

Package ‘effectsize’

January 18, 2021

Type Package

Title Indices of Effect Size and Standardized Parameters

Version 0.4.3

Maintainer Mattan S. Ben-Shachar <matanshm@post.bgu.ac.il>

License GPL-3

URL <https://easystats.github.io/effectsize/>

BugReports <https://github.com/easystats/effectsize/issues/>

Description Provide utilities to work with indices of effect size and standardized parameters for a wide variety of models (see support list of insight; Lüdtke, Waggoner & Makowski (2019) <doi:10.21105/joss.01412>), allowing computation and conversion of indices such as Cohen's d, r, odds, etc.

Depends R (>= 3.6)

Imports bayestestR (>= 0.8.0), insight (>= 0.12.0), parameters (>= 0.11.0), stats, utils

Suggests afex, BayesFactor, boot, brms, car, correlation, covr, dplyr, emmeans, gamm4, ggplot2, knitr, lavaan, lm.beta, lme4, lmerTest, MASS, mediation, modelbased, MuMIn, performance, pscl, rlang, rmarkdown, rstan, rstanarm, rstantools, see, testthat, tidy

RoxygenNote 7.1.1

Language en-GB

VignetteBuilder knitr

Encoding UTF-8

Config/testthat/edition 3

Config/testthat/parallel true

NeedsCompilation no

Author Mattan S. Ben-Shachar [aut, cre]
(<<https://orcid.org/0000-0002-4287-4801>>),
Dominique Makowski [aut] (<<https://orcid.org/0000-0001-5375-9967>>),
Daniel Lüdtke [aut] (<<https://orcid.org/0000-0002-8895-3206>>),

Indrajeet Patil [ctb] (<<https://orcid.org/0000-0003-1995-6531>>,
 @patilindrajeets),
 Ken Kelley [ctb],
 David Stanley [ctb]

Repository CRAN

Date/Publication 2021-01-18 17:10:06 UTC

R topics documented:

adjust	3
change_scale	5
chisq_to_phi	6
cohens_d	8
d_to_common_language	11
d_to_r	12
effectsize	14
effectsize-CIs	16
equivalence_test.effectsize_table	17
eta2_to_f2	19
eta_squared	20
format_standardize	26
F_to_eta2	27
hardlyworking	30
interpret	31
interpret_bf	31
interpret_d	33
interpret_direction	34
interpret_ess	35
interpret_gfi	36
interpret_oddsratio	38
interpret_omega_squared	39
interpret_p	40
interpret_parameters	41
interpret_r	42
interpret_r2	44
interpret_rope	45
is_effectsize_name	46
normalize	46
oddsratio_to_riskratio	48
odds_to_probs	49
phi	50
plot.effectsize_table	53
ranktransform	54
rank_biserial	55
rules	58
sd_pooled	59
standardize	59

<i>adjust</i>	3
standardize_info	62
standardize_parameters	64
t_to_d	67
Index	71

<code>adjust</code>	<i>Adjust data for the effect of other variable(s)</i>
---------------------	--

Description

This function can be used to adjust the data for the effect of other variables present in the dataset. It is based on an underlying fitting of regressions models, allowing for quite some flexibility, such as including factors as random effects in mixed models (multilevel partialization), continuous variables as smooth terms in general additive models (non-linear partialization) and/or fitting these models under a Bayesian framework. The values returned by this function are the residuals of the regression models. Note that a regular correlation between two "adjusted" variables is equivalent to the partial correlation between them.

Usage

```
adjust(
  data,
  effect = NULL,
  select = NULL,
  exclude = NULL,
  multilevel = FALSE,
  additive = FALSE,
  bayesian = FALSE,
  keep_intercept = FALSE
)

data_adjust(
  data,
  effect = NULL,
  select = NULL,
  exclude = NULL,
  multilevel = FALSE,
  additive = FALSE,
  bayesian = FALSE,
  keep_intercept = FALSE
)
```

Arguments

<code>data</code>	A dataframe.
<code>effect</code>	Character vector of column names to be adjusted for (regressed out). If NULL (the default), all variables will be selected.

select	Character vector of column names. If NULL (the default), all variables will be selected.
exclude	Character vector of column names to be excluded from selection.
multilevel	If TRUE, the factors are included as random factors. Else, if FALSE (default), they are included as fixed effects in the simple regression model.
additive	If TRUE, continuous variables are included as smooth terms in additive models. The goal is to regress-out potential non-linear effects.
bayesian	If TRUE, the models are fitted under the Bayesian framework using <code>rstanarm</code> .
keep_intercept	If FALSE (default), the intercept of the model is re-added. This avoids the centering around 0 that happens by default when regressing out another variable (see the examples below for a visual representation of this).

Value

A data frame comparable to `data`, with adjusted variables.

Examples

```
adjusted_all <- adjust(attitude)
head(adjusted_all)
adjusted_one <- adjust(attitude, effect = "complaints", select = "rating")
head(adjusted_one)

## Not run:
adjust(attitude, effect = "complaints", select = "rating", bayesian = TRUE)
adjust(attitude, effect = "complaints", select = "rating", additive = TRUE)
attitude$complaints_LMH <- cut(attitude$complaints, 3)
adjust(attitude, effect = "complaints_LMH", select = "rating", multilevel = TRUE)

## End(Not run)

if(require("bayestestR") && require("MASS")){
# Generate data
data <- bayestestR::simulate_correlation(n=100, r=0.7)
data$V2 <- (5 * data$V2) + 20 # Add intercept

# Adjust
adjusted <- adjust(data, effect="V1", select="V2")
adjusted_icpt <- adjust(data, effect="V1", select="V2", keep_intercept=TRUE)

# Visualize
plot(data$V1, data$V2, pch = 19, col = "blue",
      ylim=c(min(adjusted$V2), max(data$V2)),
      main = "Original (blue), adjusted (green), and adjusted - intercept kept (red) data")
abline(lm(V2 ~ V1, data = data), col = "blue")
points(adjusted$V1, adjusted$V2, pch = 19, col = "green")
abline(lm(V2 ~ V1, data = adjusted), col = "green")
points(adjusted_icpt$V1, adjusted_icpt$V2, pch = 19, col = "red")
abline(lm(V2 ~ V1, data = adjusted_icpt), col = "red")
}
```

change_scale	<i>Rescale a numeric variable</i>
--------------	-----------------------------------

Description

Rescale a numeric variable to a new range.

Usage

```
change_scale(x, ...)  
  
## S3 method for class 'numeric'  
change_scale(x, to = c(0, 100), range = NULL, verbose = TRUE, ...)  
  
## S3 method for class 'grouped_df'  
change_scale(  
  x,  
  select = NULL,  
  exclude = NULL,  
  to = c(0, 100),  
  range = NULL,  
  ...  
)  
  
## S3 method for class 'data.frame'  
change_scale(  
  x,  
  select = NULL,  
  exclude = NULL,  
  to = c(0, 100),  
  range = NULL,  
  ...  
)
```

Arguments

x	Object.
...	Arguments passed to or from other methods.
to	New range of values of the data after rescaling.
range	Initial (old) range of values. If NULL, will take the range of data.
verbose	Toggle warnings and messages on or off.
select	Character vector of column names. If NULL (the default), all variables will be selected.
exclude	Character vector of column names to be excluded from selection.

Value

A rescaled object.

See Also

`normalize()` `standardize()` `ranktransform()`

Other transform utilities: `normalize()`, `ranktransform()`, `standardize()`

Examples

```
change_scale(c(0, 1, 5, -5, -2))
change_scale(c(0, 1, 5, -5, -2), to = c(-5, 5))

head(change_scale(trees))
```

chisq_to_phi

Conversion Chi-Squared to Phi or Cramer's V

Description

Convert between Chi square, (χ^2), Cramer's V, phi (ϕ) and Cohen's w for contingency tables or goodness of fit.

Usage

```
chisq_to_phi(chisq, n, nrow, ncol, ci = 0.95, adjust = FALSE, ...)
chisq_to_cohens_w(chisq, n, nrow, ncol, ci = 0.95, adjust = FALSE, ...)
chisq_to_cramers_v(chisq, n, nrow, ncol, ci = 0.95, adjust = FALSE, ...)
phi_to_chisq(phi, n, ...)
```

Arguments

chisq	The Chi-squared statistic.
n	Sample size.
nrow, ncol	The number of rows/columns in the contingency table (ignored for Phi when <code>adjust=FALSE</code> and <code>CI=NULL</code>).
ci	Confidence Interval (CI) level
adjust	Should the effect size be bias-corrected? Defaults to FALSE.
...	Arguments passed to or from other methods.
phi	The Phi statistic.

Details

These functions use the following formulae:

$$\phi = \sqrt{\chi^2/n}$$

$$Cramer'sV = \phi / \sqrt{\min(nrow, ncol) - 1}$$

For adjusted versions, see Bergsma, 2013.

Value

A data frame with the effect size(s) between 0-1, and confidence interval(s). See `cramers_v()`.

Confidence Intervals

Unless stated otherwise, confidence intervals are estimated using the Noncentrality parameter method; These methods searches for a the best non-central parameters (ncps) of the noncentral t-, F- or Chi-squared distribution for the desired tail-probabilities, and then convert these ncps to the corresponding effect sizes. (See full [effectsize-CIs](#) for more.)

CI Contains Zero

For positive only effect sizes (Eta squared, Cramer's V, etc.; Effect sizes associated with Chi-squared and F distributions), special care should be taken when interpreting CIs with a lower bound equal to 0, and even more care should be taken when the *upper* bound is equal to 0 (Steiger, 2004; Morey et al., 2016). For example:

```
eta_squared(aov(mpg ~ factor(gear) + factor(cyl), mtcars[1:7, ]))
```

```
## Parameter | Eta2 (partial) | 90% CI
## -----|-----|-----
## factor(gear) | 0.58 | [0.00, 0.84]
## factor(cyl) | 0.46 | [0.00, 0.78]
```

Note

Cohen's *w* is equivalent to *Phi*.

References

- Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement*, 61(4), 532-574.
- Bergsma, W. (2013). A bias-correction for Cramer's V and Tschuprow's T. *Journal of the Korean Statistical Society*, 42(3), 323-328.

See Also

Other effect size from test statistic: [F_to_eta2\(\)](#), [t_to_d\(\)](#)

Examples

```
contingency_table <- as.table(rbind(c(762, 327, 468), c(484, 239, 477), c(484, 239, 477)))

chisq.test(contingency_table)
#
#      Pearson's Chi-squared test
#
# data:  ctab
# X-squared = 41.234, df = 4, p-value = 2.405e-08

chisq_to_phi(41.234,
  n = sum(contingency_table),
  nrow = nrow(contingency_table),
  ncol = ncol(contingency_table)
)
chisq_to_cramers_v(41.234,
  n = sum(contingency_table),
  nrow = nrow(contingency_table),
  ncol = ncol(contingency_table)
)
```

cohens_d

Effect size for differences

Description

Compute effect size indices for standardized differences: Cohen's *d*, Hedges' *g* and Glass's *delta*. (This function returns the **population** estimate.)

Both Cohen's *d* and Hedges' *g* are the estimated the standardized difference between the means of two populations. Hedges' *g* provides a bias correction to Cohen's *d* for small sample sizes. For sample sizes > 20, the results for both statistics are roughly equivalent. Glass's *delta* is appropriate when the standard deviations are significantly different between the populations, as it uses only the *second* group's standard deviation.

Usage

```
cohens_d(
  x,
  y = NULL,
  data = NULL,
  pooled_sd = TRUE,
  mu = 0,
  paired = FALSE,
```



```

    ci = 0.95,
    verbose = TRUE,
    ...,
    correction
)

hedges_g(
  x,
  y = NULL,
  data = NULL,
  correction = 1,
  pooled_sd = TRUE,
  mu = 0,
  paired = FALSE,
  ci = 0.95,
  verbose = TRUE,
  ...
)

glass_delta(
  x,
  y = NULL,
  data = NULL,
  mu = 0,
  ci = 0.95,
  iterations = 200,
  verbose = TRUE,
  ...,
  correction
)

```

Arguments

x	A formula, a numeric vector, or a character name of one in data.
y	A numeric vector, a grouping (character / factor) vector, a or a character name of one in data. Ignored if x is a formula.
data	An optional data frame containing the variables.
pooled_sd	If TRUE (default), a <code>sd_pooled()</code> is used (assuming equal variance). Else the mean SD from both groups is used instead.
mu	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
paired	If TRUE, the values of x and y are considered as paired. This produces an effect size that is equivalent to the one-sample effect size on x - y.
ci	Confidence Interval (CI) level
verbose	Toggle warnings and messages on or off.
...	Arguments passed to or from other methods.

correction	Type of small sample bias correction to apply to produce Hedges' <i>g</i> . Can be 1 for Hedges and Olkin's original correction (default) or 2 for Hunter and Schmidt's correction (see McGrath & Meyer, 2006).
iterations	The number of bootstrap replicates for computing confidence intervals. Only applies when <i>ci</i> is not NULL.

Details

Confidence Intervals for Glass' *delta*:

Confidence Intervals for Glass' *delta* are estimated using the bootstrap method.

Value

A data frame with the effect size (*Cohens_d*, *Hedges_g*, *Glass_delta*) and their CIs (*CI_low* and *CI_high*).

Confidence Intervals

Unless stated otherwise, confidence intervals are estimated using the Noncentrality parameter method; These methods searches for a the best non-central parameters (ncps) of the noncentral t-, F- or Chi-squared distribution for the desired tail-probabilities, and then convert these ncps to the corresponding effect sizes. (See full [effectsize-CIs](#) for more.)

Note

The indices here give the population estimated standardized difference. Some statistical packages give the sample estimate instead (without applying Bessel's correction).

References

- Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd Ed.). New York: Routledge.
- Hedges, L. V. & Olkin, I. (1985). *Statistical methods for meta-analysis*. Orlando, FL: Academic Press.
- Hunter, J. E., & Schmidt, F. L. (2004). *Methods of meta-analysis: Correcting error and bias in research findings*. Sage.
- McGrath, R. E., & Meyer, G. J. (2006). When effect sizes disagree: the case of *r* and *d*. *Psychological methods*, 11(4), 386.

See Also

[d_to_common_language\(\)](#) [sd_pooled\(\)](#)

Other effect size indices: [effectsize\(\)](#), [eta_squared\(\)](#), [phi\(\)](#), [rank_biserial\(\)](#), [standardize_parameters\(\)](#)

Examples

```

cohens_d(sleep$extra, sleep$group)
hedges_g("extra", "group", data = sleep)

cohens_d(sleep$extra[sleep$group==1], sleep$extra[sleep$group==2], paired = TRUE)

cohens_d(mpg ~ am, data = mtcars)
cohens_d(mpg ~ am, data = mtcars, pooled_sd = FALSE)
cohens_d(mpg ~ am, data = mtcars, mu = -5)
hedges_g(mpg ~ am, data = mtcars)
if (require(boot)) glass_delta(mpg ~ am, data = mtcars)

print(cohens_d(mpg ~ am, data = mtcars), append_CL = TRUE)

```

d_to_common_language *Convert Standardized Mean Difference to Common Language Effect Sizes*

Description

Convert Standardized Mean Difference to Common Language Effect Sizes

Usage

```

d_to_common_language(d)

convert_d_to_common_language(d)

```

Arguments

d Standardized difference value (Cohen's d).

Details

This function use the following formulae:

$$Cohen's U_3 = \Phi(d)$$

$$Overlap = 2 \times \Phi(-|d|/2)$$

$$Pr(superiority) = \Phi(d/\sqrt{2})$$

Value

A list of Cohen's U3, Overlap, Probability of superiority.

Note

These calculations assume that the populations have equal variance and are normally distributed.

References

- Cohen, J. (1977). Statistical power analysis for the behavioral sciences. New York: Routledge.
- Reiser, B., & Faraggi, D. (1999). Confidence intervals for the overlapping coefficient: the normal equal variance case. *Journal of the Royal Statistical Society*, 48(3), 413-418.
- Ruscio, J. (2008). A probability-based measure of effect size: robustness to base rates and other factors. *Psychological methods*, 13(1), 19-30.

See Also

[cohens_d\(\)](#)

Other convert between effect sizes: [d_to_r\(\)](#), [eta2_to_f2\(\)](#), [odds_to_probs\(\)](#), [oddsratio_to_riskratio\(\)](#)

d_to_r

Convert between d, r and Odds ratio

Description

Enables a conversion between different indices of effect size, such as standardized difference (Cohen's d), correlation r or (log) odds ratios.

Usage

`d_to_r(d, ...)`

`r_to_d(r, ...)`

`convert_d_to_r(d, ...)`

`convert_r_to_d(r, ...)`

`oddsratio_to_d(OR, log = FALSE, ...)`

`convert_oddsratio_to_d(OR, log = FALSE, ...)`

`logoddsratio_to_d(OR, log = TRUE, ...)`

`d_to_oddsratio(d, log = FALSE, ...)`

```

convert_d_to_oddsratio(d, log = FALSE, ...)
oddsratio_to_r(OR, log = FALSE, ...)
convert_oddsratio_to_r(OR, log = FALSE, ...)
logoddsratio_to_r(OR, log = TRUE, ...)
r_to_oddsratio(r, log = FALSE, ...)
convert_r_to_oddsratio(r, log = FALSE, ...)

```

Arguments

d	Standardized difference value (Cohen's d).
...	Arguments passed to or from other methods.
r	Correlation coefficient r.
OR	<i>Odds ratio</i> values in vector or data frame.
log	Take in or output the log of the ratio (such as in logistic models).

Details

Conversions between *OR* and *r* is done through these formulae.

- *d to r*: $d = \frac{2*r}{\sqrt{1-r^2}}$
- *r to d*: $r = \frac{d}{\sqrt{d^2+4}}$
- *OR to d*: $d = \frac{\log(OR) \times \sqrt{3}}{\pi}$
- *d to OR*: $\log(OR) = d * \frac{\pi}{\sqrt{3}}$

The conversion from *d* to *r* assumes equally sized groups. The resulting *r* is also called the binomial effect size display (BESD; Rosenthal et al., 1982).

Value

Converted index.

References

- Sánchez-Meca, J., Marín-Martínez, F., & Chacón-Moscoso, S. (2003). Effect-size indices for dichotomized outcomes in meta-analysis. *Psychological methods*, 8(4), 448.
- Borenstein, M., Hedges, L. V., Higgins, J. P. T., & Rothstein, H. R. (2009). Converting among effect sizes. *Introduction to meta-analysis*, 45-49.
- Rosenthal, R., & Rubin, D. B. (1982). A simple, general purpose display of magnitude of experimental effect. *Journal of educational psychology*, 74(2), 166.

See Also

Other convert between effect sizes: [d_to_common_language\(\)](#), [eta2_to_f2\(\)](#), [odds_to_probs\(\)](#), [oddsratio_to_riskratio\(\)](#)

Examples

```
r_to_d(0.5)
d_to_oddsratio(1.154701)
oddsratio_to_r(8.120534)

d_to_r(1)
r_to_oddsratio(0.4472136, log = TRUE)
oddsratio_to_d(1.813799, log = TRUE)
```

effectsize

Effect Size

Description

This function tries to return the best effect-size measure for the provided input model. See details.

Usage

```
effectsize(model, ...)

## S3 method for class 'BFBayesFactor'
effectsize(model, type = NULL, verbose = TRUE, ...)

## S3 method for class 'aov'
effectsize(model, type = NULL, ...)

## S3 method for class 'htest'
effectsize(model, type = NULL, verbose = TRUE, ...)
```

Arguments

model	An object of class <code>htest</code> , or a statistical model. See details.
...	Arguments passed to or from other methods. See details.
type	The effect size of interest. See details.
verbose	Toggle warnings and messages on or off.

Details

- For an object of class `htest`, data is extracted via `insight::get_data()`, and passed to the relevant function according to:
 - A **t-test** depending on type: `"cohens_d"` (default), `"hedges_g"`.

- A **correlation test** returns r .
- A **Chi-squared tests of independence or goodness-of-fit**, depending on type: "cramers_v" (default), "phi" or "cohens_w", "cohens_h", "oddsratio", or "riskratio".
- A **One-way ANOVA test**, depending on type: "eta" (default), "omega" or "epsilon"-squared, "f", or "f2".
- A **McNemar test** returns *Cohen's g*.
- A **Fisher's Exact test** (in the 2x2 case) returns *Odds ratio*.
- A **Wilcoxon test** returns *rank biserial correlation*.
- A **Kruskal-Wallis test** returns *rank Epsilon squared*.
- A **Friedman test** returns *Kendall's W*.
- For an object of class BFBayesFactor, using `bayestestR::describe_posterior()`,
 - A **t-test** returns *Cohen's d*.
 - A **correlation test** returns r .
 - A **contingency table test**, depending on type: "cramers_v" (default), "phi" or "cohens_w", "cohens_h", "oddsratio", or "riskratio".
- Objects of class anova, aov, or aovlist, depending on type: "eta" (default), "omega" or "epsilon"-squared, "f", or "f2".
- Other objects are passed to `standardize_parameters()`.

For statistical models it is recommended to directly use the listed functions, for the full range of options they provide.

Value

A data frame with the effect size (depending on input) and and its CIs (CI_low and CI_high).

See Also

Other effect size indices: `cohens_d()`, `eta_squared()`, `phi()`, `rank_biserial()`, `standardize_parameters()`

Examples

```
## Hypothesis Testing
## -----
contingency_table <- as.table(rbind(c(762, 327, 468), c(484, 239, 477), c(484, 239, 477)))
Xsq <- chisq.test(contingency_table)
effectsize(Xsq)
effectsize(Xsq, type = "phi")

Ts <- t.test(1:10, y = c(7:20))
effectsize(Ts)

Aov <- oneway.test(extra ~ group, data = sleep, var.equal = TRUE)
effectsize(Aov)
effectsize(Aov, type = "omega")

## Bayesian Hypothesis Testing
```

```
## -----

if (require(BayesFactor)) {
  bf1 <- ttestBF(mtcars$mpg[mtcars$am == 1], mtcars$mpg[mtcars$am == 0])
  effectsize(bf1, test = NULL)

  bf2 <- correlationBF(attitude$rating, attitude$complaints)
  effectsize(bf2, test = NULL)

  data(raceDolls)
  bf3 <- contingencyTableBF(raceDolls, sampleType = "poisson", fixedMargin = "cols")
  effectsize(bf3, test = NULL)
  effectsize(bf3, type = "oddsratio", test = NULL)
}

## Models and Anova Tables
## -----
fit <- lm(mpg ~ factor(cyl) * wt + hp, data = mtcars)
effectsize(fit)

anova_table <- anova(fit)
effectsize(anova_table)
effectsize(anova_table, type = "epsilon")
```

effectsize-CIs

Confidence Intervals

Description

More information regarding Confidence Intervals and how they are computed in effectsize.

Confidence Intervals

Unless stated otherwise, confidence intervals are estimated using the Noncentrality parameter method; These methods searches for a the best non-central parameters (ncps) of the noncentral t-, F- or Chi-squared distribution for the desired tail-probabilities, and then convert these ncps to the corresponding effect sizes. (See full [effectsize-CIs](#) for more.)

CI Contains Zero

For positive only effect sizes (Eta squared, Cramer's V, etc.; Effect sizes associated with Chi-squared and F distributions), special care should be taken when interpreting CIs with a lower bound equal to 0, and even more care should be taken when the *upper* bound is equal to 0 (Steiger, 2004; Morey et al., 2016). For example:

```
eta_squared(aov(mpg ~ factor(gear) + factor(cyl), mtcars[1:7, ]))
```



```
## Parameter | Eta2 (partial) | 90% CI
## -----
## factor(gear) | 0.58 | [0.00, 0.84]
## factor(cyl) | 0.46 | [0.00, 0.78]
```

CI Does Not Contain the Estimate

For very large sample sizes, the width of the CI can be smaller than the tolerance of the optimizer, resulting in CIs of width 0. This can also, result in the estimated CIs excluding the point estimate. For example:

```
chisq_to_cramers_v(13223.73, n = 76227, nrow = 6, ncol = 1)
```

```
## Cramer's V | 95% CI
## -----
## 0.19 | [0.20, 0.20]
```

```
t_to_d(80, df_error = 4555555)
```

```
## d | 95% CI
## -----
## 0.07 | [0.08, 0.08]
```

References

- Morey, R. D., Hoekstra, R., Rouder, J. N., Lee, M. D., & Wagenmakers, E. J. (2016). The fallacy of placing confidence in confidence intervals. *Psychonomic bulletin & review*, 23(1), 103-123.
- Steiger, J. H. (2004). Beyond the F test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods*, 9, 164-182.

```
equivalence_test.effectsize_table
Test for Practical Equivalence
```

Description

Perform a **Test for Practical Equivalence** for indices of effect size.

Usage

```
## S3 method for class 'effectsize_table'
equivalence_test(
  x,
  range = "default",
  rule = c("classic", "cet", "bayes"),
  ...
)
```

Arguments

x	An effect size table, such as returned by <code>cohens_d()</code> , <code>eta_squared()</code> , <code>F_to_r()</code> , etc.
range	The range of practical equivalence of an effect. If a single value is provided, the test is done against <code>c(-range, range)</code> . For effect sizes that cannot be negative, the lower bound is set to 0. If "default", will be set to [-.1, .1].
rule	How should acceptance and rejection be decided? See details.
...	Arguments passed to or from other methods.

Details

The CIs used in the equivalence test are the ones in the provided effect size table. For results equivalent (ha!) to those that can be obtained using the TOST approach (e.g., Lakens, 2017), appropriate CIs should be extracted using the function used to make the effect size table (`cohens_d`, `eta_squared`, `F_to_r`, etc). See examples.

The Different Rules:

- "classic" - **the classic method**:
 - If the CI is completely within the ROPE - *Accept H0*
 - Else, if the CI does not contain 0 - *Reject H0*
 - Else - *Undecided*
- "cet" - **conditional equivalence testing**:
 - If the CI does not contain 0 - *Reject H0*
 - Else, If the CI is completely within the ROPE - *Accept H0*
 - Else - *Undecided*
- "bayes" - **The Bayesian approach**, as put forth by Kruschke:
 - If the CI does is completely outside the ROPE - *Reject H0*
 - Else, If the CI is completely within the ROPE - *Accept H0*
 - Else - *Undecided*

Value

A data frame with the results of the equivalence test.

References

- Campbell, H., & Gustafson, P. (2018). Conditional equivalence testing: An alternative remedy for publication bias. *PLOS ONE*, 13(4), e0195145. <https://doi.org/10.1371/journal.pone.0195145>
- Kruschke, J. K. (2014). *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press
- Kruschke, J. K. (2018). Rejecting or accepting parameter values in Bayesian estimation. *Advances in Methods and Practices in Psychological Science*, 1(2), 270-280. doi: 10.1177/2515245918771304
- Lakens, D. (2017). Equivalence Tests: A Practical Primer for t Tests, Correlations, and Meta-Analyses. *Social Psychological and Personality Science*, 8(4), 355–362. <https://doi.org/10.1177/1948550617697177>

See Also

For more details, see [bayestestR::equivalence_test\(\)](#).

Examples

```

model <- aov(mpg ~ factor(am) * factor(cyl), data = mtcars)
es <- eta_squared(model)
equivalence_test(es, range = 0.15)

ds <- t_to_d(
  t = c(0.45, -0.65, 7, -2.2, 2.25),
  df_error = c(675, 525, 2000, 900, 1875),
  ci = 0.9
) # TOST approach
equivalence_test(ds, range = 0.2)

# Can also plot
if (require(see)) plot(equivalence_test(ds, range = 0.2))
if (require(see)) plot(equivalence_test(ds, range = 0.2, rule = "cet"))
if (require(see)) plot(equivalence_test(ds, range = 0.2, rule = "bayes"))

```

eta2_to_f2

Convert between ANOVA effect sizes

Description

Convert between ANOVA effect sizes

Usage

```
eta2_to_f2(es)
```

```
eta2_to_f(es)
```

```
f2_to_eta2(f2)
```

```
f_to_eta2(f)
```

Arguments

es	Any measure of variance explained such as Eta-, Epsilon-, Omega-, or R-Squared, partial or otherwise. See details.
f, f2	Cohen's <i>f</i> or <i>f</i> -squared.

Details

Any measure of variance explained can be converted to a corresponding Cohen's f via:

$$f^2 = \frac{\eta^2}{1 - \eta^2}$$

$$\eta^2 = \frac{f^2}{1 + f^2}$$

If a partial Eta-Squared is used, the resulting Cohen's f is a partial-Cohen's f ; If a less biased estimate of variance explained is used (such as Epsilon- or Omega-Squared), the resulting Cohen's f is likewise a less biased estimate of Cohen's f .

References

- Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd Ed.). New York: Routledge.
- Steiger, J. H. (2004). Beyond the F test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods*, 9, 164-182.

See Also

[eta_squared\(\)](#) for more details.

Other convert between effect sizes: [d_to_common_language\(\)](#), [d_to_r\(\)](#), [odds_to_probs\(\)](#), [oddsratio_to_riskratio\(\)](#)

eta_squared

Effect size for ANOVA

Description

Functions to compute effect size measures for ANOVAs, such as Eta, Omega and Epsilon squared, and Cohen's f (or their partialled versions) for `aov`, `aovlist` and `anova` models. These indices represent an estimate of how much variance in the response variables is accounted for by the explanatory variable(s).

Effect sizes are computed using the sums of squares obtained from `anova(model)` which might not always be appropriate (*Yeah... ANOVAs are hard...*). It is suggested that ANOVA models be fit with `afex` package. See details.

Usage

```
eta_squared(  
  model,  
  partial = TRUE,  
  generalized = FALSE,  
  ci = 0.9,  
  verbose = TRUE,  
  ...  
)  
  
omega_squared(model, partial = TRUE, ci = 0.9, verbose = TRUE, ...)  
  
epsilon_squared(model, partial = TRUE, ci = 0.9, verbose = TRUE, ...)  
  
cohens_f(  
  model,  
  partial = TRUE,  
  ci = 0.9,  
  squared = FALSE,  
  verbose = TRUE,  
  model2 = NULL,  
  ...  
)  
  
cohens_f_squared(  
  model,  
  partial = TRUE,  
  ci = 0.9,  
  squared = TRUE,  
  verbose = TRUE,  
  model2 = NULL,  
  ...  
)  
  
eta_squared_posterior(  
  model,  
  partial = TRUE,  
  generalized = FALSE,  
  ss_function = stats::anova,  
  draws = 500,  
  verbose = TRUE,  
  ...  
)
```

Arguments

model	A model, ANOVA object, or the result of <code>parameters::model_parameters</code> .
partial	If TRUE, return partial indices.

generalized	If TRUE, returns generalized Eta Squared, assuming all variables are manipulated. Can also be a character vector of observed (non-manipulated) variables, in which case generalized Eta Squared is calculated taking these observed variables into account. For <code>afex_aov</code> model, when <code>generalized = TRUE</code> , the observed variables are extracted automatically from the fitted model, if they were provided then.
ci	Confidence Interval (CI) level
verbose	Toggle warnings and messages on or off.
...	For Bayesian models, passed to <code>ss_function</code> . Otherwise ignored.
squared	Return Cohen's f or Cohen's f -squared?
model2	Optional second model for Cohen's f (/squared). If specified, returns the effect size for R-squared-change between the two models.
ss_function	For Bayesian models, the function used to extract sum-of-squares. Uses <code>anova()</code> by default, but can also be <code>car::Anova()</code> for simple linear models.
draws	For Bayesian models, an integer indicating the number of draws from the posterior predictive distribution to return. Larger numbers take longer to run, but provide estimates that are more stable.

Details

For `aov` and `aovlist` models, the effect sizes are computed directly with Sums-of-Squares (for `mlm` / `maov` models, effect sizes are computed for each response separately). For all other model, the model is passed to `anova()`, and effect sizes are approximated via test statistic conversion (see `F_to_eta2()` for more details.)

Type of Sums of Squares:

The sums of squares (or F statistics) used for the computation of the effect sizes is based on those returned by `anova(model)` (whatever those may be - for `aov` and `aovlist` these are *type-1* sums of squares; for `merMod` these are *type-3* sums of squares). Make sure these are the sums of squares you are interested in; You might want to pass the result of `car::Anova(mode, type = 3)`, or use the `afex` package to fit ANOVA models.

It is generally recommended to fit models with `contr.sum` *factor weights* and *centered covariates*, for sensible results. See examples and the `afex` package.

Un-Biased Estimate of Eta:

Both **Omega** and **Epsilon** are unbiased estimators of the population's **Eta**, which is especially important is small samples. But which to choose?

Though Omega is the more popular choice (Albers & Lakens, 2018), Epsilon is analogous to adjusted R² (Allen, 2017, p. 382), and has been found to be less biased (Carroll & Nordholm, 1975).

(Note that for ω_p^2 and ϵ_p^2 it is possible to compute a negative number; even though this doesn't make any practical sense, it is recommended to report the negative number and not a 0.)

Cohen's f:

Cohen's f can take on values between zero, when the population means are all equal, and an indefinitely large number as standard deviation of means increases relative to the average standard deviation within each group.

When comparing two models in a sequential regression analysis, Cohen's f for R-square change is the ratio between the increase in R-square and the percent of unexplained variance.

Cohen has suggested that the values of 0.10, 0.25, and 0.40 represent small, medium, and large effect sizes, respectively.

Eta Squared from Posterior Predictive Distribution:

For Bayesian models (fit with `brms` or `rstanarm`), `eta_squared_posterior()` simulates data from the posterior predictive distribution (ppd) and for each simulation the Eta Squared is computed for the model's fixed effects. This means that the returned values are the population level effect size as implied by the posterior model (and not the effect size in the sample data). See [rstantools::posterior_predict\(\)](#) for more info.

Value

A data frame with the effect size(s) between 0-1 (Eta2, Epsilon2, Omega2, Cohens_f or Cohens_f2, possibly with the `partial` or `generalized` suffix), and their CIs (`CI_low` and `CI_high`).

For `eta_squared_posterior()`, a data frame containing the ppd of the Eta squared for each fixed effect, which can then be passed to [bayestestR::describe_posterior\(\)](#) for summary stats.

A data frame containing the effect size values and their confidence intervals.

Confidence Intervals

Unless stated otherwise, confidence intervals are estimated using the Noncentrality parameter method; These methods searches for a the best non-central parameters (ncps) of the noncentral t-, F- or Chi-squared distribution for the desired tail-probabilities, and then convert these ncps to the corresponding effect sizes. (See full [effectsize-CIs](#) for more.)

CI Contains Zero

For positive only effect sizes (Eta squared, Cramer's V, etc.; Effect sizes associated with Chi-squared and F distributions), special care should be taken when interpreting CIs with a lower bound equal to 0, and even more care should be taken when the *upper* bound is equal to 0 (Steiger, 2004; Morey et al., 2016). For example:

```
eta_squared(aov(mpg ~ factor(gear) + factor(cyl), mtcars[1:7, ]))
```

```
## Parameter | Eta2 (partial) | 90% CI
## -----|-----|-----
## factor(gear) | 0.58 | [0.00, 0.84]
## factor(cyl) | 0.46 | [0.00, 0.78]
```

References

- Albers, C., & Lakens, D. (2018). When power analyses based on pilot data are biased: Inaccurate effect size estimators and follow-up bias. *Journal of experimental social psychology*, 74, 187-195.
- Allen, R. (2017). *Statistics and Experimental Design for Psychologists: A Model Comparison Approach*. World Scientific Publishing Company.
- Carroll, R. M., & Nordholm, L. A. (1975). Sampling Characteristics of Kelley's epsilon and Hays' omega. *Educational and Psychological Measurement*, 35(3), 541-554.
- Kelley, T. (1935) An unbiased correlation ratio measure. *Proceedings of the National Academy of Sciences*. 21(9). 554-559.
- Olejnik, S., & Algina, J. (2003). Generalized eta and omega squared statistics: measures of effect size for some common research designs. *Psychological methods*, 8(4), 434.
- Steiger, J. H. (2004). Beyond the F test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods*, 9, 164-182.

See Also

[F_to_eta2\(\)](#)

Other effect size indices: [cohens_d\(\)](#), [effectsize\(\)](#), [phi\(\)](#), [rank_biserial\(\)](#), [standardize_parameters\(\)](#)

Examples

```
library(effectsize)
mtcars$am_f <- factor(mtcars$am)
mtcars$cyl_f <- factor(mtcars$cyl)

model <- aov(mpg ~ am_f * cyl_f, data = mtcars)

eta_squared(model)
eta_squared(model, generalized = "cyl_f")
omega_squared(model)
epsilon_squared(model)
cohens_f(model)
(etas <- eta_squared(model, partial = FALSE))

if (require(see)) plot(etas)

model0 <- aov(mpg ~ am_f + cyl_f, data = mtcars) # no interaction
cohens_f_squared(model0, model2 = model)

# Recommended: Type-3 effect sizes + effects coding
# -----
if (require(car, quietly = TRUE)) {
  contrasts(mtcars$am_f) <- contr.sum
  contrasts(mtcars$cyl_f) <- contr.sum
}
```



```

model <- aov(mpg ~ am_f * cyl_f, data = mtcars)
model_anova <- car::Anova(model, type = 3)

eta_squared(model_anova)
}

# afex takes care of both type-3 effects and effects coding:
if (require(afex)) {
  data(obk.long, package = "afex")
  model <- aov_car(value ~ treatment * gender + Error(id / (phase)),
    data = obk.long, observed = "gender"
  )
  eta_squared(model)
  epsilon_squared(model)
  omega_squared(model)
  eta_squared(model, partial = FALSE)
  epsilon_squared(model, partial = FALSE)
  omega_squared(model, partial = FALSE)
  eta_squared(model, generalized = TRUE) # observed vars are pulled from the afex model.
}

## Approx. effect sizes for mixed models
## -----
if (require(lmerTest, quietly = TRUE)) {
  model <- lmer(mpg ~ am_f * cyl_f + (1 | vs), data = mtcars)
  omega_squared(model)
}

## Bayesian Models (PPD)
## -----
## Not run:
if (require(rstanarm) && require(bayestestR) && require(car)) {
  fit_bayes <- stan_glm(mpg ~ factor(cyl) * wt + qsec,
    data = mtcars,
    family = gaussian(),
    refresh = 0
  )

  es <- eta_squared_posterior(fit_bayes,
    ss_function = car::Anova, type = 3
  )
  bayestestR::describe_posterior(es)

# compare to:
fit_freq <- lm(mpg ~ factor(cyl) * wt + qsec,
  data = mtcars

```

```
)  
aov_table <- car::Anova(fit_freq, type = 3)  
eta_squared(aov_table)  
}  
  
## End(Not run)
```

format_standardize *Transform a standardized vector into character*

Description

Transform a standardized vector into character, e.g., `c("-1 SD", "Mean", "+1 SD")`.

Usage

```
format_standardize(x, reference = x, robust = FALSE, digits = NULL, ...)
```

Arguments

<code>x</code>	A standardized numeric vector.
<code>reference</code>	The reference vector from which to compute the mean and SD.
<code>robust</code>	Logical, if TRUE, centering is done by subtracting the median from the variables and dividing it by the median absolute deviation (MAD). If FALSE, variables are standardized by subtracting the mean and dividing it by the standard deviation (SD).
<code>digits</code>	Number of significant digits.
<code>...</code>	Arguments passed to or from other methods.

Examples

```
format_standardize(c(-1, 0, 1))  
format_standardize(c(-1, 0, 1, 2), reference = rnorm(1000))  
format_standardize(c(-1, 0, 1, 2), reference = rnorm(1000), robust = TRUE)
```

F_to_eta2	<i>Convert test statistics (F, t) to indices of partial variance explained (partial Eta / Omega / Epsilon squared and Cohen's f)</i>
-----------	---

Description

These functions are convenience functions to convert F and t test statistics to **partial** Eta squared, (η_p^2), Omega squared (ω_p^2), Epsilon squared (ϵ_p^2 ; an alias for the adjusted Eta squared) and Cohen's f. These are useful in cases where the various Sum of Squares and Mean Squares are not easily available or their computation is not straightforward (e.g., in liner mixed models, contrasts, etc.). For test statistics derived from lm and aov models, these functions give exact results. For all other cases, they return close approximations.

See [Effect Size from Test Statistics vignette](#).

Usage

```
F_to_eta2(f, df, df_error, ci = 0.9, ...)
t_to_eta2(t, df_error, ci = 0.9, ...)

F_to_epsilon2(f, df, df_error, ci = 0.9, ...)
t_to_epsilon2(t, df_error, ci = 0.9, ...)

F_to_eta2_adj(f, df, df_error, ci = 0.9, ...)
t_to_eta2_adj(t, df_error, ci = 0.9, ...)

F_to_omega2(f, df, df_error, ci = 0.9, ...)
t_to_omega2(t, df_error, ci = 0.9, ...)

F_to_f(f, df, df_error, ci = 0.9, squared = FALSE, ...)
t_to_f(t, df_error, ci = 0.9, squared = FALSE, ...)

F_to_f2(f, df, df_error, ci = 0.9, squared = TRUE, ...)
t_to_f2(t, df_error, ci = 0.9, squared = TRUE, ...)
```

Arguments

df, df_error	Degrees of freedom of numerator or of the error estimate (i.e., the residuals).
ci	Confidence Interval (CI) level
...	Arguments passed to or from other methods.
t, f	The t or the F statistics.
squared	Return Cohen's <i>f</i> or Cohen's <i>f</i> -squared?

Details

These functions use the following formulae:

$$\eta_p^2 = \frac{F \times df_{num}}{F \times df_{num} + df_{den}}$$

$$\epsilon_p^2 = \frac{(F - 1) \times df_{num}}{F \times df_{num} + df_{den}}$$

$$\omega_p^2 = \frac{(F - 1) \times df_{num}}{F \times df_{num} + df_{den} + 1}$$

$$f_p = \sqrt{\frac{\eta_p^2}{1 - \eta_p^2}}$$

For t , the conversion is based on the equality of $t^2 = F$ when $df_{num} = 1$.

Choosing an Un-Biased Estimate:

Both Omega and Epsilon are unbiased estimators of the population Eta. But which to choose? Though Omega is the more popular choice, it should be noted that:

1. The formula given above for Omega is only an approximation for complex designs.
2. Epsilon has been found to be less biased (Carroll & Nordholm, 1975).

Value

A data frame with the effect size(s) between 0-1 (Eta2_partial, Epsilon2_partial, Omega2_partial, Cohens_f_partial or Cohens_f2_partial), and their CIs (CI_low and CI_high). (Note that for ω_p^2 and ϵ_p^2 it is possible to compute a negative number; even though this doesn't make any practical sense, it is recommended to report the negative number and not a 0).

Confidence Intervals

Unless stated otherwise, confidence intervals are estimated using the Noncentrality parameter method; These methods searches for a the best non-central parameters (ncps) of the noncentral t-, F- or Chi-squared distribution for the desired tail-probabilities, and then convert these ncps to the corresponding effect sizes. (See full [effectsize-CIs](#) for more.)

CI Contains Zero

For positive only effect sizes (Eta squared, Cramer's V, etc.; Effect sizes associated with Chi-squared and F distributions), special care should be taken when interpreting CIs with a lower bound equal to 0, and even more care should be taken when the *upper* bound is equal to 0 (Steiger, 2004; Morey et al., 2016). For example:

```
eta_squared(aov(mpg ~ factor(gear) + factor(cyl), mtcars[1:7, ]))
```

```
## Parameter | Eta2 (partial) | 90% CI
## -----
## factor(gear) | 0.58 | [0.00, 0.84]
## factor(cyl) | 0.46 | [0.00, 0.78]
```

Note

$Adj.\eta_p^2$ is an alias for ϵ_p^2 .

References

- Albers, C., & Lakens, D. (2018). When power analyses based on pilot data are biased: Inaccurate effect size estimators and follow-up bias. *Journal of experimental social psychology*, 74, 187-195. doi: [10.31234/osf.io/b7z4q](https://doi.org/10.31234/osf.io/b7z4q)
- Carroll, R. M., & Nordholm, L. A. (1975). Sampling Characteristics of Kelley's epsilon and Hays' omega. *Educational and Psychological Measurement*, 35(3), 541-554.
- Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement*, 61(4), 532-574.
- Friedman, H. (1982). Simplified determinations of statistical power, magnitude of effect and research sample sizes. *Educational and Psychological Measurement*, 42(2), 521-526. doi: [10.1177/001316448204200214](https://doi.org/10.1177/001316448204200214)
- Mordkoff, J. T. (2019). A Simple Method for Removing Bias From a Popular Measure of Standardized Effect Size: Adjusted Partial Eta Squared. *Advances in Methods and Practices in Psychological Science*, 2(3), 228-232. doi: [10.1177/2515245919855053](https://doi.org/10.1177/2515245919855053)
- Morey, R. D., Hoekstra, R., Rouder, J. N., Lee, M. D., & Wagenmakers, E. J. (2016). The fallacy of placing confidence in confidence intervals. *Psychonomic bulletin & review*, 23(1), 103-123.
- Steiger, J. H. (2004). Beyond the F test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods*, 9, 164-182.

See Also

[eta_squared\(\)](#) for more details.

Other effect size from test statistic: [chisq_to_phi\(\)](#), [t_to_d\(\)](#)

Examples

```
if (require("afex")) {
  data(md_12.1)
  aov_ez("id", "rt", md_12.1,
    within = c("angle", "noise"),
    anova_table = list(correction = "none", es = "pes")
  )
}
# compare to:
(etas <- F_to_eta2(
```

```

    f = c(40.72, 33.77, 45.31),
    df = c(2, 1, 2),
    df_error = c(18, 9, 18)
  ))

  if (require(see)) plot(etas)

  if (require("lmerTest")) { # for the df_error
    fit <- lmer(extra ~ group + (1 | ID), sleep)
    # anova(fit)
    # #> Type III Analysis of Variance Table with Satterthwaite's method
    # #>      Sum Sq Mean Sq NumDF DenDF F value  Pr(>F)
    # #> group 12.482  12.482     1     9  16.501 0.002833 **
    # #> ---
    # #> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

    F_to_eta2(16.501, 1, 9)
    F_to_omega2(16.501, 1, 9)
    F_to_epsilon2(16.501, 1, 9)
    F_to_f(16.501, 1, 9)
  }

  ## Use with emmeans based contrasts
  ## -----
  if (require(emmeans)) {
    warp.lm <- lm(breaks ~ wool * tension, data = warpbreaks)

    jt <- joint_tests(warp.lm, by = "wool")
    F_to_eta2(jt$F.ratio, jt$df1, jt$df2)
  }

```

hardlyworking

Workers' salary and other information

Description

A sample (simulated) dataset, used in tests and some examples.

Format

A data frame with 500 rows and 5 variables:

salary Salary, in Shmekels

xtra_hours Number of overtime hours (on average, per week)

n_comps Number of compliments given to the boss (observed over the last week)

age Age in years

seniority How many years with the company

interpret	<i>Generic function for interpretation</i>
-----------	--

Description

Interpret a value based on a set of rules. See [rules\(\)](#).

Usage

```
interpret(x, ...)

## S3 method for class 'numeric'
interpret(x, rules, name = attr(rules, "rule_name"), ...)
```

Arguments

x	Vector of value break points (edges defining categories).
...	Currently not used.
rules	Set of rules() .
name	Name of the set of rules (stored as a 'rule_name' attribute).

See Also

[rules](#)

Examples

```
rules_grid <- rules(c(0.01, 0.05), c("very significant", "significant", "not significant"))
interpret(0.001, rules_grid)
interpret(0.021, rules_grid)
interpret(0.08, rules_grid)
interpret(c(0.01, 0.005, 0.08), rules_grid)

interpret(c(0.35, 0.15), c("small" = 0.2, "large" = 0.4), name = "Cohen's Rules")
interpret(c(0.35, 0.15), rules(c(0.2, 0.4), c("small", "medium", "large")))
```

interpret_bf	<i>Interpret Bayes Factor (BF)</i>
--------------	------------------------------------

Description

Interpret Bayes Factor (BF)

Usage

```
interpret_bf(
  bf,
  rules = "jeffreys1961",
  include_value = FALSE,
  protect_ratio = TRUE,
  exact = TRUE
)
```

Arguments

bf	Value or vector of Bayes factor (BF) values.
rules	Can be "jeffreys1961" (default), "raftery1995" or custom set of <code>rules()</code> (for the <i>absolute magnitude</i> of evidence).
include_value	Include the value in the output.
protect_ratio	Should values smaller than 1 be represented as ratios?
exact	Should very large or very small values be reported with a scientific format (e.g., 4.24e5), or as truncated values (as "> 1000" and "< 1/1000").

Details

Argument names can be partially matched.

Rules

Rules apply to BF as ratios, so BF of 10 is as extreme as a BF of 0.1 (1/10).

- Jeffreys (1961) ("jeffreys1961"; default)
 - **BF = 1** - No evidence
 - **1 < BF <= 3** - Anecdotal
 - **3 < BF <= 10** - Moderate
 - **10 < BF <= 30** - Strong
 - **30 < BF <= 100** - Very strong
 - **BF > 100** - Extreme.
- Raftery (1995) ("raftery1995")
 - **BF = 1** - No evidence
 - **1 < BF <= 3** - Weak
 - **3 < BF <= 20** - Positive
 - **20 < BF <= 150** - Strong
 - **BF > 150** - Very strong

References

- Jeffreys, H. (1961), Theory of Probability, 3rd ed., Oxford University Press, Oxford.
- Raftery, A. E. (1995). Bayesian model selection in social research. Sociological methodology, 25, 111-164.
- Jarosz, A. F., & Wiley, J. (2014). What are the odds? A practical guide to computing and reporting Bayes factors. The Journal of Problem Solving, 7(1), 2.

Examples

```
interpret_bf(1)
interpret_bf(c(5, 2))
```

interpret_d	<i>Interpret standardized differences</i>
-------------	---

Description

Interpretation of standardized differences using different sets of rules of thumb.

Usage

```
interpret_d(d, rules = "cohen1988", ...)
interpret_g(g, rules = "cohen1988")
interpret_delta(delta, rules = "cohen1988")
```

Arguments

d, g, delta	Value or vector of effect size values.
rules	Can be "cohen1988" (default), "gignac2016", "sawilowsky2009" or custom set of <code>rules()</code> .
...	Not directly used.

Rules

Rules apply to equally to positive and negative *d*.

- Cohen (1988) ("cohen1988"; default)
 - $d < 0.2$ - Very small
 - $0.2 \leq d < 0.5$ - Small
 - $0.5 \leq d < 0.8$ - Medium
 - $d \geq 0.8$ - Large
- Sawilowsky (2009) ("sawilowsky2009")
 - $d < 0.1$ - Tiny

- $0.1 \leq d < 0.2$ - Very small
 - $0.2 \leq d < 0.5$ - Small
 - $0.5 \leq d < 0.8$ - Medium
 - $0.8 \leq d < 1.2$ - Large
 - $1.2 \leq d < 2$ - Very large
 - $d \geq 2$ - Huge
- Gignac & Szodorai (2016) ("gignac2016", based on the `d_to_r()` conversion, see `interpret_r()`)
 - $d < 0.2$ - Very small
 - $0.2 \leq d < 0.41$ - Small
 - $0.41 \leq d < 0.63$ - Moderate
 - $d \geq 0.63$ - Large

References

- Gignac, G. E., & Szodorai, E. T. (2016). Effect size guidelines for individual differences researchers. *Personality and individual differences*, 102, 74-78.
- Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd Ed.). New York: Routledge.
- Sawilowsky, S. S. (2009). New effect size rules of thumb.

Examples

```
interpret_d(.02)
interpret_d(c(.5, .02))
```

```
interpret_direction    Interpret direction
```

Description

Interpret direction

Usage

```
interpret_direction(x)
```

Arguments

x Numeric value.

Examples

```
interpret_direction(.02)
interpret_direction(c(.5, -.02))
#
```

interpret_ess	<i>Interpret Bayesian diagnostic indices</i>
---------------	--

Description

Interpretation of Bayesian diagnostic indices, such as Effective Sample Size (ESS) and Rhat.

Usage

```
interpret_ess(ess, rules = "burkner2017")
```

```
interpret_rhat(rhat, rules = "vehtari2019")
```

Arguments

ess	Value or vector of Effective Sample Size (ESS) values.
rules	A character string (see <i>Rules</i>) or a custom set of <code>rules()</code> .
rhat	Value or vector of Rhat values.

Rules

ESS:

- Bürkner, P. C. (2017) ("burkner2017"; default)
 - **ESS < 1000** - Insufficient
 - **ESS >= 1000** - Sufficient

Rhat:

- Vehtari et al. (2019) ("vehtari2019"; default)
 - **Rhat < 1.01** - Converged
 - **Rhat >= 1.01** - Failed
- Gelman & Rubin (1992) ("gelman1992")
 - **Rhat < 1.1** - Converged
 - **Rhat >= 1.1** - Failed

References

- Bürkner, P. C. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1), 1-28.
- Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4), 457-472.
- Vehtari, A., Gelman, A., Simpson, D., Carpenter, B., & Bürkner, P. C. (2019). Rank-normalization, folding, and localization: An improved Rhat for assessing convergence of MCMC. arXiv preprint arXiv:1903.08008.

Examples

```
interpret_ess(1001)
interpret_ess(c(852, 1200))

interpret_rhat(1.00)
interpret_rhat(c(1.5, 0.9))
```

interpret_gfi	<i>Interpret of indices of CFA / SEM goodness of fit</i>
---------------	--

Description

Interpretation of indices of fit found in confirmatory analysis or structural equation modelling, such as RMSEA, CFI, NFI, IFI, etc.

Usage

```
interpret_gfi(x, rules = "default")
interpret_agfi(x, rules = "default")
interpret_nfi(x, rules = "byrne1994")
interpret_nnfi(x, rules = "byrne1994")
interpret_cfi(x, rules = "default")
interpret_rmsea(x, rules = "default")
interpret_srmr(x, rules = "default")
interpret_rfi(x, rules = "default")
interpret_ifi(x, rules = "default")
interpret_pnfi(x, rules = "default")

## S3 method for class 'lavaan'
interpret(x, ...)

## S3 method for class 'performance_lavaan'
interpret(x, ...)
```

Arguments

x	vector of values, or an object of class lavaan.
rules	Can be "default" or custom set of rules() .
...	Currently not used.

Details

Indices of fit:

- **Chisq:** The model Chi-squared assesses overall fit and the discrepancy between the sample and fitted covariance matrices. Its p-value should be $> .05$ (i.e., the hypothesis of a perfect fit cannot be rejected). However, it is quite sensitive to sample size.
- **GFI/AGFI:** The (Adjusted) Goodness of Fit is the proportion of variance accounted for by the estimated population covariance. Analogous to R^2 . The GFI and the AGFI should be $> .95$ and $> .90$, respectively.
- **NFI/NNFI/TLI:** The (Non) Normed Fit Index. An NFI of 0.95, indicates the model of interest improves the fit by 95\
- **CFI:** The Comparative Fit Index is a revised form of NFI. Not very sensitive to sample size (Fan, Thompson, & Wang, 1999). Compares the fit of a target model to the fit of an independent, or null, model. It should be $> .90$.
- **RMSEA:** The Root Mean Square Error of Approximation is a parsimony-adjusted index. Values closer to 0 represent a good fit. It should be $< .08$ or $< .05$. The p-value printed with it tests the hypothesis that RMSEA is less than or equal to $.05$ (a cutoff sometimes used for good fit), and thus should be not significant.
- **RMR/SRMR:** the (Standardized) Root Mean Square Residual represents the square-root of the difference between the residuals of the sample covariance matrix and the hypothesized model. As the RMR can be sometimes hard to interpret, better to use SRMR. Should be $< .08$.
- **RFI:** the Relative Fit Index, also known as RHO1, is not guaranteed to vary from 0 to 1. However, RFI close to 1 indicates a good fit.
- **IFI:** the Incremental Fit Index (IFI) adjusts the Normed Fit Index (NFI) for sample size and degrees of freedom (Bollen's, 1989). Over 0.90 is a good fit, but the index can exceed 1.
- **PNFI:** the Parsimony-Adjusted Measures Index. There is no commonly agreed-upon cutoff value for an acceptable model for this index. Should be > 0.50 .

See the documentation for [fitmeasures\(\)](#).

What to report:

For structural equation models (SEM), Kline (2015) suggests that at a minimum the following indices should be reported: The model **chi-square**, the **RMSEA**, the **CFI** and the **SRMR**.

Note

When possible, it is recommended to report dynamic cutoffs of fit indices. See <https://dynamicfit.app/cfa/>.

References

- Awang, Z. (2012). A handbook on SEM. Structural equation modeling.
- Byrne, B. M. (1994). Structural equation modeling with EQS and EQS/Windows. Thousand Oaks, CA: Sage Publications.
- Tucker, L. R., & Lewis, C. (1973). The reliability coefficient for maximum likelihood factor analysis. *Psychometrika*, 38, 1-10.
- Schumacker, R. E., & Lomax, R. G. (2004). A beginner's guide to structural equation modeling, Second edition. Mahwah, NJ: Lawrence Erlbaum Associates.

- Fan, X., B. Thompson, & L. Wang (1999). Effects of sample size, estimation method, and model specification on structural equation modeling fit indexes. *Structural Equation Modeling*, 6, 56-83.
- Kline, R. B. (2015). *Principles and practice of structural equation modeling*. Guilford publications.

Examples

```
interpret_gfi(c(.5, .99))
interpret_agfi(c(.5, .99))
interpret_nfi(c(.5, .99))
interpret_nnfi(c(.5, .99))
interpret_cfi(c(.5, .99))
interpret_rmsea(c(.07, .04))
interpret_srmr(c(.5, .99))
interpret_rfi(c(.5, .99))
interpret_ifi(c(.5, .99))
interpret_pnfi(c(.5, .99))

# Structural Equation Models (SEM)
if (require("lavaan")) {
  structure <- " ind60 =~ x1 + x2 + x3
               dem60 =~ y1 + y2 + y3
               dem60 ~ ind60 "
  model <- lavaan::sem(structure, data = PoliticalDemocracy)
  # interpret(model) # Not working until new performance is up
}
```

interpret_oddsratio *Interpret Odds ratio*

Description

Interpret Odds ratio

Usage

```
interpret_oddsratio(OR, rules = "chen2010", log = FALSE)
```

Arguments

OR	Value or vector of (log) odds ratio values.
rules	Can be "chen2010" (default), "cohen1988" (through transformation to standardized difference, see odds_to_d()) or custom set of rules() .
log	Are the provided values log odds ratio.

Rules

Rules apply to OR as ratios, so OR of 10 is as extreme as a OR of 0.1 (1/10).

- Chen et al. (2010) ("chen2010"; default)
 - **OR < 1.68** - Very small
 - **1.68 <= OR < 3.47** - Small
 - **3.47 <= OR < 6.71** - Medium
 - ****OR >= 6.71 **** - Large
- Cohen (1988) ("cohen1988", based on the `oddsratio_to_d()` conversion, see `interpret_d()`)
 - **OR < 1.44** - Very small
 - **1.44 <= OR < 2.48** - Small
 - **2.48 <= OR < 4.27** - Medium
 - ****OR >= 4.27 **** - Large

References

- Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd Ed.). New York: Routledge.
- Chen, H., Cohen, P., & Chen, S. (2010). How big is a big odds ratio? Interpreting the magnitudes of odds ratios in epidemiological studies. *Communications in Statistics—Simulation and Computation*, 39(4), 860-864.
- Sánchez-Meca, J., Marín-Martínez, F., & Chacón-Moscoso, S. (2003). Effect-size indices for dichotomized outcomes in meta-analysis. *Psychological methods*, 8(4), 448.

Examples

```
interpret_oddsratio(1)
interpret_oddsratio(c(5, 2))
```

```
interpret_omega_squared
      Interpret ANOVA effect size
```

Description

Interpret ANOVA effect size

Usage

```
interpret_omega_squared(es, rules = "field2013")
interpret_eta_squared(es, rules = "field2013")
interpret_epsilon_squared(es, rules = "field2013")
```

Arguments

- es Value or vector of eta / omega / epsilon squared values.
- rules Can be "field2013" (default), "cohen1992" or custom set of `rules()`.

Rules

- Field (2013) ("field2013"; default)
 - **ES < 0.01** - Very small
 - **0.01 <= ES < 0.06** - Small
 - **0.16 <= ES < 0.14** - Medium
 - ****ES >= 0.14 **** - Large
- Cohen (1992) ("cohen1992") applicable to one-way anova, or to *partial* eta / omega / epsilon squared in multi-way anova.
 - **ES < 0.02** - Very small
 - **0.02 <= ES < 0.13** - Small
 - **0.13 <= ES < 0.26** - Medium
 - **ES >= 0.26** - Large

References

- Field, A (2013) Discovering statistics using IBM SPSS Statistics. Fourth Edition. Sage:London.
- Cohen, J. (1992). A power primer. Psychological bulletin, 112(1), 155.

See Also

<http://imaging.mrc-cbu.cam.ac.uk/statswiki/FAQ/effectSize>

Examples

```
interpret_eta_squared(.02)
interpret_eta_squared(c(.5, .02), rules = "cohen1992")
```

interpret_p

Interpret p-values

Description

Interpret p-values

Usage

```
interpret_p(p, rules = "default")
```


Arguments

p	Value or vector of p-values.
rules	Can be "default", "rss" (for <i>Redefine statistical significance</i> rules) or custom set of <code>rules()</code> .

Rules

- Default
 - $p \geq 0.05$ - Not significant
 - $p < 0.05$ - Significant
- Benjamin et al. (2018) ("rss")
 - $p \geq 0.05$ - Not significant
 - $0.005 \leq p < 0.05$ - Suggestive
 - $p < 0.005$ - Significant

References

- Benjamin, D. J., Berger, J. O., Johannesson, M., Nosek, B. A., Wagenmakers, E. J., Berk, R., ... & Cesarini, D. (2018). Redefine statistical significance. *Nature Human Behaviour*, 2(1), 6-10.

Examples

```
interpret_p(c(.5, .02, 0.001))
interpret_p(c(.5, .02, 0.001), rules = "rss")
```

interpret_parameters *Interpret of standardized slopes*

Description

Automated interpretation of standardized slopes.

Usage

```
interpret_parameters(model, ...)

## S3 method for class 'lm'
interpret_parameters(
  model,
  parameters = NULL,
  interpretation = "funder2019",
  standardize_method = "refit",
  standardize_robust = FALSE,
  ...
)
```

Arguments

model	A statistical model.
...	For <code>standardize_parameters()</code> , arguments passed to <code>parameters::model_parameters</code> , such as: <ul style="list-style-type: none"> • <code>ci_method</code>, centrality for Bayesian models... • <code>df_method</code> for Mixed models ... • <code>exponentiate</code>, ... • etc.
parameters	A custom parameters table. If NULL, will use <code>standardize_parameters()</code> to get it.
interpretation	Interpretation grid (i.e., the set of rules of thumb) used to interpret the effects.
standardize_method	See <code>standardize_parameters()</code> .
standardize_robust	See <code>standardize_parameters()</code> .

 interpret_r

Interpret correlation

Description

Interpret correlation

Usage

```
interpret_r(r, rules = "funder2019")
```

Arguments

r	Value or vector of correlation coefficient.
rules	Can be "funder2019" (default), "gignac2016", "cohen1988", "evans1996" or custom set of <code>rules()</code> .

RulesRules apply positive and negative *r* alike.

- Funder & Ozer (2019) ("funder2019"; default)
 - **r < 0.05** - Tiny
 - **0.05 <= r < 0.1** - Very small
 - **0.1 <= r < 0.2** - Small
 - **0.2 <= r < 0.3** - Medium
 - **0.3 <= r < 0.4** - Large
 - **r >= 0.4** - Very large

- Gignac & Szodorai (2016) ("gignac2016")
 - $r < 0.1$ - Very small
 - $0.1 \leq r < 0.2$ - Small
 - $0.2 \leq r < 0.3$ - Moderate
 - $r \geq 0.3$ - Large
- Cohen (1988) ("cohen1988")
 - $r < 0.1$ - Very small
 - $0.1 \leq r < 0.3$ - Small
 - $0.3 \leq r < 0.5$ - Moderate
 - $r \geq 0.5$ - Large
- Evans (1996) ("evans1996")
 - $r < 0.2$ - Very weak
 - $0.2 \leq r < 0.4$ - Weak
 - $0.4 \leq r < 0.6$ - Moderate
 - $0.6 \leq r < 0.8$ - Strong
 - $r \geq 0.8$ - Very strong

References

- Funder, D. C., & Ozer, D. J. (2019). Evaluating effect size in psychological research: sense and nonsense. *Advances in Methods and Practices in Psychological Science*.
- Gignac, G. E., & Szodorai, E. T. (2016). Effect size guidelines for individual differences researchers. *Personality and individual differences*, 102, 74-78.
- Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd Ed.). New York: Routledge.
- Evans, J. D. (1996). *Straightforward statistics for the behavioral sciences*. Thomson Brooks/Cole Publishing Co.

See Also

Page 88 of APA's 6th Edition.

Examples

```
interpret_r(.015)
interpret_r(c(.5, -.02))
```

interpret_r2

*Interpret coefficient of determination (R2)***Description**

Interpret coefficient of determination (R2)

Usage

interpret_r2(r2, rules = "cohen1988")

Arguments

r2	Value or vector of R2 values.
rules	Can be "cohen1988" (default), "falk1992", "chin1998", "hair2011" or custom set of <code>rules()</code>].

Rules**For Linear Regression:**

- Cohen (1988) ("cohen1988"; default)
 - $R^2 < 0.02$ - Very weak
 - $0.02 \leq R^2 < 0.13$ - Weak
 - $0.13 \leq R^2 < 0.26$ - Moderate
 - $R^2 \geq 0.26$ - Substantial
- Falk & Miller (1992) ("falk1992")
 - $R^2 < 0.1$ - Negligible
 - $R^2 \geq 0.1$ - Adequate

For PLS / SEM R-Squared of *latent* variables:

- Chin, W. W. (1998) ("chin1998")
 - $R^2 < 0.19$ - Very weak
 - $0.19 \leq R^2 < 0.33$ - Weak
 - $0.33 \leq R^2 < 0.67$ - Moderate
 - $R^2 \geq 0.67$ - Substantial
- Hair et al. (2011) ("hair2011")
 - $R^2 < 0.25$ - Very weak
 - $0.25 \leq R^2 < 0.50$ - Weak
 - $0.50 \leq R^2 < 0.75$ - Moderate
 - $R^2 \geq 0.75$ - Substantial

References

- Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd Ed.). New York: Routledge.
- Falk, R. F., & Miller, N. B. (1992). A primer for soft modeling. University of Akron Press.
- Chin, W. W. (1998). The partial least squares approach to structural equation modeling. Modern methods for business research, 295(2), 295-336.
- Hair, J. F., Ringle, C. M., & Sarstedt, M. (2011). PLS-SEM: Indeed a silver bullet. Journal of Marketing theory and Practice, 19(2), 139-152.

Examples

```
interpret_r2(.02)
interpret_r2(c(.5, .02))
```

interpret_rope	<i>Interpret Bayesian diagnostic indices</i>
----------------	--

Description

Interpretation of Bayesian indices of percentage in ROPE.

Usage

```
interpret_rope(rope, ci = 0.9, rules = "default")
```

Arguments

rope	Value or vector of percentages in ROPE.
ci	The Credible Interval (CI) probability, corresponding to the proportion of HDI, that was used. Can be 1 in the case of "full ROPE".
rules	A character string (see details) or a custom set of <code>rules()</code> .

Rules

- Default
 - For $CI < 1$
 - * **Rope = 0** - Significant
 - * **0 < Rope < 1** - Undecided
 - * **Rope = 1** - Negligible
 - For $CI = 1$
 - * **Rope < 0.01** - Significant
 - * **0.01 < Rope < 0.025** - Probably significant
 - * **0.025 < Rope < 0.975** - Undecided
 - * **0.975 < Rope < 0.99** - Probably negligible
 - * **Rope > 0.99** - Negligible

References

[BayestestR's reporting guidelines](#)

Examples

```
interpret_rope(0, ci = 0.9)
interpret_rope(c(0.005, 0.99), ci = 1)
```

is_effectsize_name	<i>Checks if character is of a supported effect size</i>
--------------------	--

Description

For use by other functions and packages.

Usage

```
is_effectsize_name(x, ignore_case = TRUE)

get_effectsize_label(x, ignore_case = TRUE)
```

Arguments

x	A character, or a vector.
ignore_case	Should case of input be ignored?

normalize	<i>Normalize numeric variable to [0-1] range</i>
-----------	--

Description

Performs a normalization of data, i.e., it scales variables in the range 0 -

1. This is a special case of [change_scale\(\)](#).

Usage

```
normalize(x, ...)
```

```
## S3 method for class 'numeric'
normalize(x, include_bounds = TRUE, verbose = TRUE, ...)
```

```
## S3 method for class 'grouped_df'
normalize(
  x,
```

```

    select = NULL,
    exclude = NULL,
    include_bounds = TRUE,
    verbose = TRUE,
    ...
)

## S3 method for class 'data.frame'
normalize(
  x,
  select = NULL,
  exclude = NULL,
  include_bounds = TRUE,
  verbose = TRUE,
  ...
)

```

Arguments

<code>x</code>	A numeric vector, data frame, or matrix. See details.
<code>...</code>	Arguments passed to or from other methods.
<code>include_bounds</code>	Logical, if TRUE, return value may include 0 and 1. If FALSE, the return value is compressed, using Smithson and Verkuilen's (2006) formula $(x * (n - 1) + 0.5) / n$, to avoid zeros and ones in the normalized variables. This can be useful in case of beta-regression, where the response variable is not allowed to include zeros and ones.
<code>verbose</code>	Toggle warnings and messages on or off.
<code>select</code>	Character vector of column names. If NULL (the default), all variables will be selected.
<code>exclude</code>	Character vector of column names to be excluded from selection.

Details

- If `x` is a matrix, normalization is performed across all values (not column- or row-wise). For column-wise normalization, convert the matrix to a data.frame.
- If `x` is a grouped data frame (`grouped_df`), normalization is performed separately for each group.

Value

A normalized object.

References

- Smithson M, Verkuilen J (2006). A Better Lemon Squeezer? Maximum-Likelihood Regression with Beta-Distributed Dependent Variables. *Psychological Methods*, 11(1), 54–71.

See Also

Other transform utilities: [change_scale\(\)](#), [ranktransform\(\)](#), [standardize\(\)](#)

Examples

```
normalize(c(0, 1, 5, -5, -2))
normalize(c(0, 1, 5, -5, -2), include_bounds = FALSE)

head(normalize(trees))
```

oddsratio_to_riskratio

Convert between Odds ratios and Risk ratios

Description

Convert between Odds ratios and Risk ratios

Usage

```
oddsratio_to_riskratio(OR, p0, log = FALSE)
```

```
riskratio_to_oddsratio(RR, p0, log = FALSE)
```

Arguments

OR, RR	Risk ratio of $p1/p0$ or Odds ratio of $odds(p1)/odds(p0)$, possibly log-ed.
p0	Baseline risk
log	Take in or output the log of the ratio (such as in logistic models).

Value

Converted index.

References

Grant, R. L. (2014). Converting an odds ratio to a range of plausible relative risks for better communication of research findings. *Bmj*, 348, f7450.

See Also

Other convert between effect sizes: [d_to_common_language\(\)](#), [d_to_r\(\)](#), [eta2_to_f2\(\)](#), [odds_to_probs\(\)](#)

Examples

```

p0 <- 0.4
p1 <- 0.7

(OR <- probs_to_odds(p1) / probs_to_odds(p0))
(RR <- p1 / p0)

riskratio_to_oddsratio(RR, p0 = p0)
oddsratio_to_riskratio(OR, p0 = p0)

```

odds_to_probs

*Convert between Odds and Probabilities***Description**

Convert between Odds and Probabilities

Usage

```

odds_to_probs(odds, log = FALSE, ...)

## S3 method for class 'data.frame'
odds_to_probs(odds, log = FALSE, select = NULL, exclude = NULL, ...)

probs_to_odds(probs, log = FALSE, ...)

## S3 method for class 'data.frame'
probs_to_odds(probs, log = FALSE, select = NULL, exclude = NULL, ...)

convert_odds_to_probs(odds, log = FALSE, ...)

convert_probs_to_odds(probs, log = FALSE, ...)

```

Arguments

odds	The <i>Odds</i> (or $\log(\text{odds})$ when $\text{log} = \text{TRUE}$) to convert.
log	Take in or output log odds (such as in logistic models).
...	Arguments passed to or from other methods.
select	When a data frame is passed, character or list of of column names to be transformed.
exclude	When a data frame is passed, character or list of column names to be excluded from transformation.
probs	Probability values to convert.

Value

Converted index.

See Also

`stats::plogis()`

Other convert between effect sizes: `d_to_common_language()`, `d_to_r()`, `eta2_to_f2()`, `oddsratio_to_riskratio()`

Examples

```
odds_to_probs(3)
odds_to_probs(1.09, log = TRUE)

probs_to_odds(0.95)
probs_to_odds(0.95, log = TRUE)
```

phi	<i>Effect size for contingency tables</i>
-----	---

Description

Compute Cramer's V , phi (ϕ), Cohen's w (an alias of phi), Odds ratios, Risk ratios, Cohen's h and Cohen's g for contingency tables or goodness-of-fit. See details.

Usage

```
phi(x, y = NULL, ci = 0.95, adjust = FALSE, CI, ...)
cohens_w(x, y = NULL, ci = 0.95, adjust = FALSE, CI, ...)
cramers_v(x, y = NULL, ci = 0.95, adjust = FALSE, CI, ...)
oddsratio(x, y = NULL, ci = 0.95, log = FALSE, ...)
riskratio(x, y = NULL, ci = 0.95, log = FALSE, ...)
cohens_h(x, y = NULL, ci = 0.95, ...)
cohens_g(x, y = NULL, ci = 0.95, ...)
```

Arguments

x	a numeric vector or matrix. x and y can also both be factors.
y	a numeric vector; ignored if x is a matrix. If x is a factor, y should be a factor of the same length.
ci	Confidence Interval (CI) level
adjust	Should the effect size be bias-corrected? Defaults to FALSE.
CI	Deprecated in favor of ci.
...	Arguments passed to <code>stats::chisq.test()</code> , such as p. Ignored for <code>cohens_g()</code> .
log	Take in or output the log of the ratio (such as in logistic models).

Details

Cramer's V and phi (ϕ) are effect sizes for tests of independence in 2D contingency tables, or for goodness-of-fit in 1D tables. For 2-by-2 tables, they are identical, and are equal to the simple correlation between two dichotomous variables, ranging between 0 (no dependence) and 1 (perfect dependence). For larger tables, Cramer's V should be used, as it is bounded between 0-1, whereas phi can be larger than 1.

For 2-by-2 contingency tables, Odds ratios, Risk ratios and Cohen's h can also be estimated. Note that these are computed with each **column** representing the different groups, and the first column representing the treatment group and the second column baseline (or control). Effects are given as treatment / control. If you wish you use rows as groups you must pass a transposed table, or switch the x and y arguments.

Cohen's g is an effect size for dependent (paired) contingency tables ranging between 0 (perfect symmetry) and 0.5 (perfect asymmetry) (see `stats::mcnemar.test()`).

Value

A data frame with the effect size (Cramers_v, phi (possibly with the suffix `_adjusted`), Odds_ratio, Risk_ratio (possibly with the prefix `log_`), Cohens_h, or Cohens_g) and its CIs (CI_low and CI_high).

Confidence Intervals for Cohen's g, OR, RR and Cohen's h

For Cohen's g , confidence intervals are based on the proportion ($P = g + 0.5$) confidence intervals returned by `stats::prop.test()` (minus 0.5), which give a good close approximation.

For Odds ratios, Risk ratios and Cohen's h , confidence intervals are estimated using the standard normal parametric method (see Katz et al., 1978; Szumilas, 2010).

See *Confidence Intervals* and *CI Contains Zero* sections for *phi*, Cohen's w and Cramer's V .

Confidence Intervals

Unless stated otherwise, confidence intervals are estimated using the Noncentrality parameter method; These methods searches for a the best non-central parameters (ncps) of the noncentral t-, F- or Chi-squared distribution for the desired tail-probabilities, and then convert these ncps to the corresponding effect sizes. (See full [effectsize-CIs](#) for more.)

CI Contains Zero

For positive only effect sizes (Eta squared, Cramer's V , etc.; Effect sizes associated with Chi-squared and F distributions), special care should be taken when interpreting CIs with a lower bound equal to 0, and even more care should be taken when the *upper* bound is equal to 0 (Steiger, 2004; Morey et al., 2016). For example:

```
eta_squared(aov(mpg ~ factor(gear) + factor(cyl), mtcars[1:7, ]))
```

```
## Parameter | Eta2 (partial) | 90% CI
## -----
## factor(gear) | 0.58 | [0.00, 0.84]
## factor(cyl) | 0.46 | [0.00, 0.78]
```

References

- Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd Ed.). New York: Routledge.
- Katz, D. J. S. M., Baptista, J., Azen, S. P., & Pike, M. C. (1978). Obtaining confidence intervals for the risk ratio in cohort studies. *Biometrics*, 469-474.
- Szumilas, M. (2010). Explaining odds ratios. *Journal of the Canadian academy of child and adolescent psychiatry*, 19(3), 227.

See Also

[chisq_to_phi\(\)](#) for details regarding estimation and CIs.

Other effect size indices: [cohens_d\(\)](#), [effectsize\(\)](#), [eta_squared\(\)](#), [rank_biserial\(\)](#), [standardize_parameters\(\)](#)

Examples

```
M <- rbind(
  c(150, 130, 35, 55),
  c(100, 50, 10, 40),
  c(165, 65, 2, 25)
)
dimnames(M) <- list(
  Study = c("Psych", "Econ", "Law"),
  Music = c("Pop", "Rock", "Jazz", "Classic")
)
M

phi(M)

cramers_v(M)

## 2-by-2 tables
## -----
(RCT <- matrix(
  c(71, 30,
    50, 100),
  nrow = 2, byrow = TRUE,
  dimnames = list(Diagnosis = c("Sick", "Recovered"),
                  Group = c("Treatment", "Control"))
)) # note groups are COLUMNS

oddsratio(RCT)

riskratio(RCT)
```

```

cohens_h(RCT)

## Dependent (Paired) Contingency Tables
## -----
Performance <- rbind(
  c(794, 86),
  c(150, 570)
)
dimnames(Performance) <- list(
  "1st Survey" = c("Approve", "Disapprove"),
  "2nd Survey" = c("Approve", "Disapprove")
)
Performance

cohens_g(Performance)

```

plot.effectsize_table *Methods for effectsize tables*

Description

Printing, formatting and plotting methods for effectsize tables.

Usage

```

## S3 method for class 'effectsize_table'
plot(x, ...)

## S3 method for class 'effectsize_table'
print(x, digits = 2, ...)

## S3 method for class 'effectsize_table'
format(x, digits = 2, ...)

## S3 method for class 'effectsize_difference'
print(x, digits = 2, append_CL = FALSE, ...)

```

Arguments

x	Object to print.
...	Arguments passed to or from other functions.
digits	Number of significant digits.
append_CL	Should the Common Language Effect Sizes be printed as well? Only applicable to Cohen's <i>d</i> , Hedges' <i>g</i> for independent samples of equal variance (pooled sd) (See d_to_common_language())

ranktransform *(Signed) rank transformation*

Description

Transform numeric values with the integers of their rank (i.e., 1st smallest, 2nd smallest, 3rd smallest, etc.). Setting the `sign` argument to `TRUE` will give you signed ranks, where the ranking is done according to absolute size but where the sign is preserved (i.e., 2, 1, -3, 4).

Usage

```
ranktransform(x, ...)

## S3 method for class 'numeric'
ranktransform(x, sign = FALSE, method = "average", verbose = TRUE, ...)

## S3 method for class 'grouped_df'
ranktransform(
  x,
  select = NULL,
  exclude = NULL,
  sign = FALSE,
  method = "average",
  ...
)

## S3 method for class 'data.frame'
ranktransform(
  x,
  select = NULL,
  exclude = NULL,
  sign = FALSE,
  method = "average",
  ...
)
```

Arguments

<code>x</code>	Object.
<code>...</code>	Arguments passed to or from other methods.
<code>sign</code>	Logical, if <code>TRUE</code> , return signed ranks.
<code>method</code>	Treatment of ties. Can be one of "average" (default), "first", "last", "random", "max" or "min". See rank() for details.
<code>verbose</code>	Toggle warnings and messages on or off.
<code>select</code>	Character vector of column names. If <code>NULL</code> (the default), all variables will be selected.
<code>exclude</code>	Character vector of column names to be excluded from selection.

Value

A rank-transformed object.

See Also

Other transform utilities: [change_scale\(\)](#), [normalize\(\)](#), [standardize\(\)](#)

Examples

```
ranktransform(c(0, 1, 5, -5, -2))
ranktransform(c(0, 1, 5, -5, -2), sign = TRUE)

head(ranktransform(trees))
```

rank_biserial

Effect size for non-parametric (rank sum) tests

Description

Compute the rank-biserial correlation, Cliff's *delta*, rank Epsilon squared, and Kendall's *W* effect sizes for non-parametric (rank sum) tests.

Usage

```
rank_biserial(
  x,
  y = NULL,
  data = NULL,
  mu = 0,
  ci = 0.95,
  iterations = 200,
  paired = FALSE,
  verbose = TRUE,
  ...
)

cliffs_delta(
  x,
  y = NULL,
  data = NULL,
  mu = 0,
  ci = 0.95,
  iterations = 200,
  verbose = TRUE,
  ...
)
```

```
rank_epsilon_squared(x, groups, data = NULL, ci = 0.95, iterations = 200, ...)
```

```
kendalls_w(x, groups, blocks, data = NULL, ci = 0.95, iterations = 200, ...)
```

Arguments

x	Can be one of: <ul style="list-style-type: none"> • A numeric vector, or a character name of one in data. • A formula in to form of <code>x ~ groups</code> (for <code>rank_biserial()</code> and <code>rank_epsilon_squared()</code>) or <code>x ~ groups blocks</code> (for <code>kendalls_w()</code>). • A list of vectors (for <code>rank_epsilon_squared()</code>). • A matrix of blocks <code>x groups</code> (for <code>kendalls_w()</code>).
y	An optional numeric vector of data values to compare to x, or a character name of one in data. Ignored if x is not a vector.
data	An optional data frame containing the variables.
mu	a number indicating the value around which (a-)symmetry (for one-sample or paired samples) or shift (for independant samples) is to be estimated. See stats::wilcox.test .
ci	Confidence Interval (CI) level
iterations	The number of bootstrap replicates for computing confidence intervals. Only applies when ci is not NULL.
paired	If TRUE, the values of x and y are considered as paired. This produces an effect size that is equivalent to the one-sample effect size on <code>x - y</code> .
verbose	Toggle warnings and messages on or off.
...	Arguments passed to or from other methods.
groups	A vector or factor object giving the group for the corresponding elements of x, or a character name of one in data. Ignored if x is not a vector.
blocks	A vector giving the block for the corresponding elements of x, or a character name of one in data. Ignored if x is not a vector.

Details

Compute effect sizes for non-parametric (rank sum) tests.

The rank-biserial correlation is appropriate for non-parametric tests of differences - both for the one sample or paired samples case, that would normally be tested with Wilcoxon's Signed Rank Test (giving the **matched-pairs** rank-biserial correlation) and for two independent samples case, that would normally be tested with Mann-Whitney's *U* Test (giving **Glass'** rank-biserial correlation). See [stats::wilcox.test](#). In both cases, the correlation represents the difference between the proportion of favorable and unfavorable pairs / signed ranks (Kerby, 2014). Values range from -1 indicating that all values of the second sample are smaller than the first sample, to +1 indicating that all values of the second sample are larger than the first sample. (Cliff's *delta* is an alias to the rank-biserial correlation in the two sample case.)

The rank Epsilon squared is appropriate for non-parametric tests of differences between 2 or more

samples (a rank based ANOVA). See [stats::kruskal.test](#). Values range from 0 to 1, with larger values indicating larger differences between groups.

Kendall's W is appropriate for non-parametric tests of differences between 2 or more dependant samples (a rank based rmANOVA). See [stats::friedman.test](#). Values range from 0 to 1, with larger values indicating larger differences between groups.

Value

A data frame with the effect size (`r_rank_biserial`, `Kendalls_W` or `rank_epsilon_squared`) and its CI (`CI_low` and `CI_high`).

Confidence Intervals

Confidence Intervals are estimated using the bootstrap method.

References

- Cureton, E. E. (1956). Rank-biserial correlation. *Psychometrika*, 21(3), 287-290.
- Glass, G. V. (1965). A ranking variable analogue of biserial correlation: Implications for short-cut item analysis. *Journal of Educational Measurement*, 2(1), 91-95.
- Kendall, M.G. (1948) Rank correlation methods. London: Griffin.
- Kerby, D. S. (2014). The simple difference formula: An approach to teaching nonparametric correlation. *Comprehensive Psychology*, 3, 11-IT.
- King, B. M., & Minium, E. W. (2008). *Statistical reasoning in the behavioral sciences*. John Wiley & Sons Inc.
- Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological bulletin*, 114(3), 494.
- Tomczak, M., & Tomczak, E. (2014). The need to report effect size estimates revisited. An overview of some recommended measures of effect size.

See Also

Other effect size indices: [cohens_d\(\)](#), [effectsize\(\)](#), [eta_squared\(\)](#), [phi\(\)](#), [standardize_parameters\(\)](#)

Examples

```
A <- c(48, 48, 77, 86, 85, 85)
B <- c(14, 34, 34, 77)
rank_biserial(A, B)

x <- c(1.83, 0.50, 1.62, 2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
y <- c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)
rank_biserial(x, y, paired = TRUE)

x <- c(1.15, 0.88, 0.90, 0.74, 1.21)
rank_biserial(x, mu = 1)
```

```
x1 <- c(2.9, 3.0, 2.5, 2.6, 3.2) # normal subjects
x2 <- c(3.8, 2.7, 4.0, 2.4)      # with obstructive airway disease
x3 <- c(2.8, 3.4, 3.7, 2.2, 2.0) # with asbestosis
x <- c(x1, x2, x3)
g <- factor(rep(1:3, c(5, 4, 5)))
rank_epsilon_squared(x, g)

wb <- aggregate(warpbreaks$breaks,
                by = list(w = warpbreaks$wool,
                          t = warpbreaks$tension),
                FUN = mean)
kendalls_w(x ~ w | t, data = wb)
```

 rules

Interpretation Grid

Description

Create a container for interpretation rules of thumb. Usually used in conjunction with [interpret](#).

Usage

```
rules(values, labels = NULL, name = NULL, right = TRUE)
```

```
is.rules(x)
```

Arguments

values	Vector of reference values (edges defining categories or critical values).
labels	Labels associated with each category. If NULL, will try to infer it from values (if it is a named vector or a list), otherwise, will return the breakpoints.
name	Name of the set of rules (stored as a 'rule_name' attribute).
right	logical, for threshold-type rules, indicating if the thresholds themselves should be included in the interval to the right (lower values) or in the interval to the left (higher values).
x	An arbitrary R object.

See Also

[interpret](#)

Examples

```
rules(c(0.05), c("significant", "not significant"), right = FALSE)
rules(c(0.2, 0.5, 0.8), c("small", "medium", "large"))
rules(c("small" = 0.2, "medium" = 0.5), name = "Cohen's Rules")
```

sd_pooled	<i>Pooled Standard Deviation</i>
-----------	----------------------------------

Description

The Pooled Standard Deviation is a weighted average of standard deviations for two or more groups, with more "weight" given to larger sample sizes.

Usage

```
sd_pooled(x, y = NULL, data = NULL, verbose = TRUE)
```

```
mad_pooled(x, y = NULL, data = NULL, verbose = TRUE)
```

Arguments

x	A formula, a numeric vector, or a character name of one in data.
y	A numeric vector, a grouping (character / factor) vector, a or a character name of one in data. Ignored if x is a formula.
data	An optional data frame containing the variables.
verbose	Toggle warnings and messages on or off.

Value

Numeric, the pooled standard deviation.

See Also

[cohens_d\(\)](#)

Examples

```
sd_pooled(mpg ~ am, data = mtcars)
```

standardize	<i>Standardization (Z-scoring)</i>
-------------	------------------------------------

Description

Performs a standardization of data (z-scoring), i.e., centering and scaling, so that the data is expressed in terms of standard deviation (i.e., mean = 0, SD = 1) or Median Absolute Deviance (median = 0, MAD = 1). When applied to a statistical model, this function extracts the dataset, standardizes it, and refits the model with this standardized version of the dataset. The [normalize\(\)](#) function can also be used to scale all numeric variables within the 0 - 1 range.

Usage

```
standardize(  
  x,  
  robust = FALSE,  
  two_sd = FALSE,  
  weights = NULL,  
  verbose = TRUE,  
  ...  
)  
  
## S3 method for class 'numeric'  
standardize(  
  x,  
  robust = FALSE,  
  two_sd = FALSE,  
  weights = NULL,  
  verbose = TRUE,  
  ...  
)  
  
## S3 method for class 'data.frame'  
standardize(  
  x,  
  robust = FALSE,  
  two_sd = FALSE,  
  weights = NULL,  
  verbose = TRUE,  
  select = NULL,  
  exclude = NULL,  
  remove_na = c("none", "selected", "all"),  
  force = FALSE,  
  append = FALSE,  
  suffix = "_z",  
  ...  
)  
  
## Default S3 method:  
standardize(  
  x,  
  robust = FALSE,  
  two_sd = FALSE,  
  weights = TRUE,  
  verbose = TRUE,  
  include_response = TRUE,  
  ...  
)
```

Arguments

<code>x</code>	A data frame, a vector or a statistical model.
<code>robust</code>	Logical, if TRUE, centering is done by subtracting the median from the variables and dividing it by the median absolute deviation (MAD). If FALSE, variables are standardized by subtracting the mean and dividing it by the standard deviation (SD).
<code>two_sd</code>	If TRUE, the variables are scaled by two times the deviation (SD or MAD depending on <code>robust</code>). This method can be useful to obtain model coefficients of continuous parameters comparable to coefficients related to binary predictors, when applied to the predictors (not the outcome) (Gelman, 2008).
<code>weights</code>	Can be NULL (for no weighting), or: <ul style="list-style-type: none"> • For model: if TRUE (default), a weighted-standardization is carried out. • For data.frames: a numeric vector of weights, or a character of the name of a column in the data.frame that contains the weights. • For numeric vectors: a numeric vector of weights.
<code>verbose</code>	Toggle warnings and messages on or off.
<code>...</code>	Arguments passed to or from other methods.
<code>select</code>	Character vector of column names. If NULL (the default), all variables will be selected.
<code>exclude</code>	Character vector of column names to be excluded from selection.
<code>remove_na</code>	How should missing values (NA) be treated: if "none" (default): each column's standardization is done separately, ignoring NAs. Else, rows with NA in the columns selected with <code>select</code> / <code>exclude</code> ("selected") or in all columns ("all") are dropped before standardization, and the resulting data frame does not include these cases.
<code>force</code>	Logical, if TRUE, forces standardization of factors as well. Factors are converted to numerical values, with the lowest level being the value 1 (unless the factor has numeric levels, which are converted to the corresponding numeric value).
<code>append</code>	Logical, if TRUE and <code>x</code> is a data frame, standardized variables will be added as additional columns; if FALSE, existing variables are overwritten.
<code>suffix</code>	Character value, will be appended to variable (column) names of <code>x</code> , if <code>x</code> is a data frame and <code>append</code> = TRUE.
<code>include_response</code>	For a model, if TRUE (default), the response value will also be standardized. If FALSE, only the predictors will be standardized. Note that for certain models (logistic regression, count models, ...), the response value will never be standardized, to make re-fitting the model work. (For mediate models, only applies to the y model; m model's response will always be standardized.)

Value

The standardized object (either a standardize data frame or a statistical model fitted on standardized data).

Model Standardization

If `x` is a model object, standardization is done by completely refitting the model on the standardized data. Hence, this approach is equal to standardizing the variables *before* fitting the model and will return a new model object. However, this method is particularly recommended for complex models that include interactions or transformations (e.g., polynomial or spline terms). The `robust` (default to `FALSE`) argument enables a robust standardization of data, i.e., based on the median and MAD instead of the mean and SD. See [standardize_parameters\(\)](#) for other methods of standardizing model coefficients.

Transformed Variables:

When the model's formula contains transformations (e.g. $y \sim \exp(X)$) the transformation effectively takes place after standardization (e.g., $\exp(\text{scale}(X))$). Some transformations are undefined for negative values, such as `log()` and `sqrt()`. To avoid dropping these values, the standardized data is shifted by $Z - \min(Z) + 1$ or $Z - \min(Z)$ (respectively).

Generalized Linear Models

When standardizing coefficients of a generalized model (GLM, GLMM, etc), only the predictors are standardized, maintaining the interpretability of the coefficients (e.g., in a binomial model: the exponent of the standardized parameter is the OR of a change of 1 SD in the predictor, etc.)

Note

When `x` is a vector or a data frame with `remove_na = "none"`, missing values are preserved, so the return value has the same length / number of rows as the original input.

See Also

Other transform utilities: [change_scale\(\)](#), [normalize\(\)](#), [ranktransform\(\)](#)

Other standardize: [standardize_info\(\)](#), [standardize_parameters\(\)](#)

Examples

```
# Data frames
summary(standardize(swiss))

# Models
model <- lm(Infant.Mortality ~ Education * Fertility, data = swiss)
coef(standardize(model))
```

standardize_info

Get Standardization Information

Description

This function extracts information, such as the deviations (SD or MAD) from parent variables, that are necessary for post-hoc standardization of parameters. This function gives a window on how standardized are obtained, i.e., by what they are divided. The "basic" method of standardization uses

Usage

```
standardize_info(  
  model,  
  robust = FALSE,  
  two_sd = FALSE,  
  include_pseudo = FALSE,  
  ...  
)
```

Arguments

model	A statistical model.
robust	Logical, if TRUE, centering is done by subtracting the median from the variables and dividing it by the median absolute deviation (MAD). If FALSE, variables are standardized by subtracting the mean and dividing it by the standard deviation (SD).
two_sd	If TRUE, the variables are scaled by two times the deviation (SD or MAD depending on robust). This method can be useful to obtain model coefficients of continuous parameters comparable to coefficients related to binary predictors, when applied to the predictors (not the outcome) (Gelman, 2008).
include_pseudo	(For (G)LMMs) Should Pseudo-standardized information be included?
...	Arguments passed to or from other methods.

Value

A data frame with information on each parameter (see [parameters::parameters_type](#)), and various standardization coefficients for the post-hoc methods (see [standardize_parameters\(\)](#)) for the predictor and the response.

See Also

Other standardize: [standardize_parameters\(\)](#), [standardize\(\)](#)

Examples

```
model <- lm(mpg ~ ., data = mtcars)  
standardize_info(model)  
standardize_info(model, robust = TRUE)  
standardize_info(model, two_sd = TRUE)
```

 standardize_parameters

Parameters standardization

Description

Compute standardized model parameters (coefficients).

Usage

```
standardize_parameters(
  model,
  method = "refit",
  ci = 0.95,
  robust = FALSE,
  two_sd = FALSE,
  verbose = TRUE,
  parameters,
  ...
)
```

```
standardize_posteriors(
  model,
  method = "refit",
  robust = FALSE,
  two_sd = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

model	A statistical model.
method	The method used for standardizing the parameters. Can be "refit" (default), "posthoc", "smart", "basic" or "pseudo". See 'Details'.
ci	Confidence Interval (CI) level
robust	Logical, if TRUE, centering is done by subtracting the median from the variables and dividing it by the median absolute deviation (MAD). If FALSE, variables are standardized by subtracting the mean and dividing it by the standard deviation (SD).
two_sd	If TRUE, the variables are scaled by two times the deviation (SD or MAD depending on robust). This method can be useful to obtain model coefficients of continuous parameters comparable to coefficients related to binary predictors, when applied to the predictors (not the outcome) (Gelman, 2008).
verbose	Toggle warnings and messages on or off.

parameters	Deprecated.
...	For <code>standardize_parameters()</code> , arguments passed to <code>parameters::model_parameters</code> , such as: <ul style="list-style-type: none"> • <code>ci_method</code>, centrality for Bayesian models... • <code>df_method</code> for Mixed models ... • <code>exponentiate</code>, ... • <code>etc.</code>

Details

Methods::

- **refit**: This method is based on a complete model re-fit with a standardized version of the data. Hence, this method is equal to standardizing the variables before fitting the model. It is the "purest" and the most accurate (Neter et al., 1989), but it is also the most computationally costly and long (especially for heavy models such as Bayesian models). This method is particularly recommended for complex models that include interactions or transformations (e.g., polynomial or spline terms). The `robust` (default to `FALSE`) argument enables a robust standardization of data, i.e., based on the median and MAD instead of the mean and SD. [See `standardize\(\)` for more details.](#)
- **posthoc**: Post-hoc standardization of the parameters, aiming at emulating the results obtained by "refit" without refitting the model. The coefficients are divided by the standard deviation (or MAD if robust) of the outcome (which becomes their expression 'unit'). Then, the coefficients related to numeric variables are additionally multiplied by the standard deviation (or MAD if robust) of the related terms, so that they correspond to changes of 1 SD of the predictor (e.g., "A change in 1 SD of x is related to a change of 0.24 of the SD of y). This does not apply to binary variables or factors, so the coefficients are still related to changes in levels. This method is not accurate and tend to give aberrant results when interactions are specified.
- **smart** (Standardization of Model's parameters with Adjustment, Reconnaissance and Transformation - *experimental*): Similar to `method = "posthoc"` in that it does not involve model refitting. The difference is that the SD (or MAD if robust) of the response is computed on the relevant section of the data. For instance, if a factor with 3 levels A (the intercept), B and C is entered as a predictor, the effect corresponding to B vs. A will be scaled by the variance of the response at the intercept only. As a results, the coefficients for effects of factors are similar to a Glass' delta.
- **basic**: This method is similar to `method = "posthoc"`, but treats all variables as continuous: it also scales the coefficient by the standard deviation of model's matrix' parameter of factors levels (transformed to integers) or binary predictors. Although being inappropriate for these cases, this method is the one implemented by default in other software packages, such as `lm.beta::lm.beta()`.
- **pseudo** (*for 2-level (G)LMMs only*): In this (post-hoc) method, the response and the predictor are standardized based on the level of prediction (levels are detected with `parameters::check_heterogeneity()`): Predictors are standardized based on their SD at level of prediction (see also `parameters::demean()`); The outcome (in linear LMMs) is standardized based on a fitted random-intercept-model, where `sqrt(random-intercept-variance)` is used for level 2 predictors, and `sqrt(residual-variance)` is used for level 1 predictors (Hoffman 2015, page 342). A warning is given when a within-group variable is found to have access between-group variance.

Transformed Variables:

When the model's formula contains transformations (e.g. $y \sim \exp(X)$) `method = "refit"` might give different results compared to `method = "basic"` ("`posthoc`" and "`smart`" do not support such transformations): where "`refit`" standardizes the data prior to the transformation (e.g. equivalent to `exp(scale(X))`), the "`basic`" method standardizes the transformed data (e.g. equivalent to `scale(exp(X))`). See [standardize\(\)](#) for more details on how different transformations are dealt with.

Value

A data frame with the standardized parameters (`Std_*`, depending on the model type) and their CIs (`CI_low` and `CI_high`). Where applicable, standard errors (SEs) are returned as an attribute (`attr(x, "standard_error")`).

Generalized Linear Models

When standardizing coefficients of a generalized model (GLM, GLMM, etc), only the predictors are standardized, maintaining the interpretability of the coefficients (e.g., in a binomial model: the exponent of the standardized parameter is the OR of a change of 1 SD in the predictor, etc.)

References

- Hoffman, L. (2015). Longitudinal analysis: Modeling within-person fluctuation and change. Routledge.
- Neter, J., Wasserman, W., & Kutner, M. H. (1989). Applied linear regression models.
- Gelman, A. (2008). Scaling regression inputs by dividing by two standard deviations. *Statistics in medicine*, 27(15), 2865-2873.

See Also

[standardize_info\(\)](#)

Other standardize: [standardize_info\(\)](#), [standardize\(\)](#)

Other effect size indices: [cohens_d\(\)](#), [effectsize\(\)](#), [eta_squared\(\)](#), [phi\(\)](#), [rank_biserial\(\)](#)

Examples

```
library(effectsize)

model <- lm(len ~ supp * dose, data = ToothGrowth)
standardize_parameters(model, method = "refit")

standardize_parameters(model, method = "posthoc")
standardize_parameters(model, method = "smart")
standardize_parameters(model, method = "basic")

# Robust and 2 SD
standardize_parameters(model, robust = TRUE)
standardize_parameters(model, two_sd = TRUE)
```

```

model <- glm(am ~ cyl * mpg, data = mtcars, family = "binomial")
standardize_parameters(model, method = "refit")
standardize_parameters(model, method = "posthoc")
standardize_parameters(model, method = "basic", exponentiate = TRUE)

if (require("lme4")) {
  m <- lmer(mpg ~ cyl + am + vs + (1 | cyl), mtcars)
  standardize_parameters(m, method = "pseudo", df_method = "satterthwaite")
}

## Not run:
if (require("rstanarm")) {
  model <- stan_glm(rating ~ critical + privileges, data = attitude, refresh = 0)
  standardize_posteriors(model, method = "refit")
  standardize_posteriors(model, method = "posthoc")
  standardize_posteriors(model, method = "smart")
  head(standardize_posteriors(model, method = "basic"))
}

## End(Not run)

```

t_to_d	<i>Convert test statistics (t, z, F) to effect sizes of differences (Cohen's d) or association (partial r)</i>
--------	---

Description

These functions are convenience functions to convert t, z and F test statistics to Cohen's d and **partial r**. These are useful in cases where the data required to compute these are not easily available or their computation is not straightforward (e.g., in liner mixed models, contrasts, etc.).

See [Effect Size from Test Statistics vignette](#).

Usage

```
t_to_d(t, df_error, paired = FALSE, ci = 0.95, pooled, ...)
```

```
z_to_d(z, n, paired = FALSE, ci = 0.95, pooled, ...)
```

```
F_to_d(f, df, df_error, paired = FALSE, ci = 0.95, pooled, ...)
```

```
t_to_r(t, df_error, ci = 0.95, ...)
```

```
z_to_r(z, n, ci = 0.95, ...)
```

```
F_to_r(f, df, df_error, ci = 0.95, ...)
```

Arguments

t, f, z	The t, the F or the z statistics.
paired	Should the estimate account for the t-value being testing the difference between dependant means?
ci	Confidence Interval (CI) level
pooled	Deprecated. Use paired.
...	Arguments passed to or from other methods.
n	The number of observations (the sample size).
df, df_error	Degrees of freedom of numerator or of the error estimate (i.e., the residuals).

Details

These functions use the following formulae to approximate r and d :

$$r_{partial} = t / \sqrt{t^2 + df_{error}}$$

$$r_{partial} = z / \sqrt{z^2 + N}$$

$$d = 2 * t / \sqrt{df_{error}}$$

$$d_z = t / \sqrt{df_{error}}$$

$$d = 2 * z / \sqrt{N}$$

The resulting d effect size is an *approximation* to Cohen's d , and assumes two equal group sizes. When possible, it is advised to directly estimate Cohen's d , with `cohens_d()`, `emmeans::eff_size()`, or similar functions.

Value

A data frame with the effect size(s) (r or d), and their CIs (CI_low and CI_high).

Confidence Intervals

Unless stated otherwise, confidence intervals are estimated using the Noncentrality parameter method; These methods searches for a the best non-central parameters (ncps) of the noncentral t-, F- or Chi-squared distribution for the desired tail-probabilities, and then convert these ncps to the corresponding effect sizes. (See full [effectsize-CIs](#) for more.)

References

- Friedman, H. (1982). Simplified determinations of statistical power, magnitude of effect and research sample sizes. *Educational and Psychological Measurement*, 42(2), 521-526. doi: [10.1177/001316448204200214](https://doi.org/10.1177/001316448204200214)
- Wolf, F. M. (1986). *Meta-analysis: Quantitative methods for research synthesis* (Vol. 59). Sage.
- Rosenthal, R. (1994) Parametric measures of effect size. In H. Cooper and L.V. Hedges (Eds.). *The handbook of research synthesis*. New York: Russell Sage Foundation.
- Steiger, J. H. (2004). Beyond the F test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods*, 9, 164-182.
- Cumming, G., & Finch, S. (2001). A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement*, 61(4), 532-574.

See Also

Other effect size from test statistic: [F_to_eta2\(\)](#), [chisq_to_phi\(\)](#)

Examples

```
## t Tests
res <- t.test(1:10, y = c(7:20), var.equal = TRUE)
t_to_d(t = res$statistic, res$parameter)
t_to_r(t = res$statistic, res$parameter)

res <- with(sleep, t.test(extra[group == 1], extra[group == 2], paired = TRUE))
t_to_d(t = res$statistic, res$parameter, paired = TRUE)
t_to_r(t = res$statistic, res$parameter)

## Linear Regression
model <- lm(rating ~ complaints + critical, data = attitude)
library(parameters)
(param_tab <- parameters(model))

(rs <- t_to_r(param_tab$t[2:3], param_tab$df_error[2:3]))

if (require(see)) plot(rs)

# How does this compare to actual partial correlations?
if (require("correlation")) {
  correlation::correlation(attitude[, c(1, 2, 6)], partial = TRUE)[1:2, c(2, 3, 7, 8)]
}
```

```
## Use with emmeans based contrasts (see also t_to_eta2)
if (require(emmeans)) {
  warp.lm <- lm(breaks ~ wool * tension, data = warpbreaks)

  # Also see emmeans::eff_size()
  em_tension <- emmeans(warp.lm, ~tension) #'
  diff_tension <- summary(pairs(em_tension))
  t_to_d(diff_tension$t.ratio, diff_tension$df)
}
```

Index

- * **convert between effect sizes**
 - d_to_common_language, 11
 - d_to_r, 12
 - eta2_to_f2, 19
 - odds_to_probs, 49
 - oddsratio_to_riskratio, 48
- * **data**
 - hardlyworking, 30
- * **effect size from test statistic**
 - chisq_to_phi, 6
 - F_to_eta2, 27
 - t_to_d, 67
- * **effect size indices**
 - cohens_d, 8
 - effectsize, 14
 - eta_squared, 20
 - phi, 50
 - rank_biserial, 55
 - standardize_parameters, 64
- * **standardize**
 - standardize, 59
 - standardize_info, 62
 - standardize_parameters, 64
- * **transform utilities**
 - change_scale, 5
 - normalize, 46
 - ranktransform, 54
 - standardize, 59
- adjust, 3
- anova(), 22
- bayestestR::describe_posterior(), 15, 23
- bayestestR::equivalence_test(), 19
- car::Anova(), 22
- change_scale, 5, 48, 55, 62
- change_scale(), 46
- chisq_to_cohens_w(chisq_to_phi), 6
- chisq_to_cramers_v(chisq_to_phi), 6
- chisq_to_phi, 6, 29, 69
- chisq_to_phi(), 52
- cliffs_delta(rank_biserial), 55
- cohens_d, 8, 15, 24, 52, 57, 66
- cohens_d(), 12, 18, 59, 68
- cohens_f(eta_squared), 20
- cohens_f_squared(eta_squared), 20
- cohens_g(phi), 50
- cohens_h(phi), 50
- cohens_w(phi), 50
- convert_d_to_common_language(d_to_common_language), 11
- convert_d_to_odds(d_to_r), 12
- convert_d_to_oddsratio(d_to_r), 12
- convert_d_to_r(d_to_r), 12
- convert_odds_to_d(d_to_r), 12
- convert_odds_to_probs(odds_to_probs), 49
- convert_odds_to_r(d_to_r), 12
- convert_oddsratio_to_d(d_to_r), 12
- convert_oddsratio_to_r(d_to_r), 12
- convert_probs_to_odds(odds_to_probs), 49
- convert_r_to_d(d_to_r), 12
- convert_r_to_odds(d_to_r), 12
- convert_r_to_oddsratio(d_to_r), 12
- cramers_v(phi), 50
- cramers_v(), 7
- d_to_common_language, 11, 14, 20, 48, 50
- d_to_common_language(), 10, 53
- d_to_odds(d_to_r), 12
- d_to_oddsratio(d_to_r), 12
- d_to_r, 12, 12, 20, 48, 50
- d_to_r(), 34
- data_adjust(adjust), 3
- effectsize, 10, 14, 24, 52, 57, 66
- effectsize-CIs, 7, 10, 16, 16, 23, 28, 51, 69

- emmeans::eff_size(), 68
- epsilon_squared(eta_squared), 20
- equivalence_test.effectsize_table, 17
- eta2_to_f(eta2_to_f2), 19
- eta2_to_f2, 12, 14, 19, 48, 50
- eta_squared, 10, 15, 20, 52, 57, 66
- eta_squared(), 18, 20, 29
- eta_squared_posterior(eta_squared), 20
- f2_to_eta2(eta2_to_f2), 19
- F_to_d(t_to_d), 67
- F_to_epsilon2(F_to_eta2), 27
- F_to_eta2, 8, 27, 69
- f_to_eta2(eta2_to_f2), 19
- F_to_eta2(), 22, 24
- F_to_eta2_adj(F_to_eta2), 27
- F_to_f(F_to_eta2), 27
- F_to_f2(F_to_eta2), 27
- F_to_omega2(F_to_eta2), 27
- F_to_r(t_to_d), 67
- F_to_r(), 18
- fitmeasures(), 37
- format.effectsize_table
 - (plot.effectsize_table), 53
- format_standardize, 26
- get_effectsize_label
 - (is_effectsize_name), 46
- glass_delta(cohens_d), 8
- hardlyworking, 30
- hedges_g(cohens_d), 8
- insight::get_data(), 14
- interpret, 31, 58
- interpret.lavaan(interpret_gfi), 36
- interpret.performance_lavaan
 - (interpret_gfi), 36
- interpret_agfi(interpret_gfi), 36
- interpret_bf, 31
- interpret_cfi(interpret_gfi), 36
- interpret_d, 33
- interpret_d(), 39
- interpret_delta(interpret_d), 33
- interpret_direction, 34
- interpret_epsilon_squared
 - (interpret_omega_squared), 39
- interpret_ess, 35
- interpret_eta_squared
 - (interpret_omega_squared), 39
- interpret_g(interpret_d), 33
- interpret_gfi, 36
- interpret_ifi(interpret_gfi), 36
- interpret_nfi(interpret_gfi), 36
- interpret_nnfi(interpret_gfi), 36
- interpret_odds(interpret_oddsratio), 38
- interpret_oddsratio, 38
- interpret_omega_squared, 39
- interpret_p, 40
- interpret_parameters, 41
- interpret_pnfi(interpret_gfi), 36
- interpret_r, 42
- interpret_r(), 34
- interpret_r2, 44
- interpret_rfi(interpret_gfi), 36
- interpret_rhat(interpret_ess), 35
- interpret_rmsea(interpret_gfi), 36
- interpret_rope, 45
- interpret_srmr(interpret_gfi), 36
- is.rules(rules), 58
- is_effectsize_name, 46
- kendalls_w(rank_biserial), 55
- lm.beta::lm.beta(), 65
- logodds_to_d(d_to_r), 12
- logodds_to_r(d_to_r), 12
- logoddsratio_to_d(d_to_r), 12
- logoddsratio_to_r(d_to_r), 12
- mad_pooled(sd_pooled), 59
- normalize, 6, 46, 55, 62
- normalize(), 6, 59
- odds_to_d(d_to_r), 12
- odds_to_d(), 38
- odds_to_probs, 12, 14, 20, 48, 49
- odds_to_r(d_to_r), 12
- oddsratio(phi), 50
- oddsratio_to_d(d_to_r), 12
- oddsratio_to_d(), 39
- oddsratio_to_r(d_to_r), 12
- oddsratio_to_riskratio, 12, 14, 20, 48, 50
- omega_squared(eta_squared), 20
- parameters::check_heterogeneity(), 65
- parameters::demean(), 65
- parameters::model_parameters, 42, 65
- parameters::parameters_type, 63

phi, [10](#), [15](#), [24](#), [50](#), [57](#), [66](#)
phi_to_chisq(chisq_to_phi), [6](#)
plot.effectsize_table, [53](#)
print.effectsize_difference
 (plot.effectsize_table), [53](#)
print.effectsize_table
 (plot.effectsize_table), [53](#)
probs_to_odds(odds_to_probs), [49](#)

r_to_d(d_to_r), [12](#)
r_to_odds(d_to_r), [12](#)
r_to_oddsratio(d_to_r), [12](#)
rank(), [54](#)
rank_biserial, [10](#), [15](#), [24](#), [52](#), [55](#), [66](#)
rank_epsilon_squared(rank_biserial), [55](#)
ranktransform, [6](#), [48](#), [54](#), [62](#)
ranktransform(), [6](#)
riskratio(phi), [50](#)
riskratio_to_oddsratio
 (oddsratio_to_riskratio), [48](#)
rstantools::posterior_predict(), [23](#)
rules, [58](#)
rules(), [31–33](#), [35](#), [36](#), [38](#), [40–42](#), [44](#), [45](#)

sd_pooled, [59](#)
sd_pooled(), [9](#), [10](#)
standardize, [6](#), [48](#), [55](#), [59](#), [63](#), [66](#)
standardize(), [6](#), [65](#), [66](#)
standardize_info, [62](#), [62](#), [66](#)
standardize_info(), [66](#)
standardize_parameters, [10](#), [15](#), [24](#), [52](#), [57](#),
 [62](#), [63](#), [64](#)
standardize_parameters(), [15](#), [42](#), [62](#), [63](#)
standardize_posteriors
 (standardize_parameters), [64](#)
stats::chisq.test(), [50](#)
stats::friedman.test, [57](#)
stats::kruskal.test, [57](#)
stats::mcnemar.test(), [51](#)
stats::plogis(), [50](#)
stats::prop.test(), [51](#)
stats::wilcox.test, [56](#)

t_to_d, [8](#), [29](#), [67](#)
t_to_epsilon2(F_to_eta2), [27](#)
t_to_eta2(F_to_eta2), [27](#)
t_to_eta2_adj(F_to_eta2), [27](#)
t_to_f(F_to_eta2), [27](#)
t_to_f2(F_to_eta2), [27](#)

t_to_omega2(F_to_eta2), [27](#)
t_to_r(t_to_d), [67](#)

z_to_d(t_to_d), [67](#)
z_to_r(t_to_d), [67](#)