

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

3 Weihao Li^{1,2}, Dianne Cook¹, Emi Tanaka², Susan VanderPlas³ and Klaus
4 Ackermann¹

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

Summary

7 Visual assessment of residual plots is a common approach for diagnosing linear
models, but it relies on manual evaluation, which does not scale well and can
lead to inconsistent decisions across analysts. The lineup protocol, which embeds
the observed plot among null plots, can reduce subjectivity but requires even
more human effort. In today's data-driven world, such tasks are well-suited for
automation. We present a new R package that uses a computer vision model to
automate the evaluation of residual plots. An accompanying Shiny app is provided
for ease of use. Given a sample of residuals, the model predicts a visual signal
strength (VSS) and offers supporting information to help analysts assess 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`
16 (Hebbali 2024), `rockchalk` (Johnson 2022), and `ggResidpanel` (Goode & Rey 2019)

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: patrick.li@anu.edu.au

17 deliver similar graphical diagnostics, often with enhanced aesthetics or interactive
18 features. These tools collectively produce the core diagnostic plots outlined in the
19 classical text by Cook & Weisberg (1982). The `ecostats` package (Warton 2023)
20 extends these diagnostics by incorporating simulation envelopes into residual plots.
21 Meanwhile, DHARMa (Hartig 2022) compares empirical quantiles (0.25, 0.5, and 0.75)
22 of scaled residuals to their theoretical counterparts, with a strong focus on identifying
23 model violations such as heteroscedasticity, misspecified functional forms, and issues
24 specific to generalized linear and mixed-effect models, like over/under-dispersion. It
25 also provides conventional test annotations to reduce the risk of misinterpretation.

26 However, relying solely on subjective assessments of these plots can lead to issues such as
27 over-interpreting random patterns as model violations. Li et al. (2024a) demonstrated
28 that visual inference methods, particularly those using the lineup protocol (Buja
29 et al. 2009), offer more practical and reliable assessments of residual patterns than
30 conventional tests, as they are less sensitive to minor departures. Packages such as
31 `nullabor` (Wickham et al. 2020), `HLMdiag` (Loy & Hofmann 2014), and `regressinator`
32 (Reinhart 2024) support this approach by enabling users to compare observed residual
33 plots with plots generated under null hypothesis, thereby helping to quantify the
34 significance of any detected patterns.

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

43 To make the statistical testing procedure and trained computer vision model widely
44 accessible, we developed the R package `autovi` along with a companion web interface,
45 `autovi.web`, which allows users to automatically assess their residual plots using the
46 trained computer vision model.

47 The remainder of this paper is structured as follows: Section 2 introduces the definition
48 and computation of visual signal strength. Section 4 provides a detailed documentation
49 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 estimates the distance between the residual distribution of a fitted classical normal
 65 linear regression model and a reference distribution (see [Li et al. 2024b](#), for details).
 66 The distance measure is based on the Kullback-Leibler (KL) divergence:

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

67 where D_{KL} is given by:

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

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

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

75 where $\mu_z = \mathbf{R}\mathbf{Z}\beta_z$, $\mathbf{R} = \mathbf{I}_n - \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$ and \mathbf{X} is the design matrix of the
 76 regression model.

77 The computer vision model approximates this mapping from a set of residuals to
 78 its corresponding distance measure. It is trained on a large number of synthetic
 79 regression models, each designed to simulate specific violations of classical linear
 80 regression assumptions. These models incorporate non-linearity through Hermite
 81 polynomial transformations of predictors, heteroskedasticity by making the error
 82 variance a predictor-dependent function, and non-normality by drawing residuals from
 83 distributions such as discrete, uniform, and lognormal. Both simple and multiple linear
 84 regression structures are used, with controlled parameters to generate diverse and
 85 complex residual patterns. Since the data-generating process is known, the distance
 86 measure D can be explicitly calculated, enabling supervised training. The computer
 87 vision model takes a residual plot as input and outputs the corresponding distance
 88 measure, learning to quantify model violations directly from visual patterns. Additional
 89 details are provided in [Li et al. \(2024b\)](#).

90 3. Definition and simulation of null and bootstrapped residuals

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

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

102 Our computer vision model is trained to assign lower VSS to null plots and higher VSS
 103 to plots that display distinct patterns. Accordingly, statistical testing is performed
 104 by computing the proportion of null plots whose VSS equals or exceeds that of the
 105 observed residual plot. This proportion serves as a p-value for a one-sided hypothesis
 106 test.

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

115 4. R package: autovi

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

123 4.1. Motivation

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

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

140 The third example is generated using the `surreal` package ([Balamuta 2024](#)), where
 141 structure residuals are embedded in the data. In this case, a quote inspired by Tukey,
 142 “visual summaries focus on unexpected values”, is used to define the residual structure.
 143 The automated plot reading predicts the VSS to be 5.87, with a *p*-value less than
 144 0.05. Visually, the structure is strikingly clear, but a RESET test for nonlinear
 145 structure would not report a problem. (It would be detected by a Breusch-Pagan for
 146 heteroscedasticity and also Shapiro-Wilk test ([Shapiro & Wilk 1965](#)) for non-normality.)

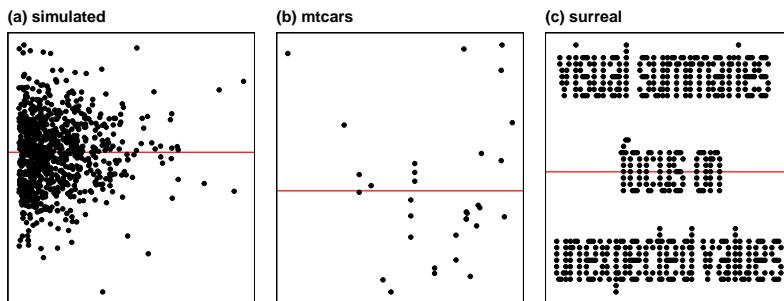


Figure 1. Reading residual plots can be a difficult task, particularly for students new to statistical modeling. The `autovi` package makes it easier. Here are three examples of residual plots, which may appear to have structure. According to `autovi`, the visual signal strengths (VSS) of these three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

147 4.2. Implementation

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

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

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

161 (Mason et al. 2022) computes scagnostics (scatter plot diagnostics), providing numerical
162 features that capture statistical properties of the plots. These scagnostics complement
163 the visual analysis by offering quantitative metrics as secondary input to the
164 computer vision model. `reticulate` (Ushey, Allaire & Tang 2024) enables seamless
165 communication between R and Python.

166 4.3. Installation

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

174 We recommend using the Shiny app `autovi.web` if users encounter installation
175 problems.

176 4.4. Usage

177 4.4.1. Numerical summary

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

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

185 The code to do this is:

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

186 It produces the following summary:

187

```

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

```

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

218 **4.4.2. Visual summary**

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

```
checker$plot_pair()
```

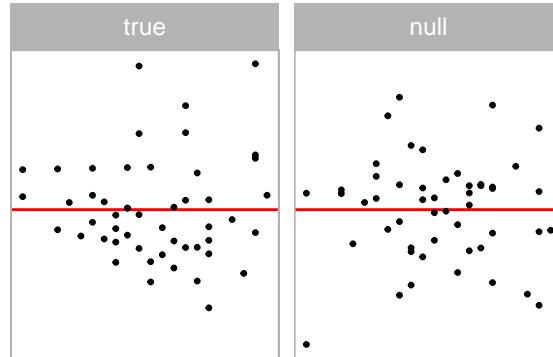


Figure 2. True plot alongside one null plot, for quick comparison.

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

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

```
checker$summary_plot()
```

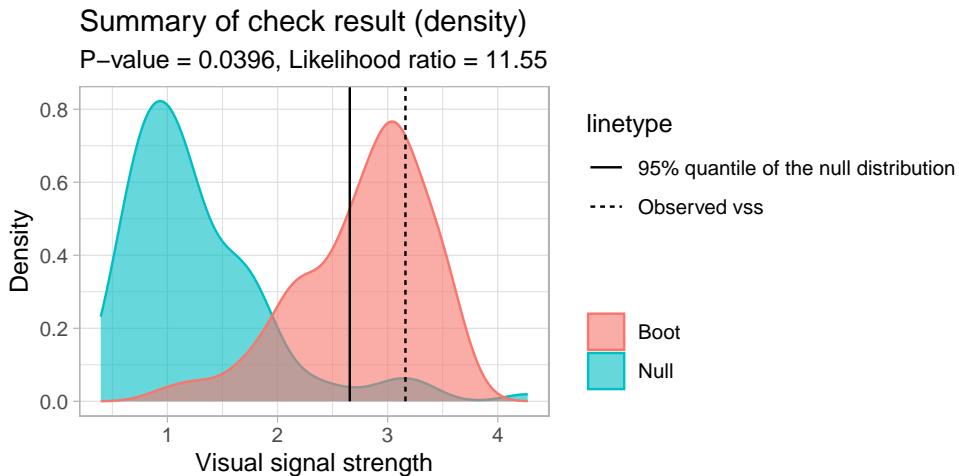


Figure 3. Summary plot comparing the densities of VSS for bootstrapped residual samples (red) relative to VSS for null plots (blue).

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

245 4.5. Modularized infrastructure

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

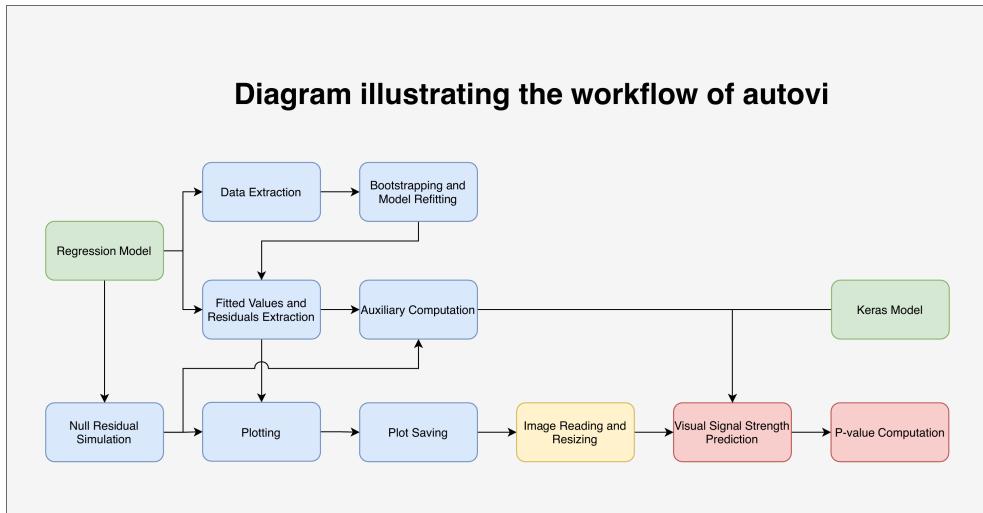


Figure 4. Diagram illustrating the infrastructure of the R package **autovi**. The modules in green are primary inputs provided by users. Modules in blue are overridable methods that can be modified to accommodate users' specific needs. The module in yellow is a pre-defined non-overridable method. The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

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

274 **4.6. Extension to Other Model Classes**

275 The `autovi` R package can be extended to accommodate other classes of models
 276 beyond linear regression, such as generalized linear models (`glm`). This is achieved
 277 by substituting the relevant overridable modules, and if needed, supplying a different
 278 `Keras` model.

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

- 281 1. Define a new class using `new_class()` with `AUTO_VI` as the parent class.
 282 2. Override the necessary methods using `register_method()`. In this example, we
 283 use Pearson residuals. To simulate null residuals, we assume the fitted model
 284 is correct and the estimated coefficients are accurate. New response values are
 285 generated accordingly, and a new model is fitted to this simulated response. Null
 286 residuals are then extracted from this refitted model.
 287 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
```

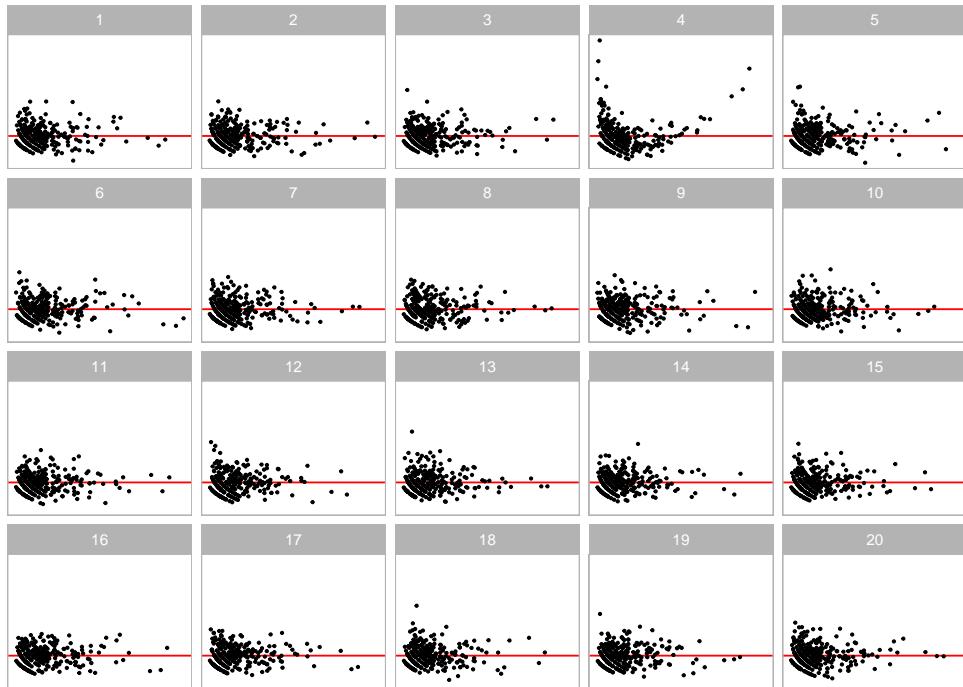
288 The resulting checker class can be employed analogously to the linear model case
 289 described in Section 4.4. For illustration, we fit a Poisson model in which the quadratic
 290 term of the predictor x is intentionally omitted. This misspecification manifests as a
 291 pronounced U-shaped pattern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

294 It is important to note, however, that the pre-trained computer vision model included
 295 in `autovi`, such as `vss_phn_32` (see `list_keras_models()` for the full list of available
 296 models), was developed specifically for diagnostics 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. In such cases, users are encouraged to train and apply their own
 304 `Keras` models. Detailed guidance on model training and discussion on extending the
 305 methodology to other model classes can be found in 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.



306

```
pois_checker$check()
```

307

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

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

324 - Quantiles:

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

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

330 5. Web interface: `autovi.web`

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

339 5.1. Implementation

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

346 Determining the best way to implement the backend was difficult. In our initial
 347 planning for `autovi.web`, we considered implementing the entire web application using
 348 the `webr` framework (Moon 2020), which would have allowed the entire application to
 349 run directly in the user's browser. However, `webr` does not support packages which use
 350 compiled fortran code, which is required by `splancs` (Rowlingson & Diggle 2023), a
 351 dependency of `autovi`. In the future, it is possible that a working Emscripten (Zakai
 352 2011) version of this package may allow full `webr` support.

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

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

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

369 Due to [shinyapps.io](#)'s resource policy, inactive servers enter sleep mode, clearing the
370 local Python environment. When reactivated for a new session, libraries must be
371 reinstalled. While this ensures a clean environment for each session, it may lead to
372 slightly longer loading times for the first user after a period of inactivity.

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

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

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

389 **5.2. Usage**

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

397 **5.2.1. Reading data and setting parameters**

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

414 **5.2.2. Results provided**

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

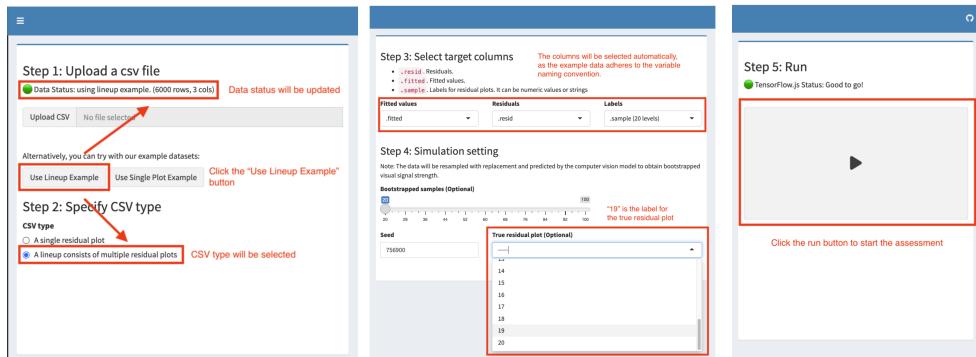


Figure 5. To begin the workflow for `autoovi` using the lineup example dataset, the user clicks the “Use Lineup Example” button (left) to load the example dataset, during which the data status and CSV type will be automatically updated. The user must manually select the label for the true residual plot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

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

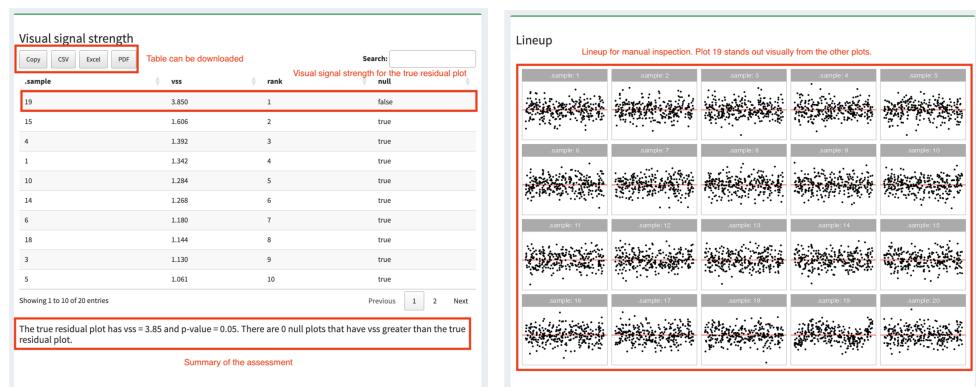


Figure 6. Results for the lineup. The VSS of the true residual plot is displayed in the first row of the table of VSS values for all the null plots (left image), with a summary text beneath the table providing the *p*-value to aid in decision-making. A lineup of residual plots allows for manual inspection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

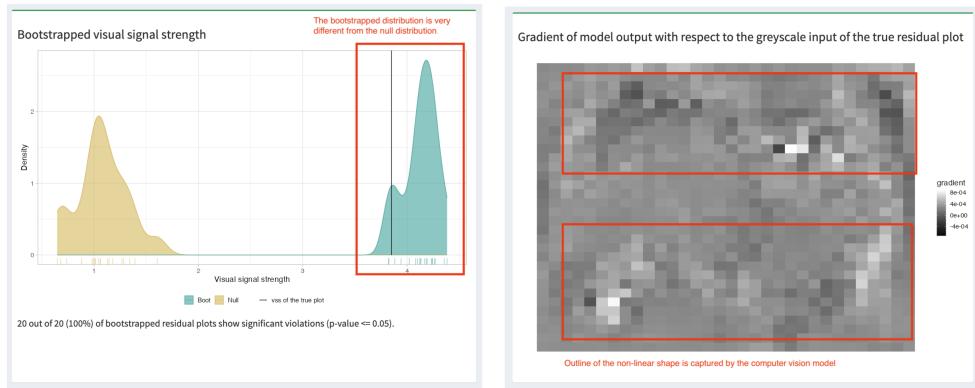


Figure 7. Summaries assessing the strength of the pattern and which elements of the plot contribute. The density plot helps verify if the bootstrapped distribution differs from the null distribution (left image). The attention map (right image) offers insights into whether the computer vision model has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

428

6. Conclusions

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

434 The **autovi** R package, introduced in this paper, automates the assessment of residual
 435 plots by incorporating a computer vision model, reducing reliance on time-consuming
 436 and potentially inconsistent human interpretation. This automation improves the
 437 efficiency of the diagnostic process and promotes consistency in model evaluation
 438 across different users and studies.

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

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

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

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

457 7. Resources and supplementary material

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

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

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

470 References

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

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

- 531 package version 2.0.0.
- 532 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.
- 533 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.
- 534 NOWOSAD, J. (2018). 'CARTOCOLORs' palettes. URL <https://nowosad.github.io/rkartocolor>. R package version 1.0.
- 535 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.
- 536 PEDERSEN, T.L. (2022). *patchwork: The composer of plots*. URL <https://CRAN.R-project.org/package=patchwork>. R package version 1.1.2.
- 537 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/>.
- 538 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.
- 539 REINHART, A. (2024). *regressinator: Simulate and Diagnose (Generalized) Linear Models*. URL <https://CRAN.R-project.org/package=regressinator>. R package version 0.2.0.
- 540 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.
- 541 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.
- 542 SHAPIRO, S.S. & WILK, M.B. (1965). An analysis of variance test for normality (complete samples). *Biometrika* **52**, 591–611.
- 543 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.
- 544 WARTON, D.I. (2023). Global simulation envelopes for diagnostic plots in regression models. *The American Statistician* **77**, 425–431.
- 545 WICKHAM, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. URL <https://ggplot2.tidyverse.org>.
- 546 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.
- 547 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.
- 548 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.
- 549 ZEILEIS, A. & HOTHORN, T. (2002). Diagnostic checking in regression relationships. *R News* **2**, 7–10.
- 550 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.