

# Automated Residual Plot Assessment with the R Package autovi and the Shiny App autovi.web

Weihao Li<sup>1</sup>, Dianne Cook<sup>1</sup>, Emi Tanaka<sup>2</sup>, Susan VanderPlas<sup>3</sup> and Klaus Ackermann<sup>1</sup>

*Monash University, The Australian National University and University of Nebraska*

## Summary

Visually assessing residual plots is a common advice for linear model diagnostics, however this approach requires manual human evaluation and thereby is not scalable for assessing many models. Human evaluation also has the potential to produce inconsistent decisions from different analysts. Using a lineup protocol, where the residual plot is embedded among null plots, can help to alleviate inconsistency, but requires even more human effort. This is the type of task that in today's world we might employ a robot to do the tedious work for a human. Here we describe a new R package that includes a computer vision model for automated assessment of residual plots, and an accompanying Shiny app for ease of use. For a user-provided sample of residuals, it predicts a measure of visual signal strength (VSS) and provides a suite of supporting information to assist the analyst decide on the appropriateness their model fit.

*Key words:* initial data analysis; statistical graphics; data visualization; visual inference; computer vision; machine learning; hypothesis testing; regression analysis; model diagnostics

## 1. Introduction

10 Regression analysis is a widely used statistical modeling technique for data in many  
11 fields. There is a vast array of software for conducting regression modeling and  
12 generating diagnostics. The package `lmtest` ([Zeileis & Hothorn 2002](#)) provides a  
13 suite of conventional tests. The `stats` package ([R Core Team 2022](#)) offers standard

---

<sup>1</sup> Department of Econometrics and Business Statistics, Monash University, Wellington Road, VIC 3800, Australia.

<sup>2</sup> Biological Data Science Institute, The Australian National University, 46 Sullivan's Creek Road, ACT 2600, Australia

<sup>3</sup> Department of Statistics, University of Nebraska, Hardin Hall, 3310 Holdrege St Suite 340, Lincoln, NE 68583, United States

Email: [weihao.li@monash.edu](mailto:weihao.li@monash.edu)

diagnostic plots such as residuals vs. fitted values, quantile-quantile (Q-Q) plots, and residuals vs. leverage plots. Packages like `jtools` (Long 2022), `olsrr` (Hebbali 2024), `rockchalk` (Johnson 2022), and `ggResidpanel` (Goode & Rey 2019) provide similar graphical diagnostics, often with alternative aesthetics or interactive features. All of these tools deliver the types of diagnostic plots outlined in the classical text by Cook & Weisberg (1982). The `ecostats` package (Warton 2023) incorporates simulation envelopes into residual plots, while DHARMA (Hartig 2022) 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. Li et al. (2024a) demonstrated that visual methods using the lineup protocol (Buja et al. 2009) for assessing residuals are more useful, and also perform more practically than conventional tests due to their reduced sensitivity to minor departures. Packages such as `nullabor` (Wickham et al. 2020), `HLMdiag` (Loy & Hofmann 2014), and `regressinator` (Reinhart 2024), 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 Li et al. (2024b), 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 3 provides a detailed documentation of the `autovi` package, including its usage and infrastructure. Section 4

48 focuses on the `autovi.web` interface, describing its design and usage, along with  
 49 illustrative examples. Finally, Section 5 presents the main conclusions of this work.

## 50 2. Definition and computation of visual signal strength

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

55 In the context of regression model diagnostics, it describes the clarity of visual patterns  
 56 on a diagnostic plot that may indicate model violations. This concept can be categorized  
 57 as weak, moderate, or strong. However, in our study, we treat it as a continuous positive  
 58 real variable. Importantly, its interpretation depends on how it is linked to a function  
 59 of the data or the underlying data generating process. Consequently, the calculation of  
 60 VSS can be vary across different model classes or within the same model, depending  
 61 on the generating function.

62 VSS is defined as an estimate of a distance measure that quantifies the disparity  
 63 between the residual distribution of a fitted classical normal linear regression model  
 64 and a reference distribution, and more details can be found in [Li et al. \(2024b\)](#). This  
 65 measure is based on the Kullback-Leibler (KL) divergence:

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

66 where  $D_{KL}$  is given by:

$$D_{KL} = \int_{\mathbb{R}^n} \log \frac{p(\mathbf{e})}{q(\mathbf{e})} p(\mathbf{e}) d\mathbf{e},$$

67 here,  $p(\cdot)$  and  $q(\cdot)$  are the probability density functions of the reference residual  
 68 distribution  $P$  and the true residual distribution  $Q$ , respectively.

69 This distance measure requires knowledge of the true residual distribution, which is  
 70 typically unknown. Therefore, a computer vision model was used for estimating the  
 71 residual distribution (see [Li et al. 2024b](#), for more details).

72

### 3. R package: autovi

73 The main purpose of **autovi** is to provide rejection decisions and  $p$ -values for testing  
 74 the null hypothesis ( $H_0$ ) that the regression model is correctly specified. The package  
 75 provides automated interpretation of residual plots using computer vision. The name  
 76 **autovi** stands for **a**utomated **v**isual **i**nference.

77 There are two ways to access the package, directly using R or through a web interface,  
 78 **autovi.web**. The web interface has the advantage that it can be used without installing  
 79 Python, R and the relevant packages locally.

80 **3.1. Motivation for usage**

81 Figure 1 shows three sets of plots of residuals against fitted values. The simulated  
 82 example in (a) might be interpreted as a heteroscedastic pattern, however the  
 83 automated reading would predict this to have a visual signal strength (VSS) of  
 84 1.53, with a corresponding  $p$ -value of 0.25. This means it would be interpreted as  
 85 a good residual plot, that there is nothing in the data to indicate a violation of  
 86 model assumptions. Skewness in the predictor variables is generating the apparent  
 87 heteroscedasticity, where the smaller variance in residuals at larger fitted values is  
 88 due to smaller sample size only. The Breusch-Pagan test (Breusch & Pagan 1979) for  
 89 heteroscedasticity would also not reject this as good residual plot.

90 The data in (b) is generated by fitting a linear model predicting `mpg` based on `hp`  
 91 using the `datasets::mtcars`. It is a small data set, and there is a hint of nonlinear  
 92 structure not captured by the model. The automated plot reading would predict a  
 93 VSS of 3.57, which has a  $p$ -value less than 0.05. That is, the nonlinear structure is  
 94 most likely real, and indicates a problem with the model. The conventional test, a  
 95 Ramsey Regression Equation Specification Error Test (RESET) (Ramsey 1969) would  
 96 also strongly detect the nonlinearity.

97 The third example is generated using the `surreal` package (Balamuta 2024) where  
 98 structured residuals are hidden in data, to be revealed if the correct model is specified.  
 99 Here a quote based on Tukey is used as the residual structure “visual summaries focus  
 100 on unexpected values”. The automated plot reading predicts the VSS to be 5.87, with  
 101 a  $p$ -value less than 0.05. This structure is blindingly obvious visually, but a RESET  
 102 test for nonlinear structure would not report a problem. (It would be detected by  
 103 a Breusch-Pagan for heteroscedasticity and also Shapiro-Wilk test (Shapiro & Wilk  
 104 1965) for non-normality.)

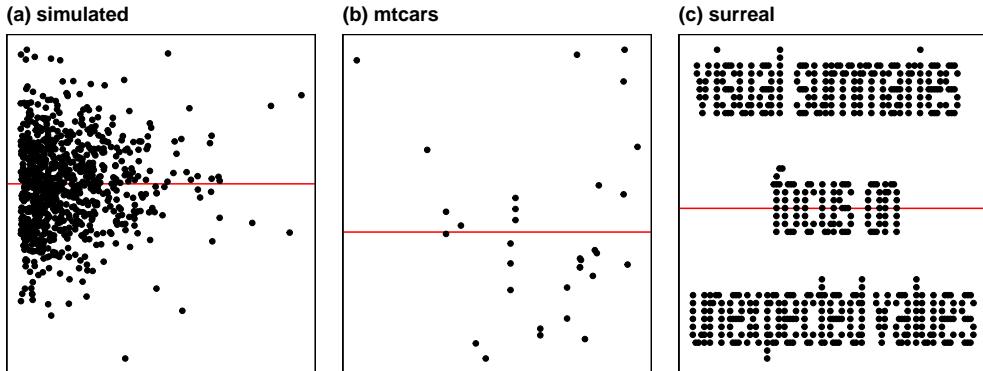


Figure 1. Reading residual plots can be a difficult task, particularly for students new to statistical modeling. The `autovi` package makes it easier. Here are three examples of residual plots, which may appear to have structure. According to `autovi`, the visual signal strengths (VSS) of these three 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.

### 105 3.2. Implementation

106 The `autovi` package is built on the `bandicoot` object-oriented programming (OOP)  
 107 system ([Li 2024](#)), marking a departure from R's traditional S3 generic system. This  
 108 OOP architecture enhances flexibility and modularity, allowing users to redefine key  
 109 functions through method overriding. While similar functionality could be achieved  
 110 using R's S3 system with generic functions, the OOP framework offers a more structured  
 111 and extensible foundation for the package.

112 The `autovi` infrastructure effectively integrates multiple programming languages and  
 113 libraries into a comprehensive analytical tool. It relies on five core libraries from  
 114 Python and R, each playing a critical role in the analysis pipeline. In Python, `pillow`  
 115 ([Clark et al. 2015](#)) handles image processing tasks such as reading and resizing PNG  
 116 files of residual plots, then converting them into input tensors for further analysis. The  
 117 `TensorFlow` ([Abadi et al. 2016](#)) library, a key component of modern machine learning,  
 118 is used to predict the VSS of these plots through a pre-trained convolutional neural  
 119 network.

120 In the R environment, `autovi` utilizes several libraries. `ggplot2` ([Wickham 2016](#))  
 121 generates the initial residual plots, saved as PNG files for visual input. The `cassowaryr`  
 122 ([Mason et al. 2022](#)) library computes scagnostics (scatter plot diagnostics), providing  
 123 numerical features that capture statistical properties of the plots. These scagnostics  
 124 complement the visual analysis by offering quantitative metrics as secondary input to  
 125 the computer vision model. The `reticulate` ([Ushey, Allaire & Tang 2024](#)) package

126 bridges R and Python, enabling seamless communication between the two languages  
 127 and supporting the integrated infrastructure.

128 **3.3. Installation**

129 The `autovi` package is available on CRAN. It is actively developed and maintained,  
 130 with the latest updates accessible on GitHub. The code discussed in this paper is  
 131 based on `autovi` version 0.4.1.

132 The package includes internal functions to check the current Python environment used  
 133 by the `reticulate` package. If the necessary Python packages are not installed in the  
 134 Python interpreter, an error will be raised. If you want to select a specific Python  
 135 environment, you can do so by calling the `reticulate::use_python()` function before  
 136 using the `autovi` package.

137 We recommend using the Shiny app `autovi.web` if users encounter installation  
 138 problems.

139 **3.4. Usage**

140 **3.4.1. Numerical summary**

141 Three steps are needed to get an automated assessment of a set of residuals and fitted  
 142 values:

- 143 1. Load the `autovi` package using the `library()` function.  
 144 2. Create a checker object with a linear regression model.  
 145 3. Call the `check()` method of the checker, which, by default, predicts the VSS for  
 146 the true residual plot, 100 null plots, and 100 bootstrapped plots, storing the  
 147 predictions internally. A concise report of the check results is then printed.

148 The code to do this is:

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

149 It produces the following summary:

150

151 -- <AUTO\_VI object>

152 Status:

```

153 - Fitted model: lm
154 - Keras model: UNKNOWN
155     - Output node index: 1
156 - Result:
157     - Observed visual signal strength: 3.162 (p-value = 0.0396)
158     - Null visual signal strength: [100 draws]
159         - Mean: 1.274
160         - Quantiles:
161
162     25%      50%      75%      80%      90%      95%      99%
163     0.8021  1.1109  1.5751  1.6656  1.9199  2.6564  3.3491
164
165     - Bootstrapped visual signal strength: [100 draws]
166         - Mean: 2.786 (p-value = 0.05941)
167         - Quantiles:
168
169     25%      50%      75%      80%      90%      95%      99%
170     2.452   2.925   3.173   3.285   3.463   3.505   3.652
171
172     - Likelihood ratio: 0.7275 (boot) / 0.06298 (null) = 11.55

```

173 The summary includes observed VSS of the true residual plot and associated  $p$ -value  
 174 of the automated visual test. The  $p$ -value is the proportion of null plots (out of the  
 175 total 100) that have VSS greater than or equal to that of the true residual plot. The  
 176 report also provides sample quantiles of VSS for null samples and bootstrapped data  
 177 plots, providing more information about the sampling variability and a likelihood of  
 178 model violations. The likelihood is computed from the proportion of values greater  
 179 than the observed VSS in both the bootstrapped data values and the simulated null  
 180 values.

### 181 **3.4.2. Visual summary**

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

```
checker$plot_pair()
```

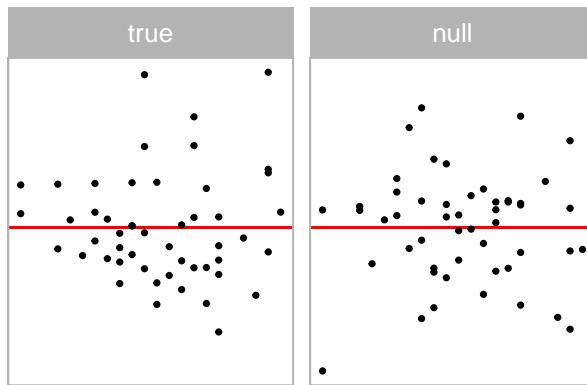


Figure 2. True plot alongside one null plot, for quick comparison.

185 The `plot_pair()` method (Figure 2) displays the true residual plot on the left and a  
 186 single null plot on the right. If a full lineup was shown, the true residual plot would  
 187 be embedded in a page of null plots. Users should look for any distinct visual patterns  
 188 in the true residual plot that are absent in the null plot. Running these functions  
 189 multiple times can help any visual suspicions, as each execution generates new random  
 190 null plots for comparison.

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

```
checker$summary_plot()
```

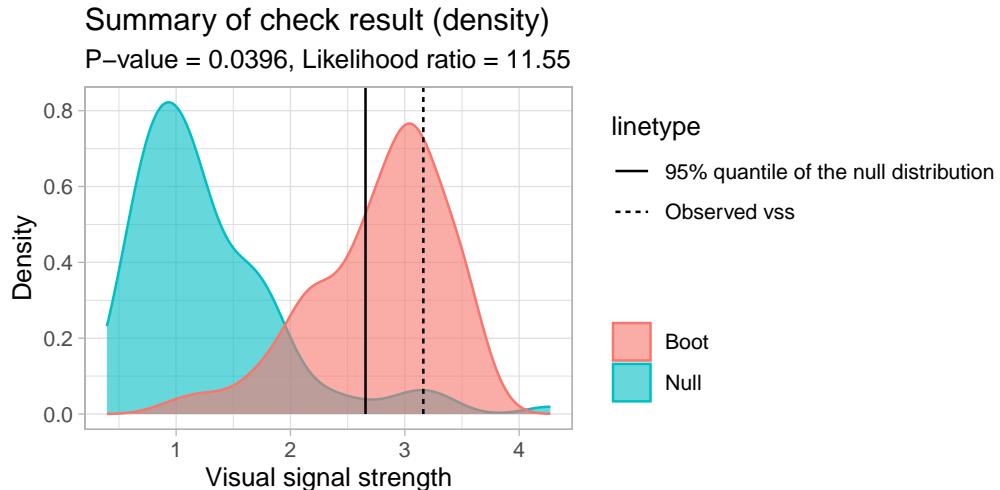


Figure 3. Summary plot comparing the densities of VSS for bootstrapped residual samples (red) relative to VSS for null plots (blue).

- 193 In the result, shown in Figure 3, the blue area represents the density of VSS for null  
 194 residual plots, while the red area shows the density for bootstrapped residual plots.  
 195 The dashed line indicates the VSS of the true residual plot, and the solid line marks  
 196 the critical value at a 95% significance level. The  $p$ -value and the likelihood ratio are  
 197 displayed in the subtitle. The likelihood ratio represents the ratio of the likelihood  
 198 of observing the VSS of the true residual plot from the bootstrapped distribution  
 199 compared to the null distribution.
- 200 Interpreting the plot involves several key aspects. If the dashed line falls to the right of  
 201 the solid line, it suggests rejecting the null hypothesis. The degree of overlap between  
 202 the red and blue areas indicates similarity between the true residual plot and null  
 203 plots; greater overlap suggests more similarity. Lastly, the portion of the red area to  
 204 the right of the solid line represents the percentage of bootstrapped models considered  
 205 to have model violations.
- 206 This visual summary provides an intuitive way to assess the model's fit and potential  
 207 violations, allowing users to quickly grasp the results of the automated analysis.

### 208 3.5. Modularized infrastructure

- 209 The initial motivation for developing `autovi` was to create a convenient interface for  
 210 sharing the models described and trained in Li et al. (2024b). However, recognizing  
 211 that the classical normal linear regression model represents a restricted class of

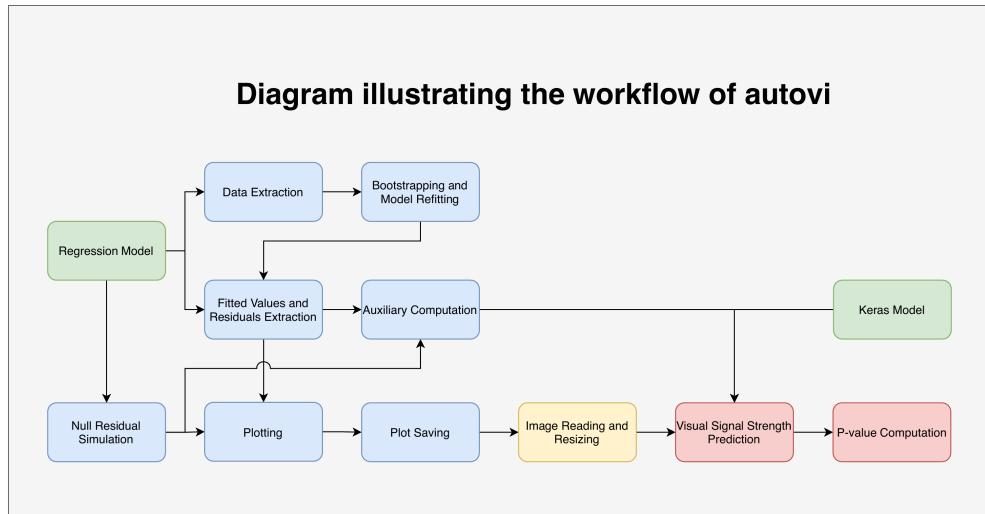


Figure 4. Diagram illustrating the infrastructure of the R package `autovi`. The modules in green are primary inputs provided by users. Modules in blue are overridable methods that can be modified to accommodate users' specific needs. The module in yellow is a pre-defined non-overridable method. The modules in red are primary outputs of the package.

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.

The infrastructure of `autovi` consists 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 on the software's website.

231

#### 4. Web interface: `autovi.web`

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

240 **4.1. Implementation**

241 The interface `autovi.web` is built using the `shiny` (Chang et al. 2022) and  
242 `shinydashboard` (Chang & Borges Ribeiro 2021) R packages. Hosted on the  
243 `shinyapps.io` domain, the application is accessible through any modern web browser.  
244 The R packages `htmltools` (Cheng et al. 2024) and `shinycssloaders` (Sali & Attali  
245 2020) are used to render markdown documentation in shiny application, and for loading  
246 animations for shiny widgets, respectively.

247 Determining the best way to implement the backend was difficult. In our initial  
248 planning for `autovi.web`, we considered implementing the entire web application using  
249 the `webr` framework (Moon 2020), which would have allowed the entire application  
250 to run directly in the user's browser. However, this approach was not feasible at the  
251 time of writing this paper. The reason is that one of the R packages `autovi` depends  
252 on the R package `splancs` (Rowlingson & Diggle 2023), which uses compiled Fortran  
253 code. A working Emscripten (Zakai 2011) version of this package, which would be  
254 required for `webr`, was not available.

255 We also explored the possibility of implementing the web interface using frameworks  
256 built on other languages, such as Python. However, server hosting domains that  
257 natively support Python servers typically do not have the latest version of R installed.  
258 Additionally, calling R from Python is typically done using the `rpy2` Python library  
259 (Gautier 2024), but this approach can be awkward when dealing with language syntax  
260 related to non-standard evaluation. Another option we considered was renting a server  
261 where we could have full control, such as those provided by cloud platforms like Google  
262 Cloud Platform (GCP) or Amazon Web Services (AWS). However, correctly setting up  
263 the server and ensuring a secure deployment requires significant expertise. Ultimately,  
264 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.web` is 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` does not use the native Python version of `TensorFlow`. Instead, it leverages `TensorFlow.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 shifts the 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 `autovi` requires downloading the pre-trained computer vision models from GitHub, these models in “.keras” file format are incompatible with `TensorFlow.js`. Therefore, we extract and store the model weights in JSON files and include them as extra resources in the Shiny application. 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 ([Attali 2021](#)) is used. This package allows calling custom 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.

## 4.2. Usage

The workflow of `autovi.web` is designed to be straightforward, 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

298 R package `datasauRus` (Davies, Locke & D'Agostino McGowan 2022). The lineup  
 299 example uses residuals from a simulated regression model that has a non-linearity  
 300 issue. We walk through the lineup example to further demonstrate the workflow of  
 301 the web application.

### 302 4.2.1. Reading data and setting parameters

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

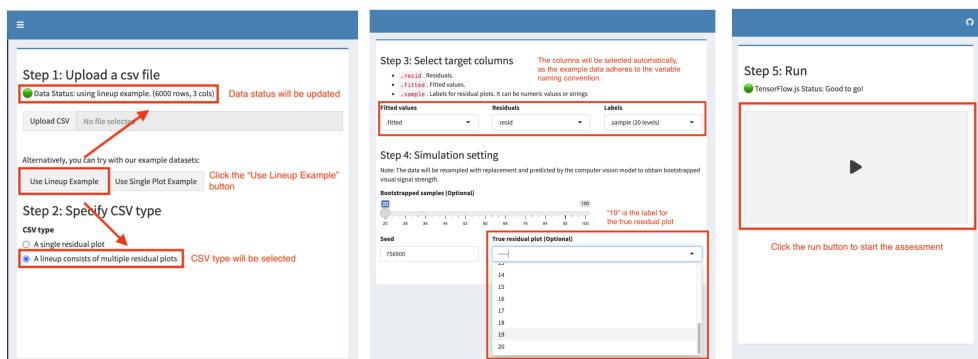


Figure 5. To begin the workflow for `autoovi` using the lineup example dataset, the user clicks the “Use Lineup Example” button (left) to load the example dataset, during which the data status and CSV type will be automatically updated. The user must manually select the label for the true residual plot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

319 **4.2.2. Results provided**

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

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

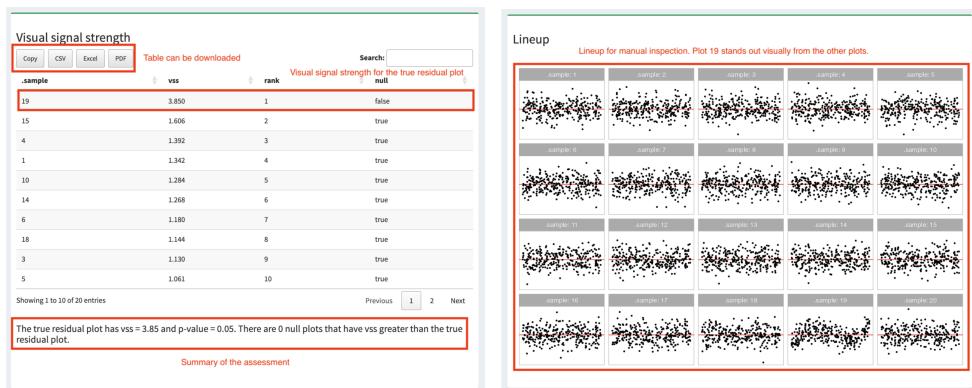


Figure 6. Results for the lineup. The VSS of the true residual plot is displayed in the first row of the table of VSS values for all the null plots (left image), with a summary text beneath the table providing the *p*-value to aid in decision-making. A lineup of residual plots allows for manual inspection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

331

## 5. Conclusions

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

337 The **autovi** R package, introduced in this paper, automates the assessment of  
 338 residual plots by incorporating a computer vision model, eliminating the need for

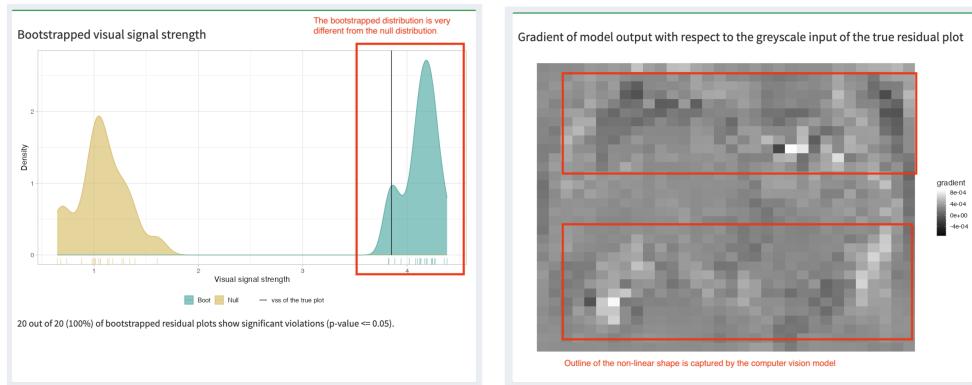


Figure 7. Summaries assessing the strength of the pattern and which elements of the plot contribute. The density plot helps verify if the bootstrapped distribution differs from the null distribution (left image). The attention map (right image) offers insights into whether the computer vision model has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

339 time-consuming and potentially inconsistent human interpretation. This automation  
 340 improves the efficiency of the diagnostic process and promotes consistency in model  
 341 evaluation across different users and studies.

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

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

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

360 **6. Resources and supplementary material**

361 The current version of `autovi` can be installed from CRAN, and source  
 362 code for both packages are available at [github.com/TengMCing/autovi](https://github.com/TengMCing/autovi) and  
 363 [github.com/TengMCing/autovi\\_web](https://github.com/TengMCing/autovi_web) respectively. The web interface is available from  
 364 [autoviweb.netlify.app](https://autoviweb.netlify.app).

365 This paper is reproducibly written using Quarto ([Allaire et al. 2024](#)) powered by  
 366 Pandoc ([MacFarlane, Krewinkel & Rosenthal 2024](#)) and pdfTeX. The full source code  
 367 to reproduce this paper is available at [github.com/TengMCing/autovi\\_paper](https://github.com/TengMCing/autovi_paper).

368 These R packages were used for the work: `tidyverse` ([Wickham et al. 2019](#)), `lmtest`  
 369 ([Zeileis & Hothorn 2002](#)), `kableExtra` ([Zhu 2021](#)), `patchwork` ([Pedersen 2022](#)),  
 370 `rcartocolor` ([Nowosad 2018](#)), `glue` ([Hester & Bryan 2022](#)), `here` ([Müller 2020](#)),  
 371 `magick` ([Ooms 2023](#)), `yardstick` ([Kuhn, Vaughan & Hvitfeldt 2024](#)) and `reticulate`  
 372 ([Ushey, Allaire & Tang 2024](#)).

373 **References**

- 374 ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G.S.,  
 375 DAVIS, A., DEAN, J., DEVIN, M. et al. (2016). Tensorflow: Large-scale machine learning on  
 376 heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .
- 377 ALLAIRE, J., TEAGUE, C., SCHEIDECKER, C., XIE, Y. & DERVIEUX, C. (2024). Quarto. doi:  
 378 10.5281/zenodo.5960048. URL <https://github.com/quarto-dev/quarto-cli>.
- 379 ATTALI, D. (2021). *shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds*.  
 380 URL <https://CRAN.R-project.org/package=shinyjs>. R package version 2.1.0.
- 381 BALAMUTA, J.J. (2024). *surreal: Create Datasets with Hidden Images in Residual Plots*. URL  
 382 <https://CRAN.R-project.org/package=surreal>. R package version 0.0.1.
- 383 BREUSCH, T.S. & PAGAN, A.R. (1979). A simple test for heteroscedasticity and random coefficient  
 384 variation. *Econometrica: Journal of the Econometric Society* , 1287–1294.
- 385 BUJA, A., COOK, D., HOFMANN, H., LAWRENCE, M., LEE, E.K., SWAYNE, D.F. & WICKHAM, H.  
 386 (2009). Statistical inference for exploratory data analysis and model diagnostics. *Philosophical  
 387 Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **367**,  
 388 4361–4383.
- 389 CHANG, W. & BORGES RIBEIRO, B. (2021). *shinydashboard: Create Dashboards with 'Shiny'*.  
 390 URL <https://CRAN.R-project.org/package=shinydashboard>. R package version 0.7.2.
- 391 CHANG, W., CHENG, J., ALLAIRE, J., SIEVERT, C., SCHLOERKE, B., XIE, Y., ALLEN, J.,  
 392 MCPHERSON, J., DIPERT, A. & BORGES, B. (2022). *shiny: Web Application Framework for  
 393 R*. URL <https://CRAN.R-project.org/package=shiny>. R package version 1.7.3.
- 394 CHENG, J., SIEVERT, C., SCHLOERKE, B., CHANG, W., XIE, Y. & ALLEN, J. (2024). *htmltools:  
 395 Tools for HTML*. URL <https://CRAN.R-project.org/package=htmltools>. R package version  
 396 0.5.8.
- 397 CLARK, A. et al. (2015). Pillow (pil fork) documentation. *readthedocs* .
- 398 COOK, R.D. & WEISBERG, S. (1982). *Residuals and influence in regression*. New York: Chapman  
 399 and Hall.

- 400 DAVIES, R., LOCKE, S. & D'AGOSTINO McGOWAN, L. (2022). *datasauRus: Datasets from the*  
 401 *Datasaurus Dozen*. URL <https://CRAN.R-project.org/package=datasauRus>. R package  
 402 version 0.1.6.
- 403 GAUTIER, L. (2024). *Python interface to the R language (embedded R)*. URL <https://pypi.org/project/rpy2/>. Version 3.5.16.
- 405 GOODE, K. & REY, K. (2019). *ggResidpanel: Panels and Interactive Versions of Diagnostic Plots*  
 406 *using 'ggplot2'*. URL <https://CRAN.R-project.org/package=ggResidpanel>. R package  
 407 version 0.3.0.
- 408 HARTIG, F. (2022). *DHARMA: Residual Diagnostics for Hierarchical (Multi-Level / Mixed)*  
 409 *Regression Models*. URL <https://CRAN.R-project.org/package=DHARMA>. R package  
 410 version 0.4.6.
- 411 HEBBALI, A. (2024). *olsrr: Tools for Building OLS Regression Models*. URL <https://CRAN.R-project.org/package=olsrr>. R package version 0.6.0.
- 413 HESTER, J. & BRYAN, J. (2022). *glue: Interpreted String Literals*. URL <https://CRAN.R-project.org/package=glue>. R package version 1.6.2.
- 415 JOHNSON, P.E. (2022). *rockchalk: Regression Estimation and Presentation*. URL <https://CRAN.R-project.org/package=rockchalk>. R package version 1.8.157.
- 417 KUHN, M., VAUGHAN, D. & HVITFELDT, E. (2024). *yardstick: Tidy Characterizations of Model*  
 418 *Performance*. URL <https://CRAN.R-project.org/package=yardstick>. R package version  
 419 1.3.1.
- 420 LI, W. (2024). *bandicoot: Light-weight python-like object-oriented system*. URL <https://CRAN.R-project.org/package=bandicoot>.
- 422 LI, W., COOK, D., TANAKA, E. & VANDERPLAS, S. (2024a). A plot is worth a thousand tests:  
 423 Assessing residual diagnostics with the lineup protocol. *Journal of Computational and*  
 424 *Graphical Statistics* **33**, 1497–1511. doi:10.1080/10618600.2024.2344612.
- 425 LI, W., COOK, D., TANAKA, E., VANDERPLAS, S. & ACKERMANN, K. (2024b). Automated  
 426 assessment of residual plots with computer vision models. *arXiv preprint arXiv:2411.01001*.
- 427 LONG, J.A. (2022). *jtools: Analysis and Presentation of Social Scientific Data*. URL <https://cran.r-project.org/package=jtools>. R package version 2.2.0.
- 429 LOY, A. & HOFMANN, H. (2014). *Hlmdiag: A suite of diagnostics for hierarchical linear models in*  
 430 *r*. *Journal of Statistical Software* **56**, 1–28.
- 431 MACFARLANE, J., KREWINKEL, A. & ROSENTHAL, J. (2024). Pandoc. URL <https://github.com/jgm/pandoc>.
- 433 MASON, H., LEE, S., LAA, U. & COOK, D. (2022). *cassowaryr: Compute Scagnostics on Pairs of*  
 434 *Numeric Variables in a Data Set*. URL <https://CRAN.R-project.org/package=cassowary>. R  
 435 package version 2.0.0.
- 436 MOON, K.W. (2020). *webr: Data and Functions for Web-Based Analysis*. URL <https://CRAN.R-project.org/package=webr>. R package version 0.1.5.
- 438 MÜLLER, K. (2020). *here: A simpler way to find your files*. URL <https://CRAN.R-project.org/package=here>. R package version 1.0.1.
- 440 NOWOSAD, J. (2018). 'CARTOCOLORs' palettes. URL <https://nowosad.github.io/rkartocolor>. R  
 441 package version 1.0.
- 442 OOMS, J. (2023). *magick: Advanced Graphics and Image-Processing in R*. URL <https://CRAN.R-project.org/package=magick>. R package version 2.7.4.
- 444 PEDERSEN, T.L. (2022). *patchwork: The composer of plots*. URL <https://CRAN.R-project.org/package=patchwork>. R package version 1.1.2.
- 446 R CORE TEAM (2022). *R: A Language and Environment for Statistical Computing*. R Foundation  
 447 for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.

- 448 RAMSEY, J.B. (1969). Tests for specification errors in classical linear least-squares regression  
449 analysis. *Journal of the Royal Statistical Society: Series B (Methodological)* **31**, 350–371.
- 450 REINHART, A. (2024). *regressinator: Simulate and Diagnose (Generalized) Linear Models*. URL  
451 <https://CRAN.R-project.org/package=regressinator>. R package version 0.2.0.
- 452 ROWLINGSON, B. & DIGGLE, P. (2023). *splancs: Spatial and Space-Time Point Pattern Analysis*.  
453 URL <https://CRAN.R-project.org/package=splancs>. R package version 2.01-44.
- 454 SALI, A. & ATTALI, D. (2020). *shinyCSSloaders: Add Loading Animations to a 'shiny' Output*  
455 While It's Recalculating. URL <https://CRAN.R-project.org/package=shinyCSSloaders>. R  
456 package version 1.0.0.
- 457 SHAPIRO, S.S. & WILK, M.B. (1965). An analysis of variance test for normality (complete samples).  
458 *Biometrika* **52**, 591–611.
- 459 USHEY, K., ALLAIRE, J. & TANG, Y. (2024). *reticulate: Interface to 'Python'*. URL <https://CRAN.R-project.org/package=reticulate>. R package version 1.35.0.
- 460 WARTON, D.I. (2023). Global simulation envelopes for diagnostic plots in regression models. *The*  
461 *American Statistician* **77**, 425–431.
- 462 WICKHAM, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York.  
463 URL <https://ggplot2.tidyverse.org>.
- 464 WICKHAM, H., AVERICK, M., BRYAN, J., CHANG, W., McGOWAN, L.D., FRANÇOIS, R.,  
465 GROLEMUND, G., HAYES, A., HENRY, L., HESTER, J., KUHN, M., PEDERSEN, T.L., MILLER,  
466 E., BACHE, S.M., MÜLLER, K., OOMS, J., ROBINSON, D., SEIDEL, D.P., SPINU, V.,  
467 TAKAHASHI, K., VAUGHAN, D., WILKE, C., WOO, K. & YUTANI, H. (2019). Welcome to  
468 the tidyverse. *Journal of Open Source Software* **4**, 1686. doi:10.21105/joss.01686.
- 469 WICKHAM, H., CHOWDHURY, N.R., COOK, D. & HOFMANN, H. (2020). *nullabor: Tools for*  
470 *Graphical Inference*. URL <https://CRAN.R-project.org/package=nullabor>. R package  
471 version 0.3.9.
- 472 ZAKAI, A. (2011). Emscripten: an llvm-to-javascript compiler. In *Proceedings of the ACM*  
473 *international conference companion on Object oriented programming systems languages and*  
474 *applications companion*. pp. 301–312.
- 475 ZEILEIS, A. & HOTHORN, T. (2002). Diagnostic checking in regression relationships. *R News* **2**,  
476 7–10.
- 477 ZHU, H. (2021). *kableExtra: Construct complex table with kable and pipe syntax*. URL  
478 <https://CRAN.R-project.org/package=kableExtra>. R package version 1.3.4.