

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

3 Weihao Li¹, Dianne Cook¹, Emi Tanaka², Susan VanderPlas³ and Klaus
4 Ackermann¹

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

Summary

7 Visual assessment of residual plots is a common approach for diagnosing linear
models, but it relies on manual evaluation, which does not scale well and can lead
to inconsistent decisions across analysts. The lineup protocol, which embeds the
observed plot among null plots, can reduce subjectivity but requires even more
human effort. In today's data-driven world, such tasks are well-suited for automation.
We present a new R package that uses a computer vision model to automate the
evaluation of residual plots. An accompanying Shiny app is provided for ease of use.
Given a sample of residuals, the model predicts a visual signal strength (VSS) and
offers supporting information to help analysts assess the adequacy of their model
fit.

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

9 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 suite
13 of conventional tests, while the `stats` package (R Core Team 2022) offers standard
14 diagnostic plots such as residuals vs. fitted values, quantile-quantile (Q-Q) plots,
15 and residuals vs. leverage plots. Additional packages like `jtools` (Long 2022), `olsrr`

¹ Department of Econometrics and Business Statistics, Monash University, Wellington Road, VIC 3800, Australia

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

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

Email: `weihao.li@monash.edu`

(Hebbali 2024), `rockchalk` (Johnson 2022), and `ggResidpanel` (Goode & Rey 2019) deliver similar graphical diagnostics, often with enhanced aesthetics or interactive features. These tools collectively produce the core diagnostic plots outlined in the classical text by Cook & Weisberg (1982). The `ecostats` package (Warton 2023) extends these diagnostics by incorporating simulation envelopes into residual plots. Meanwhile, `DHARMa` (Hartig 2022) compares empirical quantiles (0.25, 0.5, and 0.75) of scaled residuals to their theoretical counterparts, with a strong focus on identifying model violations such as heteroscedasticity, misspecified functional forms, and issues specific to generalized linear and mixed-effect models, like over/under-dispersion. It also provides conventional test annotations to reduce the risk of 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 inference methods, particularly those using the lineup protocol (Buja et al. 2009), offer more practical and reliable assessments of residual patterns than conventional tests, as they are less sensitive to minor departures. Packages such as `nullabor` (Wickham et al. 2020), `HLMdiag` (Loy & Hofmann 2014), and `regressinator` (Reinhart 2024) support this approach by enabling users to compare observed residual plots with plots generated under null hypothesis, thereby 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

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

52 2. Definition and computation of visual signal strength

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

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

68 where D_{KL} is given by:

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

69 here, $p(\cdot)$ and $q(\cdot)$ are the probability density functions of the reference residual
 70 distribution P and the true residual distribution Q , respectively.
 71 This distance measure depends on knowledge of the true residual distribution, which
 72 is unknown in practice. To compute D_{KL} for the training samples, Equation 1 takes
 73 different forms depending on the specific model violations. For instance, where necessary
 74 higher-order predictors, \mathbf{Z} , and their corresponding parameter, β_Z , are omitted from
 75 the fitted linear model, the distance measure can be expanded as follows:

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

76 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
77 regression model.

78 The computer vision model approximates this mapping from a set of residuals to its
79 corresponding distance measure. It is trained on a large number of synthetic regression
80 models, where the data-generating process is known, allowing the distance measure
81 to be explicitly calculated. The model takes a residual plot as input and outputs the
82 corresponding distance measure. Additional details are provided in [Li et al. \(2024b\)](#).

83 3. R package: autovi

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

91 3.1. Motivation

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

101 The data in (b) is generated by fitting a linear model predicting `mpg` based on `hp`
102 using the `datasets::mtcars`. It is a small data set, and there is a hint of nonlinear
103 structure not captured by the model. The automated plot reading would predict a
104 VSS of 3.57, which has a p -value less than 0.05. That is, the nonlinear structure is
105 most likely real, and indicates a problem with the model. The conventional test, a

106 Ramsey Regression Equation Specification Error Test (RESET) (Ramsey 1969) would
 107 also strongly detect the nonlinearity.

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

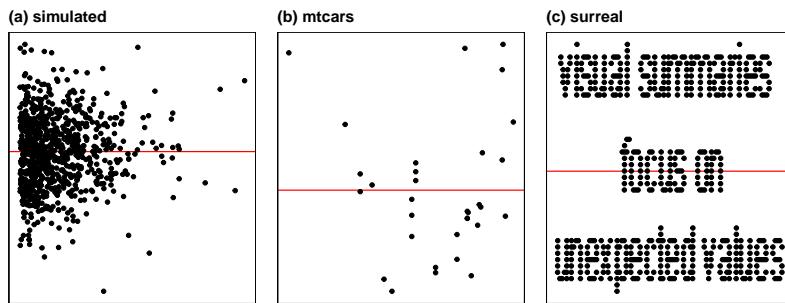


Figure 1. Reading residual plots can be a difficult task, particularly for students new to statistical modeling. The `autovi` package makes it easier. Here are three examples of residual plots, which may appear to have structure. According to `autovi`, the visual signal strengths (VSS) of these three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

116 3.2. Implementation

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

121 The `autovi` infrastructure effectively integrates multiple programming languages and
 122 libraries into a comprehensive analytical tool. It relies on five core libraries from
 123 Python and R, each playing a critical role in the analysis pipeline. In Python, `pillow`
 124 (Clark et al. 2015) handles image processing tasks such as reading and resizing PNG
 125 files of residual plots, then converting them into input tensors for further analysis.

126 TensorFlow (Abadi et al. 2016), a key component of modern machine learning, is used
 127 to predict the VSS of these plots using a pre-trained convolutional neural network.

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

135 3.3. Installation

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

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

145 3.4. Usage

146 3.4.1. Numerical summary

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

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

154 The code to do this is:

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

155 It produces the following summary:

156

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

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

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

187 **3.4.2. Visual summary**

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

```
checker$plot_pair()
```

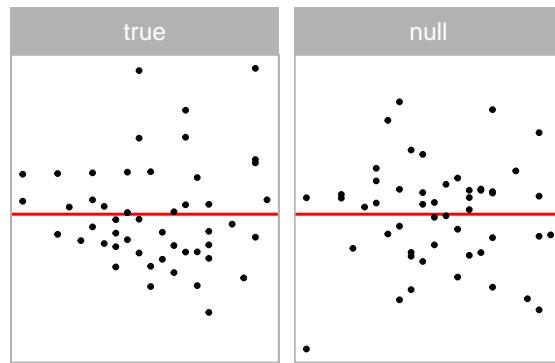


Figure 2. True plot alongside one null plot, for quick comparison.

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

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

```
checker$summary_plot()
```

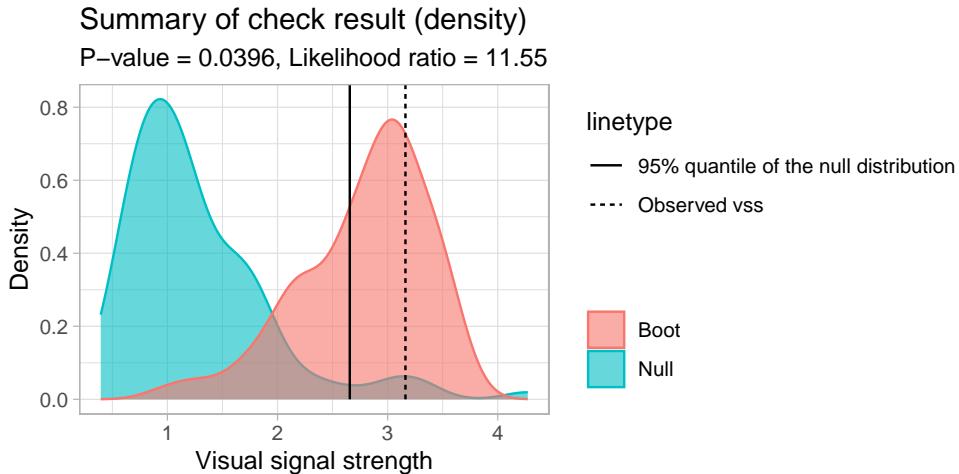


Figure 3. Summary plot comparing the densities of VSS for bootstrapped residual samples (red) relative to VSS for null plots (blue).

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

214 **3.5. Modularized infrastructure**

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

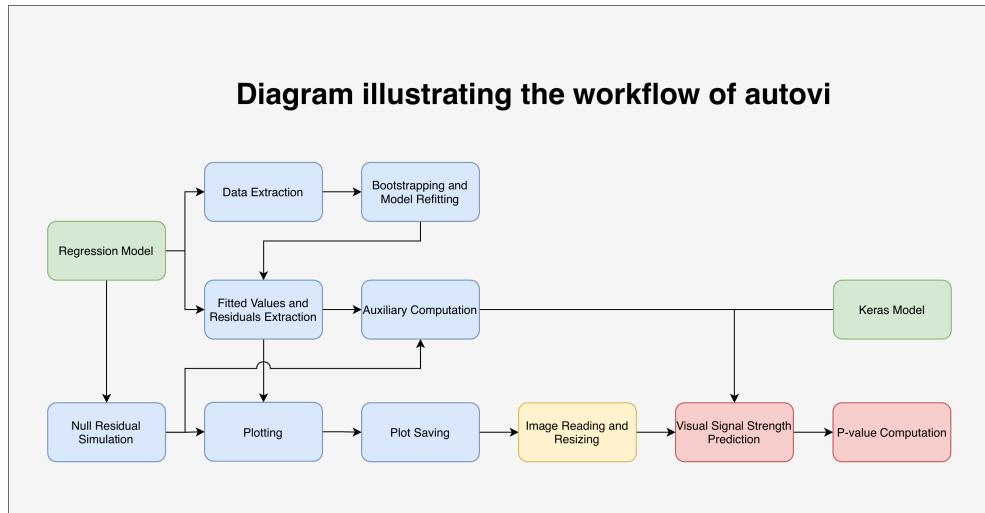


Figure 4. Diagram illustrating the infrastructure of the R package **autovi**. The modules in green are primary inputs provided by users. Modules in blue are overridable methods that can be modified to accommodate users' specific needs. The module in yellow is a pre-defined non-overridable method. The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

237

4. Web interface: `autovi.web`

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

246 **4.1. Implementation**

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

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

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

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

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

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

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

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

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

298 4.2. Usage

299 The workflow of `autovi.web` is designed to be straightforward, with numbered
300 steps displayed in each panel. There are two example datasets provided by the
301 web application. The single residual plot example uses the `dino` dataset from the
302 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.

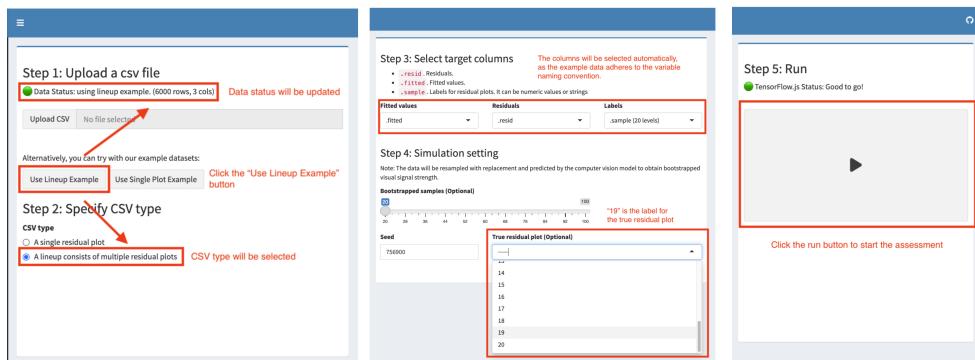


Figure 5. To begin the workflow for `autovi` using the lineup example dataset, the user clicks the “Use Lineup Example” button (left) to load the example dataset, during which the data status and CSV type will be automatically updated. The user must manually select the label for the true residual plot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

323 **4.2.2. Results provided**

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

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

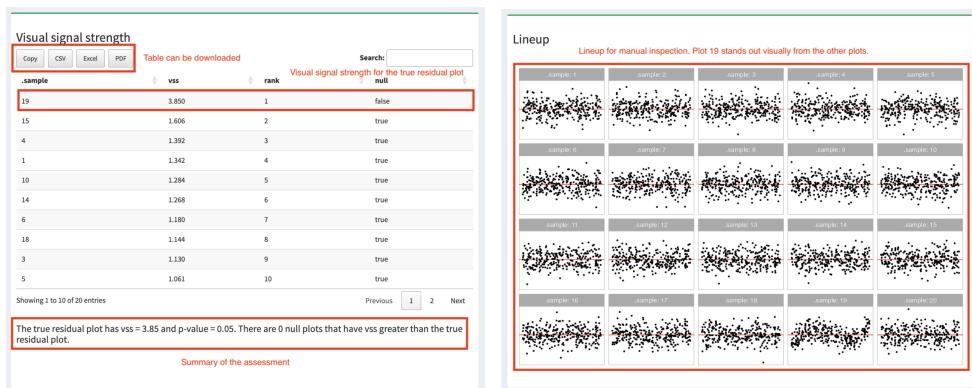


Figure 6. Results for the lineup. The VSS of the true residual plot is displayed in the first row of the table of VSS values for all the null plots (left image), with a summary text beneath the table providing the *p*-value to aid in decision-making. A lineup of residual plots allows for manual inspection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

335

5. Conclusions

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

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

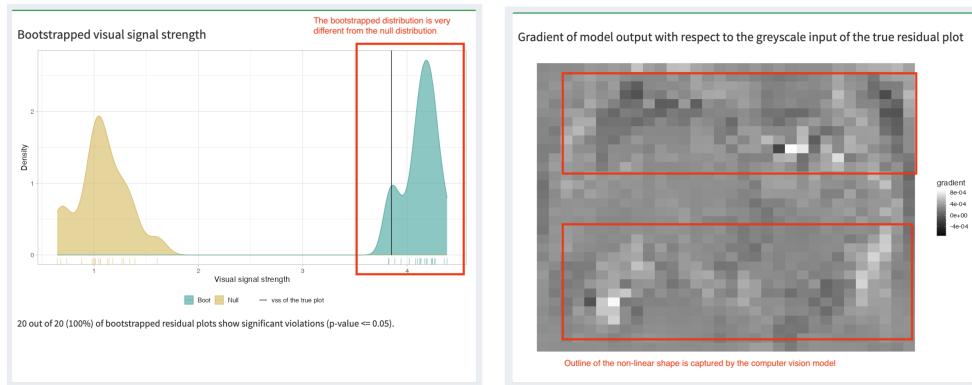


Figure 7. Summaries assessing the strength of the pattern and which elements of the plot contribute. The density plot helps verify if the bootstrapped distribution differs from the null distribution (left image). The attention map (right image) offers insights into whether the computer vision model has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

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

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

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

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

364 **6. Resources and supplementary material**

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

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

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

377 **References**

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

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

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