Journal of Statistical Software
March 2025, Volume 112, Issue 3. *doi: 10.18637/jss.v112.i03*

RESI: An R Package for Robust Effect Sizes

Megan Jones
Vanderbilt University

Kaidi Kang
Vanderbilt University

Simon Vandekar
Vanderbilt University

Abstract

Effect size indices are useful parameters that quantify the strength of association and are unaffected by sample size. There are many available effect size parameters and estimators, but it is difficult to compare effect sizes across studies as most are defined for a specific type of population parameter. We recently introduced a new robust effect size index (RESI) and confidence interval, which is advantageous because it is not model-specific. Here we present the **RESI** R package, which makes it easy to report the RESI and its confidence interval for many different model classes, with a consistent interpretation across parameters and model types. The package produces coefficient, ANOVA tables, and overall Wald tests for model inputs, appending the RESI estimate and confidence interval to each. The package also includes functions for visualization and conversions to and from other effect size measures. For illustration, we analyze and interpret three datasets using different model types.

Keywords: R, effect size, confidence intervals, CRAN, bootstrap.

1. Introduction

Standardized effect sizes are unitless indices used to describe the magnitude of an association. While unstandardized effect sizes can be informative in a given scientific context, standardized measures have the benefit of allowing communication of associations for outcomes measured without an interpretable scale and facilitating comparison across settings where outcomes are measured using different instruments. Unlike p values, which are often used to evaluate statistical significance, effect sizes do not depend on sample size (Betensky 2019). A well known criticism of p values and significance testing is that for large sample sizes, very small effects will be found as significant, even though these effects may be negligible in real-world application, as noted in Principle 5 of the American Statistical Association (ASA) statement on statistical significance and p values (Wasserstein and Lazar 2016). In contrast, effect sizes communicate the strength of the effect rather than the existence of an effect of arbitrary size, which may be more meaningful in practice (Sullivan and Feinn 2012). Although increased sample size helps improve the precision of the estimate of an effect size, the effect size is a parameter that is not dependent on sample size (Kang, Jones, Armstrong, Avery, McHugo, Heckers, and Vandekar 2023). Standardized effect sizes are also important statistical parameters for power analysis, as power is a function of the effect size, sample size, and degrees of freedom of the statistical test. Journals and statistical guidelines are increasingly encouraging authors to report effect sizes, either unstandardized or standardized, and their confidence intervals (CIs) alongside or in place of p values (Wasserstein and Lazar 2016; Wilkinson 1999; American Psychological Association 1994, 2001, 2010, 2020; Althouse, Below, Claggett, Cox, de Lemos, Deo, Duval, Hachamovitch, Kaul, Keith, Secemsky, Teixeira-Pinto, and Roger 2021). However, they are still not commonly reported (Fritz, Morris, and Richler 2012; Amaral and Line 2021) and when reported, they often do not include confidence intervals (Fritz *et al.* 2012).

There are four challenges to reporting effect sizes that limit their widespread use. First, there are many different effect size measures available (Cohen 1988; Hedges and Olkin 1985; Rosenthal 1994; Zhang and Schoeps 1997; Serdar, Cihan, Yücel, and Serdar 2021), but they are typically defined in the context of a specific population parameter, which makes comparing effects across a wide range of models difficult (Vandekar, Tao, and Blume 2020). Second, many available effect size measures do not allow for nuisance parameters or covariates (Vandekar *et al.* 2020). Third, many effect size measures do not have accurate confidence interval procedures, which precludes quantification of the uncertainty around the effect size estimate (Kang *et al.* 2023). Finally, many default model summary functions available in statistical software automatically output p values, but few also report effect sizes with confidence intervals. The **RESI** package (Jones, Kang, and Vandekar 2025) for R (R Core Team 2024) was designed to address these challenges by implementing a recently proposed effect size measure.

Methods for many common effect sizes are available in most major statistical software. In the SAS® software (SAS Institute Inc. 2020), the `GLM` procedure allows the user to compute three effect size measures with confidence intervals for linear models: the noncentrality parameter of the F statistic, the squared semipartial correlation, and the squared full partial correlation. There is also a macro (`effect_size`) for SAS software available for calculating Cohen's d from survey data following one of three designs (Kadel and Kip 2012). SPSS includes functionality to compute Cohen's d with confidence intervals for t tests, Pearson correlation, partial η^2, ω^2, and R^2 (IBM Corporation 2011). In Stata (StataCorp 2023), estimates and confidence intervals for effect size measures such as Cohen's d, Hedges's g, Glass's Δ and point-biserial correlation can be obtained using the `esize` function on raw data or the `esizei` function on summary statistics. The `estat esize` command can be used following an ANOVA model or linear regression model to compute η^2 for model variables. Confidence intervals for effect sizes based on bootstrapping are also available in Stata.

There are several R packages available for effect size calculation. For example, packages such as **MOTE** (Buchanan, Gillenwaters, Scofield, and Valentine 2019), **MBESS** (Kelley 2007, 2023), **effsize** (Torchiano 2020), **esvis** (Anderson 2020), **lsr** (Navarro 2021), **esc** (Lüdecke 2019), and **rcompanion** (Mangiafico 2024) include functionality that allows the user to manually input data or the relevant test statistics for conversion to a variety of desired effect size measures. Packages **MOTE**, **effsize** and **esc** compute confidence intervals for effect sizes using noncentral or central distributions. Package **rcompanion** utilizes bootstrapping for effect size confidence intervals, and **MBESS** implements both noncentral and bootstrapped confidence intervals. The **effectsize** package implements many effect size measures and conversions between some of them (Ben-Shachar, Lüdecke, and Makowski 2020). Package **effectsize** allows users to input test statistics, but also conveniently accepts fitted models directly to compute the desired effect size. This package computes confidence intervals in a variety of methods depending on the effect size measure, but several functions use a noncentral distribution and a few use bootstrapping. The **emmeans** package also allows post-model-fitting effect size (Cohen's d) estimation for contrasts of estimated marginal means in `emmGrid` objects and computes parametric confidence intervals (Lenth 2016, 2024). Another package, **bootES**, focuses specifically on providing bootstrapped confidence intervals for unstandardized effect sizes, Cohen's d, Pearson correlation, and Hedges's g (Gerlanc and Kirby 2023). Each of these tools can be helpful for computing effect sizes in a given data context; however, they do not fully address the general challenges to reporting and comparing effect sizes mentioned above. In particular, many confidence interval methods for effect sizes rely on chi-square, F, or t statistic implementations, which have below nominal coverage rate (Kang *et al.* 2023). There is a need for an effect size index that can be broadly applied and compared across model types and user-friendly software tools that implement such a measure to promote easy reporting of effect sizes.

The recently proposed robust effect size index (RESI) (Vandekar *et al.* 2020; Kang *et al.* 2023) addresses many of these challenges because it is broadly applicable across all common model types, it accommodates nuisance parameters, and there is an effective confidence interval procedure available (Kang *et al.* 2023). The RESI can be estimated from chi-square, F, Z, and t statistics. It is also possible to convert RESI estimates to and from other common effect size measures, such as Cohen's d, Cohen's f^2, and R^2 (Vandekar *et al.* 2020).

The **RESI** R package builds on existing infrastructure for robust standard error estimation (Zeileis 2006) and bootstrapping (Canty and Ripley 2024) allowing easy estimation, reporting, and visualization and is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=RESI`. Similarly to the **effectsize** package, **RESI** is designed to work on model inputs, so that effect size estimates can be easily obtained in tandem with common model summaries. These model-based functions also allow for a large amount of customization in the estimation and reporting process. Directly inputting test statistics and the relevant degrees of freedom and sample size is an option as well, helpful for model types that have not yet been implemented via dedicated methods in the package. The package also aims to work with other effect size measures, providing functions to convert to and from a few common effect size indices. Plotting functions are provided to allow for quick visualization of the effect size estimates present in models. With these tools, we hope to make obtaining the highly generalizable RESI simple and accessible, in order to increase ease of reporting effect sizes in research.

In this paper, we outline the theory underlying the RESI, its estimators, and confidence interval procedure in Section 2. We then discuss the **RESI** package, its structure, function arguments, and dependencies in Section 3. Finally, Section 4 provides three in-depth examples using the **RESI** package to perform analysis of effect sizes, from model creation to post-estimation visualization while Section 5 concludes the paper.

2. Statistical methods

2.1. RESI definition

The RESI is defined from the noncentrality parameter of a test statistic in the context of M-estimation, so it is broadly applicable across statistical models and parameters. A full introduction to the RESI can be found in our previous work (Vandekar *et al.* 2020; Kang *et al.* 2023).

Briefly, consider a dataset of independent observations $W = \{W_1, \dots, W_n\}$ with probability distribution $\mathcal{P}$ and let $\theta = (\alpha, \beta) \in \mathbb{R}^m$ be a vector of parameters with $\alpha \in \mathbb{R}^{m_0}$ nuisance parameters and $\beta \in \mathbb{R}^{m_1}$ the target parameters of interest. The RESI is constructed using the test statistic for the null hypothesis $H_0 : \beta = \beta_0 \in \mathbb{R}^{m_1}$, where β_0 is a reference value, usually zero (Vandekar and Stephens 2021). Assuming known variance, the usual Wald-style test statistic, centered at the reference value, $T^2 = n(\hat{\beta} - \beta_0)^\top \Sigma_\beta^{-1}(\hat{\theta})(\hat{\beta} - \beta_0)$ follows a chi-square distribution with m_1 degrees of freedom and noncentrality parameter $n(\beta - \beta_0)^\top \Sigma_\beta^{-1}(\beta - \beta_0)$. The RESI, S_β, is the square root of the component of the noncentrality parameter that does not depend on the sample size

$$S_\beta = \sqrt{(\beta - \beta_0)^\top \Sigma_\beta^{-1}(\beta - \beta_0)}.$$

2.2. RESI estimators

The RESI is very general, because its estimator can be computed for chi-square, F, Z, and t statistics (Vandekar *et al.* 2020; Kang *et al.* 2023). In this section, we review these estimators and introduce new estimators for a modified RESI using Z and t statistics, which have the advantage that the proposed modification shows the direction of the effect for univariate parameters. We also describe the use of robust covariance in the estimation of the test statistics.

The original estimator for S_β was developed using an estimator for noncentrality parameters of chi-square statistics (Vandekar *et al.* 2020)

$$\hat{S}_\beta = \left\{ \max\left[0, \frac{T^2 - m_1}{n}\right] \right\}^{\frac{1}{2}}. \tag{1}$$

Because S_β is nonnegative, the max operator ensures that the estimator is also nonnegative in finite samples.

Under normality, the finite sample distribution of the asymptotic chi-square statistic divided by its degrees of freedom is an F distribution (Mantel 1963). When this is true, a better small sample estimator can be computed using method of moments with the F distribution

$$\hat{S}_\beta = \left\{ \max\left[0, \frac{F \times (n - m - 2) - m_1 \times (n - m)}{n \times (n - m)}\right] \right\}^{\frac{1}{2}}. \tag{2}$$

The RESI is called robust because its estimator is consistent under misspecification of the variance model when estimated with a robust test statistic (Vandekar *et al.* 2020; MacKinnon and White 1985), which uses a heteroskedastic consistent sandwich estimator for Σ_β (White

1980; MacKinnon and White 1985; Long and Ervin 2000). The RESI estimator is a consistent estimator of the true effect size in contexts where the robust variance estimator yields consistent results under aspects of model misspecification, such as unknown heteroskedasticity between measurements in general linear models or a misspecified correlation structure in GEE models. When the mean model is misspecified, the RESI is a consistent estimator of the best approximation of the true model within the class of models considered (Boos and Stefanski 2013).

With Equation 1 and Equation 2, we can compute RESI estimates for chi-square and F statistics, which are easily obtained from many statistical models. However, these estimates have the feature of being nonnegative, so they do not describe the direction of an effect. While this makes them generally applicable across univariate (can be negative and positive) and multivariate (can only be positive) parameters, for univariate parameters it is also useful to be able to obtain a signed effect size estimate, showing the directionality of the effect. With this in mind, we introduce RESI estimators for Z and t statistics. We use two approaches to develop these estimators, leading to two estimators with different properties and advantages.

The first approach is the same as the development of the RESI estimators for chi-square and F statistic. We use the method of moments for the Z or t statistics to find estimators for S_β. Consider a Z statistic, whose expected value is $\mathsf{E}Z = \sqrt{n}\text{sgn}(\beta)S_\beta$, where sgn is the sign function, which leads to the signed RESI estimator

$$\hat{S}_\beta = \frac{Z}{\sqrt{n}} \tag{3}$$

For a t statistic with degrees of freedom $n - m$ and noncentrality parameter $\sqrt{n}S_\beta$, when $n-m > 1$, the expected value is $\mathsf{E}t = \sqrt{\frac{n(n-m)}{2}}\frac{\Gamma((n-m-1)/2)}{\Gamma((n-m)/2)}S_\beta$. This gives the RESI estimator

$$\hat{S}_\beta = \frac{t\sqrt{2}\Gamma((n-m)/2)}{\sqrt{n(n-m)}\Gamma((n-m-1)/2)} \tag{4}$$

The advantage of estimators in Equation 3 and Equation 4 is that both are unbiased for S.

The second approach leverages the relationship between Z and chi-square statistics and t and F statistics. Squaring a Z or t statistic gives a chi-square or F statistic, respectively. We then use Equation 1 and Equation 2 as RESI estimators for Z and t statistics by multiplying them with the sign of the test statistic. For example,

$$\hat{S}_\beta = \text{sgn}(Z)\times\left\{\max\left[0, \frac{Z^2-1}{n}\right]\right\}^{\frac{1}{2}}, \tag{5}$$

and similarly for the t estimator. These estimators are biased, but consistent and have smaller mean squared error than the estimators in Equation 3 and Equation 4. These estimators are advantageous because their estimates are equal in absolute value to the unsigned RESI estimates, whereas the estimators in Equation 3 and Equation 4 are not.

Note that the RESI estimators are based on the Wald test statistics, which are dependent on the specific modeling decisions made when fitting the data. The RESI is linear on the scale of the linear predictor (e.g., the RESI for a logistic model is linear on the log-odds scale).

Cohen's d	RESI	"Rule of thumb" interpretation
[0, 0.2]	[0, 0.1]	No effect – small
(0.2, 0.5]	(0.1, 0.25]	Small – medium
(0.5, 0.8]	(0.25, 0.4]	Medium – large
> 0.8	> 0.4	Large

Table 1: Guidelines for interpreting size of (absolute) RESI estimates based on analogous suggestions from (Cohen 1988). Note these ranges are a rule of thumb and effect sizes should always be interpreted within the scientific context.

2.3. Bootstrapping procedure for confidence intervals

In recent work, we showed that chi-square and F confidence intervals are not accurate for computing effect size confidence intervals in general. In particular, when the test statistic is estimated using a robust covariance estimator, or when the study design is observational, using a chi-square or F distribution for the RESI estimate underestimates the variance and will therefore produce confidence intervals that provide less than nominal coverage level (Kang *et al.* 2023). These chi-square and F confidence intervals with below nominal coverage are those that are implemented in most other software packages (SAS Institute Inc. 2016; Buchanan *et al.* 2019; Torchiano 2020; Lüdecke 2019; Ben-Shachar *et al.* 2020; Kelley 2023). As an alternative, we proposed a nonparametric bootstrap for the RESI confidence interval because it produces confidence intervals with nominal coverage most consistently (Kang *et al.* 2023). This is the procedure implemented in the **RESI** package. For linear models and nonlinear least squares models, a Bayesian bootstrap is also available as an option (Rubin 1981).

2.4. Meaningful RESI ranges

When interpreting the RESI estimates, it is useful to have an idea of what constitutes a "large" or "small" effect. While a meaningful effect size (standardized or unstandardized) ultimately depends on the scientific context, ranges can be posited based on recommended effect size ranges for Cohen's d assuming equal sample proportions of the two groups and equal variance (Cohen 1988, Table 1). These are guidelines based on a difference in means in behavioral sciences and the size of a meaningful effect varies by field; effort should be made to interpret estimates within the given scientific context.

3. The RESI package

RESI is available to the public on CRAN under the GPL-3 license. To download, one can use the following code:

```
R> install.packages("RESI")
```

The development version is available on GitHub at `https://github.com/statimagcoll/RESI`. This can be downloaded using the **devtools** package (Wickham, Hester, Chang, and Bryan 2022) with the following command:

```
R> devtools::install_github("statimagcoll/RESI")
```

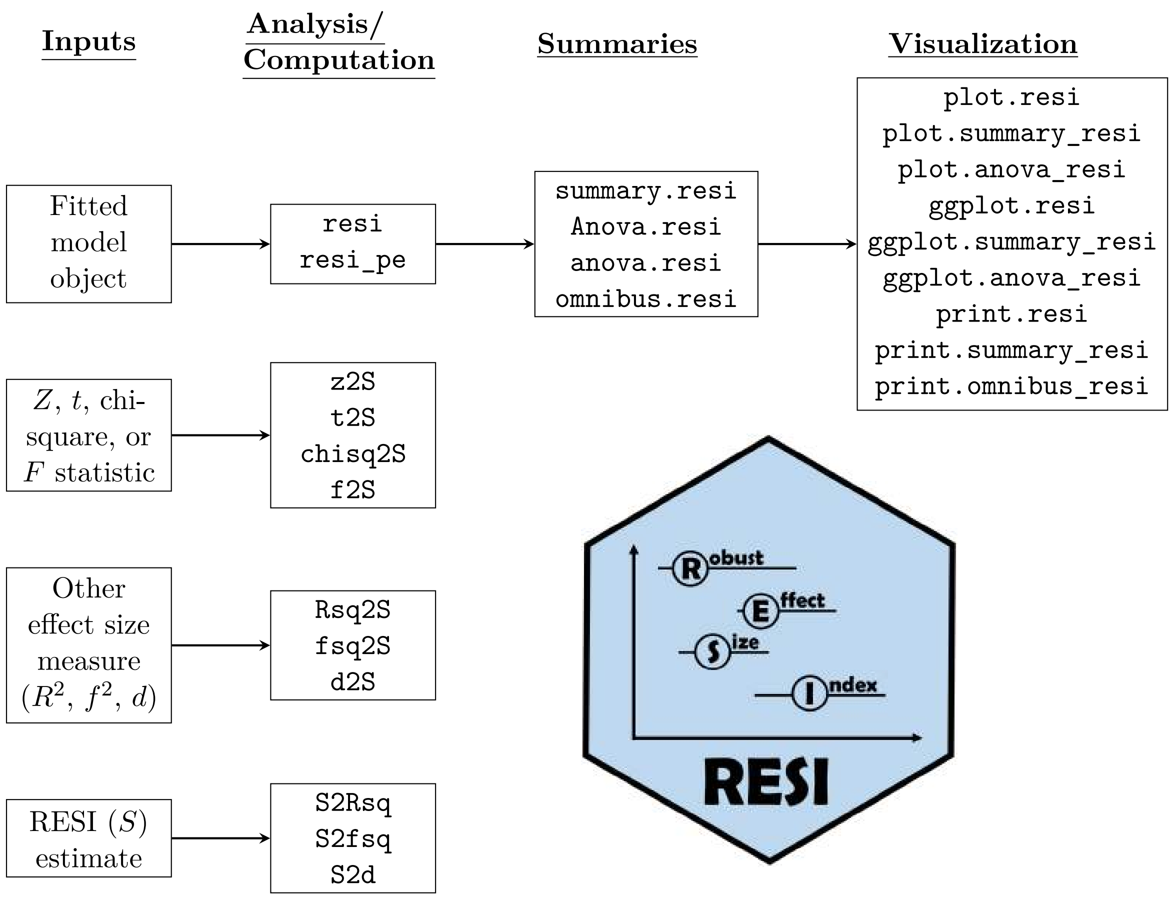

Figure 1: **RESI** package structure and logo. Inputs to package functions can be models of supported types, test statistics with relevant degrees of freedom and sample size, or effect size measures. The analysis functions compute RESI estimates with or without confidence intervals, or convert to and from other effect size indices. Summary functions provide relevant information extracted from a '`resi`' object. Post-estimation visualization functions include plotting and printing.

3.1. Operation

Users should have R version 2.10 or higher to use **RESI**. The **RESI** package is designed to easily add RESI estimates and confidence intervals to common model outputs, such as coefficient summaries and ANOVA tables. The functions in the package are split into three categories: model-based functions, conversion functions, and additional methods to other functions (Figure 1). There are also two datasets provided.

3.2. Model-based functions

The main model-based RESI estimation functions of the **RESI** package are `resi_pe()`, to obtain point estimates, and `resi()` for point estimates with confidence or credible intervals. `resi_pe()` uses standard summary and ANOVA outputs to compute the RESI point estimate. Function `resi()` uses `resi_pe()` and performs bootstrapping via the **boot** package (Canty and Ripley 2024) to produce confidence intervals for the RESI. These functions take supported fitted models as input and return a list that contains three main components: a coefficients summary table with a row for each non-reference level of each variable, an ANOVA table containing a row for each variable, and an overall RESI estimate. Details regarding functions used for table construction for the supported model types are given in Table 2. For model classes without dedicated methods, `resi()`/`resi_pe()` attempts to implement the default method and return informative messages in the case of failure.

While the user can simply run `resi()` on a supported model type and obtain a full output, there are several arguments that can be used to tailor the process. Details for all function arguments are available in the documentation, but we briefly cover important arguments here. `resi()` and `resi_pe()` both contain the following arguments. The `model.full` argument is the model to perform RESI estimation on. The `model.reduced` argument, `NULL` by default, specifies a reduced model which is used to compute an effect size estimate in comparison to the full model for a specific subset of variables that the user wishes to compare (see Section 4.3). If left as `NULL`, `resi_pe()` will compute a reduced model of the same type as the full model, but including only the intercept term. `data` is a blank argument referring to the data used to generate the model. If left blank, `resi()` pulls the data from the model. For some model types (‘`survreg`’, ‘`coxph`’, ‘`nls`’), the data is required as an input because these models objects do not store the original data frame used to fit the model. Additionally, when using some formula functions such as splines or factoring, the data needs to be input so that the spline arguments can be recomputed as they were in the original data.

The `vcovfunc` argument can be used to specify a different variance-covariance function and is important because it affects whether the effect size is robust to model misspecification (see Section 2.2). By default, **RESI** will use a robust covariance estimator. Additional arguments to the given `vcovfunc` function can be specified in list form with the `vcov.args` argument. Similarly, additional arguments to the `Anova()` function (from package **car** Fox and Weisberg 2019) can be specified with the `Anova.args` argument. The `unbiased` argument is logical (default `TRUE`) and corresponds to a choice of conversion formulas for the Z and t statistics (see Section 2.2 for details).

Function `resi()` contains additional arguments related to the bootstrap procedure. The confidence level (default 0.05) for the confidence or credible intervals can be specified with `alpha`. Multiple confidence levels can be specified using a numeric vector. For ‘`lm`’ and ‘`nls`’ models, there is a `boot.method` argument that can be specified as nonparametric (default) or Bayesian (see Section 2.3). Bootstrapping is implemented via the `boot()` function from the **boot** package (Canty and Ripley 2024), so `resi()` accepts additional arguments corresponding to that function such as `parallel` and `ncpus` to allow for increased efficiency. Finally, the `store.boot` argument (default `FALSE`) determines whether to store the complete ‘`boot`’ object as an element of the output. The `store.boot` option must be set to `TRUE` if the user wants to be able to obtain confidence intervals with different confidence levels without rerunning the bootstrap procedure.

The output of `resi()` is a list of class ‘`resi`’ that contains the three main tables (coefficients, ANOVA, and overall) with confidence intervals and several other elements to track how the functions were called. Function `resi_pe()` produces a list with these tables (without confidence intervals) and other elements about the model. Each of these elements is optionally computed and can be suppressed with an argument to `resi()`/`resi_pe()` (e.g., `anova = FALSE`).

The `overall` element of the output is a table reporting a Wald test comparing the full model to the reduced model. The test statistic is typically converted from a chi-square statistic to a RESI estimate internally using the `chisq2S()` function, which takes the number of observations from the data and degrees of freedom from the Wald test. In the case of a linear model, the RESI estimate is computed using the `f2S()` function.

The `coefficients` table is available for every model type supported by the package. This provides a RESI estimate for each model coefficient and appends it to a table resulting from one of the "Coefficients" functions in Table 2. The Z or t statistic from this function is converted to the signed RESI via `z2S()`/`t2S()`, with the logical `unbiased` argument determining if the unbiased estimator in Equation 3 and Equation 4 or the alternate version in Equation 5 is used.

The `anova` table is computed via `Anova()` from **car** (Fox and Weisberg 2019) where available (for '`geeglm`' models, `anova` is used). The `anova` argument (default = `TRUE`) in `resi()`/`resi_pe()` determines whether to compute this table. For '`lm`' models, an F-test is used. For the others, a Wald test is specified assuming chi-square statistics. Other options can be passed to `Anova()` function via `Anova.args`. Note that the `test.statistic` argument is fixed in the `resi_pe()` function, so supplying a different value for this argument will result in an error. Additionally, if the user wishes to use a different `vcov.` argument in `Anova()` function, this should be done by providing the function to the `vcovfunc` argument in `resi()` (see Section 4.1). Specifying this argument in `Anova.args` will result in an error. The resulting chi-square or F statistics are converted to RESI estimates using `chisq2S()` or `f2S()`.

The RESI for longitudinal models is still in development. Currently, the package provides point estimate and confidence interval methods for '`gee`' (from **gee** package Carey 2024) and '`geeglm`' (from **geepack** Halekoh, Højsgaard, and Yan 2006) models. For these models, both a longitudinal RESI and a per-measurement cross-sectional RESI estimate are computed for each factor in the `coefficients` table (for '`gee`' and '`geeglm`') and for each variable in the `anova` table (for '`geeglm`'). The longitudinal RESI is the estimated effect conditional on the sampling design, whereas the cross-sectional estimator is the effect if the data were collected cross-sectionally. This allows investigators to quantify the benefit conferred by considering a longitudinal design. For linear mixed effects models fit via `lme()` from package **nlme** (Pinheiro, Bates, and R Core Team 2024) and `lmerMod()` from **lme4** (Bates, Mächler, Bolker, and Walker 2015), longitudinal RESI point estimation is available in both a `coefficients` and `anova` table. The confidence interval procedure is still being evaluated for these models, so running `resi()` on a model of this type will provide point estimates only with a corresponding message.

3.3. Other package elements

The package includes `print()`, `plot()`, `ggplot()` `summary()`, `anova()`, and `car::Anova()` methods for '`resi`' objects. The `summary()` and `anova()`/`Anova()` methods are intended to isolate the corresponding elements of the '`resi`' object and allow the user to specify a different confidence level without having to rerun the bootstrapping process, if the `store.boot` option was set to `TRUE` when running `resi()`. Running `summary()` on a '`resi`' object returns the `coefficients` table as an object of class '`summary_resi`', with its own `plot()`/`ggplot()` and `print()` methods. Running `anova()` or `car::Anova()` on a '`resi`' object returns the `anova` table as an object of class '`anova_resi`' and inherited classes from `anova()`/`car::Anova()`.

Model	Package	Covariance	Coefficients	Anova	Overall
‘lm’	**stats**	`sandwich::vcovHC`	`coeftest`	`car::Anova`	`waldtest`
‘glm’	**stats**	`sandwich::vcovHC`	`coeftest`	`car::Anova`	`waldtest`
‘nls’	**stats**	`regtools::nlshc`	`coeftest`	N/A	`wald.test`
‘survreg’	**survival**	`vcov`	`coeftest`	`car::Anova`	`waldtest`
‘coxph’	**survival**	`vcov`	`coeftest`	`car::Anova`	`wald.test`
‘hurdle’	**pscl**	`sandwich::sandwich`	`coeftest`	N/A	`waldtest`
‘zeroinfl’	**pscl**	`sandwich::sandwich`	`coeftest`	N/A	`waldtest`
‘gee’	**gee**	`summary`	`summary`	N/A	N/A
‘geeglm’	**geepack**	`vcov`	`coeftest`	`anova`	`anova`
‘lme’	**nlme**	`clubSandwich::vcovCR`	`summary`	`car::Anova`	N/A
‘lmerMod’	**lme4**	`clubSandwich::vcovCR`	`summary`	`car::Anova`	N/A

Table 2: Supported model types and related functions. `coeftest` and `waldtest` are from package **lmtest** (Zeileis and Hothorn 2002). `wald.test` is from **aod** (Lesnoff and Lancelot 2023).

There are also `plot()`/`ggplot()` (Wickham 2016) methods for ‘`anova_resi`’.

The package also contains a few conversion functions from RESI to and from other common effect size measures. These are Cohen’s d, Cohen’s f^2, and R^2. Formulas for these conversions are found in Vandekar *et al.* (2020).

Lastly, the **RESI** package contains two datasets. The `insurance` dataset is adapted from the open-source repository Kaggle US Health Insurance Dataset (`https://www.kaggle.com/datasets/teertha/ushealthinsurancedataset/discussion`) and the `depression` dataset is adapted from a data analysis textbook (Agresti 2002). Full details on the datasets are provided in the **RESI** package documentation.

3.4. Important dependencies

The **RESI** package currently has dedicated methods for 11 model types (Table 2). The software function used to compute the covariance matrix varies by model type. It is possible to pass additional arguments to these covariance functions in `resi()` by using the `vcov.args` argument. Any other valid covariance function can be specified as well. Although robust covariance estimators are used as the default for most model types, the survival models (‘`survreg`’ and ‘`coxph`’) have the option for a robust covariance estimate in model setup and, when using the standard `vcov` from **stats**, they compute robust covariance matrices if the argument `robust = TRUE`. For ‘`geeglm`’ models, the robust covariance is taken from the model directly (Zeileis 2006).

Several other common analysis functions are used to obtain test statistics for RESI computation for the coefficients, ANOVA, and overall table. The functions used for different model types are found in Table 2.

3.5. Support

Users encountering problems with the package can reach out for help using the GitHub Issues page (`https://github.com/statimagcoll/RESI/issues`). The Discussions page (`https:`

//github.com/statimagcoll/RESI/discussions) can be used to seek additional support or suggest new features to be added to the package. Planned features are listed in the "Coming Soon" section of our **pkgdown** (Wickham, Hesselberth, and Salmon 2024) website at https://statimagcoll.github.io/RESI/. We ask those wishing to contribute to our software to create a new branch on our GitHub and submit a pull request describing the contribution.

4. Illustrations

To demonstrate the flexibility of the **RESI** package, we analyze a few example datasets for several different model types using different covariance estimator functions and bootstrapping options.

4.1. RESI on linear model

We first look at a linear model fit using `lm()`. After installing the package from CRAN or GitHub, we load the **RESI** library.

```
R> library("RESI")
```

We will use the `insurance` dataset in the package to fit our model. The dataset contains information on insurance charges, age, sex, body mass index (BMI), number of children, smoking status, and geographical region for 1338 individuals in the United States. We fit a linear regression of charges against region, age, BMI, and sex, with an interaction term on region and age and return the standard coefficients table using the `summary()` function.

```
R> data("insurance", package = "RESI")
R> mod_lm <- lm(charges ~ region * age + sex + bmi, data = insurance)
R> summary(mod_lm)

Call:
lm(formula = charges ~ region * age + sex + bmi, data = insurance)

Residuals:
   Min     1Q Median     3Q    Max
-14871  -7062  -4885   6235  46347

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         -5359.44    2369.09  -2.262   0.0238 *
regionnorthwest     -2339.44    2647.85  -0.884   0.3771
regionsoutheast     -3230.85    2583.12  -1.251   0.2112
regionsouthwest      -232.48    2662.84  -0.087   0.9304
age                   220.33      45.08   4.888 1.14e-06 ***
sexmale              1328.02     622.07   2.135   0.0330 *
bmi                   323.77      53.72   6.027 2.17e-09 ***
regionnorthwest:age    34.90      63.55   0.549   0.5829
```

```
regionsoutheast:age    83.64      61.65   1.357   0.1751
regionsouthwest:age   -33.63      63.74  -0.528   0.5979
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11360 on 1328 degrees of freedom
Multiple R-squared:  0.126,        Adjusted R-squared:  0.1201
F-statistic: 21.27 on 9 and 1328 DF,  p-value: < 2.2e-16
```

The p values in the standard model summary indicate that age, sex, and BMI are significantly associated with insurance charges. However, just by looking at the p values, it is hard to discern the strength of the the association. We would like to be able to see, in addition to significance, a measure of the effect size. To accomplish this, we can run `resi()` on the model object. We run it using all the default options first. This will use the `vcovHC()` function from the **sandwich** package (with default arguments) to compute robust standard error estimates (Zeileis, Köll, and Graham 2020). Since we are using the `resi()` function rather than the `resi_pe()` function, we will obtain bootstrapped confidence intervals in addition to RESI point estimates. We set the seed to ensure the results are reproducible. This function can take several seconds to run. Printing the full '`resi`' object will print several tables and notes, so to begin we just print the summary.

```
R> set.seed(0827)
R> resi_obj_lm <- resi(mod_lm)
R> summary(resi_obj_lm)

Analysis of effect sizes based on RESI:
Confidence level =  0.05
Call:  lm(formula = charges ~ region * age + sex + bmi, data = insurance)

Coefficient Table
                    Estimate Std. Error t value Pr(>|t|)   RESI   2.5%
(Intercept)         -5359.44    2175.94  -2.463    0.014 -0.067 -0.119
regionnorthwest     -2339.44    2395.15  -0.977    0.329 -0.027 -0.084
regionsoutheast     -3230.85    2643.11  -1.222    0.222 -0.033 -0.087
regionsouthwest      -232.48    2574.28  -0.090    0.928 -0.003 -0.060
age                   220.33      40.21   5.480    0.000  0.150  0.094
sexmale              1328.02     621.74   2.136    0.033  0.058  0.004
bmi                   323.77      58.08   5.574    0.000  0.152  0.105
regionnorthwest:age    34.90      57.24   0.610    0.542  0.017 -0.037
regionsoutheast:age    83.64      63.33   1.321    0.187  0.036 -0.016
regionsouthwest:age   -33.63      61.41  -0.548    0.584 -0.015 -0.070
                       97.5%
(Intercept)           -0.014
regionnorthwest        0.031
regionsoutheast        0.020
regionsouthwest        0.054
age                    0.210
```

```
sexmale                 0.108
bmi                     0.199
regionnorthwest:age     0.072
regionsoutheast:age     0.090
regionsouthwest:age     0.037
```

This output shows the `coefficients` element of the ‘`resi`’ object, as well as the model call and the confidence level (α, by default = 0.05). The coefficient table looks very similar to the standard model summary output. The estimates will remain unchanged, but the standard errors differ because `summary()` uses the model-based (naive) standard error, whereas `resi()` defaults to use a robust estimate. These standard error estimates will remain valid under heteroskedasticity. Accordingly, the t values and p values are different, but our qualitative conclusions about statistical significance are unchanged in this example. The three rightmost columns are new and represent the RESI estimates and $(1 - \alpha)\%$ confidence intervals. Note that a RESI estimate further from 0 indicates a larger effect. The sign of the RESI estimate indicates the direction of the effect. From the table we can see that BMI is estimated to have a small to moderate effect (0.152 (CI: 0.105, 0.199)) based on the ranges given in Section 2.4. Sex is estimated to have a small effect (0.058 (CI: 0.004, 0.108)). The effect size estimates are conditional on the other terms in the model. Because our model includes an interaction on age and region, the RESI estimate for age in the coefficient table is interpreted as the estimated effect size of age for those in the northeast (the reference region). This is estimated to be 0.150 (CI: 0.094, 0.210), a small to moderate effect. For these results, if the p value is less than 0.05, then the CI for the RESI does not contain 0. This will not always be the case because the RESI CI is estimated for the distribution of the effect size estimator under the alternative.

Because region is a factor variable, the test for region and its interaction with age corresponds to multiple parameters in the model. To obtain an effect size estimate for multiple parameters that correspond to a single variable, we can report the ANOVA table. We can obtain this with either the standard `anova()` function or the `car::Anova()` function on the ‘`resi`’ object.

```
R> anova(resi_obj_lm)

Analysis of Deviance Table (Type II tests)

Response: charges
           Df      F Pr(>F)   RESI  2.5% 97.5%
region      3   1.60  0.189 0.0365 0.000 0.120
age         1 117.70  0.000 0.2951 0.240 0.360
sex         1   4.56  0.033 0.0515 0.000 0.105
bmi         1  31.07  0.000 0.1498 0.101 0.197
region:age  3   1.12  0.341 0.0161 0.000 0.101
```

By default, `resi()` uses a Type II sum of squares, but this can be changed in the arguments (Papachristodoulou and Prajna 2005). This output is the same as running `car::Anova()` on the model using `sandwich::vcovHC` as the `.vcov` argument, but with the three rightmost columns added for the RESI estimates and confidence intervals. The interpretation of the

RESI is the same as the coefficient table, but we note that in the ANOVA table, the RESI estimates are all nonnegative because they are estimated from F statistics. The estimates in the ANOVA table differ for two reasons: (1) Type II sum of squares first tests main effects without their interactions in the model; for example, the RESI estimate for age is interpreted as the effect of age compared to a model that does not include age or the interaction term for age and region. (2) For variables that are tested on 1 degree of freedom, the ANOVA table estimates the absolute effect size, whereas the coefficient table uses the unbiased signed effect size by default (see Section 2.2). For example, with sex and BMI, we notice that the estimates are close in the ANOVA and coefficients tables, but not exactly equal in absolute value. This is due to using the default `unbiased = TRUE` argument, which uses the t to S estimator from Equation 4 rather than the one based on the F to S formula.

An overall Wald test is also reported in the model.

```
R> omnibus(resi_obj_lm)

Analysis of effect sizes based on RESI:
Confidence level =  0.05
Wald test

Model 1: charges ~ 1
Model 2: charges ~ region * age + sex + bmi
  Res.Df Df      F Pr(>F)  RESI  2.5% 97.5%
1   1337
2   1328  9 20.249      0 0.360 0.326 0.421
```

By default, this compares the model to a reduced model that has only the intercept. The RESI estimate represents the overall absolute effect size of the model. In this model, this is estimated to be 0.360 (CI: 0.326, 0.421). This is interpreted as a moderate to large effect.

We can also visualize the results using the `plot()` or `ggplot()` function (Figure 2). Running these functions on the '`resi`' object will plot the coefficient table. Margins will be automatically adjusted to accommodate long variable names, or this feature can be turned off with the argument `automar = FALSE`. Alternatively, the user can extract the estimates and CIs from the tables and plot using their preferred visualization tool.

```
R> library("ggplot2")
R> plot(resi_obj_lm)
R> ggplot(anova(resi_obj_lm))
```

If we want to see a plot of the ANOVA table, we can run `plot()`/`ggplot()` either directly on the `anova` element of the '`resi`' object or on `anova()` or `car::Anova()` on the '`resi`' object. These plots help us quickly visualize the RESI estimates and relative effect sizes of the variables.

If we want to use different arguments for the covariance estimator function or the ANOVA function, we can specify these using the `vcov.args` and `Anova.args` arguments, respectively, in the `resi()` function. For example, we can use the `sandwich::vcovHC()` function with the HC0 estimator instead of the default HC3 estimator (Long and Ervin 2000) and use Type III sum of squares instead of Type II as follows.

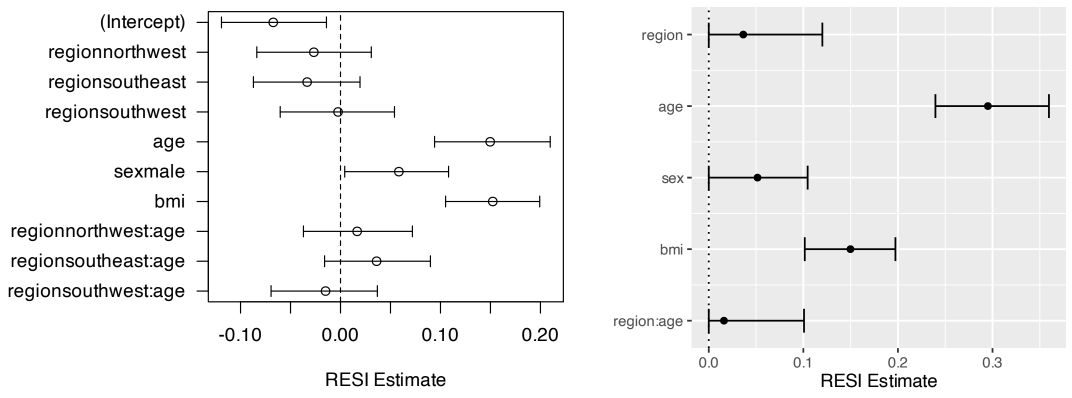

Figure 2: RESI estimates and 95% confidence intervals from linear model coefficients (left) and ANOVA tables (right).

```
R> set.seed(0827)
R> resi_obj_lm2 <- resi(mod_lm, vcov.args = list(type = "HC0"),
+    Anova.args = list(type = 3))
R> resi_obj_lm2

Analysis of effect sizes based on RESI:
Confidence level =  0.05
Call:  lm(formula = charges ~ region * age + sex + bmi, data = insurance)

Coefficient Table
                     Estimate Std. Error t value Pr(>|t|)   RESI   2.5%
(Intercept)          -5359.44    2155.75  -2.486    0.013 -0.068 -0.120
regionnorthwest      -2339.44    2372.36  -0.986    0.324 -0.027 -0.085
regionsoutheast      -3230.85    2618.64  -1.234    0.218 -0.034 -0.088
regionsouthwest       -232.48    2549.50  -0.091    0.927 -0.003 -0.061
age                    220.33      39.82   5.534    0.000  0.151  0.095
sexmale               1328.02     617.05   2.152    0.032  0.059  0.004
bmi                    323.77      57.56   5.625    0.000  0.154  0.106
regionnorthwest:age     34.90      56.68   0.616    0.538  0.017 -0.037
regionsoutheast:age     83.64      62.72   1.333    0.183  0.036 -0.016
regionsouthwest:age    -33.63      60.78  -0.553    0.580 -0.015 -0.070
                        97.5%
(Intercept)            -0.014
regionnorthwest         0.031
regionsoutheast         0.020
regionsouthwest         0.055
age                     0.212
sexmale                 0.109
bmi                     0.201
regionnorthwest:age     0.073
regionsoutheast:age     0.091
```

```
regionsouthwest:age    0.037

Analysis of Deviance Table (Type III tests)

Response: charges
            Df     F Pr(>F)  RESI  2.5% 97.5%
(Intercept)  1  6.18  0.013 0.062 0.000 0.117
region       3  0.73  0.537 0.000 0.000 0.096
age          1 30.62  0.000 0.149 0.091 0.210
sex          1  4.63  0.032 0.052 0.000 0.105
bmi          1 31.64  0.000 0.151 0.103 0.199
region:age   3  1.14  0.332 0.018 0.000 0.102

Overall RESI comparing model to intercept-only model:

  Res.Df Df      F Pr(>F)  RESI  2.5% 97.5%
1   1328  9 20.634      0 0.363 0.330 0.426

Notes:
1. The RESI was calculated using a robust covariance estimator.
2. Confidence intervals (CIs) constructed using 1000 non-parametric
bootstraps.
```

Here, we print the full output of the '`resi`' object. In addition to elements previously discussed, notes on the type of covariance estimator (robust or naive) and type and number of bootstraps are found at the bottom. As expected, we can see that the results differ slightly from our first '`resi`' object output.

4.2. RESI on nonlinear least squares

In this example, we use `resi()` on a nonlinear least squares model using `nls()`, demonstrating a helpful workaround to deal with model convergence issues in '`nls`' models when bootstrapping. For this analysis, we use the `niering` dataset in the **sars** package (Matthews, Triantis, Whittaker, and Guilhaumon 2019). This dataset provides the area (in km^2) and number of plant species for 32 islands in the Kapingamarangi Atoll (Matthews *et al.* 2019).

```
R> data("niering", package = "sars")
R> head(niering)
```

```
        a  s
1 0.00012  5
2 0.00160  7
3 0.00240  8
4 0.00280 10
5 0.00360  9
6 0.00360 11
```

The species-to-area relationship is commonly modeled using a power curve, where $Species = c \cdot Area^z$ (Preston 1962). We can fit this model using `nls()` to estimate the c and z parameters. It is well known that '`nls`' models can be sensitive to the choice of starting values. For example, the following naive guesses for the starting values produce an error due to failed convergence.

```
R> mod_nls <- nls(s ~ c * a^z, data = niering, start = list(c = 2, z = 0.5))

Error in nls(s ~ c * a^z, data = niering, start = list(c = 2, z = 0.5)) :
singular gradient
```

If we use good starting values the model converges successfully.

```
R> mod_nls <- nls(s ~ c * a^z, data = niering, start = list(c = 3, z = 0.25))
R> summary(mod_nls)

Formula: s ~ c * a^z

Parameters:
  Estimate Std. Error t value Pr(>|t|)
c 89.30789   10.11148   8.832 7.59e-10 ***
z  0.40206    0.03677  10.935 5.49e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.819 on 30 degrees of freedom

Number of iterations to convergence: 12
Achieved convergence tolerance: 2.792e-06
```

With our '`nls`' model, we can run `resi()`, making sure to provide the data argument. For this example, we will demonstrate the Bayesian bootstrap option.

```
R> set.seed(0827)
R> resi_obj_nls <- resi(mod_nls, data = niering, boot.method = "bayes")
R> resi_obj_nls

Analysis of Effect sizes (ANOES) based on RESI:
Confidence level =  0.05
Call:  nls(formula = s ~ c * a^z, data = niering, start = list(c = 3,
    z = 0.25), algorithm = "default", control = list(maxiter = 50,
    tol = 1e-05, minFactor = 0.0009765625, printEval = FALSE,
    warnOnly = FALSE, scaleOffset = 0, nDcentral = FALSE), trace = FALSE)

Coefficient Table
  Estimate Std. Error t value Pr(>|t|)   RESI   2.5%  97.5%
c  89.3079    20.5866  4.3382    1e-04 0.7475 0.6241 1.5651
z   0.4021     0.0597  6.7325    0e+00 1.1601 0.9599 2.5043
```

```
Overall RESI comparing model to intercept-only model:

              chi2 df P   RESI   2.5%   97.5%
Wald Test 142.1953  2 0 2.0931 1.2195  3.6873

Notes:
1. The RESI was calculated using a robust covariance estimator.
2. Credible intervals constructed using 1000 Bayesian bootstraps.
3. The bootstrap was successful in 744 out of 1000 attempts.
```

The `resi()` function runs without error, and we obtain a coefficients table and an overall Wald test for the model with RESI estimates and 95% credible intervals. Although the original model was able to be fit by `nls()` without issue, using this model for `resi()` does not have optimal performance. We can see from Note 3 that the bootstrap was only successful in 744 of the replicates.

The unsuccessful replicates failed to converge when attempting to update the '`nls`' model with bootstrap data. We can improve the performance of `resi()` for this model by refitting the '`nls`' model with different start values before running `resi()`. We use the estimated coefficients from the original model as the new start values.

```
R> mod_nls2 <- nls(s ~ c * a^z, data = niering,
+    start = list(c = coef(mod_nlsn)[1], z = coef(mod_nlsn)[2]))
R> set.seed(0827)
R> resi(mod_nls2, data = niering, boot.method = "bayes")

Analysis of effect sizes based on RESI:
Confidence level =  0.05
Call:  nls(formula = s ~ c * a^z, data = niering,
    start = list(c = coef(mod_nls)[1], z = coef(mod_nls)[2]),
    algorithm = "default", control = list(maxiter = 50,
    tol = 1e-05, minFactor = 0.0009765625, printEval = FALSE,
    warnOnly = FALSE, scaleOffset = 0, nDcentral = FALSE), trace = FALSE)

Coefficient Table
    Estimate Std. Error t value Pr(>|t|)  RESI  2.5% 97.5%
c.c   89.308      20.59   4.338        0 0.748 0.619 1.941
z.z    0.402       0.06   6.732        0 1.160 0.950 2.506

Overall RESI comparing model to intercept-only model:

             chi2 df P  RESI  2.5%  97.5%
overall.tab 142.2  2 0 2.093 1.120  3.557

Notes:
1. The RESI was calculated using a robust covariance estimator.
2. Credible intervals constructed using 1000 Bayesian bootstraps.
3. The bootstrap was successful in 1000 out of 1000 attempts.
```

We see that running `resi()` on this model gives us the same RESI estimates and similar credible intervals, but the performance of the bootstrap is much better. In this case all 1000 bootstrap replicates are successful, and we obtain credible intervals based on the desired number of bootstrap replicates. When using `resi()` on an '`nls`' model, consider using this strategy if the model fails to converge in many of the bootstrap samples.

4.3. RESI on survival model

As a final example, we consider a parametric survival model using the **survival** package. Following an example in the survival package documentation, we fit a Weibull model using the `lung` dataset in the **survival** package (Therneau 2023). The outcome is survival time (in days). The regressors are age, sex, and Karnofsky score.

It is important to note that for survival models (using `coxph()` or `survreg()`), the option to use a robust covariance is included in the model fitting function. The `resi()` function ignores the `vcovfunc` argument for these model types and assumes the user has specified the desired covariance method when fitting the model.

In this example we also demonstrate how the user can obtain confidence intervals for different levels of α both during and after running the `resi()` function. The `alpha` arguments allows the user to specify a vector of α levels, and the results corresponding to these levels will be output with the '`resi`' object. In the case that the user wants to produce different level confidence intervals after running the `resi()` function without rerunning the bootstrapping process the user can set `store.boot = TRUE`. This will store a '`boot`' object in the '`resi`' object called `boot.results` that includes all of the RESI estimates for each bootstrap replicate. Confidence intervals of a specific α level can then be obtained via the **boot** package, manually, or by using the `summary()` or `anova()/car::Anova()` functions.

For this example we will use the `unbiased = FALSE` option to demonstrate the alternate Z to S estimator described in Equation 5. We also specify a reduced model to compute a RESI for a subset of the model parameters, rather than using an intercept-only model. Our reduced model uses Karnofsky score as the only predictor and we use 1500 bootstrap replicates to construct CIs.

```
R> library("survival")
R> set.seed(0827)
R> mod_surv <- survreg(Surv(time, status) ~ age + sex + ph.karno,
+    data = survival::lung, dist="weibull", robust = TRUE)
R> mod_surv_reduced <- survreg(Surv(time, status) ~ ph.karno,
+                      data = survival::lung, dist="weibull",
+                      robust = TRUE)
R> resi_obj_surv <- resi(mod_surv, mod_surv_reduced, data = survival::lung,
+    unbiased = FALSE, store.boot = TRUE, alpha = c(0.05, 0.1), nboot = 1500)
R> resi_obj_surv

Analysis of effect sizes based on RESI:
Confidence level =  0.05 0.1
Full Model:survreg(formula = Surv(time, status) ~ age + sex + ph.karno,
    data = survival::lung, dist = "weibull", robust = TRUE)
Reduced Model:survreg(formula = Surv(time, status) ~ ph.karno,
     data = survival::lung, dist = "weibull", robust = TRUE)

Coefficient Table
            Estimate Std. Error z value Pr(>|z|)   RESI   2.5%     5%    95%
(Intercept)    5.326      0.685   7.771    0.000  0.512  0.338  0.360  0.669
age           -0.009      0.007  -1.217    0.223 -0.046 -0.205 -0.184  0.000
sex            0.370      0.123   3.022    0.003  0.189  0.032  0.071  0.299
ph.karno       0.009      0.006   1.587    0.112  0.082  0.000  0.000  0.268
Log(scale)    -0.281      0.067  -4.164    0.000 -0.268 -0.443 -0.422 -0.171
               97.5%
(Intercept)    0.700
age            0.000
sex            0.318
ph.karno       0.306
Log(scale)    -0.152

Analysis of Deviance Table (Type II tests)

Response: Surv(time, status)
         Df Chisq Pr(>Chisq)  RESI  2.5%    5%   95% 97.5%
age       1  1.48      0.224 0.046 0.000 0.000 0.184 0.205
sex       1  9.13      0.003 0.189 0.032 0.071 0.299 0.318
ph.karno  1  2.52      0.112 0.082 0.000 0.000 0.268 0.306

Overall RESI comparing full model to reduced model:

  Res.Df Df Chisq Pr(>Chisq)  RESI  2.5%    5%   95% 97.5%
1    222  2 10.23      0.006 0.190 0.039 0.078 0.315 0.339

Notes:
1. The RESI was calculated using a robust covariance estimator.
2. Confidence intervals (CIs) constructed using 1500 non-parametric
bootstraps.
```

The printed output reflects the modifications we made to the `resi()` arguments. The reduced model formula is displayed, which is relevant only for the overall RESI estimate. For comparison, we can look at the `overall` element of running `resi()` with an intercept-only reduced model.

```
R> set.seed(0827)
R> omnibus(resi(mod_surv, data = survival::lung,
+    unbiased = FALSE, alpha = c(0.05, 0.1), nboot = 1500))

Analysis of effect sizes based on RESI:
Confidence level =  0.05 0.1
Wald test
```

```
Model 1: Surv(time, status) ~ 1
Model 2: Surv(time, status) ~ age + sex + ph.karno
  Res.Df Df Chisq Pr(>Chisq)  RESI  2.5%    5%   95% 97.5%
1    225
2    222  3  11.5      0.009 0.194 0.062 0.098 0.381 0.408
```

The overall RESI estimate is slightly higher when comparing to an intercept-only model than the model that adjusts for Karnofsky score.

The coefficient table and ANOVA table are computed only for the full model. Because we chose the unbiased option, the RESI estimates are equal in absolute value for the coefficient and ANOVA tables. The RESI estimates for age and Karnofsky (-0.046 (95% CI: -0.205, 0) and 0.082 (95% CI: 0, 0.306) respectively) are interpreted as small effects, while the RESI estimate for sex (0.189 (95% CI: 0.032, 0.318)) is interpreted as a small to moderate effect. We see from the output that there are now four columns for the RESI confidence intervals – a lower and upper bound for each of the α levels specified. If we now want to obtain an interval with a different confidence level, we can run `summary()` and `anova()` using the `alpha` argument and specify a vector of values.

```
R> summary(resi_obj_surv, alpha = c(0.001, 0.01))

Analysis of effect sizes based on RESI:
Confidence level =  0.001 0.01
Call:  survreg(formula = Surv(time, status) ~ age + sex + ph.karno,
    data = survival::lung, dist = "weibull", robust = TRUE)

Coefficient Table
            Estimate Std. Error z value Pr(>|z|)   RESI  0.05%   0.5%  99.5%
(Intercept)    5.326      0.685   7.771    0.000  0.512  0.231  0.293  0.758
age           -0.009      0.007  -1.217    0.223 -0.046 -0.293 -0.247  0.055
sex            0.370      0.123   3.022    0.003  0.189  0.000  0.000  0.361
ph.karno       0.009      0.006   1.587    0.112  0.082 -0.072  0.000  0.383
Log(scale)    -0.281      0.067  -4.164    0.000 -0.268 -0.561 -0.517 -0.087
              99.95%
(Intercept)    0.807
age            0.105
sex            0.425
ph.karno       0.445
Log(scale)     0.000

R> anova(resi_obj_surv, alpha = c(0.001, 0.01))

         Df Chisq Pr(>Chisq)   RESI 0.05%   0.5% 97.5% 99.95%
age       1  1.48     0.2235 0.0461     0      0 0.247  0.293
sex       1  9.13     0.0025 0.1892     0      0 0.361  0.425
ph.karno  1  2.52     0.1124 0.0818     0      0 0.383  0.445
```

Note that if we try to specify different α levels with these functions on a '`resi`' object that did not use the `store.boot = TRUE` option, an error will occur with a message informing the user that this option was not used. A larger number of bootstrap samples are necessary to obtain adequate precision for smaller `alpha` levels.

5. Conclusion

The **RESI** R package aims to provide estimates and confidence intervals for the recently introduced index in a way that intuitively complements common data analysis workflow in R. Similarly to running `summary()` after fitting a model, a user can simply run `resi()` on many models and obtain several useful model summaries that include original model estimates and p values as well as RESI estimates with confidence intervals. Reporting model parameter estimates with confidence intervals is useful for communicating estimated effects within a given context. Adding RESI estimates and their confidence intervals provides the extra benefit of communicating how strong or meaningful these effects may be in a way that can be compared across many model settings.

The package provides dedicated methods for several common model types currently, with more in process. Methods for both cross-sectional and longitudinal models are available, with longitudinal methods providing both a longitudinal and a per-measurement cross-sectional RESI estimate. For models that are not currently implemented, users can manually provide the relevant information to functions within **RESI** to obtain estimates directly. The package also makes it easy to visualize RESI estimates and convert to and from other effect size indices. The RESI is a widely applicable effect size index with several advantages, including the ability to accommodate nuisance parameters and incorporate robust covariance estimates. With increasing emphasis being placed on reporting of effect sizes in research, the **RESI** package is a user-friendly tool to easily report effect sizes and confidence intervals in publications.

6. Computational details

All examples were coded using R version 4.4.1 and **RESI** version 1.3.0. The versions of relevant packages for the examples include **sandwich** 3.1-0 (Zeileis 2006), **sars** 1.3.6 (Matthews *et al.* 2019), **survival** 3.7.0 (Therneau 2023), **boot** 1.3-30 (Canty and Ripley 2024), and **ggplot2** 3.5.1 (Wickham 2016).

7. Acknowledgements

This research is funded by R01MH123563.

References

Agresti A (2002). *Categorical Data Analysis.* Wiley Series in Probability and Statistics. John Wiley & Sons, Hoboken. doi:10.1002/0471249688.

Althouse AD, Below JE, Claggett BL, Cox NJ, de Lemos JA, Deo RC, Duval S, Hachamovitch

R, Kaul S, Keith SW, Secemsky E, Teixeira-Pinto A, Roger VL (2021). "Recommendations for Statistical Reporting in Cardiovascular Medicine: A Special Report From the American Heart Association." *Circulation*, **144**(4), e70–e91. doi:10.1161/CIRCULATIONAHA.121.055393.

Amaral EdOS, Line SRP (2021). "Current Use of Effect Size or Confidence Interval Analyses in Clinical and Biomedical Research." *Scientometrics*, **126**(11), 9133–9145. doi:10.1007/s11192-021-04150-3.

American Psychological Association (1994). *Publication Manual of the American Psychological Association*. 4th edition. American Psychological Association, Washington, DC.

American Psychological Association (2001). *Publication Manual of the American Psychological Association*. 5th edition. American Psychological Association, Washington, DC.

American Psychological Association (2010). *Publication Manual of the American Psychological Association*. 6th edition. American Psychological Association, Washington, DC.

American Psychological Association (2020). *Publication Manual of the American Psychological Association*. 7th edition. American Psychological Association, Washington, DC.

Anderson D (2020). ***esvis**: Visualization and Estimation of Effect Sizes*. doi:10.32614/CRAN.package.esvis. R package version 0.3.1.

Bates D, Mächler M, Bolker B, Walker S (2015). "Fitting Linear Mixed-Effects Models Using **lme4**." *Journal of Statistical Software*, **67**(1), 1–48. doi:10.18637/jss.v067.i01.

Ben-Shachar MS, Lüdecke D, Makowski D (2020). "**effectsize**: Estimation of Effect Size Indices and Standardized Parameters." *Journal of Open Source Software*, **5**(56), 2815. doi:10.21105/joss.02815.

Betensky RA (2019). "The p Value Requires Context, Not a Threshold." *The American Statistician*, **73**(sup1), 115–117. doi:10.1080/00031305.2018.1529624.

Boos DD, Stefanski LA (2013). *Essential Statistical Inference: Theory and Methods*. Springer Texts in Statistics. Springer-Verlag, New York. doi:10.1007/978-1-4614-4818-1.

Buchanan EM, Gillenwaters A, Scofield JE, Valentine K (2019). ***MOTE**: Measure of the Effect – Package to Assist in Effect Size Calculations and Their Confidence Intervals*. R package version 1.0.2, URL http://github.com/doomlab/MOTE.

Canty A, Ripley BD (2024). ***boot**: Bootstrap R (S-PLUS) Functions*. doi:10.32614/CRAN.package.boot. R package version 1.3-30.

Carey VJ (2024). ***gee**: Generalized Estimation Equation Solver*. doi:10.32614/CRAN.package.gee. R package version 4.13-27.

Cohen J (1988). *Statistical Power Analysis for the Behavioral Sciences*. Erlbaum Associates, Hillsdale. doi:10.4324/9780203771587.

Fox J, Weisberg S (2019). *An R Companion to Applied Regression*. 3rd edition. Sage Publications, Thousand Oaks. URL https://socialsciences.mcmaster.ca/jfox/Books/Companion/.

Fritz CO, Morris PE, Richler JJ (2012). "Effect Size Estimates: Current Use, Calculations, and Interpretation." *Journal of Experimental Psychology: General*, **141**(1), 2–18. doi:10.1037/a0024338.

Gerlanc D, Kirby K (2023). ***bootES**: Bootstrap Confidence Intervals on Effect Sizes.* doi:10.32614/CRAN.package.bootES. R package version 1.3.0.

Halekoh U, Højsgaard S, Yan J (2006). "The R Package **geepack** for Generalized Estimating Equations." *Journal of Statistical Software*, **15/2**, 1–11. doi:10.18637/jss.v015.i02.

Hedges LV, Olkin I (1985). *Statistical Methods for Meta-Analysis*. Elsevier, London. doi:10.1016/C2009-0-03396-0.

IBM Corporation (2011). *IBM SPSS Statistics 20*. IBM Corporation, Armonk, NY. URL http://www-01.ibm.com/software/analytics/spss/.

Jones M, Kang K, Vandekar S (2025). ***RESI**: Robust Effect Size Index (RESI) Estimation.* doi:10.32614/CRAN.package.RESI. R package version 1.3.0.

Kadel RP, Kip KE (2012). "A SAS Macro to Compute Effect Size (Cohen's d) and its Confidence Interval from Raw Survey Data."

Kang K, Jones MT, Armstrong K, Avery S, McHugo M, Heckers S, Vandekar S (2023). "Accurate Confidence and Bayesian Interval Estimation for Non-Centrality Parameters and Effect Size Indices." *Psychometrika*, **88**(1), 253–273. doi:10.1007/s11336-022-09899-x.

Kelley K (2007). "Confidence Intervals for Standardized Effect Sizes: Theory, Application, and Implementation." *Journal of Statistical Software*, **20**(8), 1–24. doi:10.18637/jss.v020.i08.

Kelley K (2023). ***MBESS**: The **MBESS** R Package.* doi:10.32614/CRAN.package.MBESS. R package version 4.9.3.

Lenth RV (2016). "Least-Squares Means: The R Package **lsmeans**." *Journal of Statistical Software*, **69**(1), 1–33. doi:10.18637/jss.v069.i01.

Lenth RV (2024). ***emmeans**: Estimated Marginal Means, aka Least-Squares Means.* doi:10.32614/CRAN.package.emmeans. R package version 1.10.4.

Lesnoff M, Lancelot R (2023). ***aod**: Analysis of Overdispersed Data.* doi:10.32614/CRAN.package.aod. R package version 1.3.3.

Long JS, Ervin LH (2000). "Using Heteroscedasticity Consistent Standard Errors in the Linear Regression Model." *The American Statistician*, **54**(3), 217–224. doi:10.2307/2685594.

Lüdecke D (2019). ***esc**: Effect Size Computation for Meta Analysis (Version 0.5.1).* doi:10.32614/CRAN.package.esc. R package version 0.5.1.

MacKinnon JG, White H (1985). "Some Heteroskedasticity-Consistent Covariance Matrix Estimators with Improved Finite Sample Properties." *Journal of Econometrics*, **29**(3), 305–325. doi:10.1016/0304-4076(85)90158-7.

Mangiafico SS (2024). ***rcompanion**: Functions to Support Extension Education Program Evaluation.* Rutgers Cooperative Extension, New Brunswick, New Jersey. doi:10.32614/CRAN.package.rcompanion/. R package version 2.4.36.

Mantel N (1963). "Chi-Square Tests with One Degree of Freedom; Extensions of the Mantel-Haenszel Procedure." *Journal of the American Statistical Association*, **58**(303), 690–700. doi:10.2307/2282717.

Matthews T, Triantis K, Whittaker R, Guilhaumon F (2019). "**sars**: An R Package for Fitting, Evaluating and Comparing Species-Area Relationship Models." *Ecography*, **42**, 1446–1455. doi:10.1111/ecog.04271.

Navarro D (2021). *Learning Statistics with R: A Tutorial for Psychology Students and Other Beginners.* University of New South Wales, Sydney, Australia. doi:10.32614/CRAN.package.lsr. R package version 0.5.2, URL https://learningstatisticswithr.com.

Papachristodoulou A, Prajna S (2005). "A Tutorial on Sum of Squares Techniques for Systems Analysis." In *Proceedings of the 2005, American Control Conference, 2005.*, pp. 2686–2700. IEEE. doi:10.1109/ACC.2005.1470374.

Pinheiro J, Bates D, R Core Team (2024). ***nlme**: Linear and Nonlinear Mixed Effects Models.* doi:10.32614/CRAN.package.nlme. R package version 3.1-166.

Preston FW (1962). "The Canonical Distribution of Commonness and Rarity: Part I." *Ecology*, **43**(2), 185. doi:10.2307/1931976.

R Core Team (2024). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rosenthal R (1994). "Parametric Measures of Effect Size." In *The Handbook of Research Synthesis*, volume 621, pp. 231–244. Russell Sage Foundation, New York.

Rubin DB (1981). "The Bayesian Bootstrap." *The Annals of Statistics*, **9**(1), 130–134. doi:10.1214/aos/1176345338.

SAS Institute Inc (2016). *SAS/STAT 14.2 User's Guide: High-Performance Procedures.* SAS Institute Inc., Cary. URL https://support.sas.com/documentation/onlinedoc/stat/142/stathpug.pdf.

SAS Institute Inc (2020). *The SAS System, Version 15.2.* SAS Institute Inc., Cary. URL https://www.sas.com/.

Serdar CC, Cihan M, Yücel D, Serdar MA (2021). "Sample Size, Power and Effect Size Revisited: Simplified and Practical Approaches in Pre-Clinical, Clinical and Laboratory Studies." *Biochemia Medica*, **31**(1), 010502. doi:10.11613/BM.2021.010502.

StataCorp (2023). *Stata Statistical Software: Release 18.* StataCorp LLC, College Station. URL https://www.stata.com/.

Sullivan GM, Feinn R (2012). "Using Effect Size – or Why the p Value Is Not Enough." *Journal of Graduate Medical Education*, **4**(3), 279–282. doi:10.4300/JGME-D-12-00156.1.

Therneau TM (2023). *A Package for Survival Analysis in R*. doi:10.32614/CRAN.package.survival. R package version 3.7-0.

Torchiano M (2020). ***effsize**: Efficient Effect Size Computation*. doi:10.32614/CRAN.package.effsize. R package version 0.8.1.

Vandekar S, Tao R, Blume J (2020). "A Robust Effect Size Index." *Psychometrika*, **85**(1), 232. doi:10.1007/s11336-020-09698-2.

Vandekar SN, Stephens J (2021). "Improving the Replicability of Neuroimaging Findings by Thresholding Effect Sizes Instead of p Values." *Human Brain Mapping*, **42**(8), 2393–2398. doi:10.1002/hbm.25374.

Wasserstein RL, Lazar NA (2016). "The ASA's Statement on p Values: Context, Process, and Purpose." *The American Statistician*, **70**(2), 129–133. doi:10.1080/00031305.2016.1154108.

White H (1980). "A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity." *Econometrica: Journal of the Econometric Society*, pp. 817–838. doi:10.2307/1912934.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag. URL https://ggplot2.tidyverse.org/.

Wickham H, Hesselberth J, Salmon M (2024). ***pkgdown**: Make Static HTML Documentation for a Package*. doi:10.32614/CRAN.package.pkgdown. R package version 2.1.1.

Wickham H, Hester J, Chang W, Bryan J (2022). ***devtools**: Tools to Make Developing R Packages Easier*. doi:10.32614/CRAN.package.devtools. R package version 2.4.5.

Wilkinson L (1999). "Statistical Methods in Psychology Journals: Guidelines and Explanations." *American Psychologist*, **54**, 594–604. doi:10.1037/0003-066X.54.8.594.

Zeileis A (2006). "Object-Oriented Computation of Sandwich Estimators." *Journal of Statistical Software*, **16**, 1–16. doi:10.18637/jss.v016.i09.

Zeileis A, Hothorn T (2002). "Diagnostic Checking in Regression Relationships." *R News*, **2**(3), 7–10. URL https://CRAN.R-project.org/doc/Rnews/.

Zeileis A, Köll S, Graham N (2020). "Various Versatile Variances: An Object-Oriented Implementation of Clustered Covariances in R." *Journal of Statistical Software*, **95**(1), 1–36. doi:10.18637/jss.v095.i01.

Zhang Z, Schoeps N (1997). "On Robust Estimation of Effect Size Under Semiparametric Models." *Psychometrika*, **62**(2), 201–214. doi:10.1007/BF02295275.

Affiliation:

Megan Jones
Vanderbilt University
Department of Biostatistics
2525 West End Ave., Suite 1136, Nashville
Tennessee 37203, United States of America
E-mail: `megan.n.taylor@vanderbilt.edu`

Journal of Statistical Software `https://www.jstatsoft.org/`
published by the Foundation for Open Access Statistics `https://www.foastat.org/`
March 2025, Volume 112, Issue 3 *Submitted:* 2023-06-12
`doi:10.18637/jss.v112.i03` *Accepted:* 2024-05-06