

1 Automated Residual Plot Assessment with the R Package 2 autovi and 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

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 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

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

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 3 provides a detailed documentation of the `autovi` package, including its usage and infrastructure. Section 4

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

50 **2. What is visual signal strength?**

51 Visual signal strength (VSS) refers to how prominently a specific set of visual patterns
 52 appears in an image. In the context of regression model diagnostics, it describes the
 53 clarity of visual patterns on a diagnostic plot that may indicate model violations. This
 54 concept can be categorized as weak, moderate, or strong. However, in our study, we
 55 treat it as a continuous positive real variable. Importantly, its interpretation depends
 56 on how it is linked to a function of the data or the underlying data generating process.
 57 Consequently, the meaning of visual signal strength varies across different model classes
 58 and even within the same model, depending on the chosen function.

59 In Li et al. (2024b), VSS was defined as an estimate of a distance measure that
 60 quantifies the disparity between the residual distribution of a fitted classical normal
 61 linear regression model and a reference distribution. This measure is based on the
 62 Kullback-Leibler (KL) divergence:

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

63 where D_{KL} is given by:

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

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

66 This distance measure requires knowledge of the true residual distribution, which is
 67 typically unknown. Therefore, a computer vision model was used for estimating the
 68 residual distribution (see Li et al. 2024b, for more details).

69 **3. R package: autovi**

70 The main purpose of `autovi` is to provide rejection decisions and p -values for testing
 71 the null hypothesis (H_0) that the regression model is correctly specified. The package

72 provides automated interpretation of residual plots using computer vision. The name
 73 **autovi** stands for **a**utomated **v**isual **i**nference.

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

77 3.1. Motivation for usage

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

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

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

102 3.2. Implementation

103 The `autovi` package is built on the `bandicoot` object-oriented programming (OOP)
 104 system ([Li 2024](#)), marking a departure from R’s traditional S3 generic system. This

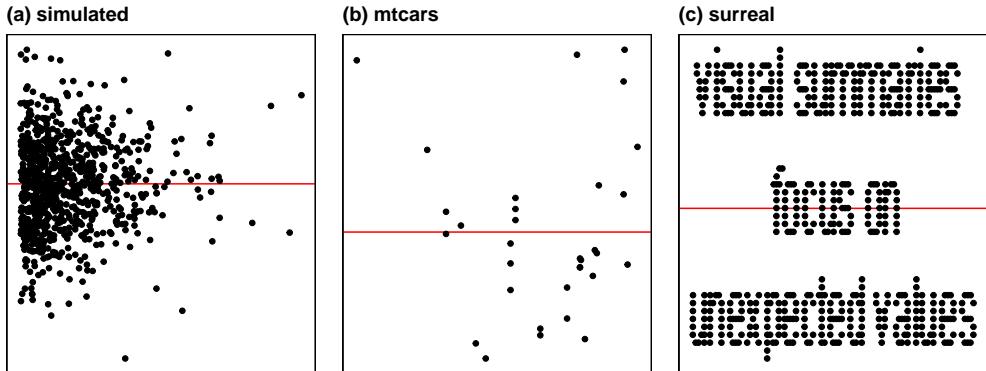


Figure 1. Reading residual plots can be a difficult task, particularly for students new to statistical modeling. The `autovi` package makes it easier. Here are three examples of residual plots, which may appear to have structure. According to `autovi`, the visual signal strengths (VSS) of these three examples are approximately (a) 1.53, (b) 3.57, (c) 5.87, resulting in (b), (c) being significant violations of good residuals, but (a) is consistent with a good residual plot.

105 OOP architecture enhances flexibility and modularity, allowing users to redefine key
 106 functions through method overriding. While similar functionality could be achieved
 107 using R's S3 system with generic functions, the OOP framework offers a more structured
 108 and extensible foundation for the package.

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

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

125 **3.3. Installation**

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

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

134 We recommend using the Shiny app `autovi.web` if users encounter installation
 135 problems.

136 **3.4. Usage**

137 **3.4.1. Numerical summary**

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

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

145 The code to do this is:

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

146 It produces the following summary:

```
-- <AUTO_VI object>
Status:
- Fitted model: lm
- Keras model: UNKNOWN
```

```

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

```

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

178 **3.4.2. Visual summary**

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

```
checker$plot_pair()
```

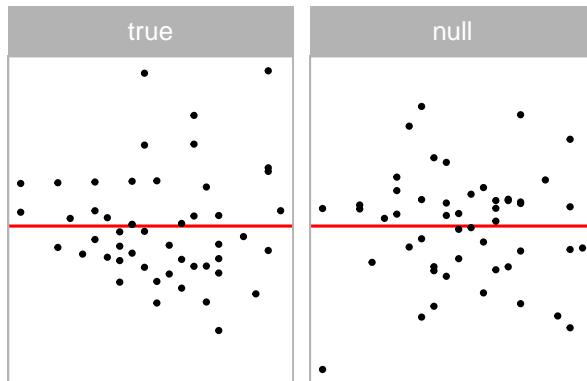


Figure 2. True plot alongside one null plot, for quick comparison.

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

```
checker$summary_plot()
```

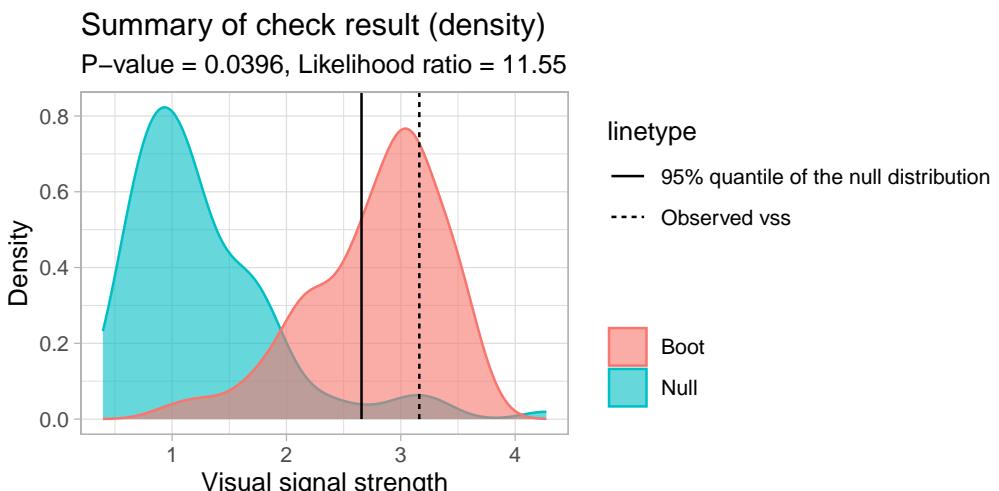


Figure 3. Summary plot comparing the densities of VSS for bootstrapped residual samples (red) relative to VSS for null plots (blue).

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

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

This visual summary provides an intuitive way to assess the model's fit and potential violations, allowing users to quickly grasp the results of the automated analysis.

3.5. Modularized infrastructure

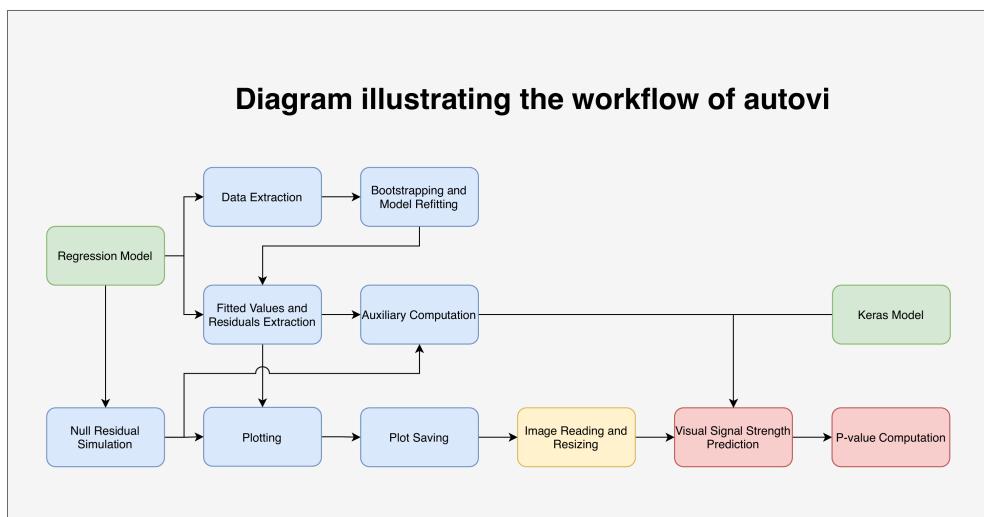


Figure 4. Diagram illustrating the infrastructure of the R package `autovi`. The modules in green are primary inputs provided by users. Modules in blue are overridable methods that can be modified to accommodate users' specific needs. The module in yellow is a pre-defined non-overridable method. The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

The initial motivation for developing `autovi` was to create a convenient interface for sharing the models described and trained in Li et al. (2024b). However, recognizing

that the classical normal linear regression model represents a restricted class of 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.

4. Web interface: `autovi.web`

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

4.1. Implementation

The interface `autovi.web` is built using the `shiny` (Chang et al. 2022) and `shinydashboard` (Chang & Borges Ribeiro 2021) R packages. Hosted on the `shinyapps.io` domain, the application is accessible through any modern web browser.

241 The R packages `htmltools` (Cheng et al. 2024) and `shinycssloaders` (Sali & Attali
242 2020) are used to render markdown documentation in shiny application, and for loading
243 animations for shiny widgets, respectively.

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

252 We also explored the possibility of implementing the web interface using frameworks
253 built on other languages, such as Python. However, server hosting domains that
254 natively support Python servers typically do not have the latest version of R installed.
255 Additionally, calling R from Python is typically done using the `rpy2` Python library
256 (Gautier 2024), but this approach can be awkward when dealing with language syntax
257 related to non-standard evaluation. Another option we considered was renting a server
258 where we could have full control, such as those provided by cloud platforms like Google
259 Cloud Platform (GCP) or Amazon Web Services (AWS). However, correctly setting up
260 the server and ensuring a secure deployment requires significant expertise. Ultimately,
261 the most practical solution was to use the `shiny` and `shinydashboard` frameworks,
262 which are well-established in the R community and offer a solid foundation for web
263 application development.

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

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

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

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

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

291 4.2. Usage

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

299 4.2.1. Reading data and setting parameters

300 The user can select to upload data as either a single set of residuals and fitted values
301 in a two (or more) column CSV file or a pre-computed lineup of residuals and null
302 datasets in a three (or more) column CSV file (i.e. multiple sets of residuals and fitted
303 values with a column indicating the set label). Here we illustrate use with lineup
304 example data sets (Figure 5). To use the lineup example data, click the “Use Lineup
305 Example” button. The data status will then update to show the number of rows and
306 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.

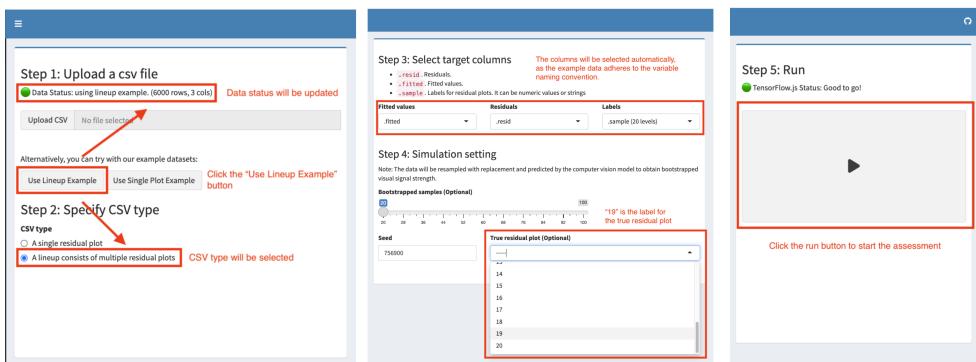


Figure 5. To begin the workflow for `autoovi` using the lineup example dataset, the user clicks the “Use Lineup Example” button (left) to load the example dataset, during which the data status and CSV type will be automatically updated. The user must manually select the label for the true residual plot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

316 4.2.2. Results provided

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

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

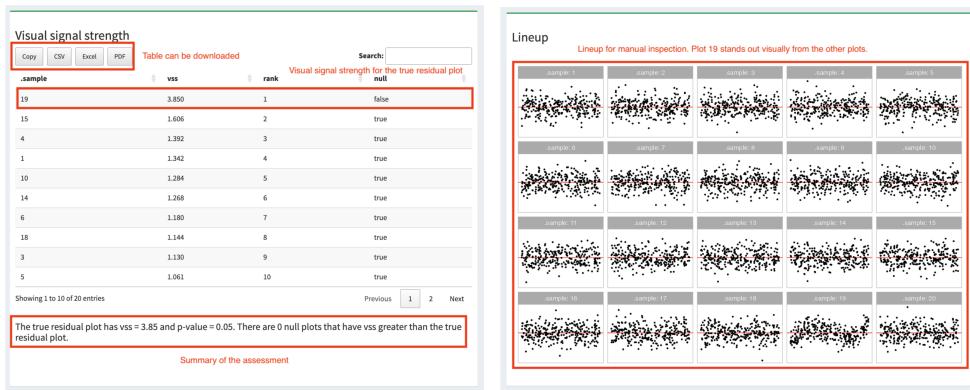


Figure 6. Results for the lineup. The VSS of the true residual plot is displayed in the first row of the table of VSS values for all the null plots (left image), with a summary text beneath the table providing the p -value to aid in decision-making. A lineup of residual plots allows for manual inspection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

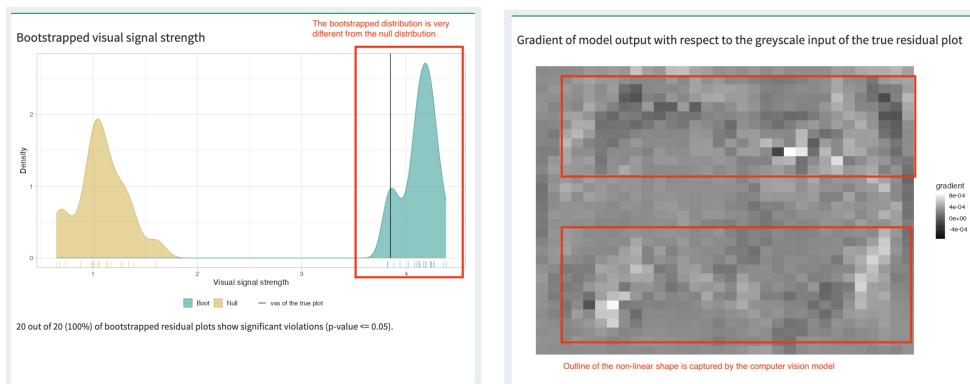


Figure 7. Summaries assessing the strength of the pattern and which elements of the plot contribute. The density plot helps verify if the bootstrapped distribution differs from the null distribution (left image). The attention map (right image) offers insights into whether the computer vision model has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

328

5. Conclusions

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

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

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

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

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

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

357 6. Resources and supplementary material

358 The current version of `autovi` can be installed from CRAN, and source code for
359 both packages are available at <https://github.com/TengMCing/autovi> and
360 https://github.com/TengMCing/autovi_web respectively. The web interface is
361 available from autoviweb.netlify.app.

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

365 These R packages were used for the work: `tidyverse` ([Wickham et al. 2019](#)), `lmtest`
366 ([Zeileis & Hothorn 2002](#)), `kableExtra` ([Zhu 2021](#)), `patchwork` ([Pedersen 2022](#)),
367 `rcartocolor` ([Nowosad 2018](#)), `glue` ([Hester & Bryan 2022](#)), `here` ([Müller 2020](#)),

368 `magick` (Ooms 2023), `yardstick` (Kuhn, Vaughan & Hvitfeldt 2024) and `reticulate`
 369 (Ushey, Allaire & Tang 2024).

370 *References*

- 371 ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G.S.,
 372 DAVIS, A., DEAN, J., DEVIN, M. et al. (2016). Tensorflow: Large-scale machine learning on
 373 heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .
- 374 ALLAIRE, J., TEAGUE, C., SCHEIDECKER, C., XIE, Y. & DERVIEUX, C. (2024). Quarto. doi:
 375 10.5281/zenodo.5960048. URL <https://github.com/quarto-dev/quarto-cli>.
- 376 ATTALI, D. (2021). *shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds*.
 377 URL <https://CRAN.R-project.org/package=shinyjs>. R package version 2.1.0.
- 378 BALAMUTA, J.J. (2024). *surreal: Create Datasets with Hidden Images in Residual Plots*. URL
 379 <https://CRAN.R-project.org/package=surreal>. R package version 0.0.1.
- 380 BREUSCH, T.S. & PAGAN, A.R. (1979). A simple test for heteroscedasticity and random coefficient
 381 variation. *Econometrica: Journal of the Econometric Society* , 1287–1294.
- 382 BUJA, A., COOK, D., HOFMANN, H., LAWRENCE, M., LEE, E.K., SWAYNE, D.F. & WICKHAM, H.
 383 (2009). Statistical inference for exploratory data analysis and model diagnostics. *Philosophical
 384 Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **367**,
 385 4361–4383.
- 386 CHANG, W. & BORGES RIBEIRO, B. (2021). *shinydashboard: Create Dashboards with 'Shiny'*.
 387 URL <https://CRAN.R-project.org/package=shinydashboard>. R package version 0.7.2.
- 388 CHANG, W., CHENG, J., ALLAIRE, J., SIEVERT, C., SCHLOERKE, B., XIE, Y., ALLEN, J.,
 389 MCPHERSON, J., DIPERT, A. & BORGES, B. (2022). *shiny: Web Application Framework for
 390 R*. URL <https://CRAN.R-project.org/package=shiny>. R package version 1.7.3.
- 391 CHENG, J., SIEVERT, C., SCHLOERKE, B., CHANG, W., XIE, Y. & ALLEN, J. (2024). *htmltools:
 392 Tools for HTML*. URL <https://CRAN.R-project.org/package=htmltools>. R package version
 393 0.5.8.
- 394 CLARK, A. et al. (2015). Pillow (pil fork) documentation. *readthedocs* .
- 395 COOK, R.D. & WEISBERG, S. (1982). *Residuals and influence in regression*. New York: Chapman
 396 and Hall.
- 397 DAVIES, R., LOCKE, S. & D'AGOSTINO McGOWAN, L. (2022). *datasauRus: Datasets from the
 398 Datasaurus Dozen*. URL <https://CRAN.R-project.org/package=datasauRus>. R package
 399 version 0.1.6.
- 400 GAUTIER, L. (2024). *Python interface to the R language (embedded R)*. URL <https://pypi.org/project/rpy2/>. Version 3.5.16.
- 401 GOODE, K. & REY, K. (2019). *ggResidpanel: Panels and Interactive Versions of Diagnostic Plots
 403 using 'ggplot2'*. URL <https://CRAN.R-project.org/package=ggResidpanel>. R package
 404 version 0.3.0.
- 405 HARTIG, F. (2022). *DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed)
 406 Regression Models*. URL <https://CRAN.R-project.org/package=DHARMa>. R package
 407 version 0.4.6.
- 408 HEBBALI, A. (2024). *olsrr: Tools for Building OLS Regression Models*. URL [https://CRAN.R-project.org/package=olsrr](https://CRAN.R-

 409 project.org/package=olsrr). R package version 0.6.0.
- 410 HESTER, J. & BRYAN, J. (2022). *glue: Interpreted String Literals*. URL [https://CRAN.R-project.org/package=glue](https://CRAN.R-

 411 project.org/package=glue). R package version 1.6.2.
- 412 JOHNSON, P.E. (2022). *rockchalk: Regression Estimation and Presentation*. URL [https://CRAN.R-project.org/package=rockchalk](https://CRAN.R-

 413 project.org/package=rockchalk). R package version 1.8.157.

- 414 KUHN, M., VAUGHAN, D. & HVITFELDT, E. (2024). *yardstick: Tidy Characterizations of Model*
 415 *Performance*. URL <https://CRAN.R-project.org/package=yardstick>. R package version
 416 1.3.1.
- 417 LI, W. (2024). *bandicoot: Light-weight python-like object-oriented system*. URL <https://CRAN.R->
 418 [project.org/package=bandicoot](https://CRAN.R-project.org/package=bandicoot).
- 419 LI, W., COOK, D., TANAKA, E. & VANDERPLAS, S. (2024a). A plot is worth a thousand tests:
 420 Assessing residual diagnostics with the lineup protocol. *Journal of Computational and*
 421 *Graphical Statistics* **33**, 1497–1511. doi:10.1080/10618600.2024.2344612.
- 422 LI, W., COOK, D., TANAKA, E., VANDERPLAS, S. & ACKERMANN, K. (2024b). Automated
 423 assessment of residual plots with computer vision models. *arXiv preprint arXiv:2411.01001* .
- 424 LONG, J.A. (2022). *jtools: Analysis and Presentation of Social Scientific Data*. URL <https://cran.r->
 425 [project.org/package=jtools](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. (????). Pandoc. URL <https://github.com/j>
 429 [gm/pandoc](https://github.com/jgm/pandoc).
- 430 MASON, H., LEE, S., LAA, U. & COOK, D. (2022). *cassowaryr: Compute Scagnostics on Pairs of*
 431 *Numeric Variables in a Data Set*. URL <https://CRAN.R-project.org/package=cassowary>. R
 432 package version 2.0.0.
- 433 MOON, K.W. (2020). *webr: Data and Functions for Web-Based Analysis*. URL <https://CRAN.R->
 434 [project.org/package=webr](https://CRAN.R-project.org/package=webr). R package version 0.1.5.
- 435 MÜLLER, K. (2020). *here: A simpler way to find your files*. URL <https://CRAN.R->
 436 [project.org/package=here](https://CRAN.R-project.org/package=here). R package version 1.0.1.
- 437 NOWOSAD, J. (2018). 'CARTOCOLORs' palettes. URL <https://nowosad.github.io/rkartocolor>. R
 438 package version 1.0.
- 439 OOMS, J. (2023). *magick: Advanced Graphics and Image-Processing in R*. URL <https://CRAN.R->
 440 [project.org/package=magick](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->
 442 [project.org/package=patchwork](https://CRAN.R-project.org/package=patchwork). R package version 1.1.2.
- 443 R CORE TEAM (2022). *R: A Language and Environment for Statistical Computing*. R Foundation
 444 for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- 445 RAMSEY, J.B. (1969). Tests for specification errors in classical linear least-squares regression
 446 analysis. *Journal of the Royal Statistical Society: Series B (Methodological)* **31**, 350–371.
- 447 REINHART, A. (2024). *regressinator: Simulate and Diagnose (Generalized) Linear Models*. URL
 448 <https://CRAN.R-project.org/package=regressinator>. R package version 0.2.0.
- 449 ROWLINGSON, B. & DIGGLE, P. (2023). *splancs: Spatial and Space-Time Point Pattern Analysis*.
 450 URL <https://CRAN.R-project.org/package=splancs>. R package version 2.01-44.
- 451 SALI, A. & ATTALI, D. (2020). *shinycssloaders: Add Loading Animations to a 'shiny' Output*
 452 *While It's Recalculating*. URL <https://CRAN.R-project.org/package=shinycssloaders>. R
 453 package version 1.0.0.
- 454 SHAPIRO, S.S. & WILK, M.B. (1965). An analysis of variance test for normality (complete samples).
 455 *Biometrika* **52**, 591–611.
- 456 USHEY, K., ALLAIRE, J. & TANG, Y. (2024). *reticulate: Interface to 'Python'*. URL <https://CRAN.R->
 457 [project.org/package=reticulate](https://CRAN.R-project.org/package=reticulate). R package version 1.35.0.
- 458 WARTON, D.I. (2023). Global simulation envelopes for diagnostic plots in regression models. *The*
 459 *American Statistician* **77**, 425–431.
- 460 WICKHAM, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York.
 461 URL <https://ggplot2.tidyverse.org>.

- 462 WICKHAM, H., AVERICK, M., BRYAN, J., CHANG, W., McGOWAN, L.D., FRANÇOIS, R.,
463 GROLEMUND, G., HAYES, A., HENRY, L., HESTER, J., KUHN, M., PEDERSEN, T.L., MILLER,
464 E., BACHE, S.M., MÜLLER, K., OOMS, J., ROBINSON, D., SEIDEL, D.P., SPINU, V.,
465 TAKAHASHI, K., VAUGHAN, D., WILKE, C., WOO, K. & YUTANI, H. (2019). Welcome to
466 the tidyverse. *Journal of Open Source Software* **4**, 1686. doi:10.21105/joss.01686.
- 467 WICKHAM, H., CHOWDHURY, N.R., COOK, D. & HOFMANN, H. (2020). *nullabor: Tools for*
468 *Graphical Inference*. URL <https://CRAN.R-project.org/package=nullabor>. R package
469 version 0.3.9.
- 470 ZAKAI, A. (2011). Emscripten: an llvm-to-javascript compiler. In *Proceedings of the ACM*
471 *international conference companion on Object oriented programming systems languages and*
472 *applications companion*. pp. 301–312.
- 473 ZEILEIS, A. & HOTHORN, T. (2002). Diagnostic checking in regression relationships. *R News* **2**,
474 7–10.
- 475 ZHU, H. (2021). *kableExtra: Construct complex table with kable and pipe syntax*. URL
476 <https://CRAN.R-project.org/package=kableExtra>. R package version 1.3.4.