ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14964344

Enhancing Regression Diagnostics: Automated Residual Analysis Using Computer Vision and Statistical Insights

Niraj Patel[1]

[1]ST. Clair College

Publication Date: 2025/03/05

Abstract: Residual analysis plays a pivotal role in validating regression models by identifying potential issues such as heteroscedasticity, non-linearity, and model misspecification. This study introduces a novel automated framework for residual diagnostics, integrating computer vision techniques with statistical inference. The proposed system evaluates residual plots, detects irregularities, and performs hypothesis testing to ensure model robustness. By combining image recog- nition algorithms with a user-friendly Shiny application, the approach eliminates subjective biases inherent in manual plot evaluation. The resulting tool enhances the scalability and reliability of regression diagnostics, offering data scientists a powerful resource for building accurate and interpretable models.

How to Cite: Niraj Patel (2025). Enhancing Regression Diagnostics: Automated Residual Analysis Using Computer Vision and Statistical Insights. *International Journal of Innovative Science and Research Technology*, 10(2), 1421-1430. https://doi.org/10.5281/zenodo.14964344

I. INTRODUCTION

Regression analysis is a widely used statistical modeling technique for data in many fields. There is a vast array of software for conducting regression modeling and generating diagnostics. The package lmtest [1] provides a suite of conventional tests. The stats package [2] offers standard diagnostic plots such as residuals vs. fitted values, quantile- quantile (Q-Q) plots, and residuals vs. leverage plots. Packages like j tools[3], olsrr[4], rockchalk[5], and gg Residpane l [6] provide similar graphical diagnostics, of- ten with alternative aesthetics or interactive features. All of these tools deliver the types of diagnostic plots outlined in the classical text by [7]. The Eco stats package [8] incorporate-rates simulation envelopes into residual plots, while Dharma [9] compares empirical quantiles (0.25, 0.5, and 0.75) of scaled residuals to their theoretical counterparts. Dharma is particularly focused on detecting model violations such as heteroscedasticity, incorrect functional forms, and issues specific to generalized linear and mixed-effect models, like over/under-dispersion. It also includes conventional test annotations to help avoid misinterpretation.

However, relying solely on subjective assessments of these plots can lead to issues such as over-interpreting random patterns as model violations. [10] demonstrated that visual methods using the lineup protocol [11] for assessing reside- duals are more useful, and also perform more practically than conventional tests due to their reduced sensitivity to minor departures. Packages such as null Abor [12], HLMdiag[13], and regressinator [14], enable users to compare observed residual plots with plots of samples from null distributions, helping to quantify the significance of any detected patterns.

However, as discussed in [15], the lineup protocol has significant limitations in large-scale applications due to dependence on human labor. Thus, a computer vision model was developed with an associated statistical testing procedure to automate the assessment of residual plots. This model takes a residual plot and a vector of auxiliary variables (such as the number of observations) as inputs and outputs the predicted visual signal strength (VSS). This strength estimates the distance between the residual distribution of the fitted regression model and the reference distribution assumed under correct model specification.

To make the statistical testing procedure and trained computer vision model widely accessible, we developed the R package autovi, and a web interface, autovi.web to make it easy for users to automatically read their residual plots with the trained computer vision model.

The remainder of this paper is structured as follows: Section II introduces the definition and computation of visual signal strength. Section III provides a detailed documentation of the autovi package, including its usage and infrastructure. Section IV focuses on the autovi.web interface, describing its design and usage, along with illustrative examples. Finally, Section V presents the main conclusions of this work.

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14964344

II. DEFINITION AND COMPUTATION OF VISUAL SIGNAL STRENGTH

To train a computer vision model, a measure of the visible pattern in a plot is needed. We call this the visual signal strength (VSS), which measures how prominently a specific set of visual patterns appears in an image. This can be computed for a training set of data, and plots, where the generating distributions are specified.

In the context of regression model diagnostics, VSS de- scribes the clarity of visual patterns on a diagnostic plot that may indicate model violations. Violations can be categorized as weak, moderate, or strong, but here we treat it as a continuous positive real variable. Importantly, its interpretation depends on how it is linked to a function of the data or the underlying data generating process Consequently, the calculation of VSS can be vary across different model classes or within the same model, depending on the generating function.

VSS is an estimate of the distance between the residual distribution of a fitted classical normal linear regression model and a reference distribution; more details can be found in [15]. The distance measure is based on the Kullback-Leibler (KL) divergence:

$$D = \log(1 + D_{KL}),$$

Where D_{KL} is given by:

$$D_{KL} = \int \log \frac{p(\mathbf{e})}{q(\mathbf{e})} p(\mathbf{e}) d\mathbf{e},$$

Where, p(.) and q(.) are the probability density functions of the reference residual distribution P and the true residual distribution Q, respectively.

This distance measure depends on knowledge of the true residual distribution, which is unknown in practice. To compute DKL for the training samples, Equation 1 takes different forms depending on the specific model violations. For instance, where necessary higher-order predictors, Z, and their corresponding parameter, βZ, are omitted from the fitted linear model, the distance measure can be expanded as follows:

$$D_{KL} = \frac{1}{2} \boldsymbol{\mu}_z^\top (\mathrm{diag}(\boldsymbol{R}\sigma^2))^{-1} \boldsymbol{\mu}_z ,$$

Where μz = RZβz , R = In X(X ⊤X) −1X ⊤ and X is the design matrix of the regression model.

The computer vision model approximates this mapping from a set of residuals to its corresponding distance measure. It is trained on a large number of synthetic regression models, where the data-generating process is known, allowing the distance measure to be explicitly calculated. The model takes a residual plot as input and outputs the corresponding distance measure. Additional details are provided in [15].

III. R PACKAGE: AUTOVI

The main purpose of autovi is to provide rejection decisions and *p*-values for testing the null hypothesis (H_0) that the regression model is correctly specified. The package provides automated interpretation of residual plots using computer vision. The name autovi stands for **auto**mated **v**isual **i**nference. This functionality can be accessed through the R package autovi, or through a web interface, autovi.web, which allows use without the full installation of R, Python, and package dependencies on the user's system. locally.

➢ *Motivation*

Figure 1 shows three sets of plots of residuals against fitted values. The simulated example in (a) might be interpreted as a heteroscedastic pattern, however the automated reading would predict this to have a visual signal strength (VSS) of 1.53, with a corresponding *p*-value of 0.25. This means it would be interpreted as a good residual plot, that there is nothing in the data to indicate a violation of model assumptions. Skewness in the predictor variables is generating the apparent heteroscedasticity, where the smaller variance in residuals at larger fitted values is due to smaller sample size only. The Breusch-Pagan test [16] for heteroscedasticity would also not reject this as good residual plot.

The data in (b) is generated by fitting a linear model predicting mpg based on hp using the datasets: :mtcars.It is a small data set, and there is a hint of nonlinear structure not captured by the model. The automated plot reading would predict a VSS of 3.57, which has a *p*-value less than 0.05. That is, the nonlinear structure is most likely real, and indicates a problem with the model. The conventional test, a Ramsey Regression Equation Specification Error Test (RESET) [17] would also strongly detect the nonlinearity.

The third example is generated using the surreal package [18] where structured residuals are hidden in data, to be revealed if the correct model is specified. Here a quote based on Tukey is used as the residual structure "visual summaries focus on unexpected values". The automated plot reading predicts the VSS to be 5.87, with a p-value less than0.05This structure isblindinglyobvious visually, but a RESET test for nonlinear structure would not report a problem. (It would be detected by a Breusch-Pagan for heteroscedasticity and also Shapiro-Wilk test [19] for nonnormality.)

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14964344

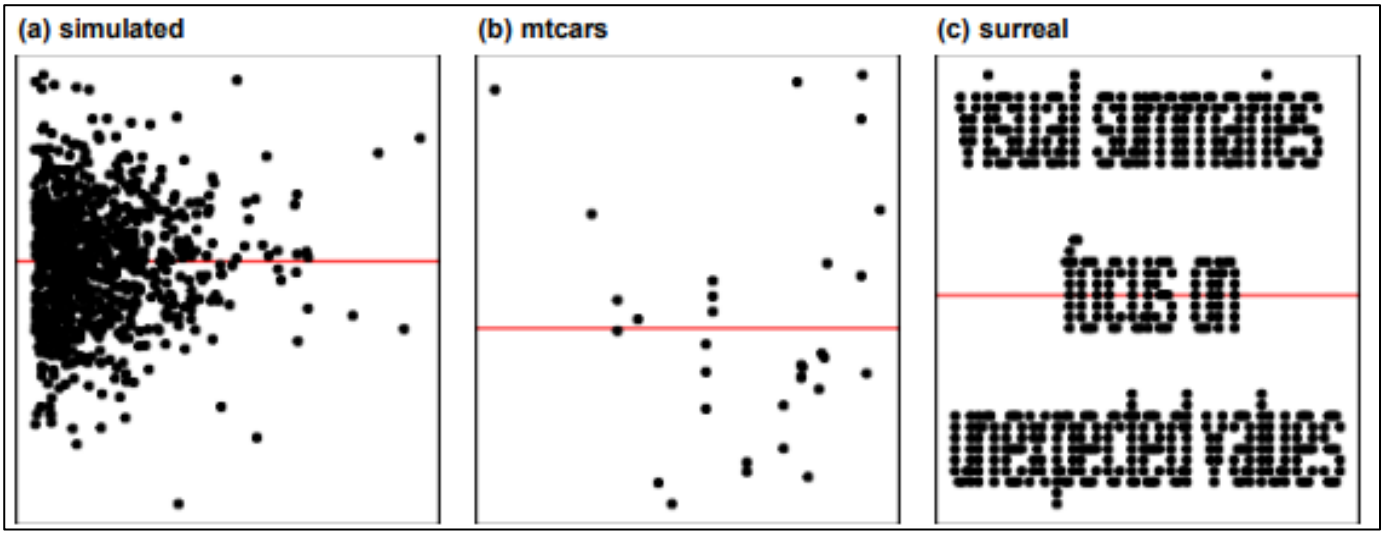

Fig 1 Simulated Mtcars Surereal

Figure 1: Reading residual plots can be a difficult task, particularly for students new to statistical modeling. The autovi package makes it easier. Here are three examples of residual plots, which may appear to have structure. According to autovi, the visual signal strengths (VSS) of these three examples are approximately (a) 1.53, (b) 3.57, (c) 5.87, resulting in (b), (c) being significant violations of good residuals, but (a) is consistent with a good residual plot.

➢ *Implementation*

The autovi package is built on the bandicoot object-oriented programming (OOP) system [20], marking a departure from R's traditional S3 generic system. This OOP architecture enhances flexibility and modularity, allowing users to redefine key functions through method overriding.

The autoviinfrastructure effectively integrates multiple programming languages and libraries into a comprehensive analytical tool. It relies on five core libraries from Python and R, each playing a critical role in the analysis pipeline. In Python, pillow [21] handles image processing tasks such as reading and resizing PNG files of residual plots, then converting them into input tensors for further analysis. TensorFlow [22], a key component of modern machine learning, is used to predict the VSS of these plots using a pre-trained convolutional neural network.

In the R environment, autovi utilizes several libraries. ggplot2 [23] generates the initial residual plots, saved as PNG files for visual input. cassowaryr [24] computes scagnostics (scatter plot diagnostics), providing numerical features that capture statistical properties of the plots. These scagnostics complement the visual analysis by offering quantitative metrics as secondary input to the computer vision model. reticulate [25] enables seamless communication between R and Python.

➢ *Installation*

The autovi package is available on CRAN. It is actively developed and maintained, with the latest updates accessi- ble on GitHub. This paper uses autovi version 0.4.1. The package includes internal functions to check the current Python environment used by the reticulate package. If the necessary Python packages are not installed in the Python interpreter, an error will be raised. If you want to select a specific Python environment, you can do so by calling the reticulate::usepython() function before using the autovipackage.

We recommend using the Shiny app autovi. webif users encounter installation problems.

➢ *Usage*

- Numerical summary: Three steps are needed to get an automated assessment of a set of residuals and fitted values:
- Load the autovi package using the library ()function.
- Create a checker object with a linear regression model.
- Call the check () method of the checker, which, by default, predicts the VSS for the true residual plot,100 null plots, and 100 bootstrapped plots. The methodstores the predictions internally and prints a concise results report.
- The code to do this is:

Listing 1: Residual Checking using AutoVI in R

```
library(autovi)
checker <- residual_checker
(lm(dist ~ speed, data = cars))
checker$check()
```

It produces the following summary:

Table 1 Summary of AUTO_VI object status and results.

```
AUTO_VI Object Status
- Fitted model: lm
- Keras model: UNKNOWN
  - Output node index: 1
- Result:
  - Observed visual signal strength: 3.162 (p =  0.0396)
  - Null visual signal strength: [100 draws]
    - Mean: 1.274
    - Quantiles:
    25%: 0.8021, 50%: 1.1109, 75%:  1.5751
    80%: 1.6656, 90%: 1.9199, 95%: 2.6564, 99%:  3.3491
  - Bootstrapped visual signal strength: [100 draws]
    - Mean: 2.786 (p = 0.05941)
    - Quantiles:
    25%: 2.452, 50%: 2.925, 75%:  3.173
    80%: 3.285, 90%: 3.463, 95%: 3.505, 99%: 3.652
  - Likelihood ratio: 0.7275 (boot) / 0.06298 (null) =  11.55
```

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14964344

The summary includes observed VSS of the true residual plot and associated *p*-value of the automated visual test. The *p*-value is the proportion of null plots (out of the total100) that have VSS greater than or equal to that of the true residual plot. The report also provides sample quantiles of VSS for null samples and bootstrapped data plots, providing more information about the sampling variability and a likelihood of model violations. The likelihood is computed from the proportion of values greater than the observed VSS in both the bootstrapped data values and the simulated null values.

➢ *Visual Summary:*

Users can visually inspect the orig- inal residual plot alongside a sample null plot using plot_pair() or a lineup of null plot plot_lineup(). This visual comparison can clarify why H_0 is either rejected or not, and help identify potential remedies.

```
checker$summary_plot()
```

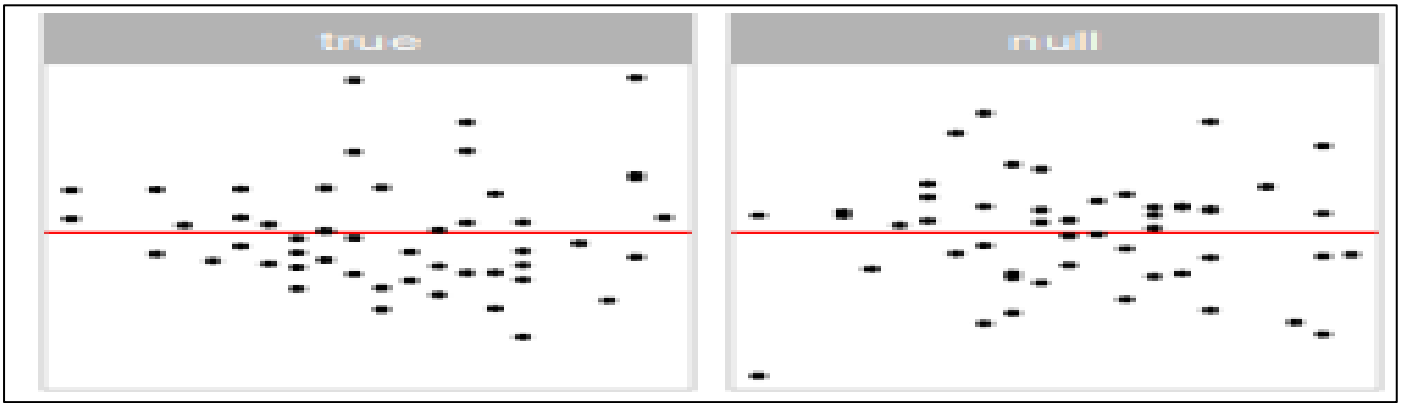

Fig 2 True Plot Alongside One Null Plot, for Quick Comparison.

The plot pair() method (Figure 2) displays the true residual plot on the left and a single null plot on the right.

If a full lineup was shown, the true residual plot would be embedded in a page of null plots. Users should look for any distinct visual patterns in the true residual plot that are absent in the null plot. Running these functions multiple times can help any visual suspicions, as each execution generates new random null plots for comparison.

The package offers a straightforward visualization of the assessment result through the summary_plot()function.

```
checker$summary_plot()
```

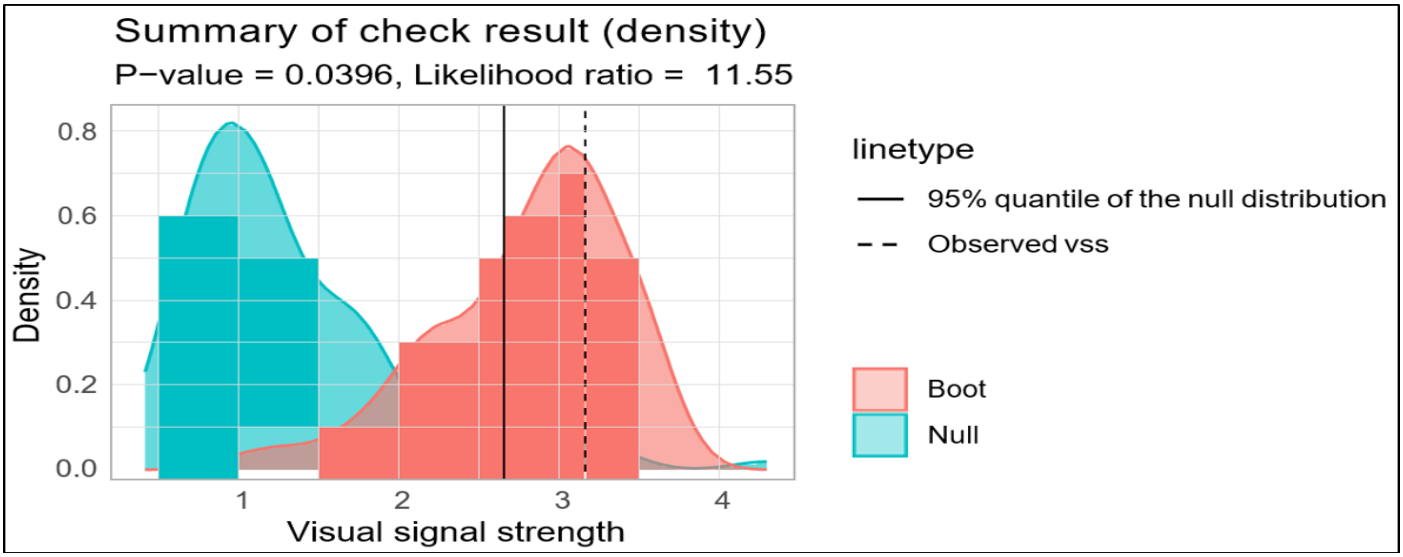

Fig 3 Summary of Check Result

Figure 3: Summary plot comparing the densities of VSS for bootstrapped residual samples (red) relative to VSS for null plots (blue).

In the result, shown in Figure 3, the blue area represents the density of VSS for null residual plots, while the red area shows the density for bootstrapped residual plots. The dashed line indicates the VSS of the true residual plot, and the solid line marks the critical value at a 95% significance level. The *p*-value and the likelihood ratio are displayed in the subtitle. The likelihood ratio represents the ratio of the likelihood of observing the VSS of the true residual plot from the bootstrapped distribution compared to the null distribution.

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14964344

Interpreting the plot involves several key aspects. If the dashed line falls to the right of the solid line, it suggests rejecting the null hypothesis. The degree of overlap between the red and blue areas indicates similarity between the true residual plot and null plots; greater overlap suggests more similarity. Lastly, the portion of the red area to the right of the solid line represents the percentage of bootstrapped models considered to have model violations.

This visual summary provides an intuitive way to assess the model's fit and potential violations, allowing users to quickly grasp the results of the automated analysis.

➢ *Modularized Infrastructure*

The initial motivation for developing autoviwas to create a convenient interface for sharing the models described and trained in [15]. However, recognizing that the classical normal linear regression model represents a restricted class of models, we sought to avoid limiting the potential for future extensions, whether by the original developers or other developers. As a result, the package was designed to function seamlessly with linear regression models with minimal modification and few required arguments, while also accommodating other classes of models through partial infrastructure substitution. This modular and customizable design allows autovi to handle a wide range of residual diagnostics tasks.

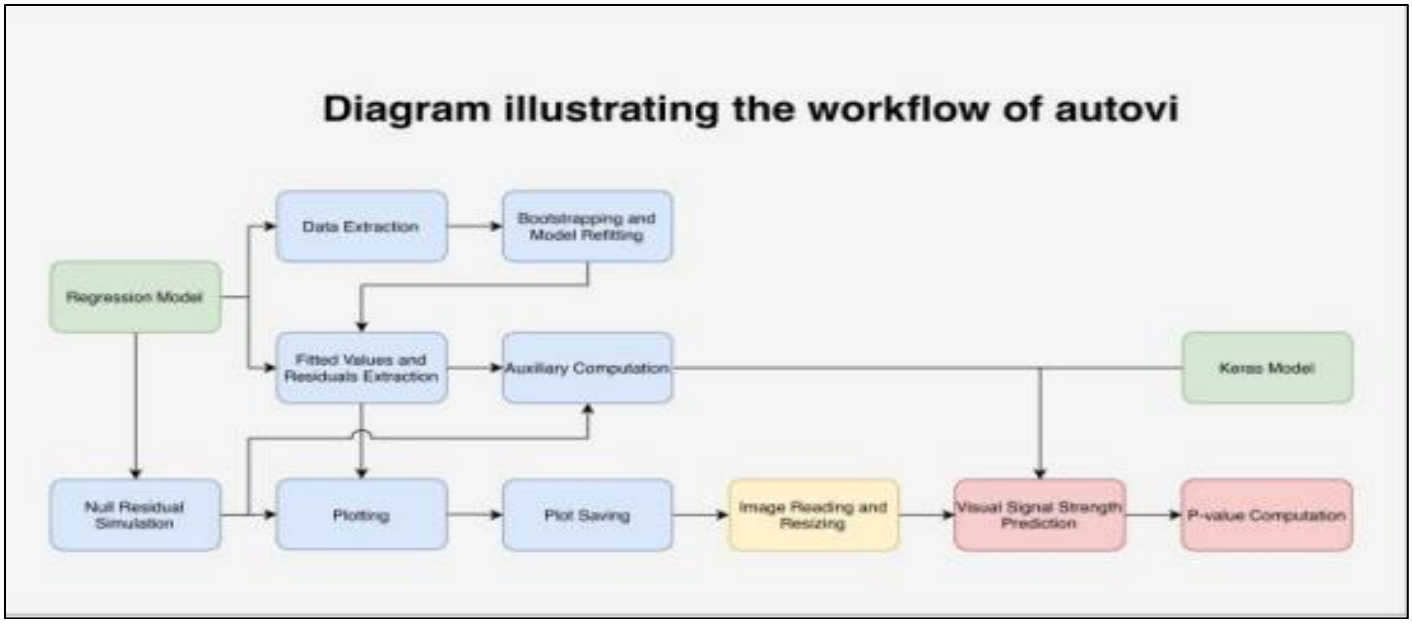

Fig 4 Diagram Illustrating the Workflow of Autovi

Figure 4: Diagram illustrating the infrastructure of the R package autovi. The modules in green are primary inputs provided by users. Modules in blue are overridable methods that can be modified to accommodate users' specific needs. The module in yellow is a pre-defined non-overridable method. The modules in red are primary outputs of the package. future extensions, whether by the original developers or other developers. As a result, the package.

The infrastructure of autoviconsists of ten core modules: data extraction, bootstrapping and model refitting, fitted values and residuals extraction, auxiliary computation, null residual simulation, plotting, plot saving, image reading and resizing, VSS prediction, and *p*-value computation. Each module is designed with minimal dependency on the preceding modules, allowing users to customize parts of the infrastructure without affecting its overall integrity. An overview of this infrastructure is illustrated in Figure 4.

The modules for VSS prediction and *p*-value computation are predefined and cannot be overridden, although users can interact with them directly through function arguments. Similarly, the image reading and resizing module is fixed but will adapt to different Keras models by checking their input shapes. The remaining seven modules are designed to be overridable, enabling users to tailor the infrastructure to their specific needs. These modules are discussed in detail in the package documentation.

IV. WEB INTERFACE: AUTOVI.WEB

The autovi.web shiny application extends the functionality of autovi by offering a user-friendly web interface for automated residual plot assessment. This eliminates the common challenges associated with software installation, so users can avoid managing Python environments or handling version requirements for R libraries. The platform is cross-platform and accessible on various devices and operating systems, making it suitable even for users without R programming experience. Additionally, updates are managed centrally, ensuring that users always have access to the latest features. This section discusses the implementation based on autovi.webversion 0.1.0.

➢ *Implementation*

The interface autovi.web is built using the shiny [26] and shiny dashboard [27] R packages. Hosted on the shinyapps.io domain, the application is accessible through

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14964344

any modern web browser. The R packages htmltools [28] and shinycssloaders [29] are used to render markdown documentation in shiny application, and for loading ani- mations for shiny widgets, respectively.

Determining the best way to implement the backend was difficult. In our initial planning for autovi.web, we considered implementing the entire web application using the webr framework [30], which would have allowed the entire application to run directly in the user's browser. However, webr does not support packages which use compiled fortran code, which is required by splancs [31], a dependency of autovi. In the future, it is possible that a working Emscripten [32] version of this package may allow full web rsupport.

We also explored the possibility of implementing the web interface using frameworks built on other languages, such as Python. However, server hosting domains that natively support Python servers typically do not have the latest version of R installed. Additionally, calling R from Python is typically done using the rpy2 Python library [33], but this approach can be awkward when dealing with language syntax related to non-standard evaluation. Another option we considered was renting a server where we could have full control, such as those provided by cloud platforms like Google Cloud Platform (GCP) or Amazon Web Services (AWS). However, deploying and maintaining the server securely requires some expertise. Ultimately, the most practical solution was to use the shiny and shinydashboard frameworks, which are well-established in the R community and offer a solid foundation for web application development.

The server-side configuration of autovi.webis carefully designed to support its functionality. Most required Python libraries, including pillow and numpy, are pre-installed on the server. These libraries are integrated into the Shiny application using the reticulate package, which provides an interface between R and Python.

Due to the resource allocation policy of shinyapps.io, the server enters a sleep mode during periods of inactivity, resulting in the clearing of the local Python virtual environment. Consequently, when the application "wakes up" for a new user session, these libraries need to be reinstalled. While this ensures a clean environment for each session, it may lead to slightly longer loading times for the first user after a period of inactivity.

In contrast to autovi, autovi.web leverages Tensor Flow .js, a JavaScript library that allows the execution of machine learning models directly in the browser. This choice enables native browser execution, enhancing compatibility across different user environments, and shiftsthe computational load from the server to the client-side. TensorFlow.js also offers better scalability and performance, especially when dealing with resource-intensive computer vision models on the web.

While autovirequires downloading the pre-trained com- puter vision models from GitHub, these models i ".keras" file format are incompatible with TensorFlow.js. There- fore, we extract and store the model weights in JSON files and include them as extra resources in the Shiny appli- cation. When the application initializes, TensorFlow.js rebuilds the computer vision model using these pre-stored weights.

To allow communication between TensorFlow.js and other components of the Shiny application, the shinyjs R package [34] is used. This package allows calling cus- tom JavaScript code within the Shiny framework. The specialized JavaScript code for initializing TensorFlow.js and calling TensorFlow.js for VSS prediction is deployed alongside the Shiny application as additional resources.

- *Usage*

The workflow of autovi.web is designed to be straight for- ward, with numbered steps displayed in each panel. There are two example datasets provided by the web application. The single residual plot example uses the dino dataset from the R package datasauRus [35]. The lineup example uses residuals from a simulated regression model that has a non-linearity issue. We walk through the lineup example to further demonstrate the workflow of the web application.

- *Reading data and setting parameters:*

The user can select to upload data as either a single set of residuals and fitted values in a two (or more) column CSV file or a pre-computed lineup of residuals and null datasets in a three (or more) column CSV file (i.e. multiple sets of residuals and fitted values with a column indicating the set label). Here we illustrate use with lineup example data sets (Figure 5). To use the lineup example data, click the "Use Lineup Example" button. The data status will then update to show the number of rows and columns in the dataset, and the CSV type will automatically be selected to the correct option. Since the example dataset follows the variable naming conventions assumed by the web application, the columns for fitted values, residuals, and labels of residual plots are automatically mapped such that the column named as.fitted is mapped to fitted values, .resid is mapped to residuals and if applicable, sampleto labels of the residual set (middle image). If the user is working with a custom dataset, these options must be set accordingly. Whenever a data containing a lineup, the user must manually select the label for the true residual plot, otherwise the web application does not provide all the results. The last step is to click the play button (right image) to start the assessment.

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14964344

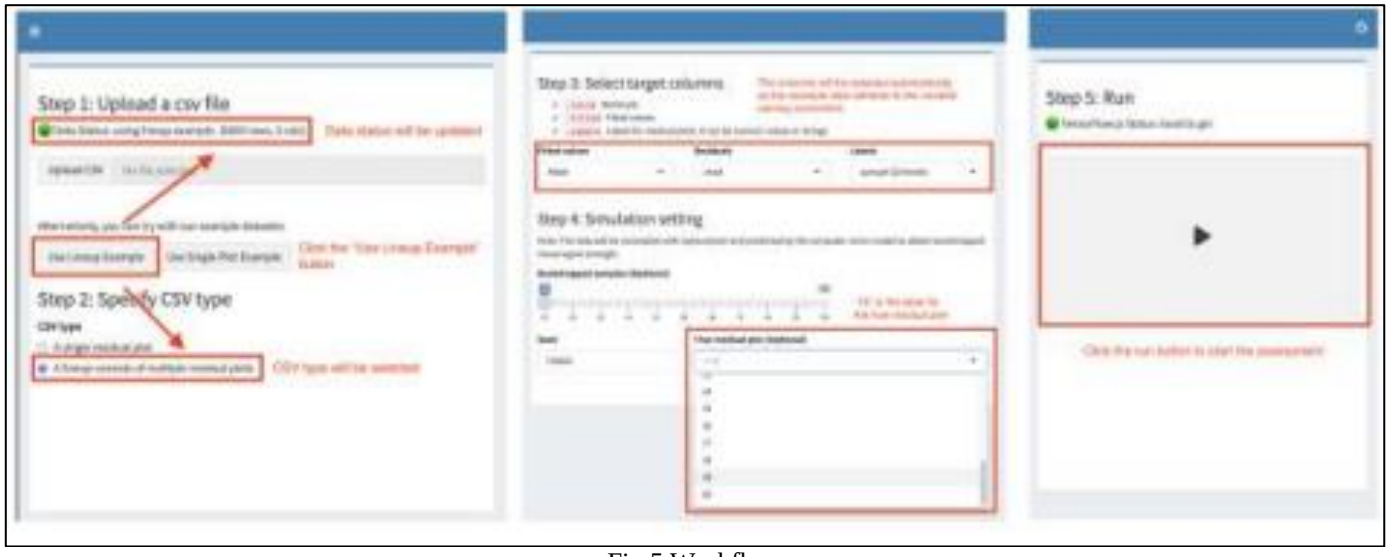

Fig 5 Workflow

Figure 5: To begin the workflow for auto vi using the lineup example dataset, the user clicks the "Use Lineup Example" button (left) to load the example dataset, during which the data status and CSV type will be automatically updated. The user must manually select the label for the true residual plot (middle) to compute further results. The user initiates the assessment of the lineup example data by clicking the run button (right).

➢ *Results Provided*

Results are provided in multiple panels. The first row of the table Figure 6 is the most crucial to check, as it provides the VSS and the rank of the true residual plot among the other plots. The summary text beneath the table provides the *p*-value, which can be used for quick decision-making. The lineup is for manual inspection, and the user should see if the true residual plot is visually distinguishable from the other plots, to confirm if the model violation is serious.

The density plot in Figure 7 offers a more robust result, al- lowing the user to compare the distribution of bootstrapped VSS with the distribution of null VSS. Finally, the grayscale attention map (right image) can be used to check if the target visual features, like the non-linearity present in the lineup example, are captured by the computer vision model, ensuring the quality of the assessment.

V. CONCLUSIONS

This paper presents new regression diagnostics software, the R package autovi and its accompanying web interface, autovi.web. It addresses a critical gap in the current landscape of statistical software. While regression tools are widely available, effective and efficient diagnostic methods have lagged behind, particularly in the field of residual plot interpretation.

The autovir package, introduced in this paper, automates the assessment of residual plots by incorporating a computer vision model, reducing reliance on time-consuming and potentially inconsistent human interpretation. This automation improves the efficiency of the diagnostic process and promotes consistency in model evaluation across different users and studies.

Fig 6 Visual Signal Strength

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14964344

Figure 6: Results for the lineup. The VSS of the true residual plot is displayed in the first row of the table of VSS values for all the null plots (left image), with a summary text beneath the table providing the *p*-value to aid in decision-making. A lineup of residual plots allows for manual inspection (right image).

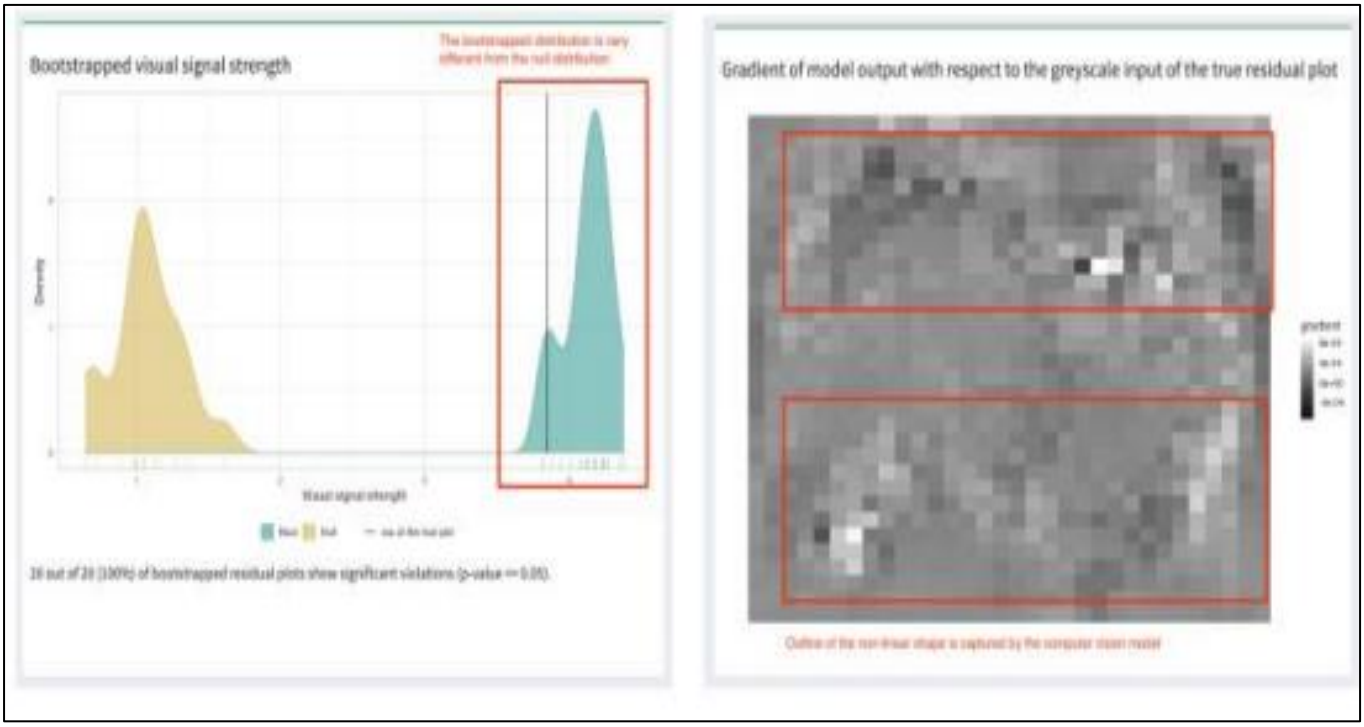

Fig 7 Bootstrapped Visual Signal Strength

Figure 7: Summaries assessing the strength of the pattern and which elements of the plot contribute. The density plot helps verify if the bootstrapped distribution differs from the null distribution (left image). The attention map (right image) offers insights into whether the computer vision model has captured the intended visual features of the true residual plot.

The development of the accompanying Shiny app, autovi.web, expands access to these advanced diagnostic tools, by providing a user-friendly interface. It makes automated residual plot assessment accessible to a broader audience, including those who may not have extensive programming experience. This web-based solution effec- tively addresses the potential barriers to adoption, such as complex dependencies and installation requirements, that are often associated with advanced statistical software.

The combination of autovi and autovi.web offers a comprehensive solution to the challenges of residual plot interpretation in regression analysis. These tools have the potential to significantly improve the quality and consistency of model diagnostics across various fields, from academic research to industry applications. By automating a critical aspect of model evaluation, they allow researchers and analysts to focus more on interpreting results and refining models, rather than grappling with the intricacies of plot assessment.

The framework established by autovi and autovi.web opens up exciting possibilities for further research and development. Future work could explore the extension of these automated assessment techniques to other types of diagnostic plots and statistical models, potentially revolutionizing how we approach statistical inference using visual displays more broadly.

VI. RESOURCES AND SUPPLEMENTARY MATERIAL

The current version of autovi can be installed from CRAN, and source code for both packages are available at github.com/TengMCing/autovi and github.com/TengMCing/autovi_web respectively. The web interface is available from autoviweb.netlify.app.

This paper is reproducibly written using Quarto [36] powered by Pandoc [37] and pdfTeX.

These R packages were used for the work: tidyverse [38], lmtest [1], kableExtra [39], patchwork [40], rcartocolor [41], glue [42], here [43], magick [44], yardstick [45] and reticulate[25].

REFERENCES

[1]. A. Zeileis and T. Hothorn, “Diagnostic checking in regression relationships,” R News, vol. 2, no. 3, pp. 7–10, 2002.
[2]. R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna,
[3]. Austria, 2022. [Online]. Available: https://www.R-project.org/

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14964344

[4]. J. A. Long, jtools: Analysis and Presentation of Social Scientific Data, 2022, r package version 2.2.0. [Online]. Available: https://cran.r-project.org/package=jtools
[5]. A. Hebbali, olsrr: Tools for Building OLS Regression Models, 2024, r package version 0.6.0. [Online]. Available: https://CRAN.R-project.org/package=olsrr
[6]. P. E. Johnson, rockchalk: Regression Estimation and Presentation, 2022, r package version 1.8.157. [Online]. Available: https://CRAN.R-project.org/package=rockchalk
[7]. K. Goode and K. Rey, ggResidpanel: Panels and Interactive Versions of Diagnostic Plots using 'ggplot2', 2019, r package version 0.3.0. [Online]. Available: https://CRAN.R-project.org/package=ggResidpanel
[8]. R. D. Cook and S. Weisberg, Residuals and influence in regres- sion. New York: Chapman and Hall, 1982.
[9]. D. I. Warton, "Global simulation envelopes for diagnostic plots in regression models," The American Statistician, vol. 77, no. 4, pp. 425–431, 2023.
[10]. F. Hartig, DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models, 2022, r package version 0.4.6. [Online]. Available: https://CRAN.R-project.org/package=DHARMa
[11]. W. Li, D. Cook, E. Tanaka, and S. VanderPlas, "A plot is worth a thousand tests: Assessing residual diagnostics with the lineup protocol," Journal of Computational and Graphical Statistics, vol. 33, pp. 1497–1511, 2024.
[12]. A. Buja, D. Cook, H. Hofmann, M. Lawrence, E.-K. Lee, D. F. Swayne, and H. Wickham, "Statistical inference for exploratory data analysis and model diagnostics," Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 367, no. 1906, pp. 4361–4383, 2009.
[13]. H. Wickham, N. R. Chowdhury, D. Cook, and H. Hofmann, nullabor: Tools for Graphical Inference, 2020, r package version 0.3.9. [Online]. Available: https://CRAN.R-project.org/package=nullabor
[14]. A. Loy and H. Hofmann, "Hlmdiag: A suite of diagnostics for hierarchical linear models in r," Journal of Statistical Software, vol. 56, pp. 1–28, 2014.
[15]. A. Reinhart, regressinator: Simulate and Diagnose (Generalized) Linear Models, 2024, r package version 0.2.0. [Online]. Available: https://CRAN.R-project.org/package=regressinator
[16]. W. Li, D. Cook, E. Tanaka, S. VanderPlas, and K. Ackermann, "Automated assessment of residual plots with computer vision models," arXiv preprint arXiv:2411.01001, 2024.
[17]. T. S. Breusch and A. R. Pagan, "A simple test for heteroscedas- ticity and random coefficient variation," Econometrica: Journal of the Econometric Society, pp. 1287–1294, 1979.
[18]. J. B. Ramsey, "Tests for specification errors in classical linear least-squares regression analysis," Journal of the Royal Statistical Society: Series B (Methodological), vol. 31, no. 2, pp. 350–371, 1969.
[19]. J. J. Balamuta, surreal: Create Datasets with Hidden Images in Residual Plots, 2024, r package version 0.0.1. [Online]. Available: https://CRAN.R-project.org/package=surreal
[20]. S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," Biometrika, vol. 52, no. 3/4, pp. 591–611, 1965.
[21]. W. Li, "bandicoot: Light-weight python-like object-oriented system," 2024. [Online]. Available: https://CRAN.R-project.org/package=bandicoot A. Clark et al., "Pillow (pil fork) documentation," readthedocs, 2015.
[22]. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al., "Tensor- flow: Large-scale machine learning on heterogeneous distributed systems," arXiv preprint arXiv:1603.04467, 2016.
[23]. H. Wickham, ggplot2: Elegant graphics for data analysis. Springer-Verlag New York, 2016. [Online]. Available: https://ggplot2.tidyverse.org
[24]. H. Mason, S. Lee, U. Laa, and D. Cook, cassowaryr: Compute Scagnostics on Pairs of Numeric Variables in a Data Set, 2022, r package version 2.0.0. [Online]. Available: https://CRAN.R-project.org/package=cassowary
[25]. K. Ushey, J. Allaire, and Y. Tang, reticulate: Interface to 'Python', 2024, r package version 1.35.0. [Online]. Available: https://CRAN.R-project.org/package=reticulate
[26]. W. Chang, J. Cheng, J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges, shiny: Web Application Framework for R, 2022, r package version 1.7.3. [Online]. Available: https://CRAN.R-project.org/package=shiny
[27]. W. Chang and B. Borges Ribeiro, shinydashboard: Create Dashboards with 'Shiny', 2021, r package version 0.7.2. [Online]. Available: https://CRAN.R-project.org/package=shinydashbo ard
[28]. J. Cheng, C. Sievert, B. Schloerke, W. Chang, Y. Xie, and J. Allen, htmltools: Tools for HTML, 2024, r package version 0.5.8. [Online]. Available: https://CRAN.R-project.org/package=htmltools
[29]. A. Sali and D. Attali, shinycssloaders: Add Loading Animations to a 'shiny' Output While It's Recalculating, 2020, r package version 1.0.0. [Online]. Available: https://CRAN.R-project.org/package=shinycssloaders
[30]. K.-W. Moon, webr: Data and Functions for Web-Based Analysis, 2020, r package version 0.1.5. [Online]. Available: https://CRAN.R-project.org/package=webr
[31]. B. Rowlingson and P. Diggle, splancs: Spatial and Space-Time Point Pattern Analysis, 2023, r package version 2.01-44. [Online]. Available: https://CRAN.R-project.org/package=splancs
[32]. A. Zakai, "Emscripten: an llvm-to-javascript compiler," in Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion, 2011, pp. 301–312.
[33]. L. Gautier, Python interface to the R language

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14964344

(embedded R), 2024, version 3.5.16. [Online]. Available: https://pypi.org/proje ct/rpy2/
[34]. D. Attali, shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds, 2021, r package version 2.1.0. [Online]. Available: https://CRAN.R-project.org/package=shinyjs
[35]. R. Davies, S. Locke, and L. D'Agostino McGowan, datasauRus: Datasets from the Datasaurus Dozen, 2022, re package version 0.1.6. [Online]. Available: https://CRAN.R-project.org/package=datasauRus
[36]. J. Allaire, C. Teague, C. Scheidegger, Y. Xie, and C. Dervieux, "Quarto," Feb. 2024. [Online]. Available: https://github.com/q uarto-dev/quarto-cli
[37]. J. MacFarlane, A. Krewinkel, and J. Rosenthal, "Pandoc," 2024. [Online]. Available: https://github.com/jgm/pandoc
[38]. H. Wickham, M. Averick, J. Bryan, W. Chang, L. D. McGowan,R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller,J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani, "Welcome to the tidyverse," Journal of Open Source Software, vol. 4, no. 43, p. 1686, 2019.
[39]. H. Zhu, kableExtra: Construct complex table with kable and pipe syntax, 2021, r package version 1.3.4. [Online]. Available: https://CRAN.R-project.org/package=kableExtra
[40]. T. L. Pedersen, patchwork: The composer of plots, 2022, rpackage version 1.1.2. [Online]. Available: https://CRAN.R-pro ject.org/package=patchwork
[41]. J. Nowosad, 'CARTOColors' palettes, 2018, r package version 1.0. [Online]. Available: https://nowosad.github.io/rcartocolor
[42]. J. Hester and J. Bryan, glue: Interpreted String Literals, 2022, r package version 1.6.2. [Online]. Available: https://CRAN.R-project.org/package=glue
[43]. K. Müller, here: A simpler way to find your files, 2020, r package version 1.0.1. [Online]. Available: https://CRAN.R-project.org/package=here
[44]. J. Ooms, magick: Advanced Graphics and Image-Processing in R, 2023, r package version 2.7.4. [Online]. Available: https://CRAN.R-project.org/package=magick
[45]. M. Kuhn, D. Vaughan, and E. Hvitfeldt, yardstick: Tidy Characterizations of Model Performance, 2024, r package version 1.3.1. [Online]. Available: https://CRAN.R-project.org/package=yardstick