

# 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 2 introduces the definition and computation of visual signal strength. Section 3 provides a detailed documentation 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 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:

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

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 This distance measure depends on knowledge of the true residual distribution, which  
 71 is often unknown. However, it can be estimated by identifying or approximating a  
 72 mapping from a set of residuals to its corresponding distance measure. The computer  
 73 vision model presented in Li et al. (2024b) approximates this mapping. It is trained  
 74 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 [Li et al. \(2024b\)](#).

### 78           3. 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 **a**utomated **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.

#### 86       3.1. 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 ([Breusch & Pagan 1979](#)) 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) ([Ramsey 1969](#)) would also strongly detect the nonlinearity.

The third example is generated using the `surreal` package ([Balamuta 2024](#)) 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 than 0.05. This structure is blindingly obvious visually, but a RESET

108 test for nonlinear structure would not report a problem. (It would be detected by  
 109 a Breusch-Pagan for heteroscedasticity and also Shapiro-Wilk test ([Shapiro & Wilk](#)  
 110 [1965](#)) for non-normality.)

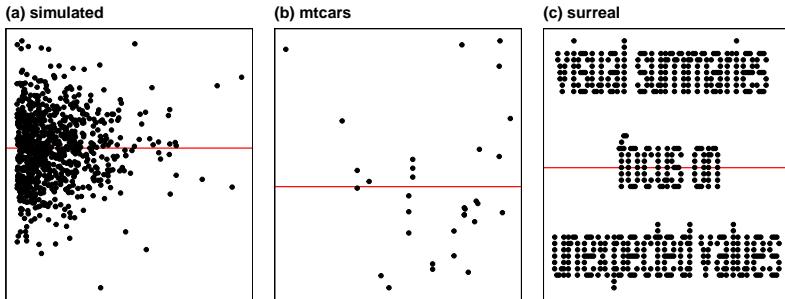


Figure 1. Reading residual plots can be a difficult task, particularly for students new to statistical modeling. The `autovi` package makes it easier. Here are three examples of residual plots, which may appear to have structure. According to `autovi`, the visual signal strengths (VSS) of these three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

### 111 3.2. Implementation

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

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

123 In the R environment, `autovi` utilizes several libraries. `ggplot2` ([Wickham 2016](#))  
 124 generates the initial residual plots, saved as PNG files for visual input. `cassowaryr`  
 125 ([Mason et al. 2022](#)) computes scagnostics (scatter plot diagnostics), providing numerical  
 126 features that capture statistical properties of the plots. These scagnostics complement  
 127 the visual analysis by offering quantitative metrics as secondary input to the

128 computer vision model. `reticulate` (Ushey, Allaire & Tang 2024) enables seamless  
 129 communication between R and Python.

130 **3.3. Installation**

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

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

140 **3.4. Usage**

141 **3.4.1. Numerical summary**

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

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

149 The code to do this is:

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

150 It produces the following summary:

```
151
152 -- <AUTO_VI object>
153 Status:
154 - Fitted model: lm
```

```

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

```

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

### 182 **3.4.2. Visual summary**

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

```
checker$plot_pair()
```

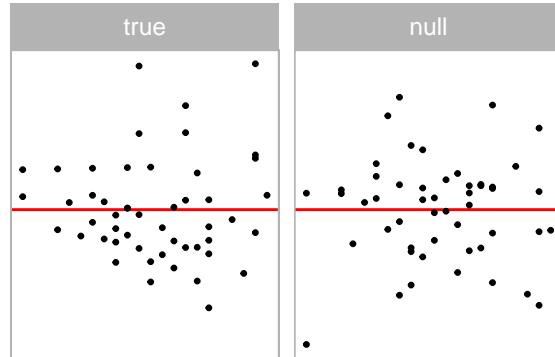


Figure 2. True plot alongside one null plot, for quick comparison.

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

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

```
checker$summary_plot()
```

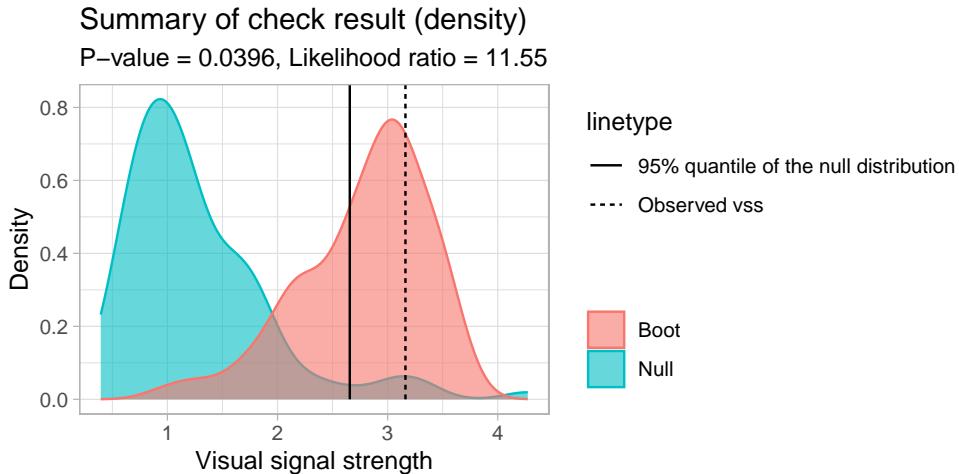


Figure 3. Summary plot comparing the densities of VSS for bootstrapped residual samples (red) relative to VSS for null plots (blue).

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

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

207 This visual summary provides an intuitive way to assess the model's fit and potential  
 208 violations, allowing users to quickly grasp the results of the automated analysis. XXX  
 209 It might be a good idea to include a basic message explaining the plot and how it's  
 210 used when it is generated, because a lot of non-statisticians aren't going to be used to  
 211 interpreting these plots XXX

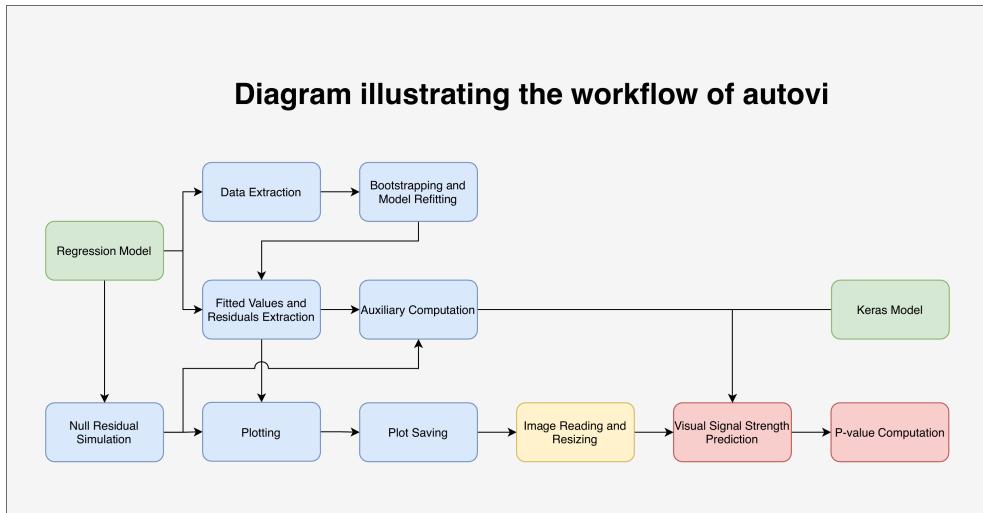


Figure 4. Diagram illustrating the infrastructure of the R package **autovi**. The modules in green are primary inputs provided by users. Modules in blue are overridable methods that can be modified to accommodate users' specific needs. The module in yellow is a pre-defined non-overridable method. The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

### 212 3.5. Modularized infrastructure

213 The initial motivation for developing **autovi** was to create a convenient interface for  
 214 sharing the models described and trained in [Li et al. \(2024b\)](#). However, recognizing  
 215 that the classical normal linear regression model represents a restricted class of  
 216 models, we sought to avoid limiting the potential for future extensions, whether by  
 217 the original developers or other developers. As a result, the package was designed to  
 218 function seamlessly with linear regression models with minimal modification and few  
 219 required arguments, while also accommodating other classes of models through partial  
 220 infrastructure substitution. This modular and customizable design allows **autovi** to  
 221 handle a wide range of residual diagnostics tasks.

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

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

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

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

##### 244 4.1. Implementation

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

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

258 We also explored the possibility of implementing the web interface using frameworks  
259 built on other languages, such as Python. However, server hosting domains that  
260 natively support Python servers typically do not have the latest version of R installed.  
261 Additionally, calling R from Python is typically done using the `rpy2` Python library

(Gautier 2024), 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.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` 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.

**296 4.2. Usage**

297 The workflow of `autovi.web` is designed to be straightforward, with numbered  
298 steps displayed in each panel. There are two example datasets provided by the  
299 web application. The single residual plot example uses the `dino` dataset from the  
300 R package `datasauRus` (Davies, Locke & D'Agostino McGowan 2022). The lineup  
301 example uses residuals from a simulated regression model that has a non-linearity  
302 issue. We walk through the lineup example to further demonstrate the workflow of  
303 the web application.

**304 4.2.1. Reading data and setting parameters**

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

**321 4.2.2. Results provided**

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

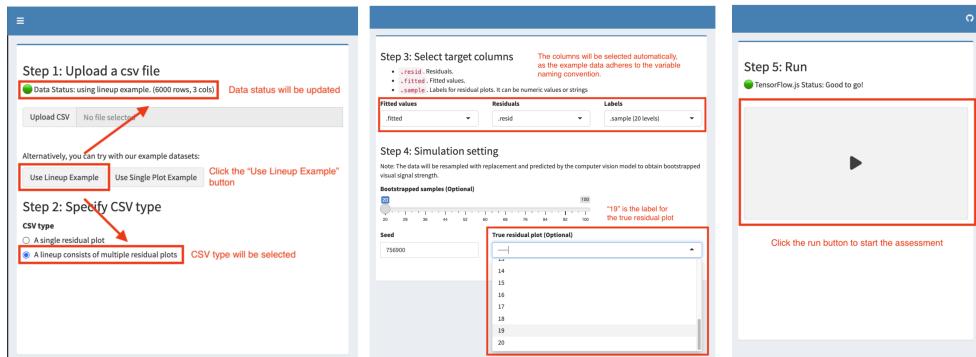


Figure 5. To begin the workflow for `autoovi` using the lineup example dataset, the user clicks the “Use Lineup Example” button (left) to load the example dataset, during which the data status and CSV type will be automatically updated. The user must manually select the label for the true residual plot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

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

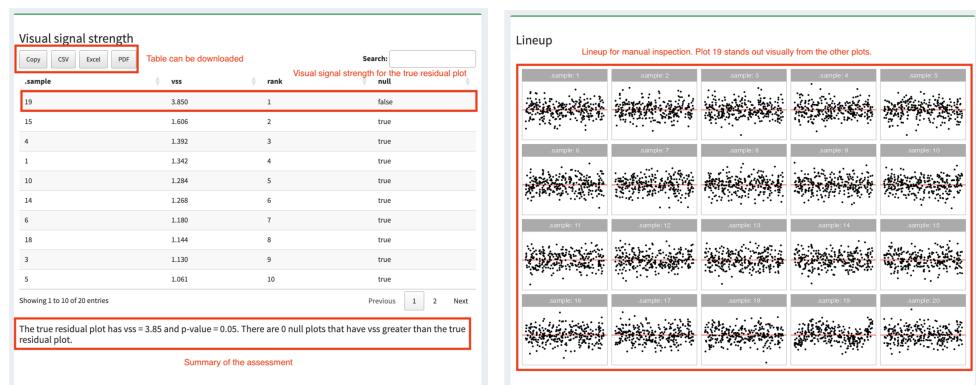


Figure 6. Results for the lineup. The VSS of the true residual plot is displayed in the first row of the table of VSS values for all the null plots (left image), with a summary text beneath the table providing the *p*-value to aid in decision-making. A lineup of residual plots allows for manual inspection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

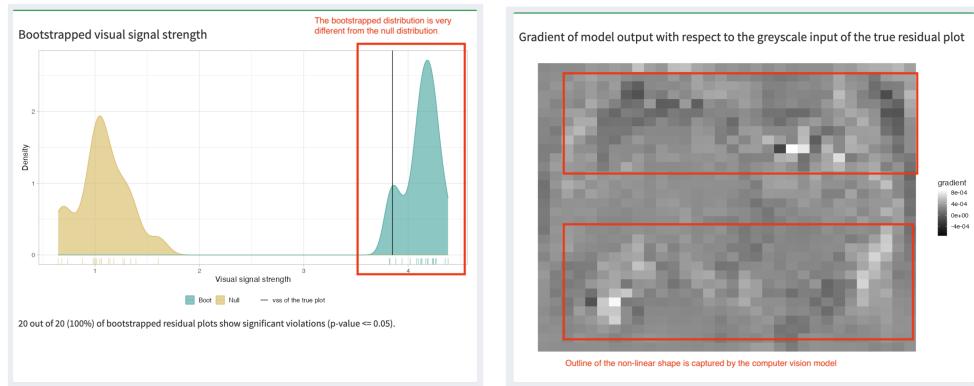


Figure 7. Summaries assessing the strength of the pattern and which elements of the plot contribute. The density plot helps verify if the bootstrapped distribution differs from the null distribution (left image). The attention map (right image) offers insights into whether the computer vision model has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

333

## 5. Conclusions

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

339 The **autovi** R package, introduced in this paper, automates the assessment of  
 340 residual plots by incorporating a computer vision model, eliminating the need for  
 341 time-consuming and potentially inconsistent human interpretation. This automation  
 342 improves the efficiency of the diagnostic process and promotes consistency in model  
 343 evaluation across different users and studies.

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

350 The combination of **autovi** and **autovi.web** offers a comprehensive solution to the  
 351 challenges of residual plot interpretation in regression analysis. These tools have the  
 352 potential to significantly improve the quality and consistency of model diagnostics

353 across various fields, from academic research to industry applications. By automating  
 354 a critical aspect of model evaluation, they allow researchers and analysts to focus more  
 355 on interpreting results and refining models, rather than grappling with the intricacies  
 356 of plot assessment.

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

## 362 6. Resources and supplementary material

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

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

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

## 375 References

- 376 ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G.S.,  
 377 DAVIS, A., DEAN, J., DEVIN, M. et al. (2016). Tensorflow: Large-scale machine learning on  
 378 heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .
- 379 ALLAIRE, J., TEAGUE, C., SCHEIDECKER, C., XIE, Y. & DERVIEUX, C. (2024). Quarto. doi:  
 380 10.5281/zenodo.5960048. URL <https://github.com/quarto-dev/quarto-cli>.
- 381 ATTALI, D. (2021). *shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds*.  
 382 URL <https://CRAN.R-project.org/package=shinyjs>. R package version 2.1.0.
- 383 BALAMUTA, J.J. (2024). *surreal: Create Datasets with Hidden Images in Residual Plots*. URL  
 384 <https://CRAN.R-project.org/package=surreal>. R package version 0.0.1.
- 385 BREUSCH, T.S. & PAGAN, A.R. (1979). A simple test for heteroscedasticity and random coefficient  
 386 variation. *Econometrica: Journal of the Econometric Society* , 1287–1294.

- 387 BUJA, A., COOK, D., HOFMANN, H., LAWRENCE, M., LEE, E.K., SWAYNE, D.F. & WICKHAM, H.  
 388 (2009). Statistical inference for exploratory data analysis and model diagnostics. *Philosophical  
 389 Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **367**,  
 390 4361–4383.
- 391 CHANG, W. & BORGES RIBEIRO, B. (2021). *shinydashboard: Create Dashboards with 'Shiny'*.  
 392 URL <https://CRAN.R-project.org/package=shinydashboard>. R package version 0.7.2.
- 393 CHANG, W., CHENG, J., ALLAIRE, J., SIEVERT, C., SCHLOERKE, B., XIE, Y., ALLEN, J.,  
 394 MCPHERSON, J., DIPERT, A. & BORGES, B. (2022). *shiny: Web Application Framework for  
 395 R*. URL <https://CRAN.R-project.org/package=shiny>. R package version 1.7.3.
- 396 CHENG, J., SIEVERT, C., SCHLOERKE, B., CHANG, W., XIE, Y. & ALLEN, J. (2024). *htmltools:  
 397 Tools for HTML*. URL <https://CRAN.R-project.org/package=htmltools>. R package version  
 398 0.5.8.
- 399 CLARK, A. et al. (2015). Pillow (pil fork) documentation. *readthedocs* .
- 400 COOK, R.D. & WEISBERG, S. (1982). *Residuals and influence in regression*. New York: Chapman  
 401 and Hall.
- 402 DAVIES, R., LOCKE, S. & D'AGOSTINO McGOWAN, L. (2022). *datasauRus: Datasets from the  
 403 Datasaurus Dozen*. URL <https://CRAN.R-project.org/package=datasauRus>. R package  
 404 version 0.1.6.
- 405 GAUTIER, L. (2024). *Python interface to the R language (embedded R)*. URL [https://pypi.org/project/rpy2/](https://pypi.org/<br/>
  406 project/rpy2/). Version 3.5.16.
- 407 GOODE, K. & REY, K. (2019). *ggResidpanel: Panels and Interactive Versions of Diagnostic Plots  
 408 using 'ggplot2'*. URL <https://CRAN.R-project.org/package=ggResidpanel>. R package version  
 409 0.3.0.
- 410 HARTIG, F. (2022). *DHARMA: Residual Diagnostics for Hierarchical (Multi-Level / Mixed)  
 411 Regression Models*. URL <https://CRAN.R-project.org/package=DHARMA>. R package  
 412 version 0.4.6.
- 413 HEBBALI, A. (2024). *olsrr: Tools for Building OLS Regression Models*. URL [https://CRAN.R-project.org/package=olsrr](https://CRAN.R-<br/>
  414 project.org/package=olsrr). R package version 0.6.0.
- 415 HESTER, J. & BRYAN, J. (2022). *glue: Interpreted String Literals*. URL [https://CRAN.R-project.org/package=glue](https://CRAN.R-<br/>
  416 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](https://CRAN.R-<br/>
  418 project.org/package=rockchalk). R package version 1.8.157.
- 419 KUHN, M., VAUGHAN, D. & HVITFELDT, E. (2024). *yardstick: Tidy Characterizations of Model  
 420 Performance*. URL <https://CRAN.R-project.org/package=yardstick>. R package version 1.3.1.
- 421 LI, W. (2024). *bandicoot: Light-weight python-like object-oriented system*. URL [https://CRAN.R-project.org/package=bandicoot](https://CRAN.R-<br/>
  422 project.org/package=bandicoot).
- 423 LI, W., COOK, D., TANAKA, E. & VANDERPLAS, S. (2024a). A plot is worth a thousand tests:  
 424 Assessing residual diagnostics with the lineup protocol. *Journal of Computational and  
 425 Graphical Statistics* **33**, 1497–1511. doi:10.1080/10618600.2024.2344612.
- 426 LI, W., COOK, D., TANAKA, E., VANDERPLAS, S. & ACKERMANN, K. (2024b). Automated  
 427 assessment of residual plots with computer vision models. *arXiv preprint arXiv:2411.01001* .
- 428 LONG, J.A. (2022). *jtools: Analysis and Presentation of Social Scientific Data*. URL [https://cran.r-project.org/package=jtools](https://cran.r-<br/>
  429 project.org/package=jtools). R package version 2.2.0.
- 430 LOY, A. & HOFMANN, H. (2014). *Hlmdiag: A suite of diagnostics for hierarchical linear models in  
 431 r*. *Journal of Statistical Software* **56**, 1–28.
- 432 MACFARLANE, J., KREWINKEL, A. & ROSENTHAL, J. (2024). Pandoc. URL [https://github.com/jgm/pandoc](https://github.com/<br/>
  433 jgm/pandoc).
- 434 MASON, H., LEE, S., LAA, U. & COOK, D. (2022). *cassowaryr: Compute Scagnostics on Pairs of  
 435 Numeric Variables in a Data Set*. URL <https://CRAN.R-project.org/package=cassowary>. R

- 436 package version 2.0.0.
- 437 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.
- 439 NOWOSAD, J. (2018). 'CARTOCOLORs' palettes. URL <https://nowosad.github.io/rkartocolor>. R package version 1.0.
- 440 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.
- 441 PEDERSEN, T.L. (2022). *patchwork: The composer of plots*. URL <https://CRAN.R-project.org/package=patchwork>. R package version 1.1.2.
- 442 R CORE TEAM (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- 443 RAMSEY, J.B. (1969). Tests for specification errors in classical linear least-squares regression analysis. *Journal of the Royal Statistical Society: Series B (Methodological)* **31**, 350–371.
- 444 REINHART, A. (2024). *regressinator: Simulate and Diagnose (Generalized) Linear Models*. URL <https://CRAN.R-project.org/package=regressinator>. R package version 0.2.0.
- 445 ROWLINGSON, B. & DIGGLE, P. (2023). *splancs: Spatial and Space-Time Point Pattern Analysis*. URL <https://CRAN.R-project.org/package=splancs>. R package version 2.01-44.
- 446 SALI, A. & ATTALI, D. (2020). *shinycssloaders: Add Loading Animations to a 'shiny' Output While It's Recalculating*. URL <https://CRAN.R-project.org/package=shinycssloaders>. R package version 1.0.0.
- 447 SHAPIRO, S.S. & WILK, M.B. (1965). An analysis of variance test for normality (complete samples). *Biometrika* **52**, 591–611.
- 448 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.
- 449 WARTON, D.I. (2023). Global simulation envelopes for diagnostic plots in regression models. *The American Statistician* **77**, 425–431.
- 450 WICKHAM, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. URL <https://ggplot2.tidyverse.org>.
- 451 WICKHAM, H., AVERICK, M., BRYAN, J., CHANG, W., McGOWAN, L.D., FRANÇOIS, R., GROLEMUND, G., HAYES, A., HENRY, L., HESTER, J., KUHN, M., PEDERSEN, T.L., MILLER, E., BACHE, S.M., MÜLLER, K., OOMS, J., ROBINSON, D., SEIDEL, D.P., SPINU, V., TAKAHASHI, K., VAUGHAN, D., WILKE, C., WOO, K. & YUTANI, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software* **4**, 1686. doi:10.21105/joss.01686.
- 452 WICKHAM, H., CHOWDHURY, N.R., COOK, D. & HOFMANN, H. (2020). *nullabor: Tools for Graphical Inference*. URL <https://CRAN.R-project.org/package=nullabor>. R package version 0.3.9.
- 453 ZAKAI, A. (2011). Emscripten: an llvm-to-javascript compiler. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*. pp. 301–312.
- 454 ZEILEIS, A. & HOTHORN, T. (2002). Diagnostic checking in regression relationships. *R News* **2**, 7–10.
- 455 ZHU, H. (2021). *kableExtra: Construct complex table with kable and pipe syntax*. URL <https://CRAN.R-project.org/package=kableExtra>. R package version 1.3.4.