring 2022, 39 percent of students became more interested in coding. However, not all outputs in the app are reproducible. For instance, the R code used to generate illustrations for probability calculations and statistical inference described in section 3 and 4, complex and lengthy code relying on the *ggplot2* and *plotly* packages, will not be produced.

When comparing the Rstats web app to commercial point-and-click statistical software such as SPSS, Minitab, and Stata, a notable distinction lies in the transparency and accessibility of the underlying code. While these commercial tools offer user-friendly interfaces for conducting statistical analyses without the need for programming knowledge, they typically do not provide users with the generated code used to perform the analyses. The Rstats web app allows users to generate R code for their statistical analyses, promoting transparency and reproducibility while serving as an educational tool for learning R programming alongside statistical analysis. Unlike costly commercial software, the openly accessible nature of the Rstats web app encourages broader adoption among students and educators. Its transparent approach fosters a deeper understanding of statistical methods and programming concepts, aligning with the educational objectives of introductory statistics courses for non-computing majors.

The source code for the web app is openly accessible on GitHub at https://github.com/esumath/Rstats, allowing users to download and deploy it to a server platform such as www.shinyapps.io. Although the app is functional, there is room for improvement. For instance, some students may desire a feature that generates a report containing the R code used in their data analysis upon completion. Therefore, suggestions and contributions aimed at enhancing the web app are highly encouraged and welcomed.

References

Arnholt, A.T. (2019). "Using a Shiny app to teach the concept of power," *Teaching Statistics*,41(3), 79-84.

Bainomugisha, E., Carreton, A.L., Cutsem, T.V., Mostinckx, S. and De Meuter, W. (2013). "A survey on reactive programming". *ACM Computing Surveys*, 45(4), 52:1–52:34. DOI: https://doi.org/10.1145/2501654.2501666

Beeley, C. (2013), *Web Application Development with R Using Shiny*, Packt Publishing, Birmingham, UK.

Cheng, J. and Sievert, C. (2021). shinymeta: Export Domain Logic from Shiny using Meta-Programming. R package version 0.2.0.3. https://cran.r-project.org/web/packages/shinymeta/index.html

Doi, J., Potter, G., Wong, J., Alcaraz, I. and Chi, P. (2016), "Web Application Teaching Tools for Statistics Using R and Shiny," *Technology Innovations in Statistics Education*, 9(1), 1-32.

Freire, S. M. (2019). "Using Shiny to illustrate the probability density function concept," *Teaching Statistics*, 41(1), 30-35.

Kwon, D., Reddy, R.R.S. and Reis, I.M. (2021), "ABCMETAapp: R shiny application for simulation-based estimation of mean and standard deviation for meta-analysis via approximate Bayesian computation," *Research Synthesis Methods*, 1-7, DOI: https://doi.org/10.1002/jrsm.1505

R Core Team (2024), "R: A language and environment for statistical computing," R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org.

Ferguson, R., Leidig, P. and Reynolds, J. (2015), "Including a Programming Course in General Education: Are We Doing Enough?", *Information Systems Education Journal*, 13, 34-42.

Henderson, H. V., and Velleman, P. F. (1981), "Building multiple regression models interactively", *Biometrics*, 37(2), 391-411. https://doi.org/10.2307/2530428

Kandlikar, G.S., Gold, Z.J., Cowen, M.C., Meyer, R.S., Freise, A.C., Kraft, N.J.B., Moberg-Parker, J., Sprague, J,, Kushner, D.J. and Curd, E.E. (2018), "ranacapa: An R package and Shiny web app to explore environmental DNA data with exploratory statistics and interactive visualizations", *F1000Res*, DOI: 10.12688/f1000research.16680.1, PMID: 30613396; PMCID: PMC6305237.

Peng, R. (2016). *R Programming for Data Science*, lulu.com, 5th ed.

Rushkoff, D. (2012), "Code Literacy: A 21st Century Requirement," Edutopia, https://www.edutopia.org/blog/code-literacy-21st-century-requirement-douglas-rushkoff.

Stratton, C., Green, J.L. and Hoegh, A. (2021), "Not just normal: Exploring power with Shiny apps", *Technology Innovations in Statistics Education*, 13(1), DOI: https://doi.org/10.5070/T513146468.

Wickham, H. (2015), "Graphics & Computing Student Paper Winners Jsm 2015". https://github.com/hadley/15-student-papers.

Wickham, H. (2015), *R Packages: Organize, Test, Document, and Share Your Code*, O'Reilly Media, Sebastopol, CA.

Wickham, H. and Grolemund, G. (2017). *R for Data Science*, O'Reilly Media.

Wickham, H. (2021). *Mastering Shiny: Build Interactive Apps, Reports, and Dashboards Powered by R*, 1st ed, O'Reilly Media.

Wickham, H., Francois, R., Henry, L. and Muller, K. (2022), "dplyr: A Grammar of Data Manipulation," https://cran.r-project.org/web/packages/dplyr.

Williams, I. J. and Williams, K. K. (2017). "Using an R shiny to enhance the learning experience of confidence intervals," *Teaching Statistics*, 40(1), 24-28.