

# Package: RESI (via r-universe)

October 31, 2024

**Type** Package

**Title** Robust Effect Size Index (RESI) Estimation

**Version** 1.2.4

**Description** Summarize model output using a robust effect size index.  
The index is introduced in Vandekar, Tao, & Blume (2020)  
<[doi:10.1007/s11336-020-09698-2](https://doi.org/10.1007/s11336-020-09698-2)>.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Collate** 'internal\_functions.R' 'summary.resi.R' 'resi.R' 'resi\_pe.R'  
'Anova.resi.R' 'data.R' 'chisq2S.R' 'f2S.R' 't2S.R' 'z2S.R'  
'Rsq2S.R' 'S2Rsq.R' 'S2d.R' 'S2fsq.R' 'd2S.R' 'fsq2S.R'  
'plot.R' 'ggplot.R' 'print.R' 'omnibus.R'

**Imports** aod, boot, car, clubSandwich, ggplot2, lmtest, nlme, sandwich

**Suggests** gee, geopack, knitr, lme4, pscl, regtools, rmarkdown,  
splines, survival, tibble, testthat (>= 3.0.0)

**RoxygenNote** 7.2.3

**Depends** R (>= 2.10)

**Config/testthat/edition** 3

**URL** <https://statimagcoll.github.io/RESI/>,  
<https://github.com/statimagcoll/RESI>

**VignetteBuilder** knitr

**BugReports** <https://github.com/statimagcoll/RESI/issues>

**Repository** <https://statimagcoll.r-universe.dev>

**RemoteUrl** <https://github.com/statimagcoll/resi>

**RemoteRef** HEAD

**RemoteSha** 3a5ecece4b608959d72c9e43009babece8f8ff73

## Contents

anova.resi . . . . .	2
chisq2S . . . . .	3
d2S . . . . .	4
depression . . . . .	5
f2S . . . . .	6
fsq2S . . . . .	7
ggplot.resi . . . . .	7
insurance . . . . .	8
omnibus . . . . .	9
plot.resi . . . . .	9
resi . . . . .	11
resi_pe . . . . .	19
Rsqr2S . . . . .	24
S2d . . . . .	25
S2fsq . . . . .	26
S2Rsqr . . . . .	27
summary.resi . . . . .	28
t2S . . . . .	29
z2S . . . . .	30
<b>Index</b>	<b>32</b>

---

anova.resi	<i>Anova method for resi objects</i>
------------	--------------------------------------

---

### Description

After running the `resi` function on a fitted model, this function can be used to print the Anova-style table component. If the `resi` function was run with the `'store.boot = TRUE'` option to store the full matrix of bootstrapped estimates, the user can specify a different alpha level for this function's confidence intervals.

### Usage

```
## S3 method for class 'resi'
anova(object, alpha = NULL, ...)
```

### Arguments

object	an object resulting from <code>resi</code> function
alpha	an optional new specification for the confidence level. Can be vector-valued
...	ignored

### Details

The `resi` function uses the `car::Anova` function to compute the Anova table.

**Value**

Returns an ‘anova’ object containing the computed Anova-style table

**Examples**

```
# fit a model
mod = lm(charges ~ bmi + sex, data = RESI::insurance)

# run resi with the store.boot = TRUE option
resi.obj = resi(mod, nboot = 100, store.boot = TRUE, alpha = 0.01)

# run anova, specifying a different alpha level if desired
anova(resi.obj, alpha = 0.05)
```

---

chisq2S	<i>Compute the robust effect size index estimate from chi-squared statistic.</i>
---------	----------------------------------------------------------------------------------

---

**Description**

This function computes the robust effect size index from Vandekar, Tao, & Blume (2020). Vector arguments are accepted. If different length arguments are passed they are dealt with in the usual way of R. For mixed effects models, RESI is conditional on the average correlation structure within subjects.

**Usage**

```
chisq2S(chisq, df, n)
```

**Arguments**

chisq	The chi-square statistic for the parameter of interest.
df	Number of degrees of freedom of the chi-square statistic.
n	Number of independent samples.

**Details**

The formula for converting a Chi-square statistic to RESI is:

$$S = \sqrt{\max(0, (chisq - df)/n)}$$

**Value**

Returns a scalar or vector argument of the the robust effect size index estimate.

**Examples**

```
# obtain Chi-sq value by fitting an lm and running a Wald test
mod = lm(charges ~ region * age + bmi + sex, data = RESI::insurance)

# run a Wald test with robust variance
wt = lmtest::waldtest(mod, vcov = sandwich::vcovHC, test = "Chisq")

# get Chi-sq value and degrees of freedom
chisq = wt$Chisq[2]
df = abs(wt$Df[2])

# run chisq2S to convert to RESI
chisq2S(chisq, df = df, n = nrow(mod$model))
```

d2S

*Covert Cohen's d to ISI***Description**

Converts Cohen's  $d$  robust effect size index (S) using the formula from Vandekar, Tao, & Blume (2020).

**Usage**

```
d2S(d, pi = 0.5)
```

**Arguments**

d	Numeric, value of Cohen's $d$ .
pi	Numeric, the sampling proportions.

**Details**

The pi parameter comes from the fact that Cohen's  $d$  doesn't account for unequal sample proportions in the population, but S does.

The default is set to a natural value 1/2, which corresponds to a case control design, for example, where sampling proportions always are controlled by the experimenter.

The formula to convert Cohen's  $d$  to S is:

$$S = d / \sqrt{(1/\pi + 1/(1 - \pi))}$$

**Value**

Returns an estimate the robust effect size index

**Examples**

```
# Consider an experiment with equal sampling proportions and a medium effect size
# corresponding to a Cohen's d of 0.5.
# convert to RESI (S)
d2S(d = 0.5)

# This corresponds to a RESI of 0.25.
```

---

depression	<i>Depression Treatment Data</i>
------------	----------------------------------

---

**Description**

A longitudinal dataset comparing two treatments for depression.

**Usage**

```
depression
```

**Format**

A data frame with 1020 rows and 5 variables:

**diagnose** diagnosed depression severity

**drug** treatment; standard or new

**id** patient id

**time** time point of treatment

**depression** depression response at time of treatment. 1 = Normal, 0 = Abnormal

**Source**

<http://static.lib.virginia.edu/statlab/materials/data/depression.csv>

**References**

Agresti, A. (2002). Categorical Data Analysis. Wiley, 2nd Edition.

f2S

*Compute the robust effect size index estimate from F-statistic***Description**

This function computes the robust effect size index from Vandekar, Tao, & Blume (2020). Vector arguments are accepted. If different length arguments are passed they are dealt with in the usual way of R.

**Usage**

```
f2S(f, df, rdf, n)
```

**Arguments**

f	The F statistic for the parameter of interest.
df	Number of degrees of freedom of the F statistic.
rdf	Model residual degrees of freedom.
n	Number of independent samples.

**Details**

The formula for converting an F statistic to S is:

$$S = \sqrt{\max(0, (f * df * (rdf - 2) / (rdf - df) / n))}$$

The estimator is derived by setting the statistic equal to the expected value of the test statistic and solving for S.

**Value**

Returns a scalar or vector argument of the the robust effect size index estimate.

**Examples**

```
# to obtain example F values, first fit a lm
mod = lm(charges ~ region * age + bmi + sex, data = RESI::insurance)

# run Anova, using a robust variance-covariance function
# get the F values and Df values
fs = car::Anova(mod, vcov. = sandwich::vcovHC)[1:5, "F"]
dfs = car::Anova(mod, vcov. = sandwich::vcovHC)[1:5, "Df"]

# get RESI estimates
f2S(fs, df = dfs, rdf = mod$df.residual, n = nrow(RESI::insurance))
```

---

`fsq2S`*Covert Cohen's  $f^2$  to  $S$* 

---

**Description**

Converts Cohen's  $f^2$  to robust effect size index ( $S$ ) using the formula from Vandekar, Tao, & Blume (2020).

**Usage**

```
fsq2S(fsq)
```

**Arguments**

`fsq` Numeric, value of Cohen's  $f^2$ .

**Details**

The formula for the conversion is:

$$S = \sqrt{f^2}$$

**Value**

Returns an estimate the robust effect size index

**Examples**

```
# consider a moderate effect size of  $f^2 = 0.3$ 
fsq2S(0.3)
# This corresponds to a RESI of 0.5477226
```

---

`ggplot.resi`*Plotting RESI Estimates and CIs*

---

**Description**

This function uses `ggplot2` graphics to plot robust effect size (RESI) estimates and confidence intervals from `'resi'`, `'summary_resi'`, and `'anova_resi'` objects.

**Usage**

```
## S3 method for class 'resi'
ggplot(data, mapping, alpha = NULL, error.bars = TRUE, ..., environment)
```

**Arguments**

<code>data</code>	Object of ‘resi’, ‘summary_resi’, or ‘anova_resi’ class
<code>mapping</code>	Ignored, included for consistency with ‘ggplot’ generic
<code>alpha</code>	Numeric, desired alpha level for confidence intervals
<code>error.bars</code>	Logical, whether to include end caps on the confidence intervals. Default = ‘TRUE’
<code>...</code>	Ignored
<code>environment</code>	Ignored, included for consistency with ‘ggplot’ generic

**Value**

Returns a ggplot of RESI point estimates

**Examples**

```
# create a resi object
resi_obj <- resi(lm(charges ~ region * age + bmi + sex, data = RESI::insurance),
  nboot = 10)

# plot ANOVA table
ggplot2::ggplot(anova(resi_obj))
```

---

insurance

*US Health Insurance Data*

---

**Description**

A dataset with 1338 observations on health insurance charges and demographic factors.

**Usage**

```
insurance
```

**Format**

A data frame with 1338 rows and 7 variables:

**age** age of primary beneficiary in years  
**sex** insurance contractor sex, male/female  
**bmi** body mass index  
**children** number of dependents  
**smoker** smoker/non-smoker  
**region** beneficiary’s region of US  
**charges** individual medical costs billed by health insurance

**Source**

<https://www.kaggle.com/datasets/teertha/ushealthinsurancedataset>



---

 omnibus

*Omnibus (Overall) Wald Test for resi objects*


---

### Description

After running the `resi` function on a fitted model, this function can be used to print the overall Wald test component. If the `resi` function was run with the `'store.boot = TRUE'` option to store the full matrix of bootstrapped estimates, the user can specify a different alpha level for this function's confidence intervals.

### Usage

```
omnibus(object, alpha = NULL, ...)
```

### Arguments

<code>object</code>	an object resulting from <code>resi</code> function
<code>alpha</code>	an optional new specification for the confidence level. Can be vector-valued
<code>...</code>	ignored

### Value

Returns a `'omnibus_resi'` object containing the computed omnibus Wald test

### Examples

```
# fit a model
mod = lm(charges ~ bmi + sex, data = RESI::insurance)

# run resi with the store.boot = TRUE option
resi_obj = resi(mod, nboot = 100, store.boot = TRUE, alpha = 0.01)

# run summary, specifying a different alpha level if desired
omnibus(resi_obj, alpha = 0.05)
```

---

 plot.resi

*Plotting RESI Estimates and CIs*


---

### Description

This function uses base graphics to plot robust effect size (RESI) estimates and confidence intervals from `'resi'`, `'summary_resi'`, and `'anova_resi'` objects.

**Usage**

```
## S3 method for class 'resi'
plot(
  x,
  alpha = NULL,
  ycex.axis = NULL,
  yaxis.args = list(),
  automar = TRUE,
  ...
)
```

**Arguments**

x	Object of 'resi', 'summary_resi', or 'anova_resi' class
alpha	Numeric, desired alpha level for confidence intervals
ycex.axis	Numeric, scale specifically for the variable name labels
yaxis.args	List, other arguments to be passed to <a href="#">axis</a> for the y-axis
automar	Logical, whether to automatically adjust the plotting margins to accommodate variable names. Default = 'TRUE'
...	Other graphical parameters passed to <a href="#">plot</a> and <a href="#">lines</a>

**Details**

This function creates a forest-like plot with RESI estimates for each variable or factor. The size of the left margin will be automatically adjusted (and returned to original after plotting) unless 'automar = FALSE'. Additional graphics parameters will be passed to the main plot function, the confidence intervals. Arguments specifically for the y-axis (variable names) can be specified using 'yaxis.args'. To manually adjust the size of the y-axis labels without affecting the x-axis, the user can specify a value for 'ycex.axis'.

**Value**

Returns a plot of RESI point estimates

**Examples**

```
# create a resi object
resi_obj <- resi(lm(charges ~ region * age + bmi + sex, data = RESI::insurance),
  nboot = 10)

# plot coefficients table, changing size of labels for both axes in the usual way
plot(resi_obj, cex.axis = 0.7)

# plot ANOVA table, changing the size of just the y-axis
plot(resi_obj, ycex.axis = 0.8)
```

---

resi	<i>Robust Effect Size Index (RESI) point and interval estimation for models</i>
------	---------------------------------------------------------------------------------

---

### Description

This function will estimate the robust effect size (RESI) from Vandekar, Tao, & Blume (2020) and its confidence interval in various ways for a fitted model object. The overall RESI is estimated via a Wald test. RESI is (optionally) estimated for each factor in coefficients-style table. RESI is (optionally) estimated for each variable/interaction in an Anova-style table for models with existing Anova methods. CIs can be calculated using either non-parametric or Bayesian bootstrapping.

### Usage

```
resi(model.full, ...)

## Default S3 method:
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  boot.method = "nonparam",
  vcovfunc = sandwich::vcovHC,
  alpha = 0.05,
  store.boot = FALSE,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  long = FALSE,
  clvar = NULL,
  ...
)

## S3 method for class 'glm'
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
```

```
nboot = 1000,  
vcovfunc = sandwich::vcovHC,  
alpha = 0.05,  
store.boot = FALSE,  
Anova.args = list(),  
vcov.args = list(),  
unbiased = TRUE,  
parallel = c("no", "multicore", "snow"),  
ncpus = getOption("boot.ncpus", 1L),  
...  
)
```

```
## S3 method for class 'lm'
```

```
resi(  
  model.full,  
  model.reduced = NULL,  
  data,  
  anova = TRUE,  
  coefficients = TRUE,  
  overall = TRUE,  
  nboot = 1000,  
  boot.method = "nonparam",  
  vcovfunc = sandwich::vcovHC,  
  alpha = 0.05,  
  store.boot = FALSE,  
  Anova.args = list(),  
  vcov.args = list(),  
  unbiased = TRUE,  
  parallel = c("no", "multicore", "snow"),  
  ncpus = getOption("boot.ncpus", 1L),  
  ...  
)
```

```
## S3 method for class 'nls'
```

```
resi(  
  model.full,  
  model.reduced = NULL,  
  data,  
  coefficients = TRUE,  
  overall = TRUE,  
  nboot = 1000,  
  boot.method = "nonparam",  
  anova = FALSE,  
  vcovfunc = r_nlshc,  
  alpha = 0.05,  
  store.boot = FALSE,  
  vcov.args = list(),  
  unbiased = TRUE,  
  ...  
)
```

```
parallel = c("no", "multicore", "snow"),
ncpus = getOption("boot.ncpus", 1L),
...
)
```

```
## S3 method for class 'survreg'
```

```
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  vcovfunc = vcov,
  alpha = 0.05,
  store.boot = FALSE,
  Anova.args = list(),
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ...
)
```

```
## S3 method for class 'coxph'
```

```
resi(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  nboot = 1000,
  vcovfunc = vcov,
  alpha = 0.05,
  store.boot = FALSE,
  Anova.args = list(),
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ...
)
```

```
## S3 method for class 'hurdle'
```

```
resi(
  model.full,
  model.reduced = NULL,
  data,
```

```
coefficients = TRUE,  
overall = TRUE,  
nboot = 1000,  
vcovfunc = sandwich::sandwich,  
anova = FALSE,  
alpha = 0.05,  
store.boot = FALSE,  
vcov.args = list(),  
unbiased = TRUE,  
parallel = c("no", "multicore", "snow"),  
ncpus = getOption("boot.ncpus", 1L),  
...  
)
```

```
## S3 method for class 'zeroinfl'
```

```
resi(  
  model.full,  
  model.reduced = NULL,  
  data,  
  coefficients = TRUE,  
  overall = TRUE,  
  nboot = 1000,  
  vcovfunc = sandwich::sandwich,  
  anova = FALSE,  
  alpha = 0.05,  
  store.boot = FALSE,  
  vcov.args = list(),  
  unbiased = TRUE,  
  parallel = c("no", "multicore", "snow"),  
  ncpus = getOption("boot.ncpus", 1L),  
  ...  
)
```

```
## S3 method for class 'geeglm'
```

```
resi(  
  model.full,  
  model.reduced = NULL,  
  data,  
  anova = TRUE,  
  coefficients = TRUE,  
  overall = TRUE,  
  nboot = 1000,  
  alpha = 0.05,  
  store.boot = FALSE,  
  unbiased = TRUE,  
  parallel = c("no", "multicore", "snow"),  
  ncpus = getOption("boot.ncpus", 1L),  
  ...  
)
```

```

)

## S3 method for class 'gee'
resi(
  model.full,
  data,
  nboot = 1000,
  alpha = 0.05,
  store.boot = FALSE,
  unbiased = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L),
  ...
)

## S3 method for class 'lme'
resi(
  model.full,
  alpha = 0.05,
  nboot = 1000,
  anova = TRUE,
  vcovfunc = clubSandwich::vcovCR,
  vcov.args = list(),
  ...
)

## S3 method for class 'lmerMod'
resi(
  model.full,
  alpha = 0.05,
  nboot = 1000,
  anova = TRUE,
  vcovfunc = clubSandwich::vcovCR,
  vcov.args = list(),
  ...
)

```

### Arguments

<code>model.full</code>	lm, glm, nls, survreg, coxph, hurdle, zeroinfl, gee, geeglm or lme model object.
<code>...</code>	Ignored.
<code>model.reduced</code>	Fitted model object of same type as <code>model.full</code> . By default 'NULL'; the same model as the full model but only having intercept.
<code>data</code>	Data.frame or object coercible to data.frame of <code>model.full</code> data (required for some model types).
<code>anova</code>	Logical, whether to produce an Anova table with the RESI columns added. By default = 'TRUE'.

<code>coefficients</code>	Logical, whether to produce a coefficients (summary) table with the RESI columns added. By default = 'TRUE'.
<code>overall</code>	Logical, whether to produce an overall Wald test comparing full to reduced model with RESI columns added. By default = 'TRUE'.
<code>nboot</code>	Numeric, the number of bootstrap replicates. By default, 1000.
<code>boot.method</code>	String, which type of bootstrap to use: 'nonparam' = non-parametric bootstrap (default); 'bayes' = Bayesian bootstrap.
<code>vcovfunc</code>	The variance estimator function for constructing the Wald test statistic. By default, <code>sandwich::vcovHC</code> (the robust (sandwich) variance estimator).
<code>alpha</code>	Numeric, significance level of the constructed CIs. By default, 0.05.
<code>store.boot</code>	Logical, whether to store all the bootstrapped estimates. By default, 'FALSE'.
<code>Anova.args</code>	List, additional arguments to be passed to <a href="#">Anova</a> function.
<code>vcov.args</code>	List, additional arguments to be passed to <code>vcovfunc</code> .
<code>unbiased</code>	Logical, whether to use the unbiased or alternative T/Z statistic to RESI conversion. By default, 'TRUE'. See details.
<code>parallel</code>	See documentation for <a href="#">boot</a> .
<code>ncpus</code>	See documentation for <a href="#">boot</a> .
<code>long</code>	Logical, whether the data is longitudinal/clustered. By default, 'FALSE'.
<code>clvar</code>	Character, the name of the cluster/id variable if data is clustered. By default, 'NULL'.

## Details

The RESI, denoted as  $S$ , is applicable across many model types. It is a unitless index and can be easily be compared across models. The RESI can also be converted to Cohen's  $d$  ([S2d](#)) under model homoskedasticity.

This function computes the RESI point estimates and bootstrapped confidence intervals based on Chi-square, F, T, or Z statistics. The robust (sandwich) variance is used by default, allowing for consistency under model-misspecification. The RESI is related to the non-centrality parameter of the test statistic. The RESI estimate is consistent for all four (Chi-square, F, T, and Z) types of statistics used. The Chi-square and F-based calculations rely on asymptotic theory, so they may be biased in small samples. When possible, the T and Z statistics are used. There are two formulas for both the T and Z statistic conversion. The first (default, `unbiased = TRUE`) are based on solving the expected value of the T or Z statistic for the RESI. The alternative is based on squaring the T or Z statistic and using the F or Chi-square statistic conversion. Both of these methods are consistent, but the alternative exhibits a notable amount of finite sample bias. The alternative may be appealing because its absolute value will be equal to the RESI based on the F or Chi-square statistic. The RESI based on the Chi-Square and F statistics is always greater than or equal to 0. The type of statistic used is listed with the output. See [f2S](#), [chisq2S](#), [t2S](#), and [z2S](#) for more details on the formulas.

For GEE (`geeglm`) models, a longitudinal RESI (L-RESI) and a cross-sectional, per-measurement RESI (CS-RESI) is estimated. The longitudinal RESI takes the specified clustering into account, while the cross-sectional RESI is estimated using a model where each measurement is its own cluster.



For most `lm` and `nls` model types, there is a Bayesian bootstrap option available as an alternative to the default, standard non-parametric bootstrap. The interpretation of a Bayesian bootstrapped interval is similar to that of a credible interval.

Certain model types require the data used for the model be entered as an argument. These are: `nls`, `survreg`, and `coxph`. Additionally, if a model includes certain functions (splines, factor, I), the data needs to be provided.

If running into convergence issues with `nls` models, it is advised to refit the `nls` model with starting values equal to the estimates provided by the model and then try rerunning `resi`.

### Value

Returns a list that includes function arguments, RESI point estimates, and confidence intervals in coefficients/anova-style tables

### Methods (by class)

- `resi(default)`: RESI point and interval estimation for models
- `resi(glm)`: RESI point and interval estimation for models
- `resi(lm)`: RESI point and interval estimation for `lm` models
- `resi(nls)`: RESI point and interval estimation for `nls` models
- `resi(survreg)`: RESI point and interval estimation for `survreg` models
- `resi(coxph)`: RESI point and interval estimation for `coxph` models
- `resi(hurdle)`: RESI point and interval estimation for hurdle models
- `resi(zeroinfl)`: RESI point and interval estimation for `zeroinfl` models
- `resi(geeglm)`: RESI point and interval estimation for GEE models
- `resi(gee)`: RESI point and interval estimation for GEE models
- `resi(lme)`: RESI point and interval estimation for LME (`nlme`) models
- `resi(lmerMod)`: RESI point and interval estimation for `lmerMod` models

### References

Vandekar S, Tao R, Blume J. A Robust Effect Size Index. *Psychometrika*. 2020 Mar;85(1):232-246. doi: 10.1007/s11336-020-09698-2.

Kang, K., Armstrong, K., Avery, S., McHugo, M., Heckers, S., & Vandekar, S. (2021). Accurate confidence interval estimation for non-centrality parameters and effect size indices. *arXiv preprint arXiv:2111.05966*.

### See Also

[resi\\_pe](#), [vcovHC](#), [f2S](#), [chisq2S](#), [z2S](#), [t2S](#)

## Examples

```

## for timing purposes, a small number of bootstrap replicates is used in the
## examples. Run them with a higher or default `nboot` argument for better performance

## RESI on a linear model
# fit linear model
mod = lm(charges ~ region * age + bmi + sex, data = RESI::insurance)

# run resi on fitted model with desired number of bootstrap replicates
# store bootstrap results for calculating different CIs later
resi_obj = resi(mod, nboot = 50, store.boot = TRUE)
# print output
resi_obj

# fit a reduced model for comparison
mod_red = lm(charges ~ bmi, data = RESI::insurance)

# running resi and including the reduced model will provide almost the exact same
# output as not including a reduced model. The difference is that the "overall"
# portion of the output will compare the full model to the reduced model.
# The "summary" and "anova" RESI estimates will be the same. (The bootstrapped
# confidence intervals may differ.)
resi(model.full = mod, model.reduced = mod_red, nboot = 10)

# used stored bootstrap results to get a different alpha-level confidence interval
summary(resi_obj, alpha = c(0.01, 0.1))
car::Anova(resi_obj, alpha = c(0.01, 0.1))

# the result of resi, as well as the summary or Anova of a `resi` object can be plotted
# if the resi object was created with the store.boot = `TRUE` option, any alpha
# can be specified
plot(resi_obj, alpha = 0.01)
# if the variable names on the y-axis are too long, you can reduce their size with
# the ycex.axis argument (or use regular common solutions like changing the margins)
plot(resi_obj, alpha = 0.01, ycex.axis = 0.5)

# for some model types and formula structures, data argument is required
if(requireNamespace("splines")){
  # fit logistic regression model with splines
  mod = glm(smoker ~ splines::ns(age, df = 3) + region, data = RESI::insurance,
    family = "binomial")

  # specify additional arguments to the variance-covariance function via vcov.args
  resi_obj = resi(mod, data = RESI::insurance, alpha = 0.01,
    vcov.args = list(type = "HC0"), nboot = 25)
  summary(resi_obj)
  car::Anova(resi_obj)}

## RESI on a survival model with alternate Z2S
if(requireNamespace("survival")){
  # fit coxph model on example data from survival package

```

```

# Note: for survival models, you need to specify robust variance in the model
# creation. resi will ignore the vcovfunc argument for this reason.
mod.coxph = survival::coxph(survival::Surv(time, status) ~ age + sex + wt.loss,
  data=survival::lung, robust = TRUE)

# run resi on the model
# to use the alternative Z to RESI formula (which is equal in absolute value to the
# chi-square to RESI (S) formula), specify unbiased = FALSE.
resi(mod.coxph, data = survival::lung, unbiased = FALSE, nboot = 10)}

```

---

resi\_pe

*Robust Effect Size Index (RESI) Point Estimation*


---

### Description

This function will estimate the robust effect size (RESI) from Vandekar, Tao, & Blume (2020). The overall RESI is estimated via a Wald test. RESI is (optionally) estimated for each factor in coefficients-style table. RESI is (optionally) estimated for each variable/interaction in an Anova-style table for models with existing Anova methods. This function is the building block for the [resi](#) function.

### Usage

```

resi_pe(model.full, ...)

## Default S3 method:
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  vcovfunc = sandwich::vcovHC,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  waldtype = 0,
  ...
)

## S3 method for class 'glm'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,

```

```
    coefficients = TRUE,
    overall = TRUE,
    vcovfunc = sandwich::vcovHC,
    Anova.args = list(),
    vcov.args = list(),
    unbiased = TRUE,
    waldtype = 0,
    ...
)

## S3 method for class 'lm'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  vcovfunc = sandwich::vcovHC,
  Anova.args = list(),
  vcov.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'nls'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  coefficients = TRUE,
  anova = FALSE,
  vcovfunc = r_nlshc,
  vcov.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'survreg'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  vcovfunc = vcov,
  Anova.args = list(),
```

```
    unbiased = TRUE,
    overall = TRUE,
    ...
  )

## S3 method for class 'coxph'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  vcovfunc = vcov,
  Anova.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'hurdle'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  coefficients = TRUE,
  anova = TRUE,
  vcovfunc = sandwich::sandwich,
  vcov.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'zeroinfl'
resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  coefficients = TRUE,
  anova = TRUE,
  vcovfunc = sandwich::sandwich,
  vcov.args = list(),
  unbiased = TRUE,
  overall = TRUE,
  ...
)

## S3 method for class 'geeglm'
```

```

resi_pe(
  model.full,
  model.reduced = NULL,
  data,
  anova = TRUE,
  coefficients = TRUE,
  overall = TRUE,
  unbiased = TRUE,
  ...
)

## S3 method for class 'gee'
resi_pe(model.full, data, unbiased = TRUE, ...)

## S3 method for class 'lme'
resi_pe(
  model.full,
  anova = TRUE,
  vcovfunc = clubSandwich::vcovCR,
  Anova.args = list(),
  vcov.args = list(),
  ...
)

## S3 method for class 'lmerMod'
resi_pe(
  model.full,
  anova = TRUE,
  vcovfunc = clubSandwich::vcovCR,
  Anova.args = list(),
  vcov.args = list(),
  ...
)

```

### Arguments

model.full	lm, glm, nls, survreg, coxph, hurdle, zeroinfl, gee, geeglm or lme model object.
...	Ignored.
model.reduced	Fitted model object of same type as model.full. By default 'NULL'; the same model as the full model but only having intercept.
data	Data.frame or object coercible to data.frame of model.full data (required for some model types).
anova	Logical, whether to produce an Anova table with the RESI columns added. By default = 'TRUE'.
coefficients	Logical, whether to produce a coefficients (summary) table with the RESI columns added. By default = 'TRUE'.

overall	Logical, whether to produce an overall Wald test comparing full to reduced model with RESI columns added. By default = 'TRUE'.
vcovfunc	The variance estimator function for constructing the Wald test statistic. By default, sandwich::vcovHC (the robust (sandwich) variance estimator).
Anova.args	List, additional arguments to be passed to Anova function.
vcov.args	List, additional arguments to be passed to vcovfunc.
unbiased	Logical, whether to use the unbiased or alternative T/Z statistic to RESI conversion. By default, 'TRUE'. See details.
waldtype	Numeric, indicates which function to use for overall Wald test. 0 (default) = lmtest::waldtest Chi-square, 1 = lmtest::waldtest F, 2 = aod::wald.test

## Details

The Robust Effect Size Index (RESI) is an effect size measure based on M-estimators. This function is called by `resi` a specified number of times to form bootstrapped confidence intervals. Called by itself, this function will only calculate point estimates.

The RESI, denoted as  $S$ , is applicable across many model types. It is a unitless index and can be easily be compared across models. The RESI can also be converted to Cohen's  $d$  ([S2d](#)) under model homoskedasticity.

The RESI is related to the non-centrality parameter of the test statistic. The RESI estimate is consistent for all four (Chi-square, F, T, and Z) types of statistics used. The Chi-square and F-based calculations rely on asymptotic theory, so they may be biased in small samples. When possible, the T and Z statistics are used. There are two formulas for both the T and Z statistic conversion. The first (default, unbiased = TRUE) are based on solving the expected value of the T or Z statistic for the RESI. The alternative is based on squaring the T or Z statistic and using the F or Chi-square statistic conversion. Both of these methods are consistent, but the alternative exhibits a notable amount of finite sample bias. The alternative may be appealing because its absolute value will be equal to the RESI based on the F or Chi-square statistic. The RESI based on the Chi-Square and F statistics is always greater than or equal to 0. The type of statistic used is listed with the output. See [f2S](#), [chisq2S](#), [t2S](#), and [z2S](#) for more details on the formulas.

For GEE (`geeglm`) models, a longitudinal RESI (L-RESI) and a cross-sectional, per-measurement RESI (CS-RESI) is estimated. The longitudinal RESI takes the specified clustering into account, while the cross-sectional RESI is estimated using a model where each measurement is its own cluster.

## Value

Returns a list containing RESI point estimates

## Methods (by class)

- `resi_pe(default)`: RESI point estimation
- `resi_pe(glm)`: RESI point estimation for generalized linear models
- `resi_pe(lm)`: RESI point estimation for linear models
- `resi_pe(nls)`: RESI point estimation for nonlinear least squares models

- `resi_pe(survreg)`: RESI point estimation for `survreg`
- `resi_pe(coxph)`: RESI point estimation for `coxph` models
- `resi_pe(hurdle)`: RESI point estimation for hurdle models
- `resi_pe(zeroinfl)`: RESI point estimation for `zeroinfl` models
- `resi_pe(geeglm)`: RESI point estimation for `geeglm` object
- `resi_pe(gee)`: RESI point estimation for `gee` object
- `resi_pe(lme)`: RESI point estimation for `lme` object
- `resi_pe(lmerMod)`: RESI point estimation for `lmerMod` object

## References

Vandekar S, Tao R, Blume J. A Robust Effect Size Index. *Psychometrika*. 2020 Mar;85(1):232-246. doi: 10.1007/s11336-020-09698-2.

## Examples

```
# This function produces point estimates for the RESI. The resi function will
# provide the same point estimates but adds confidence intervals. See resi for
# more detailed examples.

## resi_pe for a linear model
# fit linear model
mod <- lm(charges ~ region * age + bmi + sex, data = RESI::insurance)
# run resi_pe on the model
resi_pe(mod)

# if you want to have RESI estimates in the coefficient table that are equal in absolute
# value to those in the Anova table (except for those with >1 df and/or included in other
# interaction terms), you can specify unbiased = FALSE to use the alternate conversion.
resi_pe(mod, unbiased = FALSE)
```

---

Rsq2S

*Covert R<sup>2</sup> to S*

---

## Description

Converts  $R^2$ , the partial coefficient of determination, to robust effect size index (S) using the formula from Vandekar, Tao, & Blume (2020).

## Usage

```
Rsq2S(Rsq)
```

## Arguments

Rsq                      Numeric,  $R^2$



**Details**

The formula for the conversion is:

$$S = \sqrt{((-R^2)/(R^2 - 1))}$$

**Value**

Returns an estimate of  $R^2$  based on the RESI

**Examples**

```
# consider a moderate effect size of  $R^2 = 0.1$ 
Rsquared(0.1)
# this corresponds to a RESI of 0.333
```

---

S2d

---

*Convert S to Cohen's d*


---

**Description**

Converts the robust effect size index (S) to Cohen's d using the formula from Vandekar, Tao, & Blume (2020).

**Usage**

```
S2d(S, pi = 0.5)
```

**Arguments**

S                    Numeric, the robust effect size index.  
pi                    Numeric, the sampling proportions.

**Details**

The pi parameter comes from the fact that Cohen's d doesn't account for unequal sample proportions in the population, but S does.

The default is set to a natural value 1/2, which corresponds to a case control design, for example, where sampling proportions always are controlled by the experimenter.

The formula for the conversion is:

$$d = |S * \sqrt{(1/\pi + 1/(1 - \pi))}|$$

**Value**

Returns an estimate of Cohen's d based on the RESI.

**Examples**

```
# fit a simple linear regression with a binary predictor
mod = lm(charges ~ sex, data = RESI::insurance)

# calculate t-value
t = summary(mod)$coefficients[2, "t value"]

# calculate RESI (S)
S = t2S(t, n = 1338, rdf = 1336)

# determine sample proportions
pi = length(which(RESI::insurance[, "sex"]=="male"))/1338

# convert S to Cohen's d
S2d(S = S, pi = pi)
```

---

S2fsq

*Covert S to Cohen's f<sup>2</sup>*


---

**Description**

Converts robust effect size index (S) to Cohen's  $f^2$  (effect size for multiple regression) using the formula from Vandekar, Tao, & Blume (2020).

**Usage**

```
S2fsq(S)
```

**Arguments**

S                      Numeric, the robust effect size index.

**Details**

The formula for the conversion is:

$$f^2 = S^2$$

**Value**

Returns an estimate of Cohen's  $f^2$  based on the RESI

**Examples**

```
# fit a linear regression model with continuous outcome and predictor
mod = lm(charges ~ age, data = RESI::insurance)

# obtain t value for calculating RESI
t = summary(mod)$coefficients[2, "t value"]

# calculate RESI
S = t2S(t, n = 1338, rdf = 1336)

# convert to f^2
S2fsq(S)
```

---

S2Rsq

*Covert S to R^2*


---

**Description**

Converts robust effect size index (S) to R<sup>2</sup>, the partial coefficient of determination, using the formula from Vandekar, Tao, & Blume (2020).

**Usage**

```
S2Rsq(S)
```

**Arguments**

S                      Numeric, the robust effect size index.

**Details**

The formula for the conversion is:

$$R^2 = S^2 / (1 + S^2)$$

**Value**

Returns an estimate of R<sup>2</sup> based on the RESI

**Examples**

```
# fit a simple linear regression with a binary predictor
mod = lm(charges ~ sex, data = RESI::insurance)

# calculate t-value
t = summary(mod)$coefficients[2, "t value"]

# calculate RESI (S)
S = t2S(t, n = 1338, rdf = 1336)
```

```
# convert S to R^2
S2Rsq(S)
```

---

summary.resi

*Summary method for resi objects*


---

### Description

After running the `resi` function on a fitted model, this function can be used to print the coefficients table component. If the `resi` function was run with the `'store.boot = TRUE'` option to store the full matrix of bootstrapped estimates, the user can specify a different alpha level for this function's confidence intervals.

### Usage

```
## S3 method for class 'resi'
summary(object, alpha = NULL, ...)
```

### Arguments

<code>object</code>	an object resulting from <code>resi</code> function
<code>alpha</code>	an optional new specification for the confidence level. Can be vector-valued
<code>...</code>	ignored

### Value

Returns a `'summary_resi'` object containing the computed coefficients table

### Examples

```
# fit a model
mod = lm(charges ~ bmi + sex, data = RESI::insurance)

# run resi with the store.boot = TRUE option
resi_obj = resi(mod, nboot = 100, store.boot = TRUE, alpha = 0.01)

# run summary, specifying a different alpha level if desired
summary(resi_obj, alpha = 0.05)
```

---

t2S	<i>Compute the robust effect size index estimate from t statistic (default)</i>
-----	---------------------------------------------------------------------------------

---

### Description

This function computes the robust effect size index from Vandekar, Tao, & Blume (2020). Vector arguments are accepted. If different length arguments are passed they are dealt with in the usual way of R.

### Usage

```
t2S(t, rdf, n, unbiased = TRUE)
```

### Arguments

t	The t statistic for the parameter of interest.
rdf	Model residual degrees of freedom/degrees of freedom of the t statistic.
n	Number of independent samples.
unbiased	Logical, whether to use unbiased or alternative estimator. See details.

### Details

This function computes S, the RESI, from a t statistic. The formula for the unbiased estimator (default) is derived by solving the expected value of the t statistic for S. It is unbiased and consistent.

The formula for the unbiased conversion is:

$$S = (t * \sqrt{2}) * \Gamma(rdf/2) / (\sqrt{(n * rdf)} * \Gamma((rdf - 1)/2))$$

The formula for the alternative estimator is derived by squaring the t statistic and using the `f2S` formula. This estimator may be appealing for its intuitive relationship to the F statistic; the absolute value of RESI estimates using this formula will be equal to a RESI estimate using an F statistic for the same model. However, this estimator does have finite sample bias, which is an important consideration for the coverage of the bootstrapping that `resi` uses.

The formula for the alternative conversion is:

$$\sqrt{\max(0, (t^2 * (rdf - 2) / (rdf - 1) / rdf))}$$

### Value

Returns a scalar or vector argument of the the robust effect size index estimate.

### Examples

```
# to obtain t values, first fit a lm
mod = lm(charges ~ region * age + bmi + sex, data = RESI::insurance)
# run lmtest::coefstest to get t values, using a robust variance-covariance formula
ts = lmtest::coefstest(mod, vcov. = sandwich::vcovHC)[, 't value']
```

```
# get RESI estimates using unbiased estimator
t2S(ts, n = nrow(RESI::insurance), rdf = mod$df.residual)

# get RESI estimates using alternative estimator
t2S(ts, n = nrow(RESI::insurance), rdf = mod$df.residual, unbiased = FALSE)
```

---

z2S

---

*Compute the robust effect size index estimate from Z statistic*


---

### Description

This function computes the robust effect size index from Vandekar, Tao, & Blume (2020). Vector arguments are accepted. If different length arguments are passed they are dealt with in the usual way of R.

### Usage

```
z2S(z, n, unbiased = TRUE)
```

### Arguments

z	The Z statistic for the parameter of interest.
n	Number of independent samples.
unbiased	Logical, whether to use unbiased or alternative estimator. See details.

### Details

This function computes S, the RESI, from a Z statistic. The formula for the unbiased estimator (default) is derived by solving the expected value of the Z statistic for S. It is unbiased and consistent.

The formula for the unbiased conversion is:

$$S = Z / \sqrt{(n)}$$

The formula for the alternative estimator is derived by squaring the Z statistic and using the [chisq2S](#) formula. This estimator may be appealing for its intuitive relationship to the Chi-square statistic; the absolute value of RESI estimates using this formula will be equal to a RESI estimate using a Chi-square statistic for the same model. However, this estimator does have finite sample bias, which is an important consideration for the coverage of the bootstrapping that `resi` uses.

The formula for the alternative conversion is:

$$\sqrt{\max(0, (Z^2 - 1)/n)} * \text{sign}(Z)$$

### Value

Returns a scalar or vector argument of the the robust effect size index estimate.

**Examples**

```
# to obtain example z values, first fit a glm
mod = glm(charges ~ region * age + bmi + sex, data = RESI::insurance)
# run coeftest to get z values using a robust variance-covariance function
zs = lmtest::coeftest(mod, vcov. = sandwich::vcovHC)[,'z value']

# get RESI estimates using unbiased estimator
z2S(zs, n = nrow(RESI::insurance))

# get RESI estimates usng alternative estimator
z2S(zs, n = nrow(RESI::insurance), unbiased = FALSE)
```

# Index

## \* RESI functions

resi, 11

## \* datasets

depression, 5

insurance, 8

Anova, 16

anova.resi, 2

axis, 10

boot, 16

chisq2S, 3, 16, 17, 23, 30

d2S, 4

depression, 5

f2S, 6, 16, 17, 23, 29

fsq2S, 7

ggplot.resi, 7

insurance, 8

lines, 10

omnibus, 9

plot, 10

plot.resi, 9

resi, 2, 9, 11, 19, 23, 28

resi\_pe, 17, 19

Rsq2S, 24

S2d, 16, 23, 25

S2fsq, 26

S2Rsq, 27

summary.resi, 28

t2S, 16, 17, 23, 29

vcovHC, 17

z2S, 16, 17, 23, 30