

# 1 Automated Residual Plot Assessment with the R Package 2 autovi and the Shiny App autovi.web

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

5 Monash University, The Australian National University and University of  
6 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

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` (Zeileis & Hothorn 2002) provides a 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)

14 diagnostic plots such as residuals vs. fitted values, quantile-quantile (Q-Q) plots, and  
15 residuals vs. leverage plots. Packages like `jtools` (Long 2022), `olsrr` (Hebbali 2024),  
16 `rockchalk` (Johnson 2022), and `ggResidpanel` (Goode & Rey 2019) provide similar  
17 graphical diagnostics, often with alternative aesthetics or interactive features. All of  
18 these tools deliver the types of diagnostic plots outlined in the classical text by Cook  
19 & Weisberg (1982). The `ecostats` package (Warton 2023) incorporates simulation  
20 envelopes into residual plots, while DHARMA (Hartig 2022) compares empirical quantiles  
21 (0.25, 0.5, and 0.75) of scaled residuals to their theoretical counterparts. DHARMA is  
22 particularly focused on detecting model violations such as heteroscedasticity, incorrect  
23 functional forms, and issues specific to generalized linear and mixed-effect models, like  
24 over/under-dispersion. It also includes conventional test annotations to help avoid  
25 misinterpretation.

26 However, relying solely on subjective assessments of these plots can lead to issues such as  
27 over-interpreting random patterns as model violations. Li et al. (2024a) demonstrated  
28 that visual methods using the lineup protocol (Buja et al. 2009) for assessing residuals  
29 are more useful, and also perform more practically than conventional tests due to their  
30 reduced sensitivity to minor departures. Packages such as `nullabor` (Wickham et al.  
31 2020), `HLMdiag` (Loy & Hofmann 2014), and `regressinator` (Reinhart 2024), enable  
32 users to compare observed residual plots with plots of samples from null distributions,  
33 helping to quantify the significance of any detected patterns.

34 However, as discussed in Li et al. (2024b), the lineup protocol has significant limitations  
35 in large-scale applications due to dependence on human labor. Thus, a computer vision  
36 model was developed with an associated statistical testing procedure to automate  
37 the assessment of residual plots. This model takes a residual plot and a vector of  
38 auxiliary variables (such as the number of observations) as inputs and outputs the  
39 predicted visual signal strength (VSS). This strength estimates the distance between  
40 the residual distribution of the fitted regression model and the reference distribution  
41 assumed under correct model specification.

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

46 The remainder of this paper is structured as follows: Section 2 introduces the definition  
47 and computation of visual signal strength. Section 3 provides a detailed documentation  
48 of the `autovi` package, including its usage and infrastructure. Section 4 focuses on the

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

51 **2. Definition and computation of visual signal strength**

52 To train a computer vision model, a measure of the visible pattern in a plot is needed.  
 53 We call this the **visual signal strength** (VSS), which measures how prominently a  
 54 specific set of visual patterns appears in an image. This can be computed for a training  
 55 set of data, and plots, where the generating distributions are specified.  
 56 In the context of regression model diagnostics, VSS describes the clarity of visual  
 57 patterns on a diagnostic plot that may indicate model violations. Violations can be  
 58 categorized as weak, moderate, or strong, but here we treat it as a continuous positive  
 59 real variable. Importantly, its interpretation depends on how it is linked to a function  
 60 of the data or the underlying data generating process. Consequently, the calculation of  
 61 VSS can be vary across different model classes or within the same model, depending  
 62 on the generating function.  
 63 VSS is an estimate of the distance between the residual distribution of a fitted classical  
 64 normal linear regression model and a reference distribution; more details can be found  
 65 in Li et al. (2024b). The distance measure is based on the Kullback-Leibler (KL)  
 66 divergence:

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

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

$$D_{KL} = \int_{\mathbb{R}^n} \log \frac{p(\mathbf{e})}{q(\mathbf{e})} p(\mathbf{e}) d\mathbf{e}, \quad (1)$$

68 here,  $p(\cdot)$  and  $q(\cdot)$  are the probability density functions of the reference residual  
 69 distribution  $P$  and the true residual distribution  $Q$ , respectively.  
 70 Equation 1 can be adapted to different forms to account for various model violations.  
 71 For instance, in the case of a linear regression model where necessary higher-order  
 72 predictors,  $\mathbf{Z}$ , and their corresponding parameter,  $\beta_Z$ , are omitted, the distance  
 73 measure can be expanded as follows:

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

74 where  $\boldsymbol{\mu}_z = \mathbf{R}\mathbf{Z}\boldsymbol{\beta}_z$ ,  $\mathbf{R} = \mathbf{I}_n - \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$  and  $\mathbf{X}$  is the design matrix of the  
75 regression model.

76 This distance measure depends on knowledge of the true residual distribution, which  
77 is often unknown. However, it can be estimated by identifying or approximating a  
78 mapping from a set of residuals to its corresponding distance measure. The computer  
79 vision model presented in [Li et al. \(2024b\)](#) approximates this mapping. It is trained  
80 on a large number of synthetic regression models, where the data-generating process  
81 is known, allowing the distance measure to be explicitly calculated. The model takes a  
82 residual plot as input and outputs the corresponding distance measure. Additional  
83 details are provided in [Li et al. \(2024b\)](#).

### 84 3. R package: autovi

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

#### 92 3.1. Motivation

93 Figure 1 shows three sets of plots of residuals against fitted values. The simulated  
94 example in (a) might be interpreted as a heteroscedastic pattern, however the  
95 automated reading would predict this to have a visual signal strength (VSS) of  
96 1.53, with a corresponding *p*-value of 0.25. This means it would be interpreted as  
97 a good residual plot, that there is nothing in the data to indicate a violation of  
98 model assumptions. Skewness in the predictor variables is generating the apparent  
99 heteroscedasticity, where the smaller variance in residuals at larger fitted values is  
100 due to smaller sample size only. The Breusch-Pagan test ([Breusch & Pagan 1979](#)) for  
101 heteroscedasticity would also not reject this as good residual plot.

102 The data in (b) is generated by fitting a linear model predicting `mpg` based on `hp`  
103 using the `datasets::mtcars`. It is a small data set, and there is a hint of nonlinear

104 structure not captured by the model. The automated plot reading would predict a  
 105 VSS of 3.57, which has a  $p$ -value less than 0.05. That is, the nonlinear structure is  
 106 most likely real, and indicates a problem with the model. The conventional test, a  
 107 Ramsey Regression Equation Specification Error Test (RESET) (Ramsey 1969) would  
 108 also strongly detect the nonlinearity.

109 The third example is generated using the `surreal` package (Balamuta 2024) where  
 110 structured residuals are hidden in data, to be revealed if the correct model is specified.  
 111 Here a quote based on Tukey is used as the residual structure “visual summaries focus  
 112 on unexpected values”. The automated plot reading predicts the VSS to be 5.87, with  
 113 a  $p$ -value less than 0.05. This structure is blindingly obvious visually, but a RESET  
 114 test for nonlinear structure would not report a problem. (It would be detected by  
 115 a Breusch-Pagan for heteroscedasticity and also Shapiro-Wilk test (Shapiro & Wilk  
 116 1965) for non-normality.)

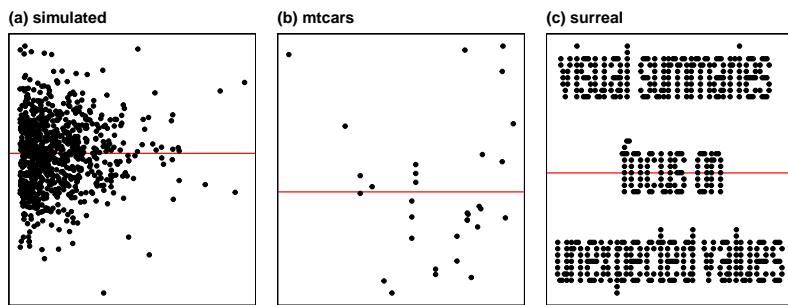


Figure 1. Reading residual plots can be a difficult task, particularly for students new to statistical modeling. The `autovi` package makes it easier. Here are three examples of residual plots, which may appear to have structure. According to `autovi`, the visual signal strengths (VSS) of these three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

### 117 3.2. Implementation

118 The `autovi` package is built on the `bandicoot` object-oriented programming (OOP)  
 119 system (Li 2024), marking a departure from R’s traditional S3 generic system. This  
 120 OOP architecture enhances flexibility and modularity, allowing users to redefine key  
 121 functions through method overriding.

122 The `autovi` infrastructure effectively integrates multiple programming languages and  
 123 libraries into a comprehensive analytical tool. It relies on five core libraries from  
 124 Python and R, each playing a critical role in the analysis pipeline. In Python, `pillow`

125 ([Clark et al. 2015](#)) handles image processing tasks such as reading and resizing PNG  
126 files of residual plots, then converting them into input tensors for further analysis.  
127 [TensorFlow](#) ([Abadi et al. 2016](#)), a key component of modern machine learning, is used  
128 to predict the VSS of these plots using a pre-trained convolutional neural network.

129 In the R environment, `autovi` utilizes several libraries. `ggplot2` ([Wickham 2016](#))  
130 generates the initial residual plots, saved as PNG files for visual input. `cassowaryr`  
131 ([Mason et al. 2022](#)) computes scagnostics (scatter plot diagnostics), providing numerical  
132 features that capture statistical properties of the plots. These scagnostics complement  
133 the visual analysis by offering quantitative metrics as secondary input to the  
134 computer vision model. `reticulate` ([Ushey, Allaire & Tang 2024](#)) enables seamless  
135 communication between R and Python.

### 136 **3.3. Installation**

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

144 We recommend using the Shiny app `autovi.web` if users encounter installation  
145 problems.

### 146 **3.4. Usage**

#### 147 **3.4.1. Numerical summary**

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

- 150 1. Load the `autovi` package using the `library()` function.
- 151 2. Create a checker object with a linear regression model.
- 152 3. Call the `check()` method of the checker, which, by default, predicts the VSS for  
153 the true residual plot, 100 null plots, and 100 bootstrapped plots. The method  
154 stores the predictions internally and prints a concise results report.

155 The code to do this is:

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

156 It produces the following summary:

157

```
158 -- <AUTO_VI object>
159 Status:
160 - Fitted model: lm
161 - Keras model: UNKNOWN
162 - Output node index: 1
163 - Result:
164 - Observed visual signal strength: 3.162 (p-value = 0.0396)
165 - Null visual signal strength: [100 draws]
166 - Mean: 1.274
167 - Quantiles:
168
169      25%    50%    75%    80%    90%    95%    99%
170 0.8021 1.1109 1.5751 1.6656 1.9199 2.6564 3.3491
171
172 - Bootstrapped visual signal strength: [100 draws]
173 - Mean: 2.786 (p-value = 0.05941)
174 - Quantiles:
175
176      25%    50%    75%    80%    90%    95%    99%
177 2.452 2.925 3.173 3.285 3.463 3.505 3.652
178
179 - Likelihood ratio: 0.7275 (boot) / 0.06298 (null) = 11.55
```

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

186 than the observed VSS in both the bootstrapped data values and the simulated null  
 187 values.

188 **3.4.2. Visual summary**

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

```
checker$plot_pair()
```

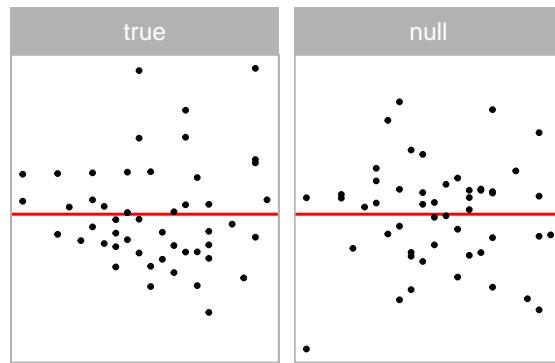


Figure 2. True plot alongside one null plot, for quick comparison.

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

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

```
checker$summary_plot()
```

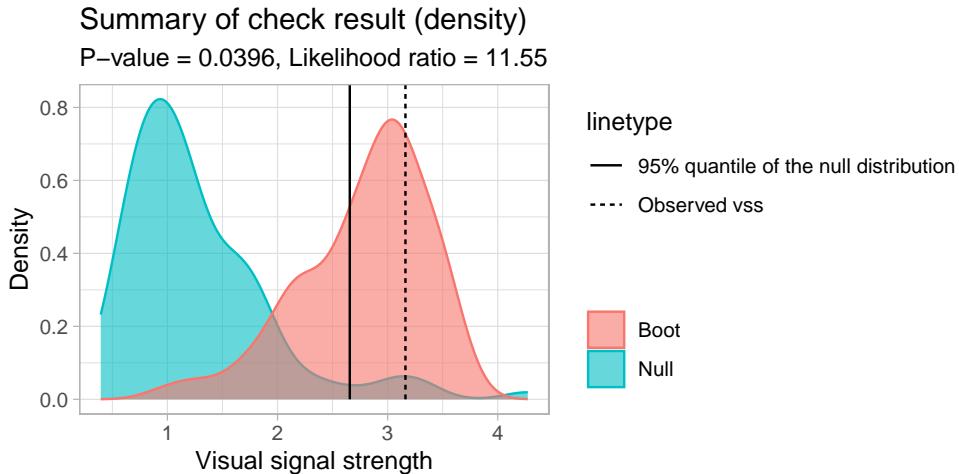


Figure 3. Summary plot comparing the densities of VSS for bootstrapped residual samples (red) relative to VSS for null plots (blue).

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

### 215 3.5. Modularized infrastructure

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

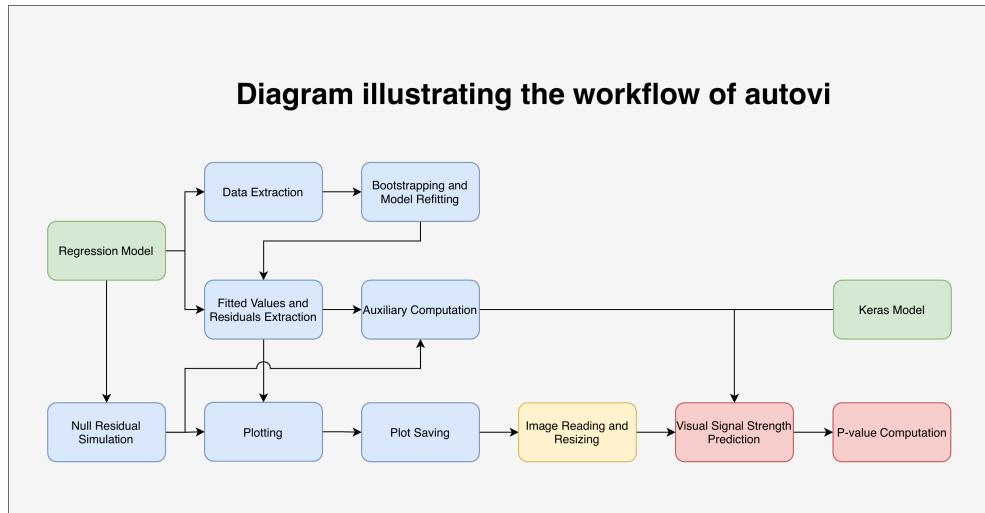


Figure 4. Diagram illustrating the infrastructure of the R package **autovi**. The modules in green are primary inputs provided by users. Modules in blue are overridable methods that can be modified to accommodate users' specific needs. The module in yellow is a pre-defined non-overridable method. The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

238

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

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

247 

### 4.1. Implementation

248 The interface `autovi.web` is built using the `shiny` (Chang et al. 2022) and  
249 `shinydashboard` (Chang & Borges Ribeiro 2021) R packages. Hosted on the  
250 `shinyapps.io` domain, the application is accessible through any modern web browser.  
251 The R packages `htmltools` (Cheng et al. 2024) and `shinycssloaders` (Sali & Attali  
252 2020) are used to render markdown documentation in shiny application, and for loading  
253 animations for shiny widgets, respectively.

254 Determining the best way to implement the backend was difficult. In our initial  
255 planning for `autovi.web`, we considered implementing the entire web application using  
256 the `webr` framework (Moon 2020), which would have allowed the entire application to  
257 run directly in the user’s browser. However, `webr` does not support packages which use  
258 compiled fortran code, which is required by `splancs` (Rowlingson & Diggle 2023), a  
259 dependency of `autovi`. In the future, it is possible that a working Emscripten (Zakai  
260 2011) version of this package may allow full `webr` support.

261 We also explored the possibility of implementing the web interface using frameworks  
262 built on other languages, such as Python. However, server hosting domains that  
263 natively support Python servers typically do not have the latest version of R installed.  
264 Additionally, calling R from Python is typically done using the `rpy2` Python library  
265 (Gautier 2024), but this approach can be awkward when dealing with language syntax  
266 related to non-standard evaluation. Another option we considered was renting a server  
267 where we could have full control, such as those provided by cloud platforms like  
268 Google Cloud Platform (GCP) or Amazon Web Services (AWS). However, deploying  
269 and maintaining the server securely requires some expertise. Ultimately, the most  
270 practical solution was to use the `shiny` and `shinydashboard` frameworks, which are

271 well-established in the R community and offer a solid foundation for web application  
272 development.

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

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

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

289 While `autovi` requires downloading the pre-trained computer vision models from  
290 GitHub, these models in “.keras” file format are incompatible with `TensorFlow.js`.  
291 Therefore, we extract and store the model weights in JSON files and include  
292 them as extra resources in the Shiny application. When the application initializes,  
293 `TensorFlow.js` rebuilds the computer vision model using these pre-stored weights.

294 To allow communication between `TensorFlow.js` and other components of the Shiny  
295 application, the `shinyjs` R package ([Attali 2021](#)) is used. This package allows calling  
296 custom JavaScript code within the Shiny framework. The specialized JavaScript  
297 code for initializing `TensorFlow.js` and calling `TensorFlow.js` for VSS prediction is  
298 deployed alongside the Shiny application as additional resources.

## 299 4.2. Usage

300 The workflow of `autovi.web` is designed to be straightforward, with numbered  
301 steps displayed in each panel. There are two example datasets provided by the  
302 web application. The single residual plot example uses the `dino` dataset from the  
303 R package `datasauRus` ([Davies, Locke & D'Agostino McGowan 2022](#)). 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.

#### 4.2.1. 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, `.sample` to 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.

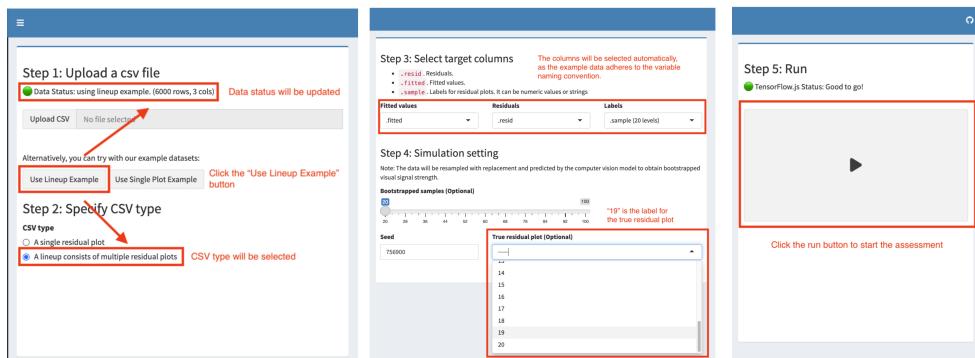


Figure 5. To begin the workflow for `autovi` using the lineup example dataset, the user clicks the “Use Lineup Example” button (left) to load the example dataset, during which the data status and CSV type will be automatically updated. The user must manually select the label for the true residual plot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

324 **4.2.2. Results provided**

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

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

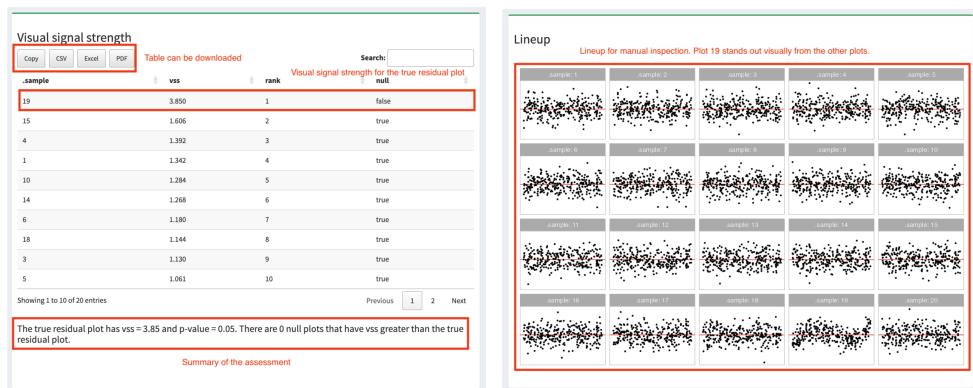


Figure 6. Results for the lineup. The VSS of the true residual plot is displayed in the first row of the table of VSS values for all the null plots (left image), with a summary text beneath the table providing the *p*-value to aid in decision-making. A lineup of residual plots allows for manual inspection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

336

## 5. Conclusions

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

342 The **autovi** R package, introduced in this paper, automates the assessment of residual  
 343 plots by incorporating a computer vision model, reducing reliance on time-consuming

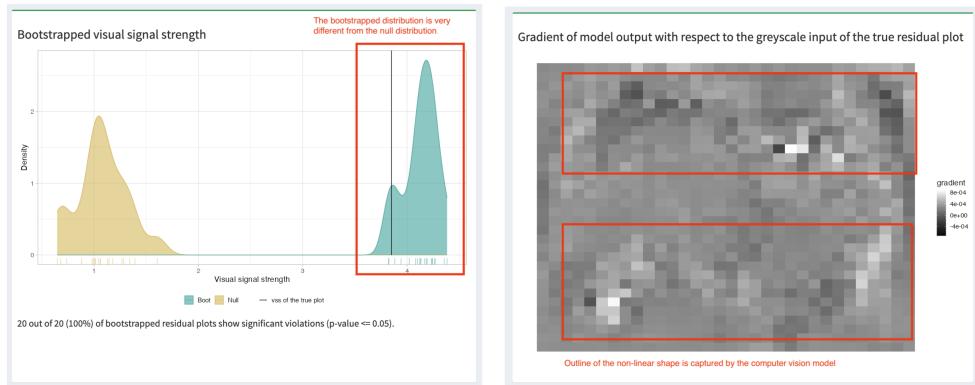


Figure 7. Summaries assessing the strength of the pattern and which elements of the plot contribute. The density plot helps verify if the bootstrapped distribution differs from the null distribution (left image). The attention map (right image) offers insights into whether the computer vision model has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

344 and potentially inconsistent human interpretation. This automation improves the  
 345 efficiency of the diagnostic process and promotes consistency in model evaluation  
 346 across different users and studies.

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

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

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

365 **6. Resources and supplementary material**

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

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

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

378 **References**

- 379 ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G.S.,  
 380 DAVIS, A., DEAN, J., DEVIN, M. et al. (2016). Tensorflow: Large-scale machine learning on  
 381 heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .
- 382 ALLAIRE, J., TEAGUE, C., SCHEIDECKER, C., XIE, Y. & DERVIEUX, C. (2024). Quarto. doi:  
 383 10.5281/zenodo.5960048. URL <https://github.com/quarto-dev/quarto-cli>.
- 384 ATTALI, D. (2021). *shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds*.  
 385 URL <https://CRAN.R-project.org/package=shinyjs>. R package version 2.1.0.
- 386 BALAMUTA, J.J. (2024). *surreal: Create Datasets with Hidden Images in Residual Plots*. URL  
 387 <https://CRAN.R-project.org/package=surreal>. R package version 0.0.1.
- 388 BREUSCH, T.S. & PAGAN, A.R. (1979). A simple test for heteroscedasticity and random coefficient  
 389 variation. *Econometrica: Journal of the Econometric Society* , 1287–1294.
- 390 BUJA, A., COOK, D., HOFMANN, H., LAWRENCE, M., LEE, E.K., SWAYNE, D.F. & WICKHAM, H.  
 391 (2009). Statistical inference for exploratory data analysis and model diagnostics. *Philosophical  
 392 Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **367**,  
 393 4361–4383.
- 394 CHANG, W. & BORGES RIBEIRO, B. (2021). *shinydashboard: Create Dashboards with 'Shiny'*.  
 395 URL <https://CRAN.R-project.org/package=shinydashboard>. R package version 0.7.2.
- 396 CHANG, W., CHENG, J., ALLAIRE, J., SIEVERT, C., SCHLOERKE, B., XIE, Y., ALLEN, J.,  
 397 MCPHERSON, J., DIPERT, A. & BORGES, B. (2022). *shiny: Web Application Framework for  
 398 R*. URL <https://CRAN.R-project.org/package=shiny>. R package version 1.7.3.
- 399 CHENG, J., SIEVERT, C., SCHLOERKE, B., CHANG, W., XIE, Y. & ALLEN, J. (2024). *htmltools:  
 400 Tools for HTML*. URL <https://CRAN.R-project.org/package=htmltools>. R package version  
 401 0.5.8.
- 402 CLARK, A. et al. (2015). Pillow (pil fork) documentation. *readthedocs* .
- 403 COOK, R.D. & WEISBERG, S. (1982). *Residuals and influence in regression*. New York: Chapman  
 404 and Hall.

- 405 DAVIES, R., LOCKE, S. & D'AGOSTINO McGOWAN, L. (2022). *datasauRus: Datasets from the*  
 406 *Datasaurus Dozen*. URL <https://CRAN.R-project.org/package=datasauRus>. R package  
 407 version 0.1.6.
- 408 GAUTIER, L. (2024). *Python interface to the R language (embedded R)*. URL <https://pypi.org/project/rpy2/>. Version 3.5.16.
- 409 GOODE, K. & REY, K. (2019). *ggResidpanel: Panels and Interactive Versions of Diagnostic Plots*  
 410 *using 'ggplot2'*. URL <https://CRAN.R-project.org/package=ggResidpanel>. R package version  
 411 0.3.0.
- 412 HARTIG, F. (2022). *DHARMA: Residual Diagnostics for Hierarchical (Multi-Level / Mixed)*  
 413 *Regression Models*. URL <https://CRAN.R-project.org/package=DHARMA>. R package  
 414 version 0.4.6.
- 415 HEBBALI, A. (2024). *olsrr: Tools for Building OLS Regression Models*. URL <https://CRAN.R-project.org/package=olsrr>. R package version 0.6.0.
- 416 HESTER, J. & BRYAN, J. (2022). *glue: Interpreted String Literals*. URL <https://CRAN.R-project.org/package=glue>. R package version 1.6.2.
- 417 JOHNSON, P.E. (2022). *rockchalk: Regression Estimation and Presentation*. URL <https://CRAN.R-project.org/package=rockchalk>. R package version 1.8.157.
- 418 KUHN, M., VAUGHAN, D. & HVITFELDT, E. (2024). *yardstick: Tidy Characterizations of Model*  
 419 *Performance*. URL <https://CRAN.R-project.org/package=yardstick>. R package version 1.3.1.
- 420 LI, W. (2024). *bandicoot: Light-weight python-like object-oriented system*. URL <https://CRAN.R-project.org/package=bandicoot>.
- 421 LI, W., COOK, D., TANAKA, E. & VANDERPLAS, S. (2024a). A plot is worth a thousand tests:  
 422 Assessing residual diagnostics with the lineup protocol. *Journal of Computational and*  
 423 *Graphical Statistics* **33**, 1497–1511. doi:10.1080/10618600.2024.2344612.
- 424 LI, W., COOK, D., TANAKA, E., VANDERPLAS, S. & ACKERMANN, K. (2024b). Automated  
 425 assessment of residual plots with computer vision models. *arXiv preprint arXiv:2411.01001*.
- 426 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.
- 427 LOY, A. & HOFMANN, H. (2014). *Hlmdiag: A suite of diagnostics for hierarchical linear models in*  
 428 *r*. *Journal of Statistical Software* **56**, 1–28.
- 429 MACFARLANE, J., KREWINKEL, A. & ROSENTHAL, J. (2024). Pandoc. URL <https://github.com/jgm/pandoc>.
- 430 MASON, H., LEE, S., LAA, U. & COOK, D. (2022). *cassowaryr: Compute Scagnostics on Pairs of*  
 431 *Numeric Variables in a Data Set*. URL <https://CRAN.R-project.org/package=cassowary>. R  
 432 package version 2.0.0.
- 433 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.
- 434 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.
- 435 NOWOSAD, J. (2018). 'CARTOCOLORs' palettes. URL <https://nowosad.github.io/rcartocolor>. R  
 436 package version 1.0.
- 437 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.
- 438 PEDERSEN, T.L. (2022). *patchwork: The composer of plots*. URL <https://CRAN.R-project.org/package=patchwork>. R package version 1.1.2.
- 439 R CORE TEAM (2022). *R: A Language and Environment for Statistical Computing*. R Foundation  
 440 for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- 441 RAMSEY, J.B. (1969). Tests for specification errors in classical linear least-squares regression  
 442 analysis. *Journal of the Royal Statistical Society: Series B (Methodological)* **31**, 350–371.

- 454 REINHART, A. (2024). *regressinator: Simulate and Diagnose (Generalized) Linear Models*. URL  
455 <https://CRAN.R-project.org/package=regressinator>. R package version 0.2.0.
- 456 ROWLINGSON, B. & DIGGLE, P. (2023). *splancs: Spatial and Space-Time Point Pattern Analysis*.  
457 URL <https://CRAN.R-project.org/package=splancs>. R package version 2.01-44.
- 458 SALI, A. & ATTALI, D. (2020). *shinycssloaders: Add Loading Animations to a 'shiny' Output  
459 While It's Recalculating*. URL <https://CRAN.R-project.org/package=shinycssloaders>. R  
460 package version 1.0.0.
- 461 SHAPIRO, S.S. & WILK, M.B. (1965). An analysis of variance test for normality (complete samples).  
*Biometrika* **52**, 591–611.
- 462 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.
- 463 WARTON, D.I. (2023). Global simulation envelopes for diagnostic plots in regression models. *The  
464 American Statistician* **77**, 425–431.
- 465 WICKHAM, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York.  
466 URL <https://ggplot2.tidyverse.org>.
- 467 WICKHAM, H., AVERICK, M., BRYAN, J., CHANG, W., MCGOWAN, L.D., FRANÇOIS, R.,  
468 GROLEMUND, G., HAYES, A., HENRY, L., HESTER, J., KUHN, M., PEDERSEN, T.L., MILLER,  
469 E., BACHE, S.M., MÜLLER, K., OOMS, J., ROBINSON, D., SEIDEL, D.P., SPINU, V.,  
470 TAKAHASHI, K., VAUGHAN, D., WILKE, C., WOO, K. & YUTANI, H. (2019). Welcome to  
471 the tidyverse. *Journal of Open Source Software* **4**, 1686. doi:10.21105/joss.01686.
- 472 WICKHAM, H., CHOWDHURY, N.R., COOK, D. & HOFMANN, H. (2020). *nullabor: Tools for  
473 Graphical Inference*. URL <https://CRAN.R-project.org/package=nullabor>. R package version  
474 0.3.9.
- 475 ZAKAI, A. (2011). Emscripten: an llvm-to-javascript compiler. In *Proceedings of the ACM  
476 international conference companion on Object oriented programming systems languages and  
477 applications companion*. pp. 301–312.
- 478 ZEILEIS, A. & HOTHORN, T. (2002). Diagnostic checking in regression relationships. *R News* **2**,  
479 7–10.
- 480 ZHU, H. (2021). *kableExtra: Construct complex table with kable and pipe syntax*. URL  
481 <https://CRAN.R-project.org/package=kableExtra>. R package version 1.3.4.