

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

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

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

## Summary

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`

<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`

(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.

As noted in Li et al. (2024b), the lineup protocol has significant limitations in large-scale applications due to its reliance on human labor. To overcome this constraint, a computer vision model was developed alongside a corresponding statistical testing procedure to automate the assessment of residual plots. The model takes as input a residual plot and a set of auxiliary variables (such as the number of observations) and outputs a predicted visual signal strength (VSS). This VSS estimates the degree of deviation between the residual distribution of the fitted model and the reference distribution expected under correct model specification.

To make the statistical testing procedure and trained computer vision model widely accessible, we developed the R package `autovi` along with a companion web interface, `autovi.web`, which allows users to automatically assess their residual plots using 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 4 provides a detailed documentation of the `autovi` package, including its usage and infrastructure. Section 5 focuses on the

50 `autovi.web` interface, describing its design and usage, along with illustrative examples.  
 51 Finally, Section 6 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,  $\beta_Z$ , 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  $\boldsymbol{\mu}_z = \mathbf{RZ}\beta_z$ ,  $\mathbf{R} = \mathbf{I}_n - \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$  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  
79 its corresponding distance measure. It is trained on a large number of synthetic  
80 regression models, each designed to simulate specific violations of classical linear  
81 regression assumptions. These models incorporate non-linearity through Hermite  
82 polynomial transformations of predictors, heteroskedasticity by making the error  
83 variance a predictor-dependent function, and non-normality by drawing residuals from  
84 distributions such as discrete, uniform, and lognormal. Both simple and multiple linear  
85 regression structures are used, with controlled parameters to generate diverse and  
86 complex residual patterns. Since the data-generating process is known, the distance  
87 measure  $D$  can be explicitly calculated, enabling supervised training. The computer  
88 vision model takes a residual plot as input and outputs the corresponding distance  
89 measure, learning to quantify model violations directly from visual patterns. Additional  
90 details are provided in [Li et al. \(2024b\)](#).

### 91 3. Definition and simulation of null and bootstrapped residuals

92 In the subsequent sections, we will frequently refer to null residuals and bootstrapped  
93 residuals, so it is helpful to first define and explain how they are generated.

94 **Null residuals** are used to generate null plots within the lineup protocol framework,  
95 serving as the foundation for the statistical testing in our automated residual plot  
96 assessment. Specifically, they represent residuals generated under the null hypothesis  
97 that the model is correctly specified. A common method for simulating null residuals  
98 in linear regression involves sampling from a normal distribution with mean zero and  
99 variance equal to the estimated variance of the error term. These simulated residuals  
100 and their corresponding plots depict what one would expect from a correctly specified  
101 model. If the true residual plot exhibits noticeable deviations from these null plots, it  
102 may suggest model misspecification.

103 Our computer vision model is trained to assign lower VSS to null plots and higher VSS  
104 to plots that display distinct patterns. Accordingly, statistical testing is performed  
105 by computing the proportion of null plots whose VSS equals or exceeds that of the

106 observed residual plot. This proportion serves as a  $p$ -value for a one-sided hypothesis  
107 test.

108 **Bootstrapped residuals** are obtained by refitting the model on bootstrap samples,  
109 which are generated by sampling individual observations with replacement from the  
110 original dataset. The residual plots obtained from these refitted models are evaluated  
111 using the same computer vision model. The predicted VSS from the bootstrapped  
112 plots provide an empirical estimate of the variation in the VSS of the observed  
113 residual plot. By examining the proportion of bootstrapped plots that also exhibit  
114 significant violations, we can assess whether the original conclusion is robust to  
115 sampling variability.

#### 116 4. R package: autovi

117 The main purpose of **autovi** is to provide rejection decisions and  $p$ -values for testing  
118 the null hypothesis ( $H_0$ ) that the regression model is correctly specified. The package  
119 provides automated interpretation of residual plots using computer vision. The name  
120 **autovi** stands for **automated visual inference**. This functionality can be accessed  
121 through the R package **autovi**, or through a web interface, **autovi.web**, which enables  
122 users to perform analyses without installing R, Python, or their associated dependencies  
123 locally.

##### 124 4.1. Motivation

125 Figure 1 shows three sets of plots of residuals against fitted values. The simulated  
126 example in (a) might be interpreted as a heteroscedastic pattern, however the  
127 automated reading would predict this to have a visual signal strength (VSS) of  
128 1.53, with a corresponding  $p$ -value of 0.25. This means it would be interpreted as  
129 a good residual plot, that there is nothing in the data to indicate a violation of  
130 model assumptions. Skewness in the predictor variables is generating the apparent  
131 heteroscedasticity, where the smaller variance in residuals at larger fitted values is  
132 due to smaller sample size only. The Breusch-Pagan test (Breusch & Pagan 1979) for  
133 heteroscedasticity would also not reject this as good residual plot.

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

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

141 The third example is generated using the `surreal` package (Balamuta 2024) where  
 142 structured residuals are hidden in data, to be revealed if the correct model is specified.  
 143 Here a quote based on Tukey is used as the residual structure “visual summaries focus  
 144 on unexpected values”. The automated plot reading predicts the VSS to be 5.87, with  
 145 a  $p$ -value less than 0.05. This structure is blindingly obvious visually, but a RESET  
 146 test for nonlinear structure would not report a problem. (It would be detected by  
 147 a Breusch-Pagan for heteroscedasticity and also Shapiro-Wilk test (Shapiro & Wilk  
 148 1965) for non-normality.)

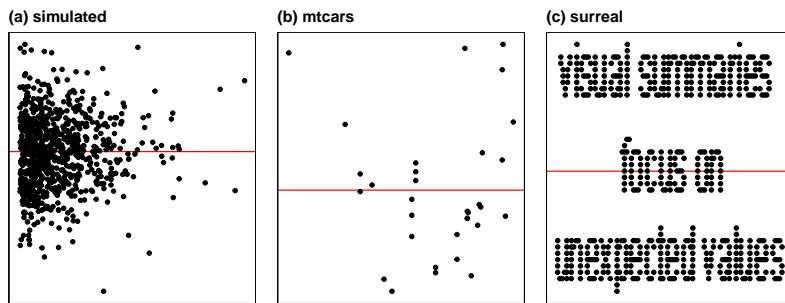


Figure 1. Reading residual plots can be a difficult task, particularly for students new to statistical modeling. The `autovi` package makes it easier. Here are three examples of residual plots, which may appear to have structure. According to `autovi`, the visual signal strengths (VSS) of these three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

149 **4.2. Implementation**

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

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

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

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

### 168 4.3. Installation

169 The `autovi` package is available on CRAN. It is actively developed and maintained,  
170 with the latest updates accessible on GitHub. This paper uses `autovi` version 0.4.2.  
171 The package includes internal functions to check the current Python environment used  
172 by the `reticulate` package. If the necessary Python packages are not installed in the  
173 Python interpreter, an error will be raised. If you want to select a specific Python  
174 environment, you can do so by calling the `reticulate::use_python()` function before  
175 using the `autovi` package.

176 We recommend using the Shiny app `autovi.web` if users encounter installation  
177 problems.

### 178 4.4. Usage

#### 179 4.4.1. Numerical summary

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

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

187 The code to do this is:

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

188 It produces the following summary:

189

```
190 -- <AUTO_VI object>
191 Status:
192 - Fitted model: lm
193 - Keras model: (None, 32, 32, 3) + (None, 5) -> (None, 1)
194     - Output node index: 1
195 - Result:
196     - Observed visual signal strength: 3.16 (p-value = 0.0396)
197     - Null visual signal strength: [100 draws]
198         - Mean: 1.274
199         - Quantiles:
200
201             25%   50%   75%   80%   90%   95%   99%
202             0.802 1.111 1.575 1.666 1.919 2.657 3.348
203
204     - Bootstrapped visual signal strength: [100 draws]
205         - Mean: 2.795 (p-value = 0.05941)
206         - Quantiles:
207
208             25%   50%   75%   80%   90%   95%   99%
209             2.455 2.941 3.177 3.300 3.474 3.537 3.668
210
211     - Likelihood ratio: 0.7333 (boot) / 0.06284 (null) = 11.67
```

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

218 than the observed VSS in both the bootstrapped data values and the simulated null  
 219 values.

220 **4.4.2. Visual summary**

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

```
checker$plot_pair()
```

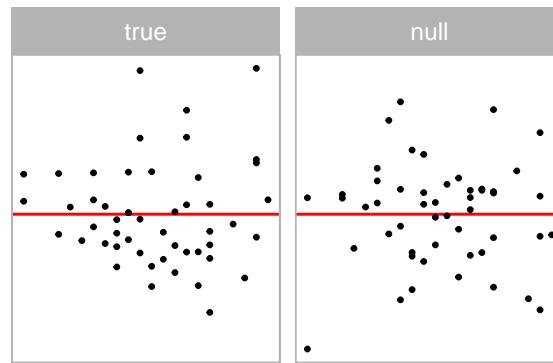


Figure 2. True plot alongside one null plot, for quick comparison.

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

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

```
checker$summary_plot()
```

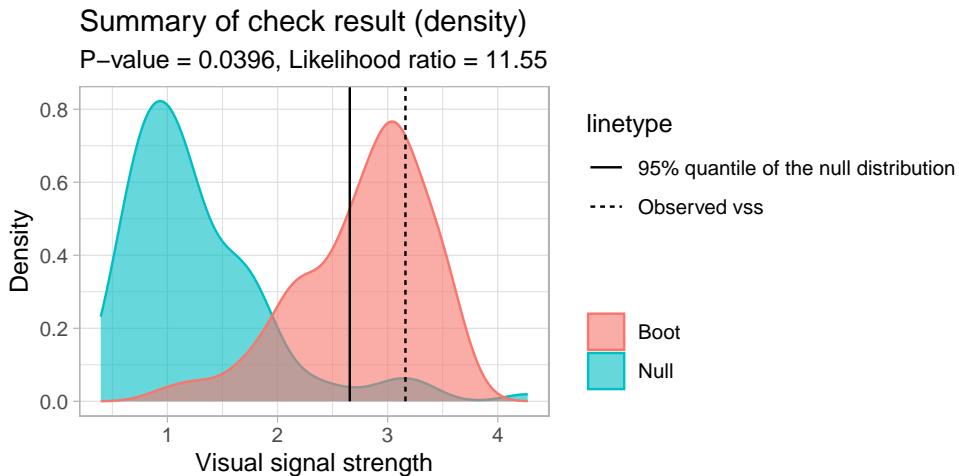


Figure 3. Summary plot comparing the densities of VSS for bootstrapped residual samples (red) relative to VSS for null plots (blue).

- 232 In the result, shown in Figure 3, the blue area represents the density of VSS for null  
 233 residual plots, while the red area shows the density for bootstrapped residual plots.  
 234 The dashed line indicates the VSS of the true residual plot, and the solid line marks  
 235 the critical value at a 95% significance level. The  $p$ -value and the likelihood ratio are  
 236 displayed in the subtitle. The likelihood ratio represents the ratio of the likelihood  
 237 of observing the VSS of the true residual plot from the bootstrapped distribution  
 238 compared to the null distribution.
- 239 Interpreting the plot involves several key aspects. If the dashed line falls to the right of  
 240 the solid line, it suggests rejecting the null hypothesis. The degree of overlap between  
 241 the red and blue areas indicates similarity between the true residual plot and null  
 242 plots; greater overlap suggests more similarity. Lastly, the portion of the red area to  
 243 the right of the solid line represents the percentage of bootstrapped models considered  
 244 to have model violations.
- 245 This visual summary provides an intuitive way to assess the model's fit and potential  
 246 violations, allowing users to quickly grasp the results of the automated analysis.

#### 247 4.5. Modularized infrastructure

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

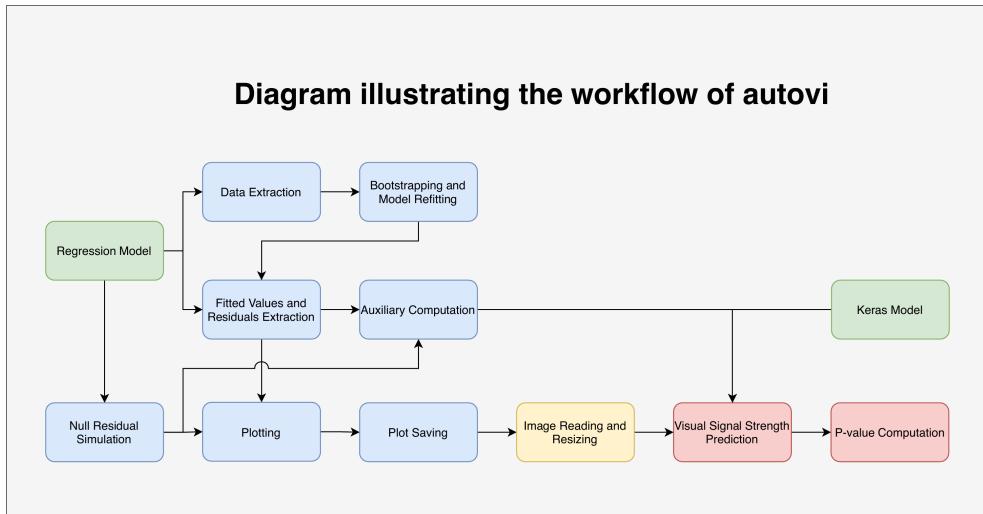


Figure 4. Diagram illustrating the infrastructure of the R package **autovi**. The modules in green are primary inputs provided by users. Modules in blue are overridable methods that can be modified to accommodate users' specific needs. The module in yellow is a pre-defined non-overridable method. The 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 package takes regression models and a **Keras** model as primary inputs. Modules for VSS prediction and *p*-value computation are fixed but accessible via function arguments, using **TensorFlow** for inference and statistical testing. The image loading module is also fixed, using **PIL** to read and resize images based on the **Keras** model's input shape. The remaining seven modules are overridable, allowing users to adapt the workflow as needed. The data extraction module extracts a **data.frame** containing variables used

in the regression model. The bootstrapping and refitting module resamples the data and refits the model. The fitted values and residuals extraction module returns these values as a `data.frame`. The auxiliary computation module calculates scagnostics such as monotonicity. The plotting module generates a `ggplot` in a standard format, and the plot saving module exports it at the same resolution as the training images. These modules are described in detail in the package documentation.

#### 4.6. Extension to Other Model Classes

The `autovi` R package can be extended to accommodate other classes of models beyond linear regression, such as generalized linear models (`glm`). This is achieved by substituting the relevant overridable modules, as illustrated in Figure 4.

We provide an example of defining a new checker class tailored for Poisson regression using the `glm` framework:

1. Define a new class using `new_class()` with `AUTO_VI` as the parent class.
2. Override the necessary methods using `register_method()`. In this example, we use Pearson residuals. To simulate null residuals, we assume the fitted model is correct and the estimated coefficients are accurate. New response values are generated accordingly, and a new model is fitted to this simulated response. Null residuals are then extracted from this refitted model.
3. Create an alias for the `instantiate()` method of the new class.

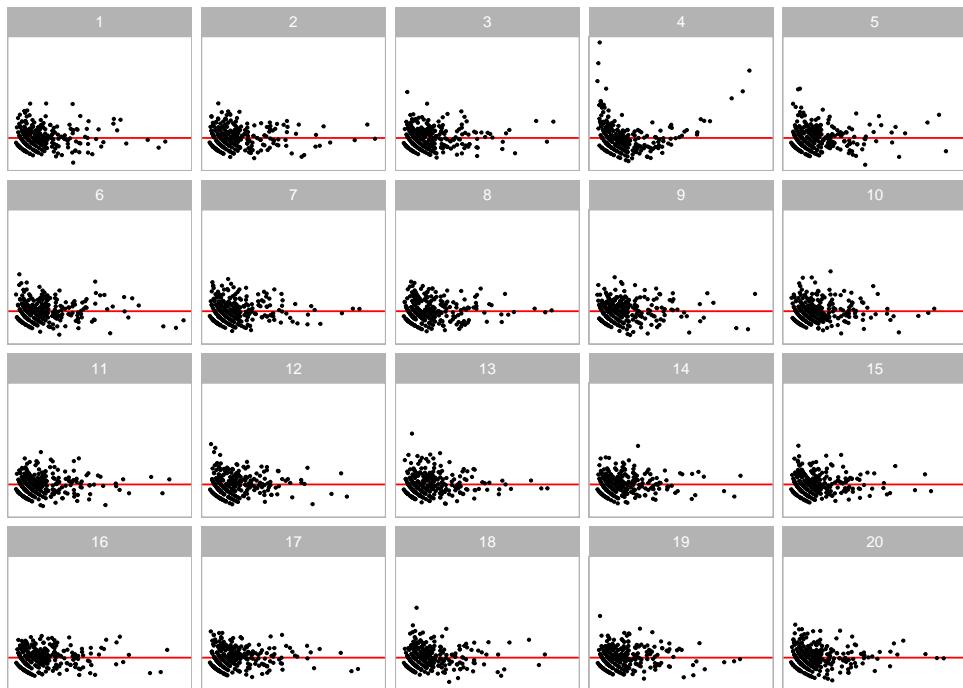
```
AUTO_POIS_VI <- new_class(AUTO_VI, class_name = "AUTO_POIS_VI")
register_method(
  AUTO_POIS_VI,
  get_fitted_and_resid = function(fitted_model = self$fitted_model) {
    tibble(.fitted = fitted(fitted_model),
          .resid = resid(fitted_model, type = "pearson"))
  },
  null_method = function(fitted_model = self$fitted_model) {
    dat <- model.frame(fitted_model)
    dat[[1]] <- rpois(nrow(dat), lambda = fitted(fitted_model))
    new_mod <- update(fitted_model, data = dat)
    return(self$get_fitted_and_resid(new_mod))
  }
)
auto_pois_vi <- AUTO_POIS_VI$instantiate
```

289 The resulting checker class can be employed analogously to the linear model case  
 290 described in Section 4.4. For illustration, we fit a Poisson model in which the quadratic  
 291 term of the predictor  $x$  is intentionally omitted. This misspecification manifests as a  
 292 pronounced U-shaped pattern in the lineup display, which is also successfully identified  
 293 by the computer vision model, yielding a p-value substantially below the conventional  
 294 threshold of 0.05.

295 It is important to note, however, that the pre-trained computer vision model was  
 296 developed specifically for diagnostic evaluation of linear regression. Its applicability  
 297 to other model classes relies on the assumption that the null residual plots exhibit  
 298 characteristics broadly consistent with those of well-behaved linear regression residuals,  
 299 that is, residuals should be approximately randomly scattered around zero, display  
 300 roughly constant variance across the range of fitted values, and exhibit no discernible  
 301 structure or curvature. If these conditions are not met, or if model violations do not  
 302 give rise to visually detectable patterns, the validity of the automated diagnostics may  
 303 be compromised. For further discussion on extending the methodology to other model  
 304 classes, refer to Li et al. (2024b).

```
x <- rnorm(300, sd = 0.5)
y <- rpois(300, lambda = exp(1 + x + x^2))
pois_checker <- auto_pois_vi(
  glm(y ~ x, family = "poisson"),
  keras_model = get_keras_model("vss_phn_32")
)
pois_checker$plot_lineup()
```

The true residual plot is at position 4.



305

```
pois_checker$check()
```

306

```
307 -- <AUTO_POIS_VI object>
308 Status:
309 - Fitted model: glm, lm
310 - Keras model: (None, 32, 32, 3) + (None, 5) -> (None, 1)
311     - Output node index: 1
312 - Result:
313     - Observed visual signal strength: 4.875 (p-value = 0.009901)
314     - Null visual signal strength: [100 draws]
315         - Mean: 1.331
316         - Quantiles:
317
318             25%    50%    75%    80%    90%    95%    99%
319             1.035  1.233  1.488  1.644  1.941  2.276  2.639
320
321     - Bootstrapped visual signal strength: [100 draws]
```

322 - Mean: 5.51 (p-value = 0.009901)

323 - Quantiles:

25%	50%	75%	80%	90%	95%	99%
5.330	5.505	5.698	5.735	5.830	5.903	6.013

328 - Likelihood ratio: 0.05096 (boot) / 0 (null) = Extremely large

## 329 5. Web interface: `autovi.web`

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

### 338 5.1. Implementation

339 The interface `autovi.web` is built using the `shiny` ([Chang et al. 2022](#)) and  
 340 `shinydashboard` ([Chang & Borges Ribeiro 2021](#)) R packages. Hosted on the  
 341 `shinyapps.io` domain, the application is accessible through any modern web browser.  
 342 The R packages `htmltools` ([Cheng et al. 2024](#)) and `shinycssloaders` ([Sali & Attali  
 343 2020](#)) are used to render markdown documentation in shiny application, and for loading  
 344 animations for shiny widgets, respectively.

345 Determining the best way to implement the backend was difficult. In our initial  
 346 planning for `autovi.web`, we considered implementing the entire web application using  
 347 the `webr` framework ([Moon 2020](#)), which would have allowed the entire application to  
 348 run directly in the user's browser. However, `webr` does not support packages which use  
 349 compiled fortran code, which is required by `splancs` ([Rowlingson & Diggle 2023](#)), a  
 350 dependency of `autovi`. In the future, it is possible that a working Emscripten ([Zakai  
 351 2011](#)) version of this package may allow full `webr` support.

352 We also explored the possibility of implementing the web interface using frameworks  
 353 built on other languages, such as Python. However, server hosting domains that  
 354 natively support Python servers typically do not have the latest version of R installed.

355 Additionally, calling R from Python is typically done using the `rpy2` Python library  
356 ([Gautier 2024](#)), but this approach can be awkward when dealing with language syntax  
357 related to non-standard evaluation. Another option we considered was renting a server  
358 where we could have full control, such as those provided by cloud platforms like  
359 Google Cloud Platform (GCP) or Amazon Web Services (AWS). However, deploying  
360 and maintaining the server securely requires some expertise. Ultimately, the most  
361 practical solution was to use the `shiny` and `shinydashboard` frameworks, which are  
362 well-established in the R community and offer a solid foundation for web application  
363 development.

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

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

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

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

385 To allow communication between `TensorFlow.js` and other components of the Shiny  
386 application, the `shinyjs` R package ([Attali 2021](#)) is used. This package allows calling  
387 custom JavaScript code within the Shiny framework. The specialized JavaScript

388 code for initializing `TensorFlow.js` and calling `TensorFlow.js` for VSS prediction is  
389 deployed alongside the Shiny application as additional resources.

390 **5.2. Usage**

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

398 **5.2.1. Reading data and setting parameters**

399 The user can select to upload data as either a single set of residuals and fitted values  
400 in a two (or more) column CSV file or a pre-computed lineup of residuals and null  
401 datasets in a three (or more) column CSV file (i.e. multiple sets of residuals and fitted  
402 values with a column indicating the set label). Here we illustrate use with lineup  
403 example data sets (Figure 5). To use the lineup example data, click the “Use Lineup  
404 Example” button. The data status will then update to show the number of rows and  
405 columns in the dataset, and the CSV type will automatically be selected to the correct  
406 option. Since the example dataset follows the variable naming conventions assumed  
407 by the web application, the columns for fitted values, residuals, and labels of residual  
408 plots are automatically mapped such that the column named as `.fitted` is mapped  
409 to fitted values, `.resid` is mapped to residuals and if applicable, `.sample` to labels of  
410 the residual set (middle image). If the user is working with a custom dataset, these  
411 options must be set accordingly. Whenever a data containing a lineup, the user must  
412 manually select the label for the true residual plot, otherwise the web application does  
413 not provide all the results. The last step is to click the play button (right image) to  
414 start the assessment.

415 **5.2.2. Results provided**

416 Results are provided in multiple panels. The first row of the table Figure 6 is the most  
417 crucial to check, as it provides the VSS and the rank of the true residual plot among  
418 the other plots. The summary text beneath the table provides the *p*-value, which can  
419 be used for quick decision-making. The lineup is for manual inspection, and the user

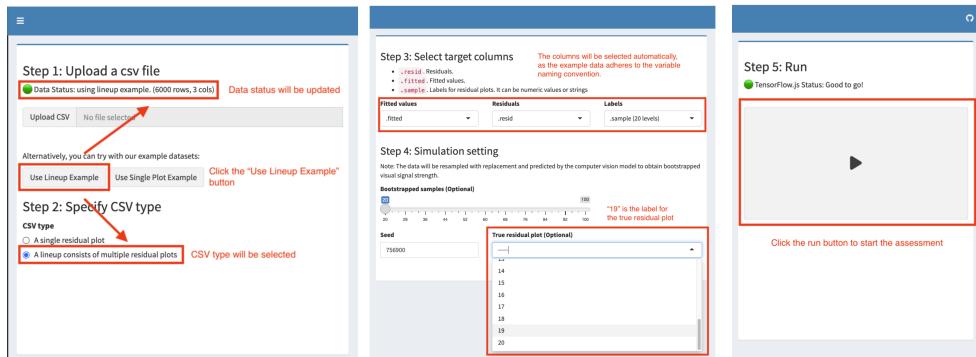


Figure 5. To begin the workflow for `autoovi` using the lineup example dataset, the user clicks the “Use Lineup Example” button (left) to load the example dataset, during which the data status and CSV type will be automatically updated. The user must manually select the label for the true residual plot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

420 should see if the true residual plot is visually distinguishable from the other plots, to  
421 confirm if the model violation is serious.

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

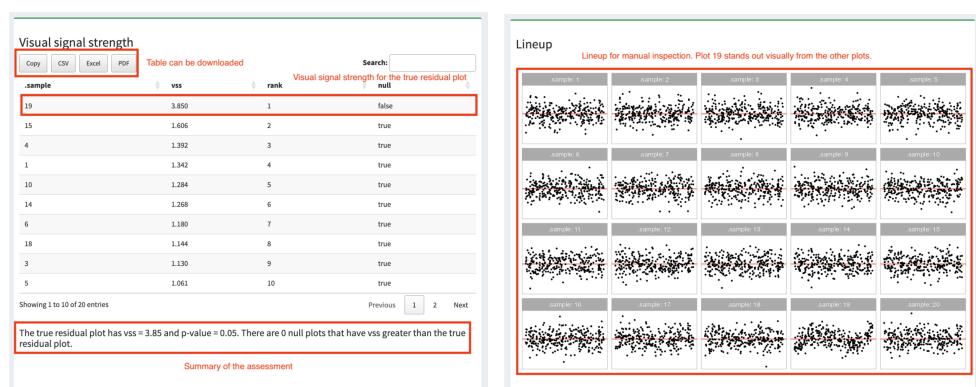


Figure 6. Results for the lineup. The VSS of the true residual plot is displayed in the first row of the table of VSS values for all the null plots (left image), with a summary text beneath the table providing the *p*-value to aid in decision-making. A lineup of residual plots allows for manual inspection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

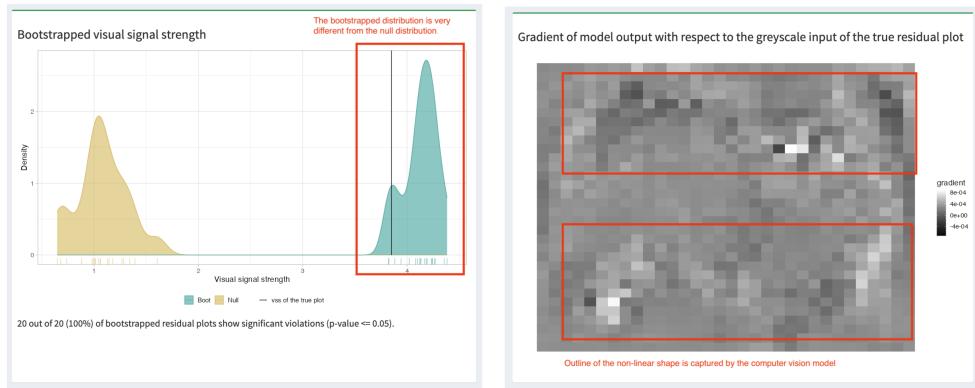


Figure 7. Summaries assessing the strength of the pattern and which elements of the plot contribute. The density plot helps verify if the bootstrapped distribution differs from the null distribution (left image). The attention map (right image) offers insights into whether the computer vision model has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

427

## 6. Conclusions

- 428 This paper presents new regression diagnostics software, the R package **autovi** and  
 429 its accompanying web interface, **autovi.web**. It addresses a critical gap in the current  
 430 landscape of statistical software. While regression tools are widely available, effective  
 431 and efficient diagnostic methods have lagged behind, particularly in the field of residual  
 432 plot interpretation.
- 433 The **autovi** R package, introduced in this paper, automates the assessment of residual  
 434 plots by incorporating a computer vision model, reducing reliance on time-consuming  
 435 and potentially inconsistent human interpretation. This automation improves the  
 436 efficiency of the diagnostic process and promotes consistency in model evaluation  
 437 across different users and studies.
- 438 The development of the accompanying Shiny app, **autovi.web**, expands access to these  
 439 advanced diagnostic tools, by providing a user-friendly interface. It makes automated  
 440 residual plot assessment accessible to a broader audience, including those who may not  
 441 have extensive programming experience. This web-based solution effectively addresses  
 442 the potential barriers to adoption, such as complex dependencies and installation  
 443 requirements, that are often associated with advanced statistical software.
- 444 The combination of **autovi** and **autovi.web** offers a comprehensive solution to the  
 445 challenges of residual plot interpretation in regression analysis. These tools have the  
 446 potential to significantly improve the quality and consistency of model diagnostics

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

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

## 456 7. Resources and supplementary material

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

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

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

## 469 References

- 470 ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G.S.,  
 471 DAVIS, A., DEAN, J., DEVIN, M. et al. (2016). Tensorflow: Large-scale machine learning on  
 472 heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .
- 473 ALLAIRE, J., TEAGUE, C., SCHEIDECKER, C., XIE, Y. & DERVIEUX, C. (2024). Quarto. doi:  
 474 10.5281/zenodo.5960048. URL <https://github.com/quarto-dev/quarto-cli>.
- 475 ATTALI, D. (2021). *shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds*.  
 476 URL <https://CRAN.R-project.org/package=shinyjs>. R package version 2.1.0.
- 477 BALAMUTA, J.J. (2024). *surreal: Create Datasets with Hidden Images in Residual Plots*. URL  
 478 <https://CRAN.R-project.org/package=surreal>. R package version 0.0.1.
- 479 BREUSCH, T.S. & PAGAN, A.R. (1979). A simple test for heteroscedasticity and random coefficient  
 480 variation. *Econometrica: Journal of the Econometric Society* , 1287–1294.

- 481 BUJA, A., COOK, D., HOFMANN, H., LAWRENCE, M., LEE, E.K., SWAYNE, D.F. & WICKHAM, H.  
482 (2009). Statistical inference for exploratory data analysis and model diagnostics. *Philosophical  
483 Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **367**,  
484 4361–4383.
- 485 CHANG, W. & BORGES RIBEIRO, B. (2021). *shinydashboard: Create Dashboards with 'Shiny'*.  
486 URL <https://CRAN.R-project.org/package=shinydashboard>. R package version 0.7.2.
- 487 CHANG, W., CHENG, J., ALLAIRE, J., SIEVERT, C., SCHLOERKE, B., XIE, Y., ALLEN, J.,  
488 MCPHERSON, J., DIPERT, A. & BORGES, B. (2022). *shiny: Web Application Framework for  
489 R*. URL <https://CRAN.R-project.org/package=shiny>. R package version 1.7.3.
- 490 CHENG, J., SIEVERT, C., SCHLOERKE, B., CHANG, W., XIE, Y. & ALLEN, J. (2024). *htmltools:  
491 Tools for HTML*. URL <https://CRAN.R-project.org/package=htmltools>. R package version  
492 0.5.8.
- 493 CLARK, A. et al. (2015). Pillow (pil fork) documentation. *readthedocs*.
- 494 COOK, R.D. & WEISBERG, S. (1982). *Residuals and influence in regression*. New York: Chapman  
495 and Hall.
- 496 DAVIES, R., LOCKE, S. & D'AGOSTINO McGOWAN, L. (2022). *datasauRus: Datasets from the  
497 Datasaurus Dozen*. URL <https://CRAN.R-project.org/package=datasauRus>. R package  
498 version 0.1.6.
- 499 GAUTIER, L. (2024). *Python interface to the R language (embedded R)*. URL [https://pypi.org/project/rpy2/](https://pypi.org/<br/>500 project/rpy2/). Version 3.5.16.
- 501 GOODE, K. & REY, K. (2019). *ggResidpanel: Panels and Interactive Versions of Diagnostic Plots  
502 using 'ggplot2'*. URL <https://CRAN.R-project.org/package=ggResidpanel>. R package version  
503 0.3.0.
- 504 HARTIG, F. (2022). *DHARMA: Residual Diagnostics for Hierarchical (Multi-Level / Mixed)  
505 Regression Models*. URL <https://CRAN.R-project.org/package=DHARMA>. R package  
506 version 0.4.6.
- 507 HEBBALI, A. (2024). *olsrr: Tools for Building OLS Regression Models*. URL [https://CRAN.R-project.org/package=olsrr](https://CRAN.R-<br/>508 project.org/package=olsrr). R package version 0.6.0.
- 509 HESTER, J. & BRYAN, J. (2022). *glue: Interpreted String Literals*. URL [https://CRAN.R-project.org/package=glue](https://CRAN.R-<br/>510 project.org/package=glue). R package version 1.6.2.
- 511 JOHNSON, P.E. (2022). *rockchalk: Regression Estimation and Presentation*. URL [https://CRAN.R-project.org/package=rockchalk](https://CRAN.R-<br/>512 project.org/package=rockchalk). R package version 1.8.157.
- 513 KUHN, M., VAUGHAN, D. & HVITFELDT, E. (2024). *yardstick: Tidy Characterizations of Model  
514 Performance*. URL <https://CRAN.R-project.org/package=yardstick>. R package version 1.3.1.
- 515 LI, W. (2024). *bandicoot: Light-weight python-like object-oriented system*. URL [https://CRAN.R-project.org/package=bandicoot](https://CRAN.R-<br/>516 project.org/package=bandicoot).
- 517 LI, W., COOK, D., TANAKA, E. & VANDERPLAS, S. (2024a). A plot is worth a thousand tests:  
518 Assessing residual diagnostics with the lineup protocol. *Journal of Computational and  
519 Graphical Statistics* **33**, 1497–1511. doi:10.1080/10618600.2024.2344612.
- 520 LI, W., COOK, D., TANAKA, E., VANDERPLAS, S. & ACKERMANN, K. (2024b). Automated  
521 assessment of residual plots with computer vision models. *arXiv preprint arXiv:2411.01001*.
- 522 LONG, J.A. (2022). *jtools: Analysis and Presentation of Social Scientific Data*. URL [https://cran.r-project.org/package=jtools](https://cran.r-<br/>523 project.org/package=jtools). R package version 2.2.0.
- 524 LOY, A. & HOFMANN, H. (2014). *Hlmdiag: A suite of diagnostics for hierarchical linear models in  
525 r*. *Journal of Statistical Software* **56**, 1–28.
- 526 MACFARLANE, J., KREWINKEL, A. & ROSENTHAL, J. (2024). Pandoc. URL [https://github.com/jgm/pandoc](https://github.com/<br/>527 jgm/pandoc).
- 528 MASON, H., LEE, S., LAA, U. & COOK, D. (2022). *cassowaryr: Compute Scagnostics on Pairs of  
529 Numeric Variables in a Data Set*. URL <https://CRAN.R-project.org/package=cassowary>. R

- 530 package version 2.0.0.
- 531 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.
- 532 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.
- 533 NOWOSAD, J. (2018). 'CARTOCOLORs' palettes. URL <https://nowosad.github.io/rkartocolor>. R package version 1.0.
- 534 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.
- 535 PEDERSEN, T.L. (2022). *patchwork: The composer of plots*. URL <https://CRAN.R-project.org/package=patchwork>. R package version 1.1.2.
- 536 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/>.
- 537 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.
- 538 REINHART, A. (2024). *regressinator: Simulate and Diagnose (Generalized) Linear Models*. URL <https://CRAN.R-project.org/package=regressinator>. R package version 0.2.0.
- 539 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.
- 540 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.
- 541 SHAPIRO, S.S. & WILK, M.B. (1965). An analysis of variance test for normality (complete samples). *Biometrika* **52**, 591–611.
- 542 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.
- 543 WARTON, D.I. (2023). Global simulation envelopes for diagnostic plots in regression models. *The American Statistician* **77**, 425–431.
- 544 WICKHAM, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. URL <https://ggplot2.tidyverse.org>.
- 545 WICKHAM, H., AVERICK, M., BRYAN, J., CHANG, W., McGOWAN, L.D., FRANÇOIS, R., GROLEMUND, G., HAYES, A., HENRY, L., HESTER, J., KUHN, M., PEDERSEN, T.L., MILLER, E., BACHE, S.M., MÜLLER, K., OOMS, J., ROBINSON, D., SEIDEL, D.P., SPINU, V., TAKAHASHI, K., VAUGHAN, D., WILKE, C., WOO, K. & YUTANI, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software* **4**, 1686. doi:10.21105/joss.01686.
- 546 WICKHAM, H., CHOWDHURY, N.R., COOK, D. & HOFMANN, H. (2020). *nullabor: Tools for Graphical Inference*. URL <https://CRAN.R-project.org/package=nullabor>. R package version 0.3.9.
- 547 ZAKAI, A. (2011). Emscripten: an llvm-to-javascript compiler. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*. pp. 301–312.
- 548 ZEILEIS, A. & HOTHORN, T. (2002). Diagnostic checking in regression relationships. *R News* **2**, 7–10.
- 549 ZHU, H. (2021). *kableExtra: Construct complex table with kable and pipe syntax*. URL <https://CRAN.R-project.org/package=kableExtra>. R package version 1.3.4.