

# 1 Automated Residual Plot Assessment with the R Package 2 autovi and 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 suffer from the potential for 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 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 are 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)

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 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 provides a detailed documentation of the `autovi` package, including its usage and infrastructure. Section 3

48 focuses on the `autovi.web` interface, describing its design and usage, along with  
49 illustrative examples. Finally, Section 4 presents the main conclusions of this work.

50 **2. R package: autovi**

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

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

58 **2.1. Motivation for usage**

59 Figure 1 shows three sets of plots of residuals against fitted values. The simulated  
60 example in (a) might be interpreted as a heteroscedastic pattern, however the  
61 automated reading would predict this to have a visual signal strength (VSS) of  
62 1.53, with a corresponding  $p$ -value of 0.25. This means it would be interpreted as  
63 a good residual plot, that there is nothing in the data to indicate a violation of  
64 model assumptions. Skewness in the predictor variables is generating the apparent  
65 heteroscedasticity, where the smaller variance in residuals at larger fitted values is  
66 due to smaller sample size only. The Breusch-Pagan test (Breusch & Pagan 1979) for  
67 heteroscedasticity would also not reject this as good residual plot.

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

75 The third example is generated using the `surreal` package (Balamuta 2024) where  
76 structured residuals are hidden in data, to be revealed if the correct model is specified.  
77 Here a quote based on Tukey is used as the residual structure “visual summaries focus  
78 on unexpected values”. The automated plot reading predicts the VSS to be 5.87, with  
79 a  $p$ -value less than 0.05. This structure is blindingly obvious visually, but a RESET

80 test for nonlinear structure would not report a problem. (It would be detected by  
 81 a Breusch-Pagan for heteroscedasticity and also Shapiro-Wilk test ([Shapiro & Wilk](#)  
 82 [1965](#)) for non-normality.)

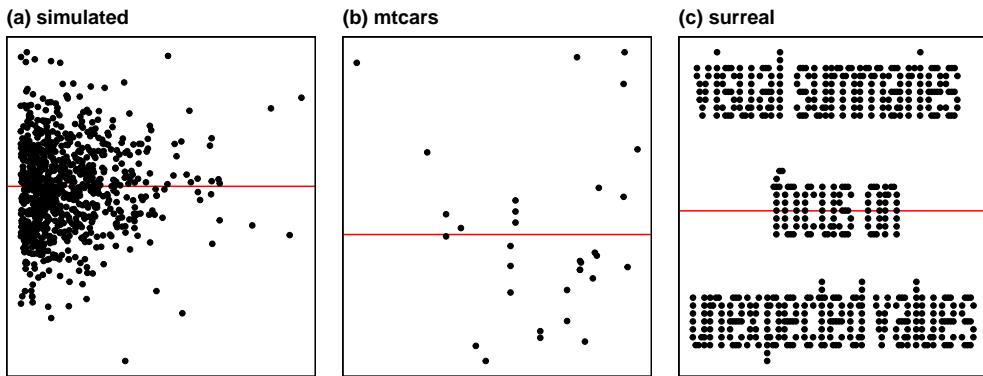


Figure 1. Reading residual plots can be a difficult task, particularly for students new to statistical modeling. The `autovi` package makes it easier. Here are three examples of residual plots, which may appear to have structure. According to `autovi`, the visual signal strengths (VSS) of these three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

### 83 2.2. Implementation

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

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

98 In the R environment, `autovi` utilizes several libraries. `ggplot2` ([Wickham 2016](#))  
 99 generates the initial residual plots, saved as PNG files for visual input. The `cassowaryr`

100 ([Mason et al. 2022](#)) library computes scagnostics (scatter plot diagnostics), providing  
 101 numerical features that capture statistical properties of the plots. These scagnostics  
 102 complement the visual analysis by offering quantitative metrics as secondary input to  
 103 the computer vision model. The `reticulate` ([Ushey, Allaire & Tang 2024](#)) package  
 104 bridges R and Python, enabling seamless communication between the two languages  
 105 and supporting the integrated infrastructure.

106 **2.3. Installation**

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

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

115 We recommend using the Shiny app `autovi.web` if users encounter installation  
 116 problems.

117 **2.4. Usage**

118 **2.4.1. Numerical Summary**

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

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

126 The code to do this is:

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

127 It produces the following summary:

128

```

129 -- <AUTO_VI object>
130 Status:
131 - Fitted model: lm
132 - Keras model: UNKNOWN
133 - Output node index: 1
134 - Result:
135 - Observed visual signal strength: 3.162 (p-value = 0.0396)
136 - Null visual signal strength: [100 draws]
137 - Mean: 1.274
138 - Quantiles:

139
140      25%    50%    75%    80%    90%    95%    99%
141 0.8021 1.1109 1.5751 1.6656 1.9199 2.6564 3.3491
142
143 - Bootstrapped visual signal strength: [100 draws]
144 - Mean: 2.786 (p-value = 0.05941)
145 - Quantiles:

146
147      25%    50%    75%    80%    90%    95%    99%
148 2.452 2.925 3.173 3.285 3.463 3.505 3.652
149
150 - Likelihood ratio: 0.7275 (boot) / 0.06298 (null) = 11.55

```

151 The summary includes observed VSS of the true residual plot and associated *p*-value  
152 of the automated visual test. The *p*-value is the proportion of null plots (out of the  
153 total 100) that have VSS greater than or equal to that of the true residual plot. The  
154 report also provides sample quantiles of VSS for null samples and bootstrapped data  
155 plots, providing more information about the sampling variability and a likelihood of  
156 model violations. The likelihood is computed from the proportion of values greater  
157 than the observed VSS in both the bootstrapped data values and the simulated null  
158 values.

159 **2.4.2. Visual Summary**

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

```
checker$plot_pair()
```

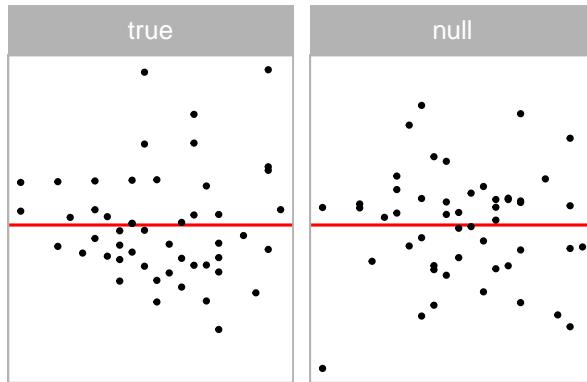


Figure 2. True plot alongside one null plot, for quick comparison.

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

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

```
checker$summary_plot()
```

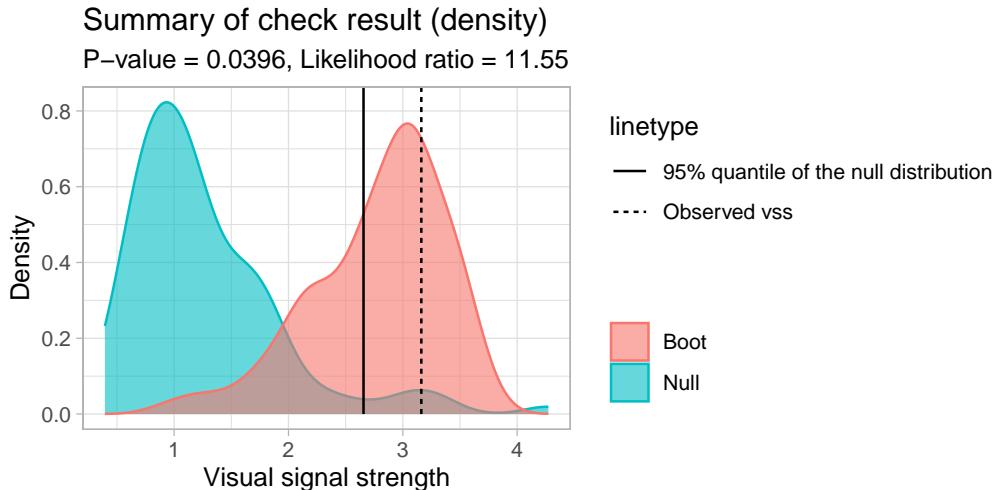


Figure 3. Summary plot comparing the densities of VSS for bootstrapped residual samples (red) relative to VSS for null plots (blue).

- 171 In the result, shown in Figure 3, the blue area represents the density of VSS for null  
 172 residual plots, while the red area shows the density for bootstrapped residual plots.  
 173 The dashed line indicates the VSS of the true residual plot, and the solid line marks  
 174 the critical value at a 95% significance level. The  $p$ -value and the likelihood ratio are  
 175 displayed in the subtitle. The likelihood ratio represents the ratio of the likelihood  
 176 of observing the VSS of the true residual plot from the bootstrapped distribution  
 177 compared to the null distribution.
- 178 Interpreting the plot involves several key aspects. If the dashed line falls to the right of  
 179 the solid line, it suggests rejecting the null hypothesis. The degree of overlap between  
 180 the red and blue areas indicates similarity between the true residual plot and null  
 181 plots; greater overlap suggests more similarity. Lastly, the portion of the red area to  
 182 the right of the solid line represents the percentage of bootstrapped models considered  
 183 to have model violations.
- 184 This visual summary provides an intuitive way to assess the model's fit and potential  
 185 violations, allowing users to quickly grasp the results of the automated analysis.

## 186 2.5. Modularized Infrastructure

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

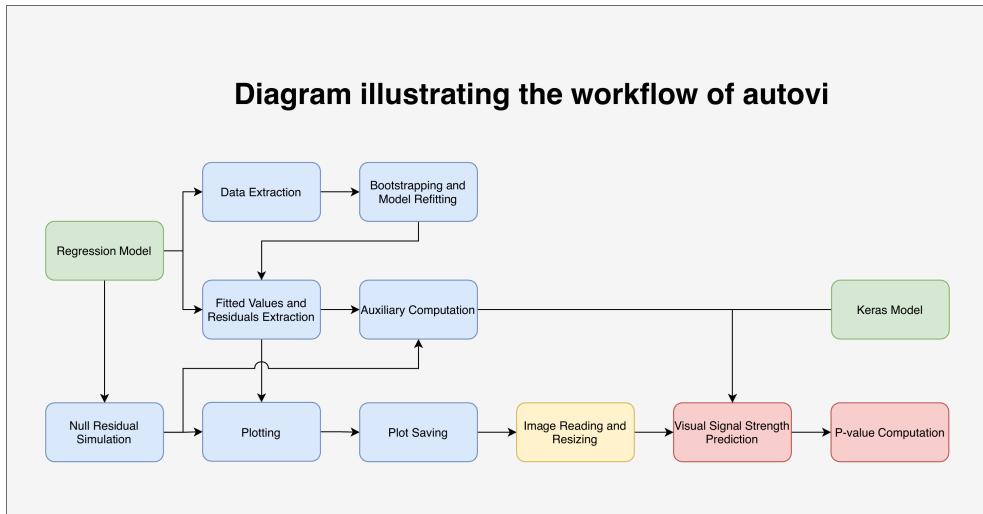


Figure 4. Diagram illustrating the infrastructure of the R package **autovi**. The modules in green are primary inputs provided by users. Modules in blue are overridable methods that can be modified to accommodate users' specific needs. The module in yellow is a pre-defined non-overridable method. The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

209

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

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

218 **3.1. Implementation**

219 The package `autovi.web` is built using the `shiny` (Chang et al. 2022) and  
220 `shinydashboard` (Chang & Borges Ribeiro 2021) R packages. Hosted on the  
221 `shinyapps.io` domain, the application is accessible through any modern web browser.  
222 The R packages `htmltools` (Cheng et al. 2024) and `shinycssloaders` (Sali & Attali  
223 2020) are used to render markdown documentation in shiny application, and for loading  
224 animations for shiny widgets, respectively.

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

233 We also explored the possibility of implementing the web interface using frameworks  
234 built on other languages, such as Python. However, server hosting domains that  
235 natively support Python servers typically do not have the latest version of R installed.  
236 Additionally, calling R from Python is typically done using the `rpy2` Python library  
237 (Gautier 2024), but this approach can be awkward when dealing with language syntax  
238 related to non-standard evaluation. Another option we considered was renting a server  
239 where we could have full control, such as those provided by cloud platforms like Google  
240 Cloud Platform (GCP) or Amazon Web Services (AWS). However, correctly setting up  
241 the server and ensuring a secure deployment requires significant expertise. Ultimately,  
242 the most practical solution was to use the `shiny` and `shinydashboard` frameworks,

243 which are well-established in the R community and offer a solid foundation for web  
244 application development.

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

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

255 In contrast to `autovi`, `autovi.web` does not use the native Python version of  
256 `TensorFlow`. Instead, it leverages `TensorFlow.js`, a JavaScript library that allows  
257 the execution of machine learning models directly in the browser. This choice enables  
258 native browser execution, enhancing compatibility across different user environments,  
259 and shifts the computational load from the server to the client-side. `TensorFlow.js`  
260 also offers better scalability and performance, especially when dealing with resource-  
261 intensive computer vision models on `shinyapps.io`.

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

267 To allow communication between `TensorFlow.js` and other components of the Shiny  
268 application, the `shinyjs` R package is used. This package allows calling custom  
269 JavaScript code within the Shiny framework. The specialized JavaScript code for  
270 initializing `TensorFlow.js` and calling `TensorFlow.js` for VSS prediction is deployed  
271 alongside the Shiny application as additional resources.

### 272 3.2. Usage

273 The workflow of `autovi.web` is designed to be straightforward, with numbered  
274 steps displayed in each panel. There are two example datasets provided by the web  
275 application. The single residual plot example uses the `dino` dataset from the R package

276 `datasauRus` (Davies, Locke & D'Agostino McGowan 2022). The lineup example uses  
 277 residuals from a simulated regression model that has a non-linearity issue. We will  
 278 walk through the lineup example to further demonstrate the workflow of the web  
 279 application.

280 **3.2.1. Reading data and setting parameters**

281 The user can select to upload data as either a single set of residuals or a pre-computed  
 282 lineup of residuals and nulls. Here we illustrate use with lineup example data sets  
 283 (Figure 5). To use the lineup example data, click the “Use Lineup Example” button.  
 284 The data status will then update to show the number of rows and columns in the  
 285 dataset, and the CSV type will automatically be selected to the correct option. Since  
 286 the example dataset follows the variable naming conventions assumed by the web  
 287 application, the columns for fitted values, residuals, and labels of residual plots are  
 288 set automatically (middle image). If the user is working with a custom dataset, these  
 289 options must be set accordingly. Regardless of the dataset, the user must manually  
 290 select the label for the true residual plot, whenever a lineup file is provided, as the  
 291 web application allows assessments without this label. The last step is to click the play  
 292 button (right image) to start the assessment.

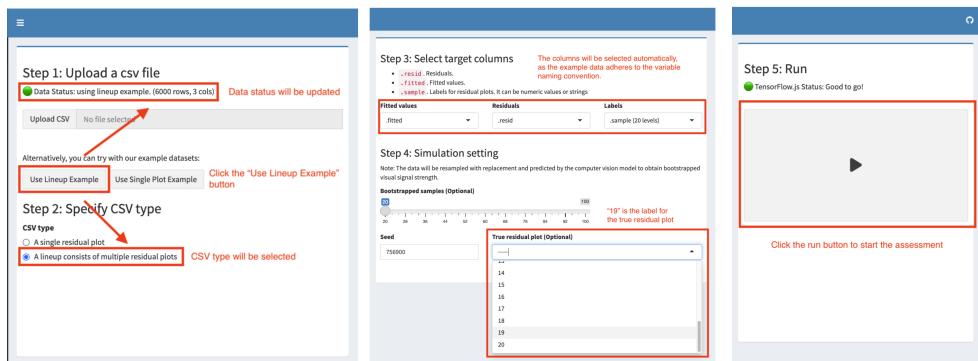


Figure 5. To begin the workflow for `autoovi` using the lineup example dataset, the user clicks the “Use Lineup Example” button (left) to load the example dataset, during which the data status and CSV type will be automatically updated. The target columns are selected automatically, though the user must manually select the label for the true residual plot (middle), as the web application permits assessment without this label. The user initiates the assessment of the lineup example data by clicking the run button (right).

293 **3.2.2. Results provided**

294 Results are provided in multiple panels. The first row of the table Figure 6 is the most  
 295 crucial to check, as it provides the VSS and the rank of the true residual plot among

296 the other plots. The summary text beneath the table provides the  $p$ -value, which can  
 297 be used for quick decision-making. The lineup is for manual inspection, and the user  
 298 should see if the true residual plot is visually distinguishable from the other plots, to  
 299 confirm if the model violation is serious.

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

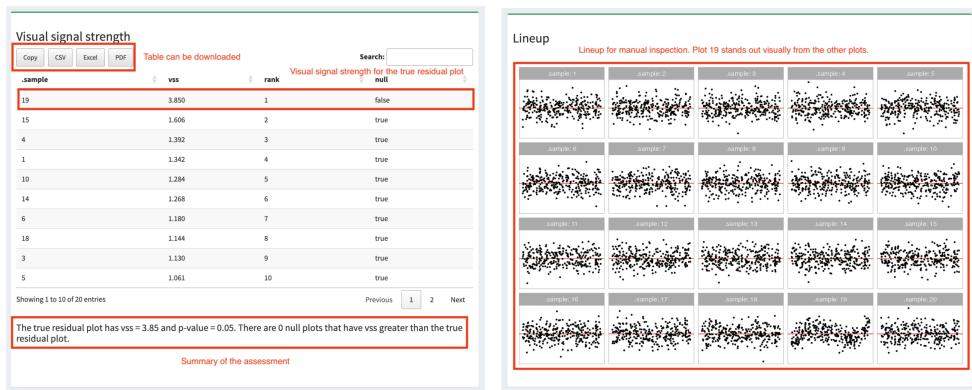


Figure 6. Results for the lineup. The VSS of the true residual plot is displayed in the first row of the table of VSS values for all the null plots (left image), with a summary text beneath the table providing the  $p$ -value to aid in decision-making. A lineup of residual plots allows for manual inspection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

305

#### 4. Conclusions

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

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

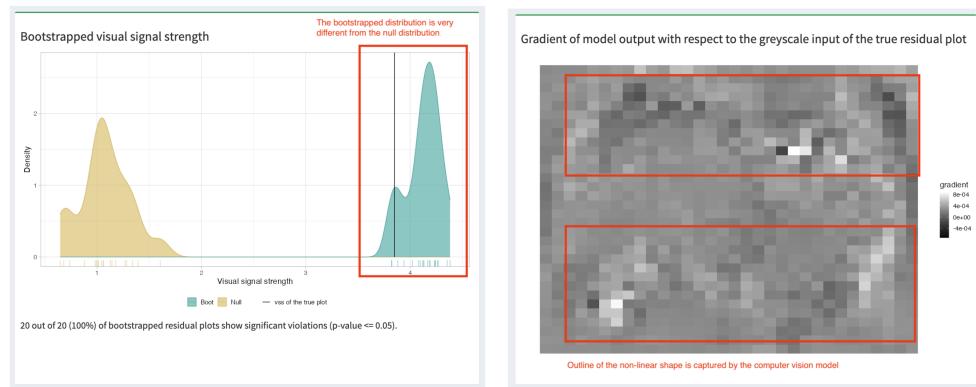


Figure 7. Summaries assessing the strength of the pattern and which elements of the plot contribute. The density plot helps verify if the bootstrapped distribution differs from the null distribution (left image). The attention map (right image) offers insights into whether the computer vision model has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

- 316 The development of the accompanying Shiny app, **autovi.web**, expands access to these  
 317 advanced diagnostic tools, by providing a user-friendly interface. It makes automated  
 318 residual plot assessment accessible to a broader audience, including those who may not  
 319 have extensive programming experience. This web-based solution effectively addresses  
 320 the potential barriers to adoption, such as complex dependencies and installation  
 321 requirements, that are often associated with advanced statistical software.
- 322 The combination of **autovi** and **autovi.web** offers a comprehensive solution to the  
 323 challenges of residual plot interpretation in regression analysis. These tools have the  
 324 potential to significantly improve the quality and consistency of model diagnostics  
 325 across various fields, from academic research to industry applications. By automating  
 326 a critical aspect of model evaluation, they allow researchers and analysts to focus more  
 327 on interpreting results and refining models, rather than grappling with the intricacies  
 328 of plot assessment.
- 329 The framework established by **autovi** and **autovi.web** opens up exciting possibilities  
 330 for further research and development. Future work could explore the extension of these  
 331 automated assessment techniques to other types of diagnostic plots and statistical  
 332 models, potentially revolutionizing how we approach statistical inference using visual  
 333 displays more broadly.

334

## 5. Resources and Supplementary Material

- 335 The the current version of `autovi` can be installed from CRAN, and source code for  
 336 both packages are available at <https://github.com/TengMCing/autovi>. The web  
 337 interface is available from [autoviweb.netlify.app](https://autoviweb.netlify.app).
- 338 These R packages were used for the work: `tidyverse` (Wickham et al. 2019), `lmtest`  
 339 ([Zeileis & Hothorn 2002](#)), `kableExtra` ([Zhu 2021](#)), `patchwork` ([Pedersen 2022](#)),  
 340 `rcartocolor` ([Nowosad 2018](#)), `glue` ([Hester & Bryan 2022](#)), `here` ([Müller 2020](#)),  
 341 `magick` ([Ooms 2023](#)), `yardstick` ([Kuhn, Vaughan & Hvitfeldt 2024](#)) and `reticulate`  
 342 ([Ushey, Allaire & Tang 2024](#)).

343

## References

- 344 ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G.S.,  
 345 DAVIS, A., DEAN, J., DEVIN, M. et al. (2016). Tensorflow: Large-scale machine learning on  
 346 heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .
- 347 BALAMUTA, J.J. (2024). *surreal: Create Datasets with Hidden Images in Residual Plots*. URL  
 348 <https://CRAN.R-project.org/package=surreal>. R package version 0.0.1.
- 349 BREUSCH, T.S. & PAGAN, A.R. (1979). A simple test for heteroscedasticity and random coefficient  
 350 variation. *Econometrica: Journal of the Econometric Society* , 1287–1294.
- 351 BUJA, A., COOK, D., HOFMANN, H., LAWRENCE, M., LEE, E.K., SWAYNE, D.F. & WICKHAM, H.  
 352 (2009). Statistical inference for exploratory data analysis and model diagnostics. *Philosophical  
 353 Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **367**,  
 354 4361–4383.
- 355 CHANG, W. & BORGES RIBEIRO, B. (2021). *shinydashboard: Create Dashboards with 'Shiny'*.  
 356 URL <https://CRAN.R-project.org/package=shinydashboard>. R package version 0.7.2.
- 357 CHANG, W., CHENG, J., ALLAIRE, J., SIEVERT, C., SCHLOERKE, B., XIE, Y., ALLEN, J.,  
 358 MCPHERSON, J., DIPERT, A. & BORGES, B. (2022). *shiny: Web Application Framework for  
 359 R*. URL <https://CRAN.R-project.org/package=shiny>. R package version 1.7.3.
- 360 CHENG, J., SIEVERT, C., SCHLOERKE, B., CHANG, W., XIE, Y. & ALLEN, J. (2024). *htmltools:  
 361 Tools for HTML*. URL <https://CRAN.R-project.org/package=htmltools>. R package version  
 362 0.5.8.
- 363 CLARK, A. et al. (2015). Pillow (pil fork) documentation. *readthedocs* .
- 364 COOK, R.D. & WEISBERG, S. (1982). *Residuals and influence in regression*. New York: Chapman  
 365 and Hall.
- 366 DAVIES, R., LOCKE, S. & D'AGOSTINO McGOWAN, L. (2022). *datasauRus: Datasets from the  
 367 Datasaurus Dozen*. URL <https://CRAN.R-project.org/package=datasauRus>. R package  
 368 version 0.1.6.
- 369 GAUTIER, L. (2024). *Python interface to the R language (embedded R)*. URL <https://pypi.org/project/rpy2/>. Version 3.5.16.
- 370 GOODE, K. & REY, K. (2019). *ggResidpanel: Panels and Interactive Versions of Diagnostic Plots  
 371 using 'ggplot2'*. URL <https://CRAN.R-project.org/package=ggResidpanel>. R package  
 373 version 0.3.0.
- 374 HARTIG, F. (2022). *DHARMA: Residual Diagnostics for Hierarchical (Multi-Level / Mixed)  
 375 Regression Models*. URL <https://CRAN.R-project.org/package=DHARMA>. R package

- 376 version 0.4.6.
- 377 HEBBALI, A. (2024). *olsrr: Tools for Building OLS Regression Models*. URL <https://CRAN.R-project.org/package=olsrr>. R package version 0.6.0.
- 378 HESTER, J. & BRYAN, J. (2022). *glue: Interpreted String Literals*. URL <https://CRAN.R-project.org/package=glue>. R package version 1.6.2.
- 379 JOHNSON, P.E. (2022). *rockchalk: Regression Estimation and Presentation*. URL <https://CRAN.R-project.org/package=rockchalk>. R package version 1.8.157.
- 380 KUHN, M., VAUGHAN, D. & HVITFELDT, E. (2024). *yardstick: Tidy Characterizations of Model Performance*. URL <https://CRAN.R-project.org/package=yardstick>. R package version 1.3.1.
- 381 LI, W. (2024). *bandicoot: Light-weight python-like object-oriented system*. URL <https://CRAN.R-project.org/package=bandicoot>.
- 382 LI, W., COOK, D., TANAKA, E. & VANDERPLAS, S. (2024a). A plot is worth a thousand tests: Assessing residual diagnostics with the lineup protocol. *Journal of Computational and Graphical Statistics* **33**, 1497–1511. doi:10.1080/10618600.2024.2344612.
- 383 LI, W., COOK, D., TANAKA, E., VANDERPLAS, S. & ACKERMANN, K. (2024b). Automated assessment of residual plots with computer vision models. *arXiv preprint arXiv:2411.01001*.
- 384 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.
- 385 LOY, A. & HOFMANN, H. (2014). *Hlmdiag: A suite of diagnostics for hierarchical linear models in r*. *Journal of Statistical Software* **56**, 1–28.
- 386 MASON, H., LEE, S., LAA, U. & COOK, D. (2022). *cassowaryr: Compute Scagnostics on Pairs of Numeric Variables in a Data Set*. URL <https://CRAN.R-project.org/package=cassowaryr>. R package version 2.0.0.
- 387 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.
- 388 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.
- 389 NOWOSAD, J. (2018). 'CARTOCOLORs' palettes. URL <https://nowosad.github.io/rkartocolor>. R package version 1.0.
- 390 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.
- 391 PEDERSEN, T.L. (2022). *patchwork: The composer of plots*. URL <https://CRAN.R-project.org/package=patchwork>. R package version 1.1.2.
- 392 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/>.
- 393 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.
- 394 REINHART, A. (2024). *regressinator: Simulate and Diagnose (Generalized) Linear Models*. URL <https://CRAN.R-project.org/package=regressinator>. R package version 0.2.0.
- 395 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.
- 396 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.
- 397 SHAPIRO, S.S. & WILK, M.B. (1965). An analysis of variance test for normality (complete samples). *Biometrika* **52**, 591–611.
- 398 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.

- 425 WARTON, D.I. (2023). Global simulation envelopes for diagnostic plots in regression models. *The  
426 American Statistician* **77**, 425–431.
- 427 WICKHAM, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York.  
428 URL <https://ggplot2.tidyverse.org>.
- 429 WICKHAM, H., AVERICK, M., BRYAN, J., CHANG, W., McGOWAN, L.D., FRANÇOIS, R.,  
430 GROLEMUND, G., HAYES, A., HENRY, L., HESTER, J., KUHN, M., PEDERSEN, T.L., MILLER,  
431 E., BACHE, S.M., MÜLLER, K., OOMS, J., ROBINSON, D., SEIDEL, D.P., SPINU, V.,  
432 TAKAHASHI, K., VAUGHAN, D., WILKE, C., WOO, K. & YUTANI, H. (2019). Welcome to  
433 the tidyverse. *Journal of Open Source Software* **4**, 1686. doi:10.21105/joss.01686.
- 434 WICKHAM, H., CHOWDHURY, N.R., COOK, D. & HOFMANN, H. (2020). *nullabor: Tools for  
435 Graphical Inference*. URL <https://CRAN.R-project.org/package=nullabor>. R package  
436 version 0.3.9.
- 437 ZAKAI, A. (2011). Emscripten: an llvm-to-javascript compiler. In *Proceedings of the ACM  
438 international conference companion on Object oriented programming systems languages and  
439 applications companion*. pp. 301–312.
- 440 ZEILEIS, A. & HOTHORN, T. (2002). Diagnostic checking in regression relationships. *R News* **2**,  
441 7–10.
- 442 ZHU, H. (2021). *kableExtra: Construct complex table with kable and pipe syntax*. URL  
443 <https://CRAN.R-project.org/package=kableExtra>. R package version 1.3.4.